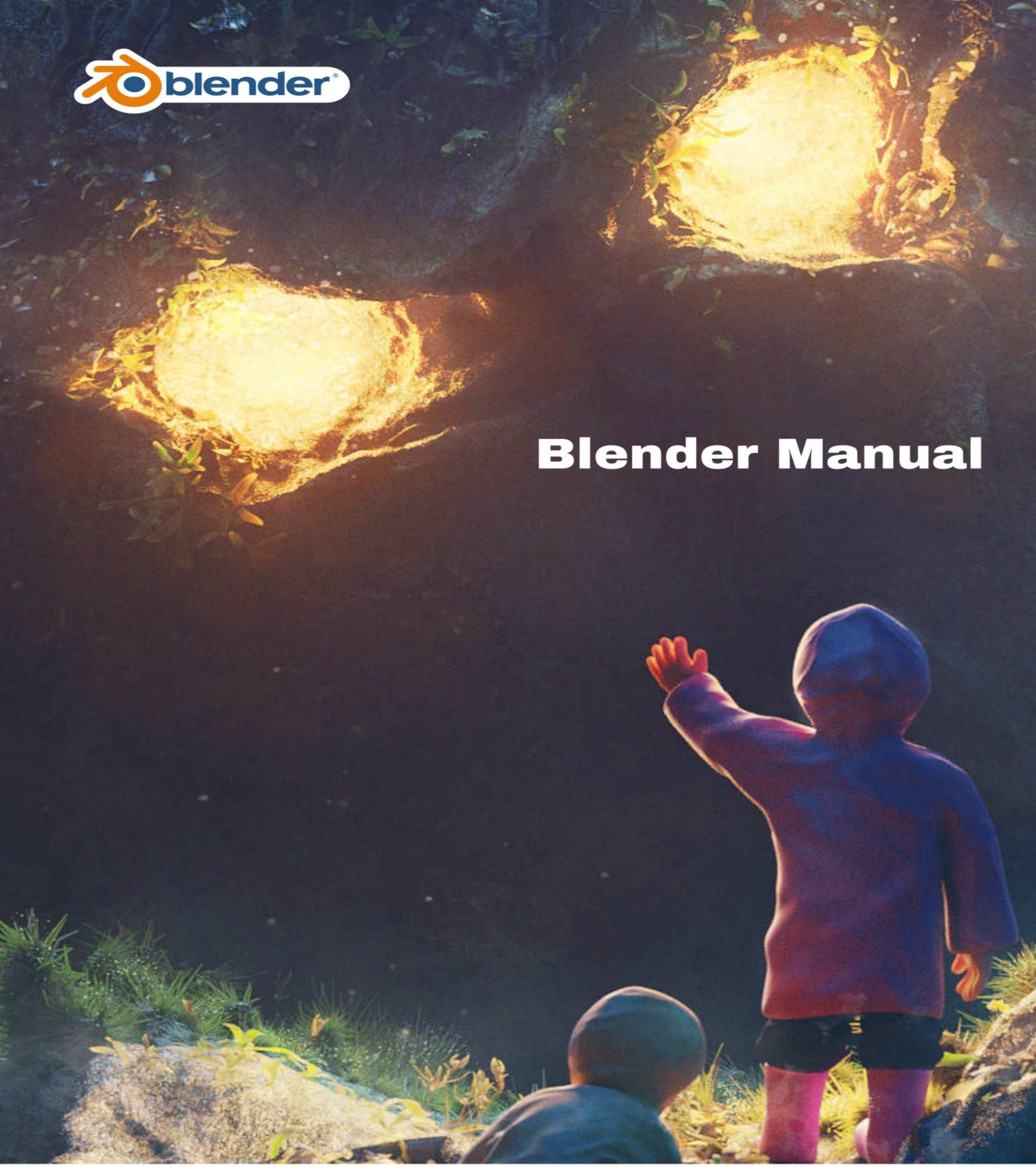


Blender Manual

Blender Documentation Team

Blender Foundation

Blender 4.3 Reference Manual

Welcome to the manual for Blender, the free and open source 3D creation suite.

Getting Started

Getting Started

- About Blender
- Installing Blender
- Configuring Blender
- Help System

Sections

Sections

- User Interface
  - Window System
  - Keymap
  - UI Elements
  - Tools & Operators
  - Nodes
- Editors
  - 3D Viewport
  - Image Editor
  - UV Editor
  - Compositor
  - Texture Nodes
  - Geometry Node Editor
  - Shader Editor
  - Video Sequencer
  - Movie Clip Editor

    - Dope Sheet
    - Timeline
    - Graph Editor
    - Drivers Editor
    - Nonlinear Animation
    - Text Editor
    - Python Console
    - Info Editor
    - Outliner
    - Properties
    - File Browser
    - Asset Browser
    - Spreadsheet
    - Preferences
- Scenes & Objects
    - Scenes
    - Objects
    - Collections
    - View Layers
- Modeling
    - Introduction
    - Meshes
    - Curves
    - Curves (New)
    - Surfaces
    - Metaball
    - Text
    - Volumes
    - Empties
    - Modifiers
    - Geometry Nodes
    - Transform
- Sculpting & Painting
    - Introduction
    - Brushes
    - Selection & Visibility
    - Navigation

    - Modes
- Grease Pencil
    - Introduction
    - Structure
    - Primitives
    - Properties
    - Modifiers
    - Visual Effects
    - Materials
    - Multiframe
    - Animation
    - Object Modes
- Animation & Rigging
    - Introduction
    - Keyframes
    - Armatures
    - Lattice
    - Constraints
    - Actions
    - Drivers
    - Markers
    - Shape Keys
    - Motion Paths
- Physics
    - Introduction
    - Rigid Body
    - Cloth
    - Soft Body
    - Fluid
    - Particle System
    - Dynamic Paint
    - Forces
    - Collision
    - Baking Physics Simulations
    - Simulation Nodes
- Rendering
    - Introduction

    - EEVEE
    - Cycles
    - Workbench
    - Cameras
    - Lights
    - Materials
    - Shader Nodes
    - Color Management
    - Freestyle
    - Layers & Passes
    - Render Output
- Compositing
    - Introduction
    - Sidebar
    - GPU Compositor
    - Node Types
- Motion Tracking & Masking
    - Motion Tracking
    - Masking
- Video Editing
    - Introduction
    - Setup Your Project
    - Edit Your Project
- Assets, Files, & Data System
    - Introduction
    - Blender File
    - Data-Blocks
    - Custom Properties
    - Linked Libraries
    - Asset Libraries
    - Media Formats
    - Importing & Exporting Files
- Add-ons
    - Add-ons Category Listings
- Advanced
    - Using Blender From The Command Line
    - Scripting & Extending Blender

    - Creating Extensions
    - Application Templates
    - Keymap Customization
    - Working Limits
    - Operators
    - Blender's Directory Layout
    - Deploying Blender in Production
    - Appendices
- Troubleshooting
    - Startup
    - 3D Viewport
    - Graphics Hardware
    - Crashes
    - Python Errors
    - Recovering Data
    - Compatibility
- Glossary
- Manual Index

Get Involved

This manual is maintained largely by volunteers.

Please consider to join the effort and Contribute to this Manual.

About Blender

Welcome to Blender! Blender is a free and open-source 3D creation suite.

With Blender, you can create 3D visualizations such as still images, 3D animations and VFX shots. You can also edit videos. It is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process.

Being a cross-platform application, Blender runs on Linux, macOS, as well as Windows systems. It also has relatively small memory and drive requirements compared to other 3D creation suites. Its interface uses OpenGL to provide a consistent experience across all supported hardware and platforms.



Who uses Blender?

Blender has a wide variety of tools making it suitable for almost any sort of media production. Professionals, hobbyists, and studios around the world

use it for creating animations, game assets, motion graphics, TV shows, concept art, story-boarding, commercials, and feature films.

Check out the User Stories page on the Blender website for more examples.

Key Features

- Blender is a fully integrated 3D content creation suite, offering a broad range of essential tools, including Modeling, Rendering, Animation & Rigging, Video Editing, VFX, Compositing, Texturing, and many types of Simulations.
- It is cross platform, with an OpenGL GUI that is uniform on all major platforms (and customizable with Python scripts).
- It has a high-quality 3D architecture, enabling fast and efficient creation workflow.
- It boasts active community support. See blender.org/community for an extensive list of sites.
- It can be installed into and run from any directory without modifying the system.

You can download the latest version of Blender here.



A rendered image being post-processed.

Blender makes it possible to perform a wide range of tasks, and it may seem daunting when first trying to grasp the basics. However, with a bit of motivation and the right learning material, it is possible to familiarize yourself with Blender after a few hours of practice.

This manual is a good start, though it serves more as a reference. There are also many online video tutorials from specialized websites.

Despite everything Blender can do, it remains a tool. Great artists do not create masterpieces by pressing buttons or manipulating brushes, but by learning and practicing subjects such as human anatomy, composition, lighting, animation principles, etc.

3D creation software such as Blender have an added technical complexity and jargon associated with the underlying technologies. Terms like UV maps, materials, shaders, meshes, and "subdivs" are the media of the digital artist, and understanding them, even broadly, will help you to use Blender to its best.

So keep reading this manual, learn the great tool that Blender is, and keep your mind open to other artistic and technological areas – and you, too, can become a great artist.

Further Reading

- Blender's History
  - Version/Revision Milestones
- About Free Software and the GPL
- The Blender Community
  - Independent Sites
  - Getting Support
  - Development
  - Chat
  - Other Useful Links

Blender's History

In 1988, Ton Roosendaal co-founded the Dutch animation studio NeoGeo. NeoGeo quickly became the largest 3D animation studio in the Netherlands and one of the leading animation houses in Europe. NeoGeo created award-winning productions (European Corporate Video Awards 1993 and 1995) for large corporate clients such as the multinational electronics company Philips. Within NeoGeo, Ton was responsible for both art direction and internal software development. After careful deliberation, Ton decided that the current in-house 3D tool set for NeoGeo was too old and cumbersome to maintain, and needed to be rewritten from scratch. In 1995 this rewrite began and was destined to become the 3D software creation we all know as Blender. As NeoGeo continued to refine and improve Blender, it became apparent to Ton that Blender could be used as a tool for other artists outside of NeoGeo.

In 1998, Ton decided to found a new company called Not a Number (NaN) as a spin-off of NeoGeo to further market and develop Blender. At the core of NaN was a desire to create and distribute a compact, cross-platform 3D application for free. At the time, this was a revolutionary concept as most commercial 3D applications cost thousands of dollars. NaN hoped to bring professional level 3D modeling and animation tools within the reach of the general computing public. NaN's business model involved providing commercial products and services around Blender. In 1999 NaN attended its first SIGGRAPH conference in an effort to more widely promote Blender. Blender's first SIGGRAPH convention was a huge success and gathered a tremendous amount of interest from both the press and attendees. Blender was a hit and its huge potential confirmed!

Following the success of the SIGGRAPH conference in early 2000, NaN secured financing of €4.5M from venture capitalists. This large inflow of cash enabled NaN to rapidly expand its operations. Soon NaN boasted as many as 50 employees working around the world trying to improve and promote Blender. In the summer of 2000, Blender 2.0 was released. This version of Blender added the integration of a game engine to the 3D

application. By the end of 2000, the number of users registered on the NaN website exceeded 250,000.

Unfortunately, NaN's ambitions and opportunities did not match the company's capabilities and the market realities of the time. This over-extension resulted in restarting NaN with new investor funding and a smaller company in April 2001. Six months later NaN's first commercial software product, Blender Publisher was launched. This product was targeted at the emerging market of interactive web-based 3D media. Due to disappointing sales and the ongoing difficult economic climate, the new investors decided to shut down all NaN operations. The shutdown also included discontinuing the development of Blender. Although there were clearly shortcomings in the then current version of Blender, such as a complex internal software architecture, unfinished features and a non-standard way of providing the GUI, the enthusiastic support from the user community and customers who had purchased Blender Publisher in the past, meant that Ton could not justify leaving Blender to fade into insignificance. Since restarting a company with a sufficiently large team of developers was not feasible, Ton Roosendaal founded the non-profit organization, Blender Foundation, in March 2002.

The Blender Foundation's primary goal was to find a way to continue developing and promoting Blender as a community-based open source project. In July 2002, Ton managed to get the NaN investors to agree to a unique Blender Foundation plan to attempt to release Blender as open source. The "Free Blender" campaign sought to raise €100,000 so that the Foundation could buy the rights to the Blender source code and intellectual property rights from the NaN investors and subsequently release Blender to the open source community. With an enthusiastic group of volunteers, among them several ex-NaN employees, a fundraising campaign was launched to "Free Blender". To everyone's surprise and delight the campaign reached the €100,000 goal in only seven short weeks. On Sunday, October 13, 2002, Blender was released to the world under the terms of the GNU GPL. Blender development continues to this day, driven by a team of dedicated volunteers from around the world led by Blender's original creator, Ton Roosendaal.

Version/Revision Milestones

The start!

- **1.00 – January 1994:** Blender in development at animation studio NeoGeo.
- **1.23 – January 1998:** SGI version published on the web, IrisGL.
- **1.30 – April 1998:** Linux and FreeBSD version, port to OpenGL and X11.
- **1.3x – June 1998:** NaN founded.
- **1.4x – September 1998:** Sun and Linux Alpha version released.
- **1.50 – November 1998:** First Manual published.
- **1.60 – April 1999:** C-key (new features behind a lock, $95), Windows version released.
- **1.6x – June 1999:** BeOS and PPC version released.
- **1.80 – June 2000:** End of C-key, Blender full freeware again.
- **2.00 – August 2000:** Interactive 3D and real-time engine.
- **2.10 – December 2000:** New engine, physics, and Python.
- **2.20 – August 2001:** Character animation system.
- **2.21 – October 2001:** Blender Publisher launch.
- **2.2x – December 2001:** macOS version.

Blender goes Open Source

13 October 2002:
Blender goes Open Source, 1st Blender Conference.

2.25 – October 2002:
Blender Publisher becomes freely available, and the experimental tree of Blender is created, a coder's playground.

2.26 – February 2003:
The first truly open source Blender release.

2.27 – May 2003:
The second open source Blender release.

2.28x – July 2003:
First of the 2.28x series.

2.30 – October 2003:
Preview release of the 2.3x UI makeover presented at the 2nd Blender Conference.

2.31 – December 2003:
Upgrade to stable 2.3x UI project.

2.32 – January 2004:
A major overhaul of internal rendering capabilities.

2.33 – April 2004:
Game Engine returns, ambient occlusion, new procedural textures.

2.34 – August 2004:
Particle interactions, LSCM UV mapping, functional YafRay integration, weighted creases in subdivision surfaces, ramp shaders, full OSA, and many (many) more.

2.35 – November 2004:
Another version full of improvements: object hooks, curve deforms and curve tapers, particle duplicators and much more.

2.36 – December 2004:
A stabilization version, much work behind the scenes, normal and displacement mapping improvements.

2.37 – June 2005:
Transformation tools and widgets, soft bodies, force fields, deflections, incremental subdivision surfaces, transparent shadows, and multi-threaded rendering.

2.40 – December 2005:
Full rework of armature system, shape keys, fur with particles, fluids, and rigid bodies.

2.41 – January 2006:
Lots of fixes, and some Game Engine features.

2.42 – July 2006:
The nodes release, Array modifier, vector blur, new physics engine, rendering, lip sync, and many other features. This was the release following Project Orange.

2.43 – February 2007:
Multiresolution meshes, multi-layer UV textures, multi-layer images and multi-pass rendering and baking, sculpting, retopology, multiple additional mattes, distort and filter nodes, modeling and animation improvements, better painting with multiple brushes, fluid particles, proxy objects, Sequencer rewrite, and post-production UV texturing.

2.44 – May 2007:
The big news, in addition to two new modifiers and re-awakening the 64-bit OS support, was the addition of subsurface scattering, which simulates light scattering beneath the surface of organic and soft objects.

2.45 – September 2007:
Serious bug fixes, with some performance issues addressed.

2.46 – May 2008:
The Peach release was the result of a huge effort of over 70 developers providing enhancements to provide hair and fur, a new particle system, enhanced image browsing, cloth, a seamless and non-intrusive physics cache, rendering improvements in reflections, AO, and render baking, a Mesh Deform modifier for muscles and such, better animation support via armature tools and drawing, skinning, constraints and a colorful Action Editor, and much more. It contained the results of Project Peach.

2.47 – August 2008:
Bugfix release.

2.48 – October 2008:

The Apricot release, cool GLSL shaders, lights and GE improvements, snap, sky simulator, Shrinkwrap modifier, and Python editing improvements. This contained the results Project Apricot.

2.49 – June 2009:
Node-based textures, armature sketching (called Etch-a-Ton), Boolean mesh operation improvements, JPEG2000 support, projection painting for direct transfer of images to models, and a significant Python script catalog. GE enhancements included video textures, where you can play movies in-game, upgrades to the Bullet physics engine, dome (fisheye) rendering, and more API GE calls made available.

Blender 2.5x – The Recode!

2.5x – From 2009 to August 2011:
This series released four pre-version (from Alpha 0 in November 2009 to Beta in July 2010) and three stable versions (from 2.57 - April 2011 to 2.59 - August 2011). It was one of the most important development projects, with a total refactor of the software with new functions, redesign of the internal window manager and event/tool/data handling system, and new Python API. The final version of this project was Blender 2.59 in August 2011.

Blender 2.6x to 2.7x – Improvements & Stabilizing

2.60 – October 2011:
Internationalization of the UI, improvements in the animation system and the GE, vertex weight groups modifiers, 3D audio and video, and bug fixes.

2.61 – December 2011:
The Cycles renderer was added to the trunk, the camera tracker was added, dynamic paint for modifying textures with mesh contact/approximation, the Ocean modifier to simulate ocean and foam, new add-ons, bug fixes, and more extensions added for the Python API.

2.62 – February 2012:

The Carve library was added to improve Boolean operations, support for object tracking was added, the Remesh modifier was added, many improvements in the GE, matrices and vectors in the Python API were improved, plus new add-ons, and many bug fixes.

2.63 – April 2012:
Bmesh was merged with the trunk, with full support for n-sided polygons, sculpt hiding, a panoramic camera for Cycles, mirror ball environment textures and float precision textures, render layer mask layers, ambient occlusion and viewport display of background images and render layers. New import and export add-ons were added, and 150 bug fixes.

2.64 – October 2012:
A mask editor was added, along with an improved motion tracker, OpenColorIO, Cycles improvements, Sequencer improvements, better mesh tools (Inset and Bevel were improved), new keying nodes, sculpt masking, Collada improvements, a new Skin modifier, a new compositing nodes backend, and the fixing of many bugs.

2.65 – December 2012:
Fire and smoke improvements, anisotropic shader for Cycles, modifier improvements, the Bevel tool now includes rounding, new add-ons, and over 200 bug fixes.

2.66 – February 2013:
Dynamic topology, rigid body simulation, improvements in UI and usability (including retina display support), Cycles now supports hair, the Bevel tool now supports individual vertex beveling, new Mesh Cache modifier and the new UV Warp modifier, new SPH particle fluid solver. More than 250 bug fixes.

2.67 – May 2013:
Freestyle was added, paint system improvements, subsurface scattering for Cycles, Ceres library in the motion tracker, new custom Python nodes, new mesh modeling tools, better support for UTF-8 text and improvements in Text editors, new add-ons for 3D printing, over 260 bug fixes.

2.68 – July 2013:
New and improved modeling tools, three new Cycles nodes, big improvements in the motion tracker, Python scripts and drivers are disabled by default when loading files for security reasons, and over 280 bug fixes.

2.69 – October 2013:
Even more modeling tools, Cycles improved in many areas, plane tracking is added to the motion tracker, better support for FBX import/export, and over 270 bugs fixed.

2.70 – March 2014:
Cycles gets basic volumetric support on the CPU, more improvements to the motion tracker, two new modeling modifiers, some UI consistency improvements, and more than 560 bug fixes.

2.71 – June 2014:
Deformation motion blur and fire/smoke support is added to Cycles, UI pop-ups are now draggable. There are performance optimizations for sculpting mode, new interpolation types for animation, many improvements to the GE, and over 400 bug fixes.

2.72 – October 2014:
Cycles gets volume and SSS support on the GPU, pie menus are added and tooltips greatly improved, the Intersection modeling tool is added, new Sun Beam node for the Compositor, Freestyle now works with Cycles, texture painting workflow is improved, and more than 220 bug fixes.

2.73 – January 2015:
Cycles gets improved volumetric support, major upgrade to Grease Pencil, Windows gets Input Method Editors (IMEs) and general improvements to painting, Freestyle, Sequencer and add-ons.

2.74 – March 2015:
Support for custom normals, viewport compositing and improvements to hair dynamics.

2.75 – July 2015:
Integrated stereo/multi-view pipeline, Smooth Corrective modifier and new developmental dependency graph.

2.76 – November 2015:
Pixar OpenSubdiv support, Viewport and File Browser performance boost, node auto-offset, and a text effect strip for the Sequencer.

2.77 – March 2016:
OpenVDB support for caching of smoke/volumetric simulations, improved Cycles subsurface scattering, Grease Pencil stroke sculpting and improved workflow, and reworked library handling to manage missing and deleted data-blocks.

2.78 – September 2016:
Cycles support for spherical stereo images for VR, Grease Pencil works more similar to other 2D drawing software, Alembic import and export support, and improvements to Bendy Bones for easier and simpler rigging.

2.79 – September 2017:
New Cycles features: Denoising, Shadow catcher, and new Principled shader. Other improvements were made to Grease Pencil and Alembic. Support was also added for application templates.

Blender 2.8 – Revamped UI

2.80 – July 2019:
A totally redesigned UI for easier navigation; improved viewport, gizmos, and tools. With EEVEE a new physically based real-time render engine was created. The Grease Pencil got a big overhaul and is now a full 2D drawing and animation system. Replacing the old layers, collections are a powerful way to organize objects. Other improvements: Cycles, Modeling, Animation, Import/Export, Dependency Graph.

2.81 – November 2019:

Revamped sculpting tools, Cycles OptiX accelerated rendering, denoising, many EEVEE improvements, library overrides, UI improvements and much more.

2.82 – February 2020:
UDIM and USD support, Mantaflow for fluids and smoke simulation, AI denoising, Grease Pencil improvements, and much more.

2.83 – June 2020:
3D Viewport virtual reality scene inspection, new volume object type, Cycles adaptive sampling, Cycles viewport denoising, sculpting improvements, and much more.

Blender 2.9 – Refining 2.8

2.90 – August 2020:
Improved sky texture, EEVEE motion blur, sculpting improvements, revamped modifier UI, improved modeling tools, and faster motion blur in Cycles.

2.91 – November 2020:
Outliner improvements, property search, improved mesh Boolean operations, animation curves, volume object and display improvements, and more refined sculpting tools.

2.92 – February 2021:
Geometry nodes, primitive add tool, sculpting improvements, Grease Pencil curve editing, Cycles Color Attribute baking, APIC fluid simulations, Video Sequencer improvements, and much more.

2.93 – June 2021:
New geometry nodes, sculpting improvements, Grease Pencil Line Art modifier along with other improvements, an improved DOF for the EEVEE render engine, redesigned Cryptomatte workflow, and more.

Blender 3.0 – Optimizing Performance

3.0 – December 2021
Asset Browser added, Cycles X, EEVEE Attributes, New geometry nodes, animation update, Grease Pencil line art improvements, pose library, Open Image Denoising 2-8x faster, additional support for AMD on linux.

3.1 – March 2022
Major point clouds improvements, Cycles Apple Metal GPU support, Subdivision GPU support, image editor handles larger images, Major performance gains for geometry nodes, context aware search for geometry nodes.

3.2 – June 2022
Light groups for Cycles, true Shadow caustics, volume motion blur, GLTF improvements, AMD GPU Rendering on Linux, painting in sculpt mode, WEBp image support.

3.3 – September 2022
New hair object, procedural UV nodes, line art shadow and contour, Intel GPU rendering support via oneAPI, and improvements to library overrides.

3.4 – December 2022
Cycles path guiding, sculpting auto masking improvements, even more geometry nodes, UV Editing improvements and Wayland support on Linux.

3.5 – March 2023
New generative hair assets, vector displacement maps for sculpting, viewport compositor, and Cycle's light trees.

About Free Software and the GPL



When one hears about “free software”, the first thing that comes to mind might be “no cost”. While this is often true, the term “free software” as used by the Free Software Foundation (originators of the GNU Project and creators of the GNU General Public License) is intended to mean “free as in freedom” rather than in the sense of “no cost” (which is usually referred to as “free as in free beer” or *gratis*). Free software in this sense is software which you are free to use, copy, modify, redistribute, with no limit. Contrast this with the licensing of most commercial software packages, where you are allowed to load the software on a single computer, are allowed to make no copies, and never see the source code. Free software allows incredible freedom to the end user. Since the source code is universally available, there are also many more chances for bugs to be caught and fixed.

When a program is licensed under the GNU General Public License (the GPL):

- You have the right to use the program for any purpose.
- You have the right to modify the program and have access to the source codes.
- You have the right to copy and distribute the program.
- You have the right to improve the program, and release your own versions.

In return for these rights, you have some responsibilities if you distribute a GPL’d program. These responsibilities are designed to protect your freedoms and the freedoms of others:

- You must provide a copy of the GPL with the program, so that recipients are aware of their rights under the license.
- You must include the source code or make the source code freely available.
- If you modify the code and distribute the modified version, you must license your modifications available under the GPL (or a compatible license).
- You may not restrict the licensing of the program beyond the terms of the GPL (you may not turn a GPL'd program into a proprietary product).

For more on the GPL, check its page on the GNU Project website.

Note

The GPL only applies to the Blender application and **not** the artwork you create with it; for more info see the Blender License.

The Blender Community

Being freely available from the start, even while it was closed source, considerably helped Blender's adoption by the community. A large, stable, and active community of users has gathered around Blender since 1998. The community showed its support for Blender in 2002 when they helped raise €100,000 in seven weeks to enable Blender to go Open Source under the GNU GPL License.

Independent Sites

There are several independent websites such as forums, blogs, news, and tutorial sites dedicated to Blender.

One of the largest community forums is Blender Artists, where Blender users gather to show off their creations, get feedback, ask and offer help and, in general, discuss Blender.

Getting Support

Blender's community is one of its greatest features, so apart from this user manual, there are many different ways to get support from other users, such as Chat, Stack Exchange, and Reddit.

For studios and organizations there is Enterprise support, and for studios looking to add Blender to their pipeline, Blender Studio contains documentation and training material around this topic. If you think you have found an issue with Blender, please report a bug.

More details about support can be found on the support page.

Development

Being open source, Blender welcomes development from volunteers. Communication between developers is done mostly through three platforms:

- The projects.blender.org system
- Developer Forum
- Online Chat (see below)

If you are interested in helping develop Blender, see the Get Involved page.

Chat

For real-time discussion, we have chat.blender.org which uses Blender ID for authentication.

You can join these channels:

- #general For general chat with the community.
- #blender-coders For developers to discuss Blender development.
- #python For support for developers using the Python API.
- #docs For discussion related to Blender's documentation.

Other Useful Links

- Blender FAQ (Can I use Blender commercially? What is GPL/GNU? …)
- Demo and benchmark files
- Developer's Ask Us Anything!

Installing Blender

Blender is released approximately every three months. You can keep up to date with the latest changes through the release notes.

System Requirements

Blender is available for download on Windows, macOS, and Linux. Always check that the graphics drivers are up to date and that OpenGL is well supported. Blender has a set of minimum and recommended requirements; so make sure these are met before trying to install Blender.

Support for other hardware such as graphic tablets and 3D mice are covered later in Configuring Hardware.

Download Blender

Blender offers a variety of different binary packages to choose from depending on their level of stability. Each package has the trade off of newest feature versus stability. The package that is right for you depends on your requirements for those two. A studio for example might want to have *long-term support*, while a hobbyist may want newer features, while others may just want to test upcoming features. Each package described below has something just right for everyone.

Stable Release
: A package that contains the latest features and is considered stable without regressions. A new stable version is available about every three months.

Long-term Support
: A package designed for long-lasting projects requiring a very stable version of Blender. LTS releases are supported for two years and will

not have any new features, API changes or improvements. A new long-term support version is available every year.

Daily Builds
: A package updated daily to include the newest changes in development. These versions are not as thoroughly tested as the stable release, and might break, although they are official and usually not highly experimental.

Note

Blender's source code is available for free to either reference or to Build from Source. While normal users are **not** expected to compile Blender, it does have advantages:

- Blender is always up to date.
- It allows access to any version or branch where a feature is being developed.
- It can be freely customized.

The procedure for installing a binary, either the latest stable release or a daily build, is the same. Follow the steps for your platform.

Note

Blender doesn't have a built-in updating system. This means you will need to update Blender yourself by following the upgrade steps described in the sections below.

Installation Guides

- Installing on Linux
- Installing on macOS
- Installing on Windows
- Installing from Steam

Installing on Linux

Check the Downloading Blender page to find the minimum requirements and the different versions that are available for Blender (if you have not done so yet).

Install from blender.org

Download the Linux version for your architecture and decompress the file to the desired location (e.g. `~/software` or `/usr/local`).

Blender can now be launched by double-clicking the executable.

When using this method of installation, it is possible to have multiple versions of Blender installed.

For ease of access, you may wish to add a menu entry and create blend-file associations for the file-browser. This can be done by Registering Blender.

To make the installation and configuration fully self-contained, set up a Portable Installation.

Install from Package Manager

Some Linux distributions may have a specific package for Blender in their repositories.

Installing Blender via the distribution's native mechanisms ensures consistency with other packages on the system and may provide other features (given by the package manager), such as listing of packages, update notifications and automatic menu configuration. Be aware, though, that the package may be outdated compared to the latest official release, or not include some features of Blender. For example, some distributions do

not build Blender with Cycles GPU rendering support, for licensing or other reasons.

If there is a specific package for your distribution, you may choose what is preferable and most convenient, otherwise, the official binary is available on blender.org.

Install from Snap

Snap is a universal package manager designed to work across a range of distributions. Assuming snap is already installed, Blender can be installed through snap with:

```
snap install blender --classic
```

Installing from this method has a benefit that updates to Blender are automatically installed. Blender from Snap should have a more consistent distribution then individual package managers.

Running from the Terminal

See Launching from the terminal.

Graphics System (X11 & Wayland)

Blender supports both X11 and Wayland, see Linux Windowing Environment for details.

Avoiding Alt-Mouse Conflict

Some window managers default to `Alt-LMB` and `Alt-RMB` for moving and resizing windows.

Blender uses these for various operations, notably:

- Emulate 3 Button Mouse.

- Select Edge Loops.
- Changing multiple properties at once.

To access Blender's full feature set, you can change the window manager settings to use the *Meta* key instead (also called *Super* or *Windows* key):

Gnome
: Enter the following in a command line (effective at next login):

```
gsettings set org.gnome.desktop.wm.preferences mouse-button-modifier '<Super>'
```

KDE
: *System Settings* ▸ *Window Management* ▸ *Window Behavior* ▸ *Window Actions*, Switch from 'Alt' to 'Meta' key.

Updating on Linux

On Linux there are various ways of updating Blender. This section covers the most common approaches.

Updating from blender.org

When an update for Blender is released, it can be downloaded directly from the Blender website and installed using the steps described in the section Install from blender.org.

Updating with a Package Manager

Many Linux distributions have packages for Blender available, which can be installed using the distribution's package manager. After installation, Blender can be updated using the same steps as updating any other application.

See also

The Splash screen Defaults page for information about importing settings from previous Blender versions and other quick settings.

Known Limitations

Archive Extraction

Extracting Blender's archive using **7-zip** is not supported. TAR must be used instead, see: #104070.

Installing on macOS

Check the Downloading Blender page to find the minimum requirements and the different versions that are available for Blender (if you have not done so yet).

Important

Blender supports both Intel and Apple Silicon architectures on macOS. Make sure to download a variant that is compatible with your CPU's architecture.

Install from DMG

Blender for macOS is distributed as disk images (dmg-files). To mount the disk image, double-click on the dmg-file. Then drag `Blender.app` into the Applications folder.

Depending on the Security and Privacy preferences of your Mac, macOS will request your approval before opening Blender for the first time.

To make the installation and configuration fully self-contained, set up a Portable Installation.

Updating on macOS

On macOS there are various ways of updating Blender. This section covers the most common approach.

Updating with DMG

When an update for Blender is released, it can be downloaded directly from the Blender website. Install the new version by overwriting the current

`Blender.app` in the Applications folder. You can rename `Blender.app` or place it in a different folder to have more than one version at a time.

See also

The Splash screen Defaults page for information about importing settings from previous Blender versions and other quick settings.

Installing on Windows

Check the Downloading Blender page to find the minimum requirements and the different versions that are available for Blender (if you have not done so yet).

Download the zip-file or Windows Installer File.

Important

Blender supports both x64 and arm64 architectures on Windows. Make sure to download a variant that is compatible with your CPU's architecture.

Install from Windows Installer File

The Windows installer will let you choose an installation folder, and will create an entry in the start menu as well as associate blend-files with Blender. It requires administrator rights.

Install from Zip

When choosing the zip-file, you have to manually extract Blender to the desired folder, where you can double-click the executable to run Blender.

No start menu item will be created and no blend-file association will be registered, but there is also no need for administrator rights. You can register the file association manually by clicking *Make Default* on the System tab of the Preferences. Alternatively, you can run `blender -r` from the Command Line.

To make the installation and configuration fully self-contained, set up a Portable Installation.

Install from Microsoft Store

Blender can be installed from the Microsoft Store by searching for Blender in the Microsoft Store and installing it.

Blender can now be launched from the Windows Start menu.

Updating on Windows

On Windows there are various ways of updating Blender. This section covers the most common approaches.

Updating from Windows Installer File

When an update for Blender is released, it can be downloaded directly from the Blender website. The Windows installer can then be run to install the updated version of Blender. To remove a previously installed version of Blender, use Windows settings or control panel to uninstall the desired version.

Updating from Zip

When an update for Blender is released, it can be downloaded directly from the Blender website and extracted to the desired folder, where you can double-click the executable to run Blender. For more information on creating a portable version of Blender, see the section Install from Zip.

Note, you do not have to overwrite your existing Blender installation. It's perfectly possible to have multiple versions installed side by side.

Updating from the Microsoft Store

When an update for Blender is available on the Microsoft Store, it will be downloaded and installed automatically.

See also

The Splash screen Defaults page for information about importing settings from previous Blender versions and other quick settings.

Installing from Steam

Steam is a software distribution platform. Blender can be downloaded and updated using the Steam client by following the steps described below on Linux, macOS, or Windows.

Download the Steam client for your operating system. Once installed, open the client and login to your Steam account, or create one if you haven't already. Once logged in, navigate to the *Store* tab, search for "Blender", and press the green installation button. Blender should now be available in the *Library* tab of the Steam client.

See also

When installing Blender from Steam on Linux and Windows, the `.blend` filename extension will not be automatically associated with Blender. To associate blend-files with Blender, see the processes described on the Linux and Windows installation pages.

Updating with Steam

When an update for Blender is available on Steam, Steam will automatically update Blender for you.

Linux Windowing Environment

On Linux Blender supports both X11 and Wayland for official releases.

When Wayland is detected, it is the preferred system, otherwise X11 will be used.

Hint

The current "Windowing Environment" is listed in *File ‣ About*.

X11

This is the windowing environment that has been used most widely on Linux & Unix systems.

There are no near-term plans to deprecate or remove X11 support.

Wayland

Support for Wayland is a more recent addition, so there may be configurations that have not been tested yet. Please report a bug if you experience problems.

Blender has been tested with Gnome-Shell (mutter), KDE (plasma) & SWAY (wlroots) based compositors.

Requirements

Gnome-Shell
: Under Gnome-Shell the `libdecor` library is required. This is available as a package on most Linux distribution.

If the library isn't found X11 will be used as a fallback.

Troubleshooting

Detailed Wayland output can help to track down problems. Launch Blender from the command-line with additional arguments:

Blender's Wayland Logging

```
blender --log "ghost.wl.*" --log-level 2
```

Wayland Built-In Logging

```
WAYLAND_DEBUG=1 blender
```

Disable Wayland (forcing X11)

```
WAYLAND_DISPLAY="" blender
```

Disable `libdecor` (forcing borderless windows under Gnome-Shell)
: Uninstall `libdecor`, then run Blender with an empty X11 display variable.

```
DISPLAY="" blender
```

Environment Variables

`XCURSOR_THEME`
: The cursor theme to use (must refer to a locally installed cursor).

`XCURSOR_SIZE`
: The cursor size, defaults to 28, you may wish to increase the size on Hi-DPI displays.

Known Limitations

Gnome Shell's Fractional Scaling (before version 44)
: Versions of Gnome-Shell prior to 44 don't fully support fractional scaling.

Using fractional under older versions of Gnome-Shell may result in glitches such as a small cursor size.

NVidia GPU

Currently NVidia drivers don't fully support features needed for Wayland. Graphical glitches and flickering are common problems. In some cases, there can be crashes on startup. This is not specific to Blender, so NVidia users may want to use X11 until driver support improves.

Feature Comparison

Feature	X11	Wayland	Notes
Smooth Scroll	✗	✓	Smooth scrolling with track-pads.
Multi-Touch Gestures	✗	✓	Track-pad and tablet support for pinch to zoom, pan and orbit.
Reliable Cursor Warping	✗ *1	✓	Cursor warping is used while transforming and orbiting the viewport for e.g.
Window Positioning	✓	✗ *2	Needed for dragging between windows and restoring window positions on file load.

Other features which both systems support such as Hi-DPI, 3D-mouse, tablet input, … etc. have been left out of this list.

[*1] In X11 fast cursor motion may exit the window bounds while the cursor is grabbed (transforming for e.g.).
[*2] Wayland doesn't support setting the window position, as this is a design decision it's unlikely to be supported (see issues for position).

Configuring Blender

- Introduction
  - Language
  - Input
  - File and Paths
  - Save & Load
- Configuring Peripherals
  - Displays
  - Input Devices
  - Head-Mounted Displays (Virtual Reality)
- Defaults
  - Import Preferences From Previous Version
  - Create New Preferences
  - Saving Defaults
  - Loading Factory Settings

See also

See Preferences for a complete list of options.

Introduction

Here are some preferences that you may wish to set initially. See the section Preferences for the complete list of available settings.

Language

Enable *Edit* ▸ *Preferences* ▸ *Interface* ▸ *Translation*, and choose the *Language* and what to translate from *Interface*, *Tooltips* and *New Data*.

See Language for details.

Input

If you have a compact keyboard without a separate number pad, enable *Preferences* ▸ *Input* ▸ *Keyboard* ▸ *Emulate Numpad*. This gives you the 3D view shortcuts regularly used on the number pad.

If you do not have a middle mouse button, you can enable *Preferences* ▸ *Input* ▸ *Mouse* ▸ *Emulate 3 Button Mouse*. This allows you to hold the `Alt` or `OSKey` key while dragging with the mouse, to orbit.

See Input Preferences for details.

File and Paths

At *Preferences* ▸ *File Paths* you can set options such as what *Image Editor* (GIMP, Krita…) and *Animation Player* to use.

The Temporary Directory sets where to store files such as temporary renders and auto-saves.

Tip

The `//` at the start of each path in Blender means the directory of the currently opened blend-file, used to reference relative paths.

See File Preferences for details.

Save & Load

If you trust the source of your blend-files, you can enable *Auto Run Python Scripts*. This option is meant to protect you from malicious Python scripts in blend-files that you got from someone else. Many users turn this option on, as advanced rigs tend to use scripts of some sort.

See Save & Load Auto Run Python Scripts Preference.

Configuring Peripherals

Displays

A full HD display or higher is recommended. Multi-monitor setups are supported, and workspaces can be configured to span multiple monitors.



Example of Blender's multi-monitor support.

Input Devices

Blender supports various types of input devices:

- Keyboard (recommended: keyboard with numeric keypad, English layout works best)
- Mouse (recommended: three button mouse with scroll wheel)
- Graphic Tablet
- Touchpad
- NDOF Device (also known as *3D Mouse*)

Note

If you don't have a middle mouse button or numpad, you can emulate these in the Input Preferences.

Mouse

Mouse Button Emulation

If you do not have a 3 button mouse, you will need to emulate it by checking the option in the Preferences.

The following table shows the combinations used:

3-button Mouse	LMB	MMB	RMB
2-button Mouse	LMB	Alt–LMB	RMB

Keyboard

Numpad Emulation

If you do not have a numpad on the side of your keyboard, you may want to emulate one. You can then use the number row at the top of the keyboard instead, but will no longer have access to these keys' original functions (such as switching between vertex/edge/face selection in Edit Mode).

See also

Read more about *Numpad Emulation* in the Preferences.

Non-English Keyboards

If you use a keyboard with a non-English layout, you may still benefit from switching to the UK or US layout while working with Blender.

Note

You can also change the keymap from the Preferences. However, this manual assumes you are using the default keymap.

Graphic Tablet

Graphics tablets can be used to provide a more traditional method of controlling the mouse cursor using a pen. This can help provide a more familiar experience for artists who are used to painting and drawing with similar tools, as well as provide additional controls such as pressure sensitivity.

Note

If you are using a graphic tablet instead of a mouse and pressure sensitivity does not work properly, try to place the mouse pointer in the Blender window and then unplug/replug your graphic tablet. This might help.

Touchpad

Touchpad controls are available on Windows, macOS and Linux with Wayland. If you are working from a laptop without a mouse, you can emulate controls using multi-touch gestures with the trackpad from Preferences.

Supported multi-touch gestures

Gesture	Effect

Gesture	Effect
Pan	Hold the `Shift` key while dragging two fingers on the pad.
Zoom	Hold the `Ctrl` or `OSKey` key while dragging two fingers on the pad.
Orbit	Drag two fingers on the pad.
Emulate right-click	Tap two fingers on the pad.

NDOF (3D Mouse)

3D mice or NDOF devices are hardware that you can use to navigate a scene in Blender. Currently only devices made by 3Dconnexion are supported. These devices allow you to explore a scene, and make Fly/Walk Navigation easier to control. The NDOF device can be configured in the Preferences. These settings can also be accessed directly from the viewport using the `NDOFMenu` button on the NDOF device.

See also

See Input Preference for more information on configuring peripherals.

Head-Mounted Displays (Virtual Reality)

HMDs make it possible to place users in an interactive, virtual environment. Attached to the head, they track head movements to project a seemingly

surrounding world onto small screens in front of the user's eyes. If the system works well, they experience the virtual environment as if they were really inside of it.

Supported Platforms

Virtual reality support in Blender is implemented through the multi-platform OpenXR standard. This standard is new and therefore support for it is still limited.

OpenXR compatible platforms.

Platform	Operating System	Notes
HTC Vive Cosmos	Windows	Developer Preview
HTC Vive Focus 3	Windows	Developer Preview
Monado	GNU/Linux	*Not* recommended for general use yet.
Meta (formerly Oculus) (Rift and Quest)	Windows	Requires Oculus v31 Software Update. Oculus Link required for Quest.
SteamVR	Windows, GNU/Linux	Requires SteamVR 1.16 or greater.
Varjo	Windows	–

Platform	Operating System	Notes
Windows Mixed Reality	Windows	Requires Windows 10 May 2019 Update (1903).

Getting Started

The following subsections describe how an HMD can be set up for usage with the supported platforms. If this is not done, Blender will report an error when trying to start a virtual reality session.

HTC Vive Cosmos

The dedicated platform for the HTC Vive Cosmos is currently targeted at developers and may lack features found in other platforms.

- Follow the steps from the Vive Developer Forums.
- Enable the VR Scene Inspection add-on in Blender.

HTC Vive Focus 3

The dedicated platform for the HTC Vive Focus 3 is currently targeted at developers and may lack features found in other platforms.

- Follow the steps from the Vive Developer Forums.
- Enable the VR Scene Inspection add-on in Blender.

Monado

Monado is a free and open source XR platform for Linux. It is not yet ready for production usage and should only be used for testing purposes.

- Packages are available for the following distributions:

    - Ubuntu (Eoan, Focal)
    - Debian (bullseye, sid)

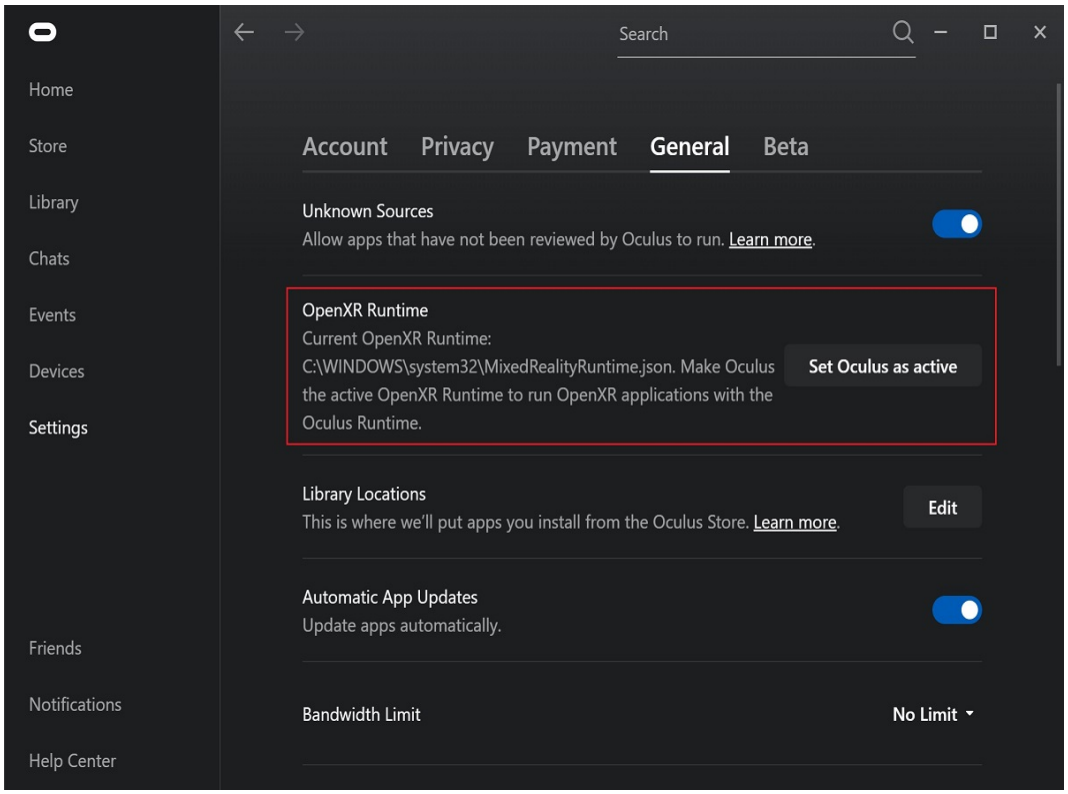
  For other systems, it has to be compiled from source, which in this case is not recommended for people with little experience in compiling software. Follow the Getting Started Guides from Monado to do so nevertheless.

- Enable the VR Scene Inspection add-on in Blender.

Meta (formerly Oculus)

Meta (formerly Oculus) provides full support for OpenXR as of the Oculus v31 Software Update.

- Download and install the Oculus Rift/Oculus Link software.
- Set Oculus as the active OpenXR runtime via the *General* tab in the Oculus App Settings.



- Enable the VR Scene Inspection add-on in Blender.

Passthrough Support

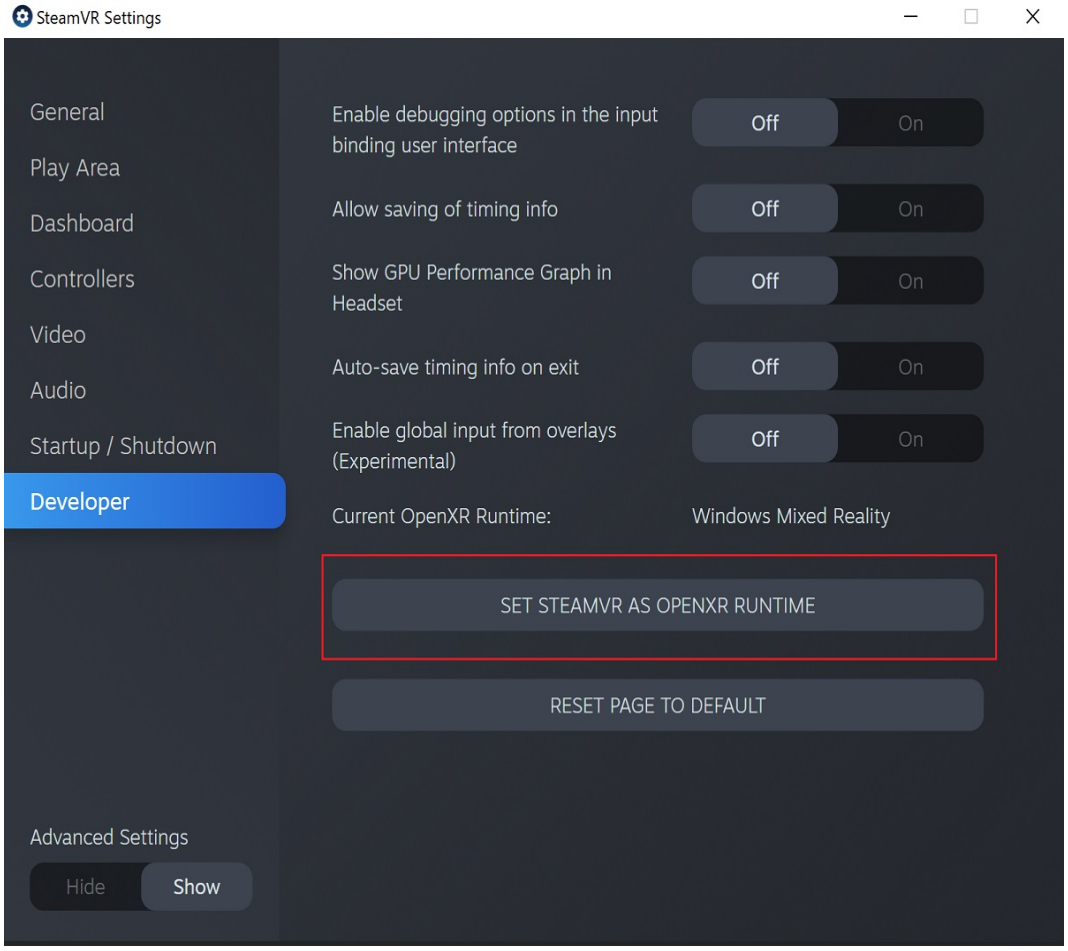
Currently, passthrough support over OpenXR is disabled by default in the Quest Link app, and must be manually enabled in it's settings to use this feature.

The performance of the passthrough render varies with the quality of the connection between the headset and the computer. For better results, connecting the headset directly through USB to the PC, or at least connecting the computer to the local network over ethernet, is recommended.

SteamVR

SteamVR provides full support for OpenXR as of SteamVR 1.16.

- Set SteamVR as the active OpenXR runtime via the *Developer* tab in the SteamVR Settings.



- Enable the VR Scene Inspection add-on in Blender.

Note

The SteamVR runtime can also be used for HTC Vive Cosmos, Oculus, and Windows Mixed Reality HMDs.

Varjo

Varjo includes full OpenXR support with its required Varjo Base software.

- Enable the VR Scene Inspection add-on in Blender.

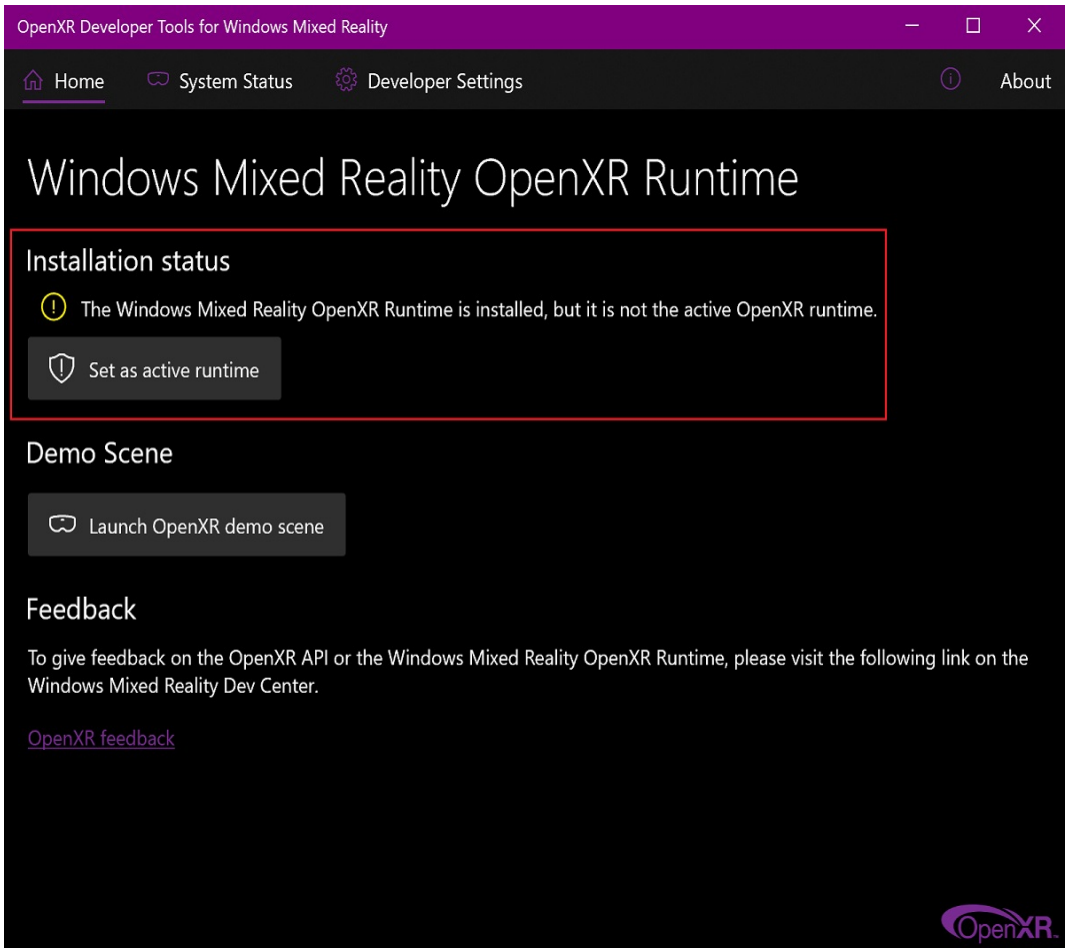
Windows Mixed Reality

Windows Mixed Reality provides full support for OpenXR. To check if a PC meets the requirements to run the software, Microsoft offers the Windows Mixed Reality PC Check application.

- Make sure the Windows 10 May 2019 Update (1903) is installed.
- If the system meets all requirements, the Mixed Reality Portal should already be installed. It is also available in the Microsoft Store.
- Launch the Mixed Reality Portal. Click the menu button **...** in the lower left corner. In the menu it opens, select the *Set up OpenXR*.
- Enable the VR Scene Inspection add-on in Blender.

Note

To switch to Windows Mixed Reality from another OpenXR runtime (e.g. SteamVR), download the OpenXR Developer Tools from the Microsoft Store and set Windows Mixed Reality as the active runtime.


OpenXR Developer Tools for Windows Mixed Reality
Home
System Status
Developer Settings
About
Windows Mixed Reality OpenXR Runtime
Installation status
The Windows Mixed Reality OpenXR Runtime is installed, but it is not the active OpenXR runtime.
Set as active runtime
Demo Scene
Launch OpenXR demo scene
Feedback
To give feedback on the OpenXR API or the Windows Mixed Reality OpenXR Runtime, please visit the following link on the Windows Mixed Reality Dev Center.
OpenXR feedback
OpenXR

Defaults

When you start Blender for the first time or update to a new version, the interactive region of the Splash Screen is replaced with a couple of initial preferences to configure how you interact with Blender.

Note

These options can always be changed later in the Preferences.

Import Preferences From Previous Version

This is where you can copy preferences from an older version of Blender. Doing so will copy preferences and startup files from the previous version of Blender and then loads them.

The preferences need to be imported from previous versions because the configuration files of each Blender version are stored in separate folders. Refer to the Blender's Directory Layout page for the location of these folders.

If you would like to start fresh with the new version, continue to Create New Preferences.

Create New Preferences

Language
: The language used in the user interface. The list is broken up into categories determining how complete the translations are. More language preferences can be set in the Translation Preferences.

Theme

Choose between a light or dark theme for Blender. Themes can be customized more in the Preferences.

Keymap

Presets for the default keymap for Blender. Note that this manual assumes that you use the "Blender" keymap.

Blender:: This is the default keymap. Read more about this keymap here.

Blender 2.7x:: This keymap is intended to match an older series of Blender versions and is designed for people upgrading who do not want to learn the updated keymap.

Industry Compatible:: This keymap is intended to match common creation software and is intended for people who use many different such applications. Read more about this keymap here.

Mouse Select

Controls which mouse button, either right or left, is used to select items in Blender.

Spacebar Action

Controls the action of `Spacebar`. These and other shortcuts can be modified in the keymap preferences.

Play:: Starts playing through the Timeline. This option is good for animation or video editing.

Tools:: Opens the Toolbar underneath the cursor to quickly change the active tool. This option is good if doing a lot of modeling or rigging.

Search:: Opens up the Menu Search. This option is good for someone who is new to Blender and is unfamiliar with its menus and shortcuts.

Save New Preferences

Saves the preferences set above and opens the regular Splash Screen.

Saving Defaults

The preferences are automatically saved when changed.

Changing the default startup file can be done via *File ▸ Defaults ▸ Save Startup File*. See Startup File.

There are two areas where Blender's defaults are stored:

Preferences
: The Preferences file stores keymap, add-ons theme and other options.

Startup File
: The Startup File stores the scene and UI setup which are displayed at startup and when creating a new file (*File ▸ New*).

Loading Factory Settings

You can revert your customizations to Blender's defaults:

Preferences
: The Preferences Load Factory Settings.

Startup File & Preferences
: *File ▸ Defaults ▸ Load Factory Settings*.

Note

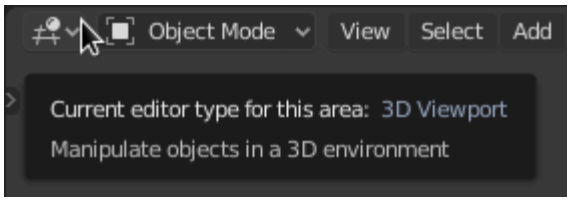
After loading the factory settings, the preferences won't be auto-saved.

See Managing Preferences for details.

Help System

Blender has a range of built-in and web-based help options.

Tooltips



Tooltip of the Renderer selector in the Info Editor.

When hovering the mouse cursor over a button or setting, after a few instants a tooltip appears.

Elements

The context-sensitive Tooltip might contain some of these elements:

Short Description
: Related details depending on the control.

Shortcut
: A keyboard or mouse shortcut associated to the tool.

Value
: The value of the property.

  Hovering over a color property will display a large swatch preview of the color and the color's hexadecimal, RGBA, and HSVA values.

Library
: Source file of the active object. See also Linked Libraries.

Disabled (red)
: The reason why the value is not editable.

Python
: When Python Tooltips are enabled, a Python expression is displayed for scripting (usually an operator or property).

Context-Sensitive Manual Access

Reference

Mode::	All modes
Menu::	*Context menu* ‣ *Online Manual*
Shortcut::	`F1`

You may want to access help for a tool or area from within Blender.

To do so; hover the cursor over the tool or button you need help with and use the keyboard shortcut or context menu item to visit pages of this reference manual from within Blender. This opens a web page relating to the button under the cursor, supporting both tool and value buttons.

Note

We do not currently have 100% coverage. You may see an alert in the info header if a tool does not have a link to the manual.

Other times, buttons may link to more general sections of the documentation.

Help Menu

Web Links

The first options of this menu provide direct links to Blender-related websites. The same links can also be found in the Splash Screen.

Manual
: This is a link to the Official Blender Manual (which you are now reading).

Release Notes
: Link to the release notes for the current Blender version.

Tutorials
: Multiple tutorials to help you learn to use Blender.

Support
: Links to various sites, providing both community and professional support.

User Communities
: Lists of many different community sites and support venues.

Report a Bug
: The Blender Bug Tracker (registration needed).

Save System Info

This extracts system information which can be useful for including in bug reports, inspecting the configuration, or diagnosing problems.

You will be prompted to save a text file called `system-info.txt`.

It contains the following sections:

Blender
: This section shows you the Blender version, details about the build configuration, and the path in which Blender is running.

Python

The version and path of your Python installation.

Directories
: Paths used for scripts, data files, presets and temporary files.

  Those directories are configured using the Preferences Editor.

GPU
: Shows the GPU vendor, version and the capabilities of your hardware and driver.

Enabled Add-Ons
: Lists add-ons currently in use.

User Interface

- Window System
  - Introduction
  - Splash Screen
  - Topbar
  - Workspaces
  - Status Bar
  - Areas
  - Regions
  - Tabs & Panels
- Keymap
  - Common Shortcuts
  - Default Keymap
  - Industry Compatible Keymap
- UI Elements
  - Buttons
  - Input Fields
  - Menus
  - Eyedropper
  - Decorators
  - Data-Block Menu
  - List View
  - Color Picker
  - Color Ramp Widget
  - Color Palette
  - Curve Widget
- Tools & Operators
  - Tool System
  - Operators
  - Undo & Redo
  - Annotations
  - Selecting
- Nodes
  - Introduction

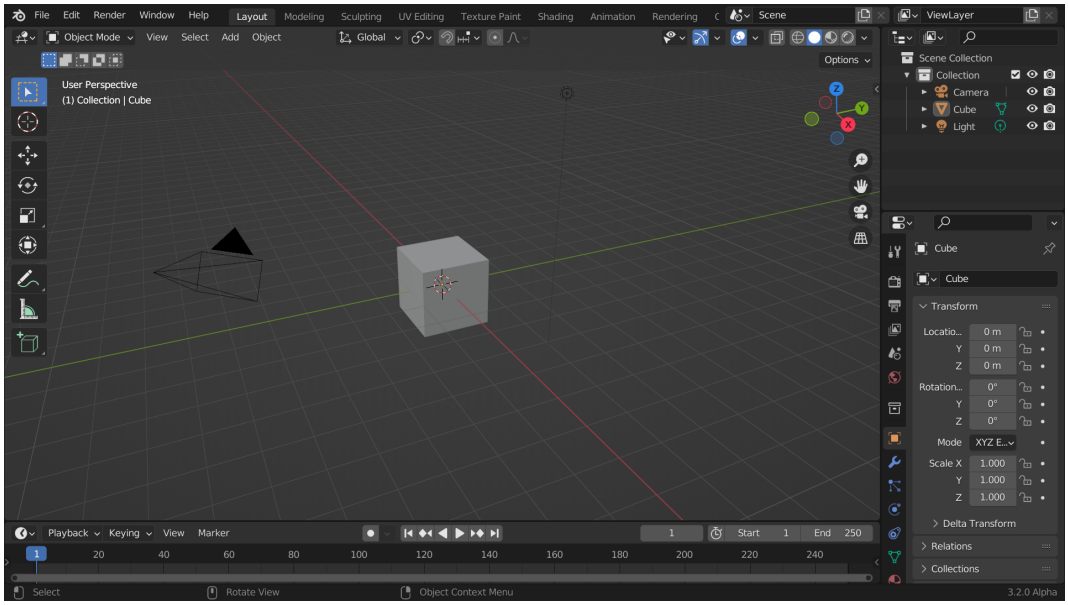
- Node Parts
- Selecting
- Arranging Nodes
- Editing
- Sidebar
- Node Groups
- Frame Node
- Reroute Node

Window System

- Introduction
- Splash Screen
- Topbar
- Workspaces
- Status Bar
- Areas
- Regions
- Tabs & Panels

Window System Introduction

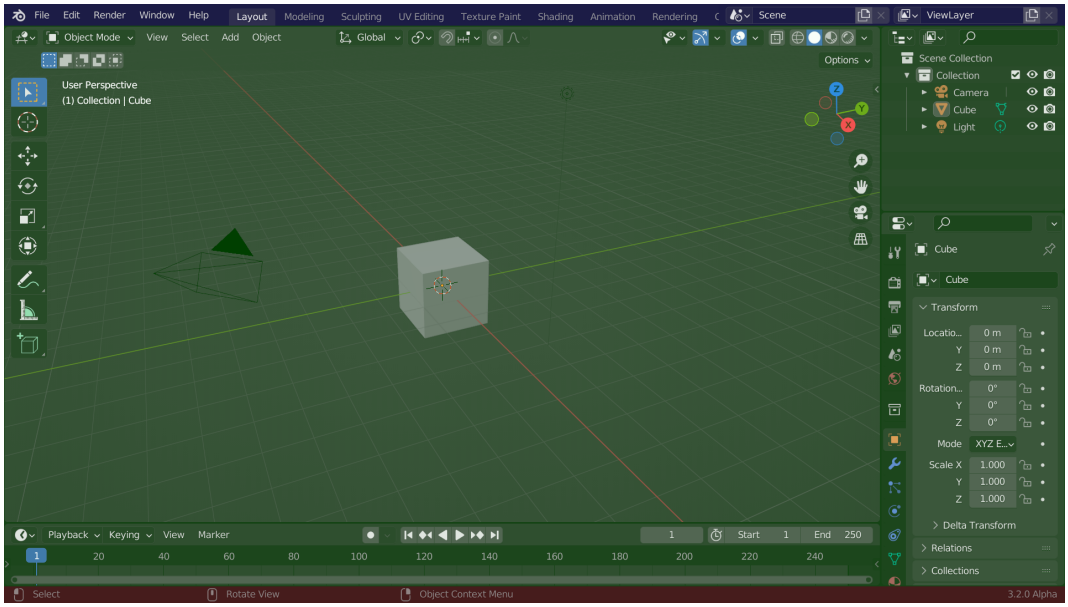
After starting Blender and closing the Splash Screen, the Blender window should look similar to the image below.



The default startup Blender window.

Blender's interface is separated into three main parts:

- The Topbar at the very top, consists of the main menu, which is used for saving, importing and exporting files, configuring settings, and rendering among other functions.
- Areas in the middle, which is the main workspace
- The Status Bar at the bottom, which displays shortcut suggestions and relevant statistics.



Blender's default Screen Layout. Topbar (blue), Areas (green) and Status Bar (red).

Customization

Keyboard shortcuts

Blender makes heavy use of keyboard shortcuts to speed up work. These can be customized in the Keymap Editor.

Theme colors

Blender allows for most of its interface colors to be changed to suit the needs of the user. If you find that the colors you see on screen do not match those in the Manual, it could be that your default theme has been altered. Creating a new theme or selecting/altering a preexisting one can be done by opening the Preferences and clicking on the *Themes* tab.

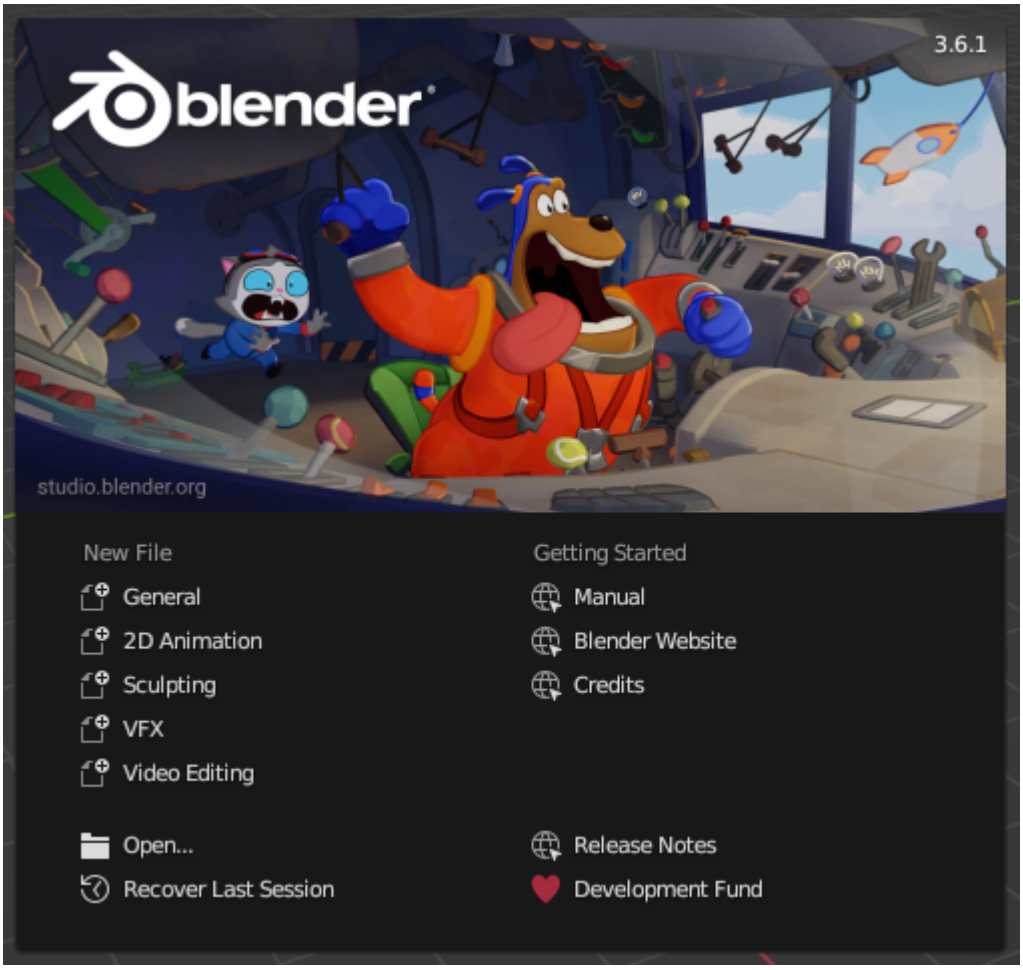
Accessibility

Blender has several options for visibility customization, including resolution scale, and the ability to load custom fonts. These settings can be

configured in the Interface Preferences.

Splash Screen

When starting Blender, the splash screen appears in the center of the window. It contains options to create new projects or open recent ones. A more detailed description can be found below.



Blender Splash Screen.

To close the splash screen and start a new project, click anywhere outside the splash screen (but inside the Blender window) or press `Esc`. The splash screen will disappear revealing the default screen. To reopen the splash screen, click on the Blender icon in the Topbar and select *Splash Screen*.

Note

When starting Blender for the first time or updating to a new version, the "interactive region" contains a Quick Set Up Process.

Splash Image

The upper part of the splash screen contains the splash image with the Blender version in the top right.

Interactive Region

The interactive region is the bottom half of the splash screen.

New File
: Start a new project based on a template.

Recent Files
: Your most recently opened blend-files. This gives quick and easy access to your recent projects.

Open
: Allows opening an existing blend-file.

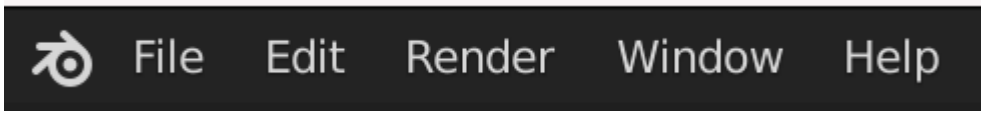
Recover Last Session
: Blender will try to recover the last session based on temporary files. See Recovering Data.

Donate
: Open Blender's Development Fund website.

What's New
: Open the latest release notes.

Topbar

Menus



Blender Menu

Splash Screen
: Open the Splash Screen.

About Blender
: Opens a menu displaying the following information about Blender:

- **Version**: The Blender version.
- **Date**: Date when Blender was compiled.
- **Hash**: The Git Hash of the build. This can be useful to give to support personnel when diagnosing a problem.
- **Branch**: Optional branch name.
- **Windowing Environment**: On Linux, this will show either Wayland or X11 depending on the windowing environment that Blender is running on.
- **Donate**: Open Blender's Development Fund website.
- **What's New**: Open the latest release notes.
- **Credits**: Open the credits webpage.
- **License**: Open the license webpage.
- **Blender Store**: Open the Blender Store website.
- **Blender Website**: Open main Blender website.

Install Application Template
: Install a new application template.

File Menu

The options to manage files are:

New `Ctrl-N`
: Clears the current scene and loads the selected application template.

Open `Ctrl-O`
: Open a blend-file.

Open Recent `Shift-Ctrl-O`
: Displays a list of the most recently opened blend-files. Hovering over items will show a preview, and information about the blend-file. Select any of the file names in the list to open that blend-file.

  Clear Recent Files List
  : Removes items from the recent files list.

Revert
: Reopens the current file to its last saved version.

Recover
: Recover Last Session
  : This will load a blend-file that Blender automatically saves just before exiting. So this option enables you to recover your last work session, e.g. if you closed Blender by accident.

  Recover Auto Save
  : This will open an automatically saved file to recover it.

Save `Ctrl-S`
: Save the current blend-file.

Save As… `Shift-Ctrl-S`
: Opens the File Browser to specify file name and location of save.

Save Copy…
: Saves a copy of the current file.

Save Incremental `Ctrl-Alt-S`
: Save the current Blender file with a numerically incremented name that does not overwrite any existing files.

Link…
: Links data from an external blend-file (library) to the current one. The editing of that data is only possible in the external library. *Link* and *Append* are used to load in only selected parts from another file. See Linked Libraries.

Append…
: Appends data from an external blend-file to the current one. The new data is copied from the external file, and completely unlinked from it.

Data Previews
: Tools for managing data-block previews.

Import
: Blender can use information stored in a variety of other format files which are created by other graphics programs. See Import/Export.

Export
: Normally you save your work in a blend-file, but you can export some or all of your work to a format that can be processed by other graphics programs. See Import/Export.

Export All Collections
: Invokes all configured exporters for all collection.

External Data
: External data, like texture images and other resources, can be stored inside the blend-file (packed) or as separate files (unpacked). Blender keeps track of all unpacked resources via a relative or absolute path. See pack or unpack external data.

Automatically Pack Resources

Pack all currently used external files into the blend-file and automatically pack any files that are added later. Unchecking this option will only stop the automatic packing for new files; it won't unpack existing ones.

Pack Resources

Pack all used external files into the blend-file. After running this operator and saving the blend-file, the external files will no longer be used – any changes in them will no longer be reflected in the blend-file, and you are free to move or delete them.

Unpack Resources

Export previously packed files back to external ones. You can choose whether to reuse existing external files or overwrite them.

Pack Linked Libraries

Pack data-blocks that are linked from an external blend-file into the current one.

Unpack Linked Libraries

Export previously packed data-blocks back to external blend-files. Existing blend-files are overwritten.

Make Paths Relative

Make all paths to external files relative to the current blend-file.

Make Paths Absolute

Make all paths to external files absolute (= full path from the system's root).

Report Missing Files

This option is useful to check if there are links to unpacked files that no longer exist. After selecting this option, a warning message will appear in the Info editor's header. If no warning is shown, there are no missing external files.

Find Missing Files

In case you have broken links in a blend-file, this can help you to fix the problem. A File Browser will show up. Select the desired directory (or a file within that directory), and a search will be performed in it, recursively in all contained directories. Every missing file found in the search will be recovered. Those recoveries will be done as absolute paths, so if you want to have relative paths you will need to select *Make Paths Relative*.

Note

Recovered files might need to be reloaded. You can do that one by one, or you can save the blend-file and reload it again, so that all external files are reloaded at once.

Clean Up

Purge Unused Data

Opens a dialog to remove unused data-blocks from both the current blend-file or any Linked Data (cannot be undone). See the Outliner for more information.

Manage Unused Data

Opens a pop-up window of the Outliner in Unused Data mode which lists data-blocks and other data that are unused and/or will be lost when the file is reloaded. It includes data-blocks which have only a fake user. You can add/remove the Fake User by clicking on cross/tick icon on the right side of the Outliner.

Defaults

This menu manages the startup file which is used to store the default scene, workspace, and interface displayed when creating a new file.

Initially this contains the startup scene included with Blender. This can be replaced by your own customized setup.

Save Startup File

Saves the current blend-file as the startup file.

Load Factory Settings
: Restores the default startup file and preferences.

When an Application Templates is in use the following operators are shown:

Load Factory Blender Settings
: Loads the default settings to the original Blender settings without the changes made from the current application template.

Load Factory (Application Template Name) Settings
: Loads the default settings to the original application template.

See also

Managing Preferences.

Quit `Ctrl-Q`
: Closes Blender. The current scene is saved to a file called "quit.blend" in Blender's temporary directory (which can be found on the "File Paths" tab of the Preferences).

Edit Menu

Undo, Redo, Undo History
: See Undo & Redo.

Adjust Last Operation, Repeat Last, Repeat History
: See Undo & Redo.

Menu Search
: Find a menu based on its name.

Operator Search
: Execute an operator based on its name (Developer Extras only).

Rename Active Item
: Rename the active object or node; see Rename tool for more information.

Batch Rename
: Renames multiple data types at once; see Batch Rename tool for more information.

Lock Object Modes
: Prevents selecting objects that are in a different mode than the current one.

  Note

  This option can prevent accidental mode changes, such as when you're trying to select a bone in Pose Mode to animate it, but instead click a piece of background scenery (which would normally select that piece and switch to Object Mode).

  You may want to disable *Lock Object Modes* for example when weighting rigged objects or sculpting/painting where you intentionally want to switch between objects in different modes.

Preferences `Ctrl-Comma`
: Open the Preferences window.

Render Menu

Render Image `F12`
: Render the active scene at the current frame.

Render Animation `Ctrl-F12`
: Render the animation of the active scene.

  See also

  - Rendering Animations for details.

Render Audio
: Mix the scene's audio to a sound file.

See also

- Rendering audio for details.

View Render `F11`
: Show the Render window. (Press again to switch back to the main Blender window.)

View Animation `Ctrl-F11`
: Playback rendered animation in a separate player.

See also

- Animation player for details.
- Preferences for selecting a different animation player than the default one.

Lock Interface
: Lock interface during rendering in favor of giving more memory to the renderer.

Window Menu

New Window
: Create a new window by copying the current window.

New Main Window
: Create a new window with its own workspace and scene selection.

Toggle Window Fullscreen
: Toggle the current window fullscreen.

Next Workspace
: Switch to the next workspace.

Previous Workspace
: Switch to the previous workspace.

Show Status Bar
: Choose whether the Status Bar at the bottom of the window should be displayed.

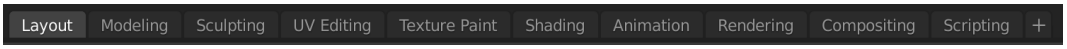
Save Screenshot
: Capture a picture of the current Blender window. A File Browser will open to choose where the screenshot is saved.

Save Screenshot (Editor)
: Capture a picture of the selected Editor. Select the Editor by clicking `LMB` within its area after running the operator. A File Browser will open to choose where the screenshot is saved.

Help Menu

See Help Menu.

Workspaces



This set of tabs is used to switch between Workspaces, which are essentially predefined window layouts.

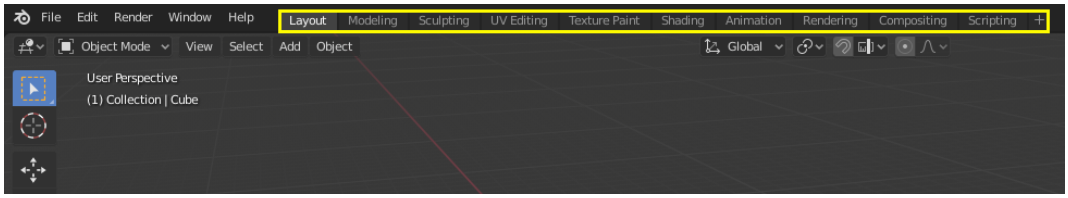
Scenes & Layers



These data-block menus are used to select the current Scene and View Layer.

Workspaces

Workspaces are essentially predefined window layouts. Each Workspace consists of a set of Areas containing Editors, and is geared towards a specific task such as modeling, animating, or scripting. You'll typically switch between multiple Workspaces while working on a project.



Workspaces are located at the Topbar.

Controls

Tabs
: Click on the tabs to switch between the workspaces. You can also use the keyboard shortcuts `Ctrl-PageUp` and `Ctrl-PageDown`. Double-click a tab to rename the workspace.

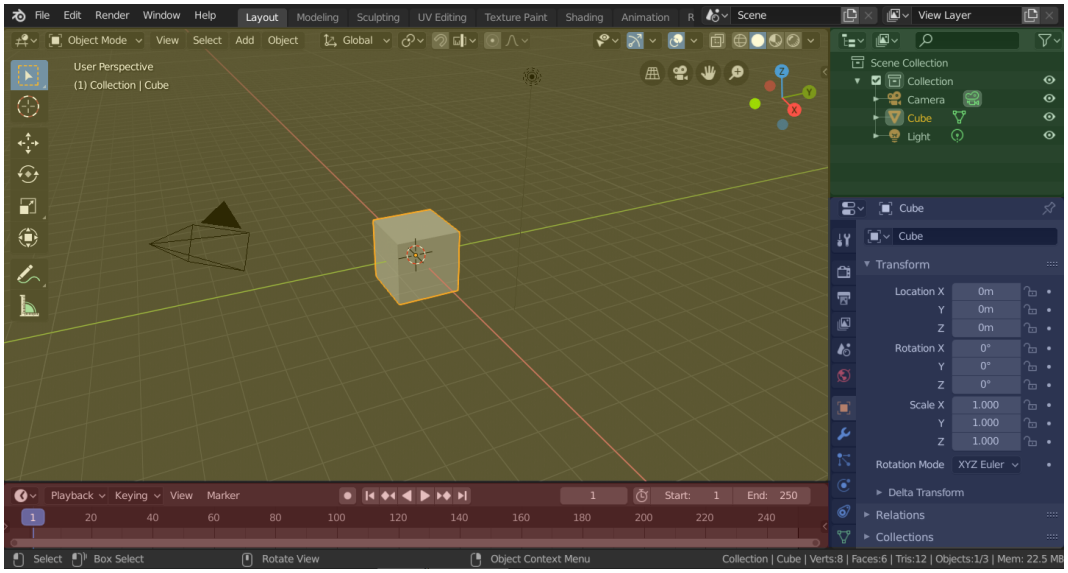
Add +
: Click on the *Add* button to add a new workspace.

Context menu `RMB`
: The context menu contains options to duplicate, delete and reorder workspaces.

Default Workspaces

Blender's default startup shows the "Layout" workspace in the main area. This workspace is a general workspace to preview your scene and contains the following Editors:

- 3D Viewport on top left.
- Outliner on top right.
- Properties on bottom right.
- Timeline on bottom left.



Blender's 'Layout' Workspace with four editors.

3D Viewport (yellow), Outliner (green), Properties (blue) and Timeline (red).

Blender also has several other workspaces added by default:

Modeling::	For modification of geometry by modeling tools.
Sculpting::	For modification of meshes by sculpting tools.
UV Editing::	For mapping of image texture coordinates to 3D surfaces.
Texture Paint::	For coloring image textures in the 3D Viewport.
Shading::	For specifying material properties for rendering.
Animation::	For making properties of objects dependent on time.
Rendering::	For viewing and analyzing rendering results.
Compositing::	For combining and post-processing of images and rendering information.
Geometry Nodes::	For procedural modeling using Geometry Nodes.
Scripting::	

For interacting with Blender's Python API and writing scripts.

Additional Workspaces

Blender has a couple additional Workspaces to choose from when adding a new Workspace:

2D Animation

2D Animation:: General workspace to work with Grease Pencil.
2D Full Canvas:: Similar to "2D Animation" but contains a larger canvas.

VFX

Masking:: For creating 2D masks for compositing or video editing.
Motion Tracking:: For calculating camera motion and stabilizing video footage.

Video Editing

Video Editing:: For sequencing together media into one video.

Save and Override

The workspaces are saved in the blend-file. When you open a file, enabling Load UI in the File Browser indicates that Blender should use the file's screen layout rather than the current one.

A custom set of workspaces can be saved as a part of the Defaults.

Workspace Settings

Reference

Editor:: Properties
Panel:: *Tool tab ▸ Workspace*

Pin Scene
: When enabled, the current workspace will remember the currently selected scene. Then, whenever you activate the workspace, it'll automatically switch back to that scene.

Mode
: Switch to this Mode when activating the workspace.

Filter Add-ons
: Determines which add-ons are enabled in the active workspace. When unchecked, the global add-ons will be used. When checked, you can enable individual add-ons in the list below.

Status Bar

The Status Bar is located at the bottom of the Blender window and displays contextual information such as keyboard shortcuts, messages, and statistical information. The Status Bar can be hidden by disabling *Show Status Bar* in Window menu or by dragging from the top edge down.



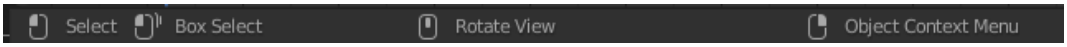
Status Bar.

Keymap Information

The left side of the Status Bar displays mouse button shortcuts and the keymap of the active tool. In editors with a Toolbar, tapping `Alt` (or `Option` on macOS) shows the hotkeys to change to a desired tool.

Tip

This functionality can be disabled with the *Alt Click Tool Prompt* preference in the Keymap Preferences).



Status Messages

The middle of the Status Bar displays information about in-progress operations.



Running Task

Shows the progress of the currently running task (such as rendering or baking). Hovering the mouse pointer over the progress bar will display a time estimate. The task can be aborted by clicking the cancel button ().

Report Message
: Informational messages or warnings, such as after saving a file. They disappear after a short time. Click them to show the full message in the Info Editor.

Resource Information

The right side of the Status Bar displays information about the Blender instance. Which information is shown can be chosen by `RMB` on the Status Bar or in the Preferences.



Scene Statistics
: Shows information about the data in the active scene.

- **Collection**: The name of the active Collection.
- **Active Object**: The name of the active selected object.
- **Geometry**: Information about the current scene depending on the mode and object type. This can be the number of vertices, faces, triangles, or bones.
- **Objects**: The number of selected objects and the total count of objects.

Scene Duration
: Shows the total amount of time of the playback along with the current frame number and total frame count. The format of the duration text is determined by the Timecode Style.

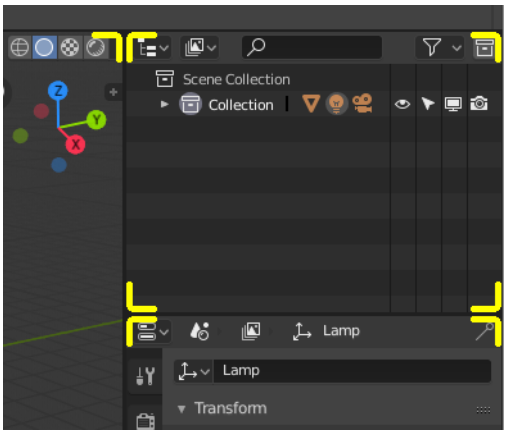
System Memory
: Shows an estimate of Blender's RAM consumption. On a single-instance single-machine scenario, this estimate provides a measurement against the hardware limit of the machine.

Extensions Updates
: Shows the number of extensions with available updates.

Blender Version
: Shows the version number of Blender that is currently running.

Areas

The Blender window is divided into a number of rectangles called Areas. Areas reserve screen space for Editors, such as the 3D Viewport or the Outliner. Each editor offers a specific piece of functionality.



Area boundaries are indicated by rounded corners (yellow highlights).

Areas are grouped into Workspaces, which are geared towards particular tasks (modeling, animating and so on).

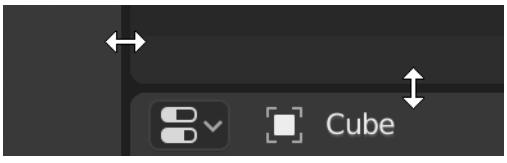
Note

While some keyboard shortcuts in Blender are global (such as `Ctrl-S` for saving), many depend on which editor the mouse cursor is hovering over.

As an example, say you just selected two objects in the Outliner and want to join them. If you pressed the shortcut for this (`Ctrl-J`) while the cursor is still in the Outliner, nothing would happen as the shortcut isn't valid there; you first need to move your cursor to the 3D Viewport.

Resizing

You can resize areas by dragging their borders with `LMB`. Move your mouse cursor over the border between two areas, so that the cursor changes to a double-headed arrow, and then click and drag.



Tip

Hold `Ctrl` to snap the size of areas to convenient sizes.

Docking

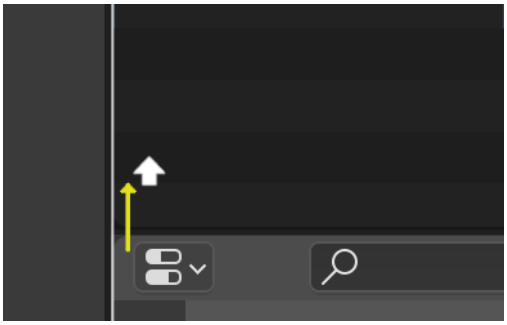
Docking describes several ways an area a user can interactively manipulate the size and location of areas along with splitting an area into new areas.

To start the interactive process, placing the mouse cursor in an area corner will change the cursor to a cross (+). Once the cursor is a cross, press and hold `LMB` to preform any of the following actions:

If you press `Esc` or `RMB` before releasing the mouse, the operation will be canceled.

Joining

Dragging from an area corner into the space of a second area will *join* two areas. The areas that will be joined will be displayed brighter.



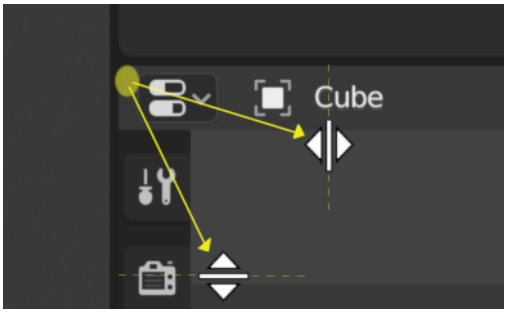
Properties is being joined to the Outliner.

Splitting

Splitting an area will create a new area. Dragging from an area corner left/right will split the area vertically, to split the area horizontally drag up/down.

You can split and join areas at once by dragging a split operation into a separate area.

Dragging an area into the middle of an second area will replace the second area with the first area.



Area Options

`RMB` on the border opens the *Area Options*.

Vertical/Horizontal Split
: Shows an indicator line that lets you select the area and position where to split. `Tab` switches between vertical/horizontal.

Join Up/Down/Left/Right
: Shows the join direction overlay.

Swap Areas
: Swaps this area with the adjacent one.

Swapping Contents

You can swap the contents of two areas by pressing `Ctrl-LMB` on one of the corners of the initial area, dragging towards the target area, and releasing the mouse there. The two areas do not need to be side-by-side, though they must be inside the same window.

Duplicate Area into new Window

Reference

Menu:: *View ▸ Area ▸ Duplicate Area into new Window*

A new floating window containing an area can be created from *View ▸ Area ▸ Duplicate Area into new Window*. (Not available in some editors.)

The new window is a fully functional window, which is part of the same instance of Blender. This can be useful, e.g. if you have multiple monitors.

You can also create a new window from an existing area by pressing `Shift-LMB` on an area corner, then dragging outward slightly.

Toggle Maximize Area

Reference

Menu::	*View ▸ Area ▸ Toggle Maximize Area*
Shortcut::	`Ctrl-Spacebar`

Expands the Area so it fills the whole window (while keeping the Topbar and Status Bar visible). To return to normal size, use the keyboard shortcut again or click the *Back to Previous* button in the Topbar.

Toggle Fullscreen Area

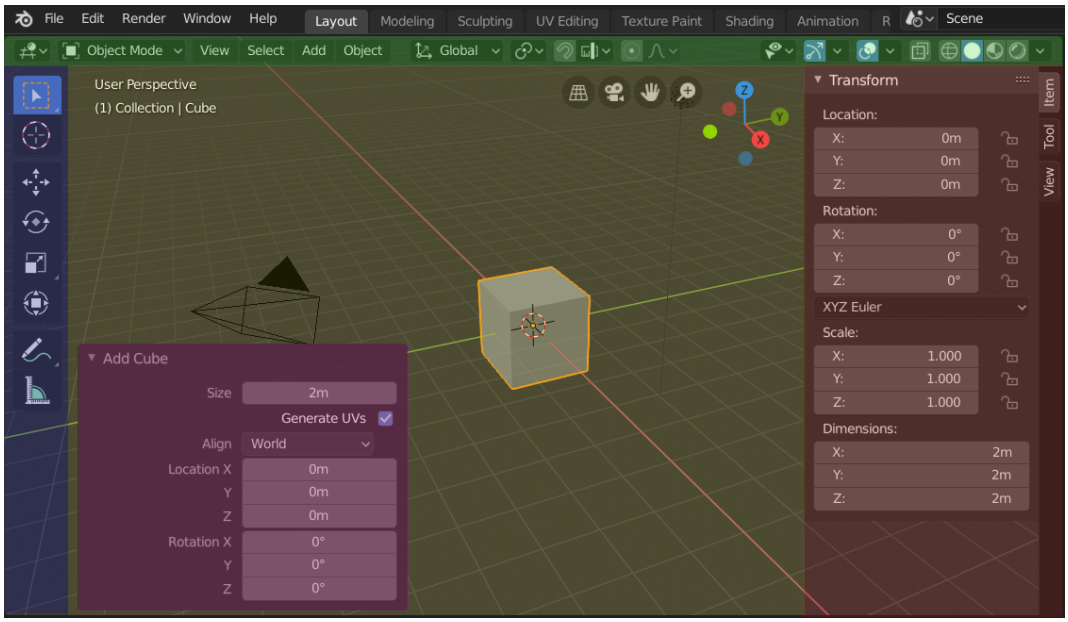
Reference

Menu::	*View ▸ Area ▸ Toggle Fullscreen Area*
Shortcut::	`Ctrl-Alt-Spacebar`

Expands the Area so it fills the whole window, hiding the Topbar, Status Bar, and even the secondary regions (toolbars etc.) of the Area's own editor. To return to normal size, use the keyboard shortcut again or click the icon in the Area's top right corner (only becomes visible when hovering).

Regions

Every Editor in Blender is divided into Regions. Regions can have smaller structuring elements like tabs and panels with buttons, controls and widgets placed within them.



The regions of the 3D Viewport showing the Sidebar and the Adjust Last Operation panel after adding a Cube.

Header (green), Main region (yellow), Toolbar (blue), Sidebar (red) and Adjust Last Operation panel (pink).

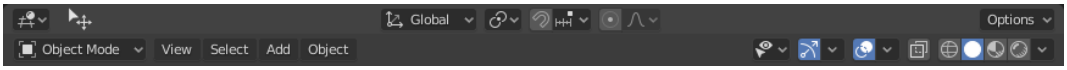
Main Region

At least one region is always visible. It is called the Main region and is the most prominent part of the editor.

Each editor has a specific purpose, so the main region and the availability of additional regions are different between editors. See specific documentation about each editor in the Editors chapter.

Header

A header is a small horizontal strip, which sits either at the top or bottom of an area. All editors have a header acting as a container for menus and commonly used tools. Menus and buttons will change with the editor type and the selected object and mode.



The Header of the 3D Viewport.

Context Menu

RMB on a header reveals a context menu with a couple options:

Show Header
: Toggles the visibility of the header. If a header is hidden, it can be made visible again by clicking or dragging the small arrow that appears at the top/bottom right of the editor.

Show Tool Settings
: Toggles the visibility of the Tool Settings.

Show Menus
: Toggles whether the Menus are collapsed or not.

Flip to Bottom/Top
: Toggles whether the header or Tool Settings appear on the top or bottom of the editor.

Vertical/Horizontal Split
: Shows an indicator line that lets you select the area and position where to split. Tab switches between vertical/horizontal.

Maximize/Full Screen Area
: See Toggle Maximize Area.

Duplicate Area into New Window
: See Duplicate Area into new Window.

Close Area
: Closes the area and replaces it with the expansion of a neighboring area.

Toolbar

The Toolbar (on the left side of the editor area) contains a set of interactive tools. `T` toggles the visibility of the Toolbar.

Tool Settings

A horizontal strip at the top or bottom of the editor (similar to the header) containing settings for the currently selected tool. Just like the header, it can be hidden and moved through its context menu.

Adjust Last Operation

Adjust Last Operation is a region that allows tweaking an operator after running it. For example, if you just added a cube, you can use this region to tweak its size.

Sidebar

The *Sidebar* (on the right side of the editor area) contains Panels with settings of objects within the editor and the editor itself. `N` toggles the visibility of the Sidebar.

Footer

Some editors show a bar (on top/bottom of the editor area) that displays information about for example the active tool or operator.

Arranging

Scrolling

A region can be scrolled vertically and/or horizontally by dragging it with the `MMB`. If the region has no zoom level, it can also be scrolled by using the `Wheel` while the mouse hovers over it.

Some regions, in particular animation timelines, have scrollbars with added control points to adjust the vertical or horizontal range of the region. These special scrollbars will have added widgets at the ends, as shown in the following image:



Scrollbars with zoom widgets.

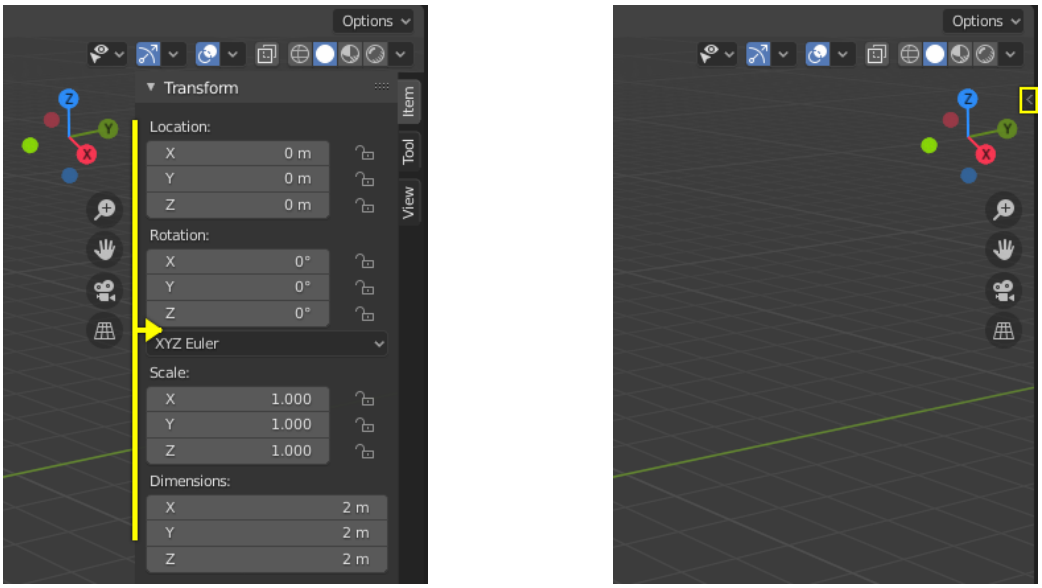
This can be used to stretch or compress the range to show more or less detail within the available screen space. Simply drag one of the dots to either increase or decrease the displayed range. You can also quickly adjust both the horizontal and vertical range by dragging in the editor with `Ctrl-MMB`.

Changing the Size and Hiding

Resizing regions works by dragging their border, the same way as Areas.

To hide a region, resize it down to nothing. A hidden region leaves a little arrow sign. `LMB` on this icon to make the region reappear.

Hiding and showing the Sidebar.



Scaling

The scale of certain regions (such as the Toolbar) can be changed by dragging inside them with `Ctrl-MMB`, or using `NumpadPlus` and `NumpadMinus` while hovering the mouse cursor over them. Press `Home` to reset the scale to the default.

Asset Shelf

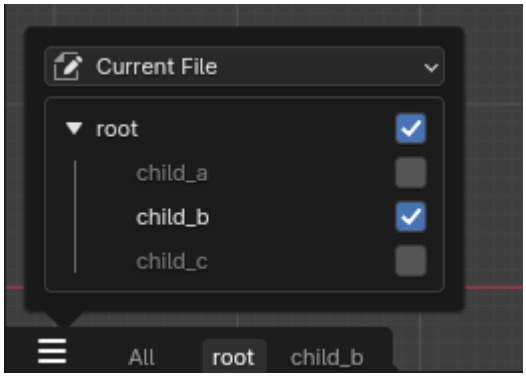


The Asset Shelf of the 3D View, showing material assets.

Search

To search for assets, hover your mouse over the Asset Shelf, then press Ctrl-F and type a search query. This will filter the poses to match what you typed.

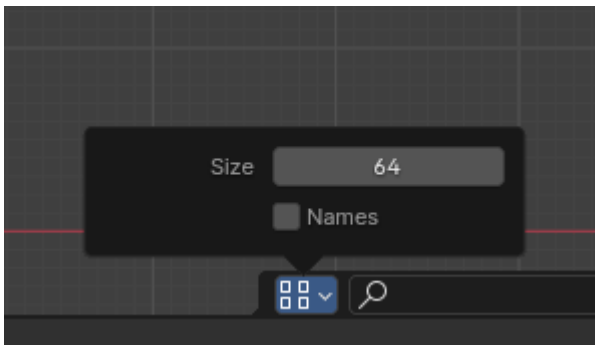
Tabs



The usage of catalogs as tabs.

Catalogs can be shown as individual tabs. Each tab will only show its content, and the content of its children. That makes it easy to filter down to a certain set of assets.

Display Options



Display options available for the asset shelf.

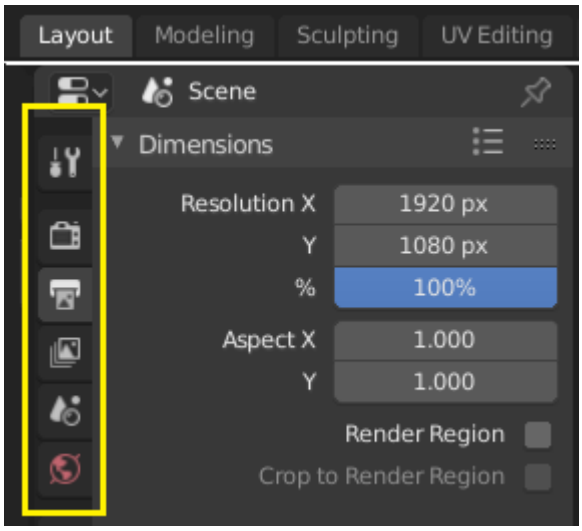
It is possible to change the size of items on the shelf by using the size property.

By toggling on the “Names” checkbox, the asset names will be shown in the shelf. Alternatively it is also possible to hover over an item to show its name.

Tabs & Panels

Tabs

Tabs are used to control overlapping sections in the user interface. The content of only one Tab is visible at a time. Tabs are listed in a *Tab header*, which can be horizontal or vertical.



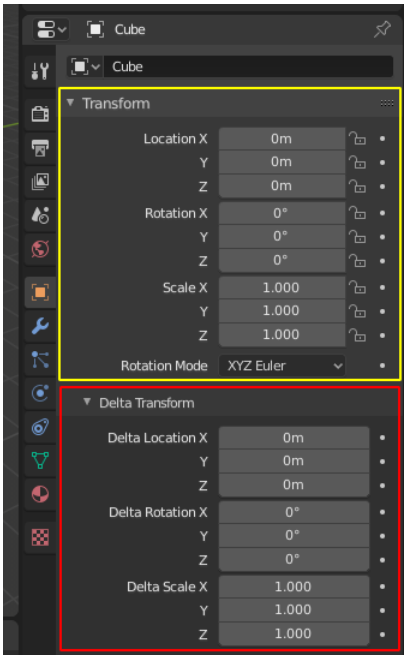
Top: Horizontal Tab header in the Topbar. Bottom: Vertical Tab header shows tab icons in the Properties.

Switching/Cycling

Vertical tabs can be switched with `Ctrl-Wheel` from anywhere in the tab. You can also cycle through tabs with `Ctrl-Tab` and `Shift-Ctrl-Tab`, or press down `LMB` and move the mouse over the tab header icons. (This does not apply to Workspace tabs; see Workspace controls.)

Panels

The smallest organizational unit in the user interface is a panel. The panel header shows the title of the panel. It is always visible. Some panels also include subpanels.



Panels in Properties.

A panel is highlighted in yellow and a subpanel in red.

Collapsing and Expanding

A panel can either be expanded to show its contents, or collapsed to hide its contents. An expanded panel is indicated by a down-arrow (▼) in the panel header, while a collapsed panel is shown with a right-arrow (▶).

- Clicking `LMB` on the panel header expands or collapses it.
- Pressing `A` expands/collapses the panel under the mouse pointer.
- Clicking `Ctrl-LMB` on the header of a collapsed panel will expand it and collapse all others.
- Clicking `Ctrl-LMB` on the header of an expanded panel will expand/collapse all its subpanels.
- Dragging with `LMB` over the headers will expand or collapse many at once.

Position

You can change the position of a panel within its region by clicking and dragging the grip widget (::::) on the right side of its header.

Pinning

Sometimes it is desirable to view panels from different tabs at the same time. Like, for instance, having access to a camera's properties, while other objects are selected. This has been solved by making panels pinnable.

A pinned panel remains visible regardless of which tab has been selected. You can pin a panel by clicking on the pin icon in its header. Panels that do not have a pin icon can be pinned by `RMB` on the panel header and selecting *Pin*, or by pressing `Shift-LMB`.

Note

Pinning is not available for all panels. For example, it's available in the Sidebar but not in the Properties editor.

Presets



Example Presets menu.

Selector
: A list of available presets. A selection will override the included properties.

Add +
: New presets can be added based on the currently applied set of properties, which will be saved for later reuse. A pop-up opens where you can set a name, after which you can select it from the list and in some cases additional settings.

Remove –
: Deletes the selected preset.

Keymap

- Common Shortcuts
- Default Keymap
- Industry Compatible Keymap

Common Shortcuts

Conventions Used in This Manual

Keyboards

Hotkey letters are shown in this manual like they appear on a keyboard; for example:

`G`
: refers to the lowercase `g`.

`Shift`, `Ctrl`, `Alt`
: are specified as modifier keys.

`Ctrl-W`, `Shift-Alt-A`, ...
: indicates that these keys should be pressed simultaneously.

`Numpad0` to `Numpad9`, `NumpadPlus`
: refer to the keys on the separate numeric keypad.

Other keys are referred to by their names, such as `Esc`, `Tab`, `F1` to `F12`. Of special note are the arrow keys, `Left`, `Right` and so on.

Mouse

This manual refers to mouse buttons as:

`LMB`
: Left Mouse Button

`RMB`
: Right Mouse Button

`MMB`

Middle Mouse Button

`Wheel`, `WheelUp` & `WheelDown`
: Scrolling the wheel.

Note

Blender has two main selection modes: left-click select and right-click select. See the Select with Mouse Button preference.

While left-click select is the default as it's the most common in other applications, right-click select does have its advantages. See: Learn the benefits of right-click select.

Hovering

While hovering (when the cursor is held over a button).

Properties

- `Ctrl-C` – Copy the (single) value of the button.
- `Ctrl-V` – Paste the (single) value of the button.
- `Ctrl-Alt-C` – Copy the entire vector or color of the field.
- `Ctrl-Alt-V` – Paste the entire vector or color of the field.
- `RMB` – Open the context menu.
- `Backspace` – Clear the value (sets to zero or clears a text field).
- `Minus` – Negate number values (multiply by -1.0).
- `Ctrl-Wheel` – Change the value incremental steps.

  For pop-up option menus buttons, this cycles the value.

- `Return` – Activates menus or toggles the value.
- `Alt` – Hold while editing values to apply the change to all selected items (objects, bones, sequence-strips).

  This can be used for number fields and toggles.

Animation

- `I` – Insert a keyframe.
- `Alt-I` – Clear the keyframe.
- `Shift-Alt-I` – Clear all keyframes (removing all F-Curves).
- `Ctrl-D` – Assign a driver.
- `Ctrl-Alt-D` – Clear the driver.
- `K` – Add a Keying Set.
- `Alt-K` – Clear the Keying Set.

Python Scripting

- `Ctrl-C` over any Operator Buttons copies their Python command into the clipboard.

  This can be used in the Python Console or in the Text editor when writing scripts.

- `Shift-Ctrl-C` over property buttons copies their data path for this property (also available from the context menu).

  Useful when writing drivers or scripts.

- `Shift-Ctrl-Alt-C` over property buttons copies their *full* data path for the data-block and property.

  Note that in most cases it is best to access values based on the context, instead of by name.

Dragging

- `Ctrl` – While dragging, snap to discrete steps.
- `Shift` – Gives precision control over the value.
- `Shift-Ctrl` – Precise snap will move the object with high precision along with the snapping constraint.

Text Editing

- `Home` – Go to the start of the line.
- `End` – Go to the end of the line.
- `Left`, `Right` – Move the cursor a single character.
- `Ctrl-Left`, `Ctrl-Right` – Move the cursor an entire word.
- `Backspace`, `Delete` – Delete characters.
- `Ctrl-Backspace`, `Ctrl-Delete` – Delete words.
- `Shift` – Select while holding the key and moving the cursor.
- `Ctrl-A` – Select all text.
- `Ctrl-C` – Copy the selected text.
- `Ctrl-X` – Cut the selected text.
- `Ctrl-V` – Paste text at the cursor position.

Confirm & Cancel

- `Esc`, `RMB` – Cancel.
- `Return`, `LMB` – Confirm.

Default Keymap

While this isn't a comprehensive list, this page shows common keys used in Blender's default keymap.

Global Keys

`Ctrl-O`	Open file.
`Ctrl-S`	Save file.
`Ctrl-N`	New file.
`Ctrl-Z`	Undo.
`Shift-Ctrl-Z`	Redo.
`Ctrl-Q`	Quit.
`F1`	Help *(context sensitive)*.
`F2`	Rename active item.
`F3`	Menu Search.

`F4`	File context menu.
`F5` - `F8`	*Reserved for user actions*.
`F9`	Adjust last operation.
`F10`	*Reserved for user actions*.
`F11`	Show render window.
`F12`	Render the current frame.
`Q`	Quick access (favorites).
`Ctrl-Spacebar`	Toggle Maximize Area.
`Ctrl-PageUp` / `Ctrl-PageDown`	Next/previous Workspace.
`Spacebar`	User configurable. **Play::** Toggle animation playback. **Tools::** Tool switching with hotkeys (`Shift-Spacebar` for play). **Search::**

	Search for actions (`Shift-Spacebar` for play).
`Shift-Ctrl-Spacebar`	Playback animation (reverse).

Common Editing Keys

`X`	Deletes the selected item, requires a confirmation dialog.
`Delete`	Deletes the selected item, does not require a confirmation dialog.

Common Editor Keys

These keys are shared across editors such as the 3D Viewport, UV and Graph editor.

`A`	Select all.
`Alt-A`	Select none.
`Ctrl-I`	Invert selection.
`H`	Hide selection.

`Alt-H`	Reveal hidden items.
`T`	Toggle Toolbar.
`N`	Toggle Sidebar.

3D Viewport Keys

`Tab`	Toggle Edit mode.
`Ctrl-Tab`	Toggle Pose mode for armatures, or show a mode switching pie menu for others.
`1 - 3`	In Edit Mode, switch between editing vertices (`1`), edges (`2`), or faces (`3`). Hold `Shift` to toggle one of these without disabling the others. Hold `Ctrl` to alter how the selection is transformed from the old mode to the new. See Mesh Selection Modes for details.
`AccentGrave`	Show 3D Viewport navigation pie menu.
`Ctrl-AccentGrave`	Toggle gizmos.

`Shift-AccentGrave`	Start Walk/Fly Navigation.

Platform Specific Keys

macOS

The `Cmd` key can be used instead of `Ctrl` on macOS for all but a few exceptions which conflict with the operating system.

List of additional macOS specific keys:

`Cmd-Comma`	Preferences.

Keymap Preferences

Select with Mouse Button
: Controls which mouse button, either left or right, is used to select items in Blender. If *Left* is selected, the `RMB` will be a context sensitive menu. If *Right* is selected, the `LMB` will place the 3D Cursor.

Spacebar Action
: Controls the action of `Spacebar`. These and other shortcuts can be modified in the keymap preferences.

  Play:: Starts playing through the Timeline. This option is good for animation or video editing work.

  Tools:: Opens the Toolbar underneath the cursor to quickly change the active tool. This option is good if you are doing a lot of modeling or rigging work.

  Search::

Opens up the Menu Search. This option is good for someone who is new to Blender and is unfamiliar with the menus and shortcuts.

Activate Gizmo Event
: The activation event for gizmos that support drag motion. This option is only available when Left click *Select with Mouse Button* is chosen.

Press:: The gizmo's operation gets initiated (and additional options become available in the Status Bar) the moment you press down the mouse button on the gizmo.

Drag:: The operation only gets initiated once you start dragging the gizmo.

Tool Keys
: The method of keys to activate tools such as move, rotate, and scale.

Immediate:: Activate actions immediately.

Active Tool:: Activate the tool for editors that support tools.

Alt Click Tool Prompt
: Tapping `Alt` shows a prompt in the status bar prompting a second keystroke to activate the tool. Note that this option is not available when using Emulate 3 Button Mouse.

Alt Tool Access
: Hold `Alt` to use the Active Tool when the gizmo would normally be required. (For example, with the Move tool selected, you can hold `Alt` and drag the mouse anywhere in the viewport to move the selected object, rather than having to drag its gizmo.) This option is only available when *Select with Mouse Button* is set to Left click and Emulate 3 Button Mouse is disabled.

Select All Toggles
: Causes selection shortcut `A` to deselect all when any selection exists.

Region Toggle Pie

`N` opens a pie menu to toggle Regions, rather than toggling a single region.

3D Viewport

Grave Accent / Tilde Action

Navigate:: Navigation pie menu, useful on systems without a numeric keypad.

Gizmos:: Transform gizmos pie menu, useful for quickly switching between transform gizmos. Note that this doesn't apply to tools that force a certain gizmo (Move, Rotate, Scale and Transform); if you have such a tool selected, the gizmo will stay the same no matter what you choose in the pie menu.

Middle Mouse Action

The action when `MMB` dragging in the viewport, this also applies to trackpads.

Orbit:: Rotates the view around a pivot point, `Shift-MMB` is used for panning the view.

Pan:: Shifts the view towards the mouse, `Shift-MMB` is used for orbiting the view.

Alt Middle Mouse Drag Action

Relative:: Set the view axis where each mouse direction maps to an axis relative to the current orientation.

Absolute:: Set the view axis where each mouse direction always maps to the same axis.

Tab for Pie Menu

Causes `Tab` to open a pie menu (swaps `Tab` and `Ctrl-Tab`).

Pie Menu on Drag

This allows keys to have a secondary drag action.

`Tab`

tap:: Toggles Edit Mode.
drag:: Object Mode pie menu.

`Z`
tap:: Toggles wireframe view.
drag:: Display mode pie menu.

`AccentGrave`
tap:: First person Fly/walk Navigation.
drag:: View axis pie menu.

Extra Shading Pie Menu Items
Show additional items in the shading menu (`Z` key).

Transform Navigation with Alt
During transformations, use `Alt` to navigate the 3D Viewport.

When disabled, hotkeys for proportional editing, automatic constraints, and auto IK chain length will require holding `Alt`.

File Browser

Open Folders on Single Click
Navigate into folders by clicking on them once instead of twice.

Industry Compatible Keymap

While this is not a comprehensive list, this page shows common keys used in the industry compatible keymap.

General

`1 - 3`	Switch Selection mode
`4``	Object Mode
`5``	Modes Pie Menu
`RMB`	Context menu
`Tab`	Menu Search.
`Shift-Tab`	Quick access (favorites)
`Return`	Rename
`Ctrl-Return`	Render
`Ctrl-[`	Toggle Toolbar

`Ctrl-]`	Toggle Sidebar

Common Editing Keys

`Backspace`	Deletes the selected item, requires a confirmation dialog.
`Delete`	Deletes the selected item, does not require a confirmation dialog.
`Ctrl-D`	Duplicate
`P`	Set Parent
`B`	Proportional Editing / Soft Selection

Viewport

`Alt-LMB`	Orbit View
`Alt-MMB`	Pan View
`Alt-RMB`	Zoom View
`F1 - F4`	Front/Side/Top/Camera Viewpoints

`F`	Frame Selected
`Shift F`	Center View to Mouse
`A`	Frame All

Selection

`LMB`	Select
`Ctrl-A`	Select All
`Shift-Ctrl-A`	Deselect All
`Ctrl-I`	Select Inverse
`Up`	Select More
`Down`	Select Less
Double `LMB`	Select Loop
Double `Alt-LMB`	Select Ring

`Ctrl L`	Select Linked

Tools

`W, E, R`	Transform Tools
`Q`	Select Tools
`D`	Annotate
`C`	Cursor Tool

Edit Mode Tools

`Ctrl-E`	Extrude
`Ctrl-B`	Bevel
`I`	Inset
`K`	Knife
`Alt-C`	Loop Cut

Animation

`Spacebar`	Play/Pause
`S`	Set Location + Rotation + Scale keyframe
`Shift-S`	Insert Keyframe menu
`Shift-W`	Set Location Key
`Shift-E`	Set Rotation Key
`Shift-R`	Set Scale Key

Platform Specific Keys

macOS

The `Cmd` key can be used instead of `Ctrl` on macOS for all but a few exceptions which conflict with the operating system.

UI Elements

- Buttons
- Input Fields
- Menus
- Eyedropper
- Decorators
- Data-Block Menu
- List View
- Color Picker
- Color Ramp Widget
- Color Palette
- Curve Widget

Buttons

Operator Buttons

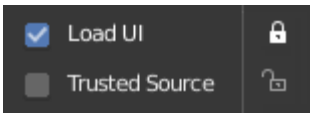
Operator buttons execute an Operator which in summary execute an action when clicked with LMB. Operator buttons may be an icon, text, or text with an icon.



Operator button.

Checkboxes & Toggle Buttons

These controls are used to activate or deactivate options. Use LMB to change their state. A tick is shown on checkboxes when the option is activated. Active status on toggle buttons is indicated either by color on the icon background, or a change in icon graphics.



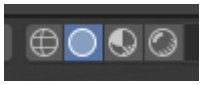
Checkboxes and Toggle buttons.

Dragging

To change many values at once on or off, you can press down LMB and drag over multiple buttons. This works for checkboxes, toggles and to select a radio button value.

Radio Buttons

Radio buttons are used to choose one option from a selection of options. The active button is indicated by a colored background.



Radio buttons.

Cycling

Use `Ctrl-Wheel` while hovering the mouse over radio buttons to cycle between the options.

Direction Buttons

Clicking with `LMB` in the sphere and dragging the mouse cursor lets the user change the direction by rotating the sphere.



Direction buttons.

Shortcuts

- `LMB` (drag) rotates the direction.
- `Ctrl` (while dragging) snaps to vertical & diagonal directions.

Input Fields

Text & Search Fields



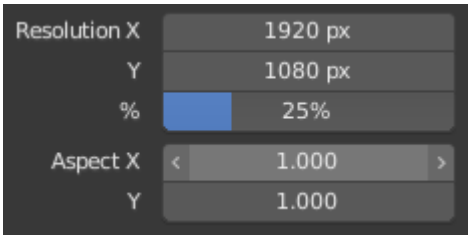
A text and a search field.

Text fields show a rounded rectangular border, and optionally an icon and/or text inside the border. Text fields store text strings, and provide the means to edit text by standard text editing shortcuts.

For text fields with an icon and pop-ups, see Data ID.

Number Fields

Number fields store values and units.



Number fields.

The first type of number field shows triangles pointing left (<) and right (>) on the sides of the field when mouse pointer is over the field.

Sliders, a second type of number field, have a colored bar in the background to display values over a range, e.g. percentage values.

The value can be edited in several ways:

Incremental Steps
: To change the value in unit steps, click `LMB` on the small triangles (not available for sliders). You can also use `Ctrl-Wheel` while hovering over the field to edit the value.

Dragging
: To change the value with the mouse, hold down LMB and drag to left or right.

  Hold Ctrl to snap to the discrete steps while dragging or Shift for precision input.

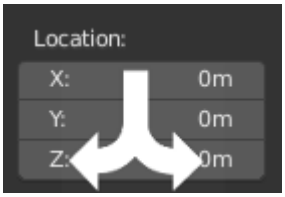
Keyboard Input
: Press LMB or Return to enter value by typing it with keyboard.

  When entering values by keyboard, number fields work like text fields:

  - Press Return or LMB outside the field to apply the change.
  - Press Esc or RMB to cancel.
  - Press Tab to jump to the next field or Shift-Tab to go to the previous field.
  - Press Minus while hovering over a number field to negate the value.

Multi-Value Editing

You can edit multiple number fields at once by pressing down LMB on the first field, and then dragging vertically over the fields you want to edit. Finally you can either drag left or right to adjust value with the mouse, or release the LMB and type in a value.



Multi-value editing.

Value Limits

Most numerical values are restricted by “soft limit” and “hard limit” value ranges. Changing values by dragging with the mouse is restricted to the “soft limit” value range. Input via keyboard will allow the use of wider value ranges, but never wider than the “hard limit”.

Expressions

You can enter mathematical expressions into any number field. For example, enter `3*2` or `10/5+4` instead of `6`. Even constants like `pi` (3.142) or functions like `sqrt(2)` (square root of 2) may be used.

See also

These expressions are evaluated by Python; for all available math expressions see: Math module reference.

Expressions as Drivers

You may want your expression to be re-evaluated after it is entered. Blender supports this using Drivers (a feature of the animation system).

Expressions beginning with `#` have a special use. Instead of evaluating the value and discarding the expression, a driver is added to the property with the expression entered.

The expression `#frame` is a quick way to map a value to the current frame, but more complex expressions like `#fmod(frame, 24) / 24` are also supported.

This is simply a convenient shortcut to add drivers which can also be added via the `RMB` menu.

Units

As well as expressions, you can specify numbers and units. If no unit is given, then a default unit is applied. The unit system can be changed in scene settings.

You can use either the unit abbreviation or the full name after the value.

Examples of valid usage of length units include:

- `1cm`
- `1m 3mm`
- `1m, 3mm`
- `2ft`
- `3ft/0.5km`
- `2.2mm + 5' / 3" - 2yards`

Note

Using Units

- Decimal separator is optional.
- You can mix units, e.g. metric and imperial even though you can only show one at a time.
- Plurals of the names are recognized too, so `meter` and `meters` can both be used.

Color Fields

The color field stores a color value. Clicking on it with `LMB` opens the Color Picker.



Color fields. With and without alpha.

Color fields with an alpha channel are divided in half: on the left, the color is shown without an alpha channel, and on the right, it's shown with an alpha channel over a checker pattern.

Colors can be copied to other color fields by dragging and dropping.

Hovering over a color property will display a large swatch preview of the color and the color's hexadecimal, RGBA, and HSVA values.

Menus

Blender uses a variety of different menus for accessing options and Operators. Menus can be interacted with in the following ways:

Mouse selection
: `LMB` on the desired item.

Numerical selection
: You can use the number keys or numpad to input an item in the list to select. For example, `Numpad1` will select the first item and so on.

If a menu is too large to fit on the screen, a small scrolling triangle appears on the top or bottom. Scrolling is done by moving the mouse above or below this triangle.

Shortcuts

- Use `Wheel` while hovering with the mouse.
- Arrow keys can be used to navigate.
- Each menu item has an underlined character which can be pressed to activate it.
- Number keys or numpad can also be used to activate menu items. (`1` for the first menu item, `2` for the second etc. For larger menus, `Alt-1` activates the 11th and so on, up to `Alt-0` for the 20th.)
- Press `Return` to activate the selected menu item.
- Press `Esc` to close the menu without activating any menu item. This can also be done by moving the mouse cursor far away from the menu, or by `LMB` clicking anywhere outside of it.

Popup Menus

Popup menus list Operators which can be executed by selecting with `LMB` or using the generated shortcut indicated by the underlined character of the

operator name. All menu entries show any relevant shortcut keys, which can be executed without opening the menu.

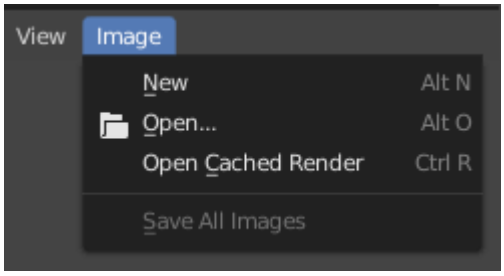


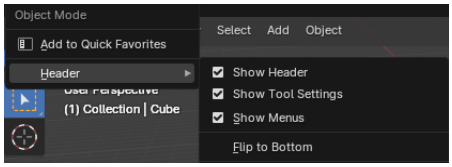
Image menu in the Header of the Image editor.

All popup menus can be searched by pressing `Spacebar` and typing the name of the operator in the menu. If a popup menu has "Search" as one of the items, the menu can be searched without having to press `Spacebar` first.

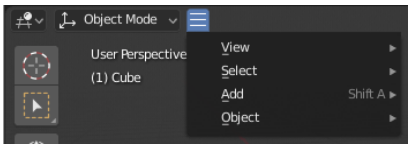
All popup menus of an editor can be searched using the Menu Search feature.

Collapsing Menus

Sometimes it's helpful to gain some extra horizontal space in the header by collapsing menus. This can be accessed from the header context menu: click `RMB` on the header and uncheck the *Show Menus checkbox*.



Right-click on any of the header menus.

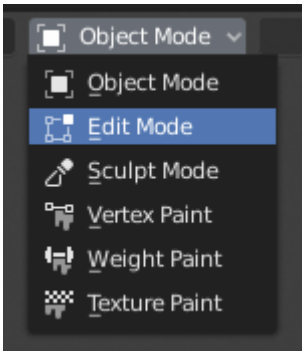


Access the menu from the collapsed icon.

Select Menus

A Select menu (or "selector" for short) lets you choose between a set of options. It appears as an icon and/or text with a down arrow on the right side. To use it, click the button with `LMB` to show the available options, then

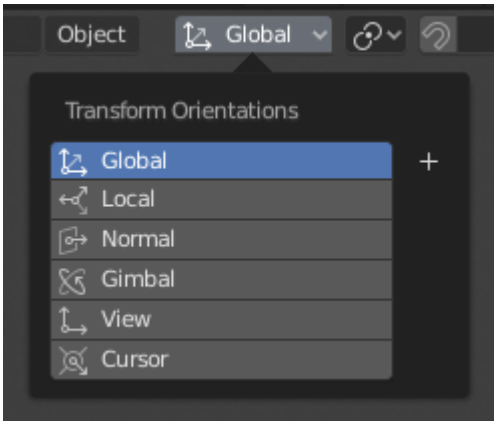
click the desired option (once selected, it'll appear inside the button). You can also use `Ctrl-Wheel` to cycle through the options without opening the menu.



The 3D Viewport Mode Select menu.

Popover Menus

Popover menus are similar to Select Menus, but can show more varied content such as a title, buttons, sliders, etc.



The Transform Orientations popover menu.

Context Menu

Context menus are pop-ups that can be opened with `RMB`. In most editors, it's also possible to use the `Menu` key. The contents of the menu depend on the location of the mouse pointer.

When invoked in an editor, the menu contains a list of operators sensitive to the editor's mode. When invoked over buttons and properties, common

options include:

Single
: Apply the change to a single value of a set (e.g. only the X coordinate of an object's Location).

All
: Apply the change to all values in a set (e.g. all coordinates of an object's Location).

Reset to Default Value(s) `Backspace`
: Replaces the current value by the default.

Copy Data Path `Shift-Ctrl-C`
: Copies the Python property data path, relative to the data-block. Useful for Python scripting.

Copy Full Data Path `Shift-Ctrl-Alt-C`
: Copies the full Python property data path including any needed context information.

Copy As New Driver
: Creates a new driver using this property as input, and copies it to the clipboard. Use *Paste Driver* to add the driver to a different property, or *Paste Driver Variables* to extend an existing driver with a new input variable.

Copy To Selected
: Copies the property value to the selected object's corresponding property. A use case is if the Properties context is pinned.

Assign Shortcut
: Lets you define a keyboard or mouse shortcut for an operation. To define the shortcut you must first move the mouse cursor over the button that pops up. When "Press a key" appears you must press and/or click the desired shortcut. Press `Esc` to cancel.

See also

Common Shortcuts.

Change Shortcut
: Lets you redefine the shortcut.

Remove Shortcut
: Unlinks the existing shortcut.

Open File Location, Open Location Externally
: Opens the containing folder using the operating system's file manager.

Online Manual `F1`
: Opens an online page of the Blender Manual in a web browser.

Online Python Reference
: Context-sensitive access to the Python API Reference.

Edit Source
: For UI development – Creates a text data-block with the source code associated with the control, in case the control is based on a Python script. In the Text Editor it points at the code line where the element is defined.

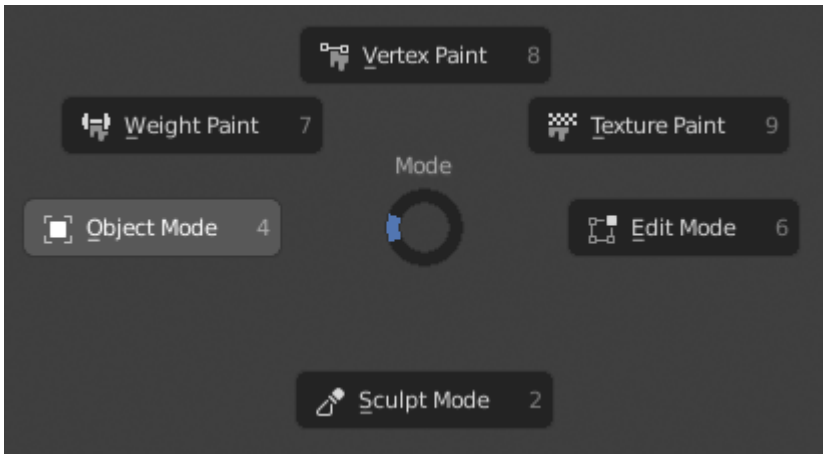
Edit Translation
: For UI development – Points at the translation code line.

Specials Menu

The Specials pop-up menu is similar to a context menu, but is opened using a button consisting of a down arrow on a dark background .

Pie Menus

A pie menu is a menu whose items are spread radially around the mouse.



The 3D Viewport Mode Pie menu.

Tip

The fastest way to operate a Pie menu is to press down the key(s) that invoke the menu, move the mouse slightly towards a selection, and release the key(s) to activate the selection.

Releasing the key without moving the mouse will keep the menu open so you can click the desired item. If you do move the mouse before releasing, the item closest to the mouse will be activated instantly.

An open disc widget at the center of the pie menu shows the current direction of the pie menu. The selected item is also highlighted. A pie menu will only have a valid direction for item selection if the mouse is touching or extending beyond the disc widget at the center of the menu.

Pie menu items support key accelerators, which are the letters underlined on each menu item. Number keys can also be used.

If there are sub-pies available, it is indicated by a plus icon.

See also

Pie menu settings.

Eyedropper

The eyedropper (pipette icon) allows you to sample from anywhere in the Blender window. The eyedropper can be used to select different kinds of data:

Color
: This is the most common usage. The eyedropper is used to sample a pixel's color from anywhere within Blender.

  Note

  The View Transform of the color management affects the color. In order to get consistent results, it should be set to *Standard*. If it's set to any other option, the eyedropper may return an inaccurate color.

Color Ramp
: Dragging the cursor over the window to sample a line which is converted into a color ramp.

Objects/Object-Data
: This is used with object buttons (such as parent, constraints or modifiers) to select an object from the 3D Viewport or Outliner, rather than having to select it from a drop-down.

Bones
: This is used when a subtarget to an armature can be chosen. It is possible to choose a bone from the 3D Viewport or from the outliner. Only bones that belong to the armature that was chosen as a target can be picked.

  Note

In the 3D Viewport, bones can only be picked if the armature is in Pose Mode or in Edit Mode.

Camera Depth

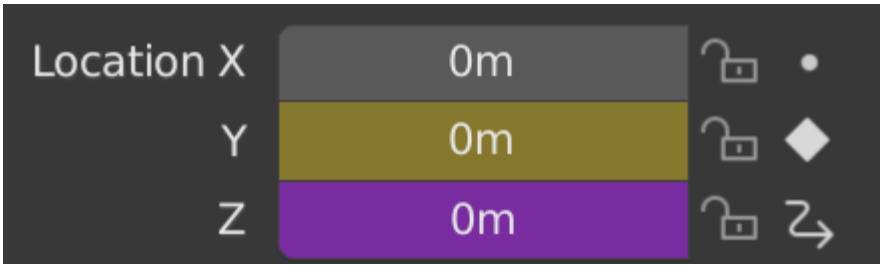
Number fields effecting distance can also use the eyedropper.

This is used to set the camera's depth of field so the depth chosen is in focus.

- `E` will activate the eyedropper while hovering over a button.
- `LMB` dragging will mix the colors you drag over, which can help when sampling noisy imagery.
- `Spacebar` resets and starts mixing the colors again.

Decorators

Decorators are small buttons that appear to the right of other buttons and show the state of the property. Decorators may appear next to number fields, menus, and checkboxes to indicate the property can be animated.



Decorators indicating different property states.

Clicking on the decorator dot icon will add a Keyframe to that property. Clicking the rhombus icon again will remove the keyframe. A solid rhombus icon indicates there is a keyframe on the current frame, while a non-solid rhombus icon indicates that the property has a keyframe on another frame. Clicking the non-solid rhombus icon will create a keyframe on the current frame with the current property value.

If a property is being driven by another, the decorator shows the driver icon.

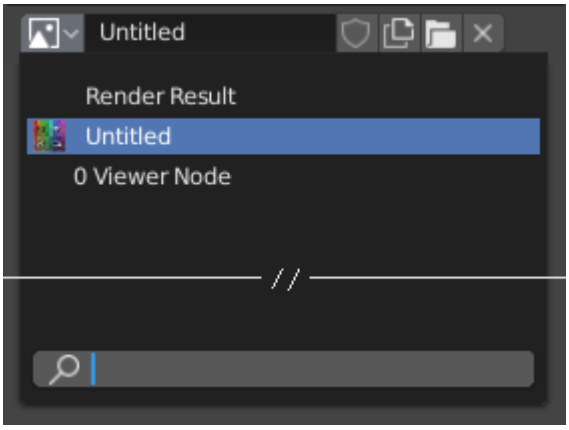
Decorators make it quick and easy to glance over properties and see their state.

See also

State Colors

Data-Block Menu

Lets you select a Data-Block (such as a material) in order to link it to something else (such as an object).



A data-block menu with a search field.

Type
: Shows an icon indicating the data-block type. Clicking the image or the down arrow opens the popup menu. Dragging the image lets you apply the data-block to something else. (For example, you can drag a material onto an object in the 3D Viewport to assign it. Dragging onto Data ID fields is also possible.)

List
: A list of data-blocks available in the current blend-file, or a link to select an item from. The menu may show a preview besides the items and a search field to search the items in the list by name.

  Note

  Data-blocks with names that begin with `.` are hidden from the list, unless a string that also starts with `.` is entered into the search field, or the Show Hidden Files/Data-Blocks user preference is enabled.

Name
: Displays, and allows editing of, the name of the selected data-block.

User Count

Displays the number of users of the data (if there's more than one user). Clicking it will create a single-user copy.

As an example, if three separate objects referenced the same material, the material's user count would be 3. Changing the material would affect all three objects. If you now selected an object and clicked the user count, the object would receive its very own copy of the material, which can be modified independently of the original that's still used by the other two.

Fake User (shield icon)

If a data-block has no real users, it'll normally be cleaned up (deleted) when saving the blend-file. To prevent this, you can give it a fake user; that way, it's guaranteed to "survive." Data-blocks with a fake user have an "F" prefix in the drop-down list.

The Outliner can show an overview of all data-blocks without real users in the blend-file. Simply change its Display Mode to Orphan Data.

New/Add (files icon)

Creates a new data-block (or duplicates the current one) and selects it.

Open File (folder icon)

Opens the File Browser, for importing an image for example.

Unpack File (bin icon)

Unpack the file packed into the current blend-file to an external one.

(Unlink Data-block)

Clears the link. `Shift-LMB` to set the users to zero allowing the data to be fully deleted from the blend-file.

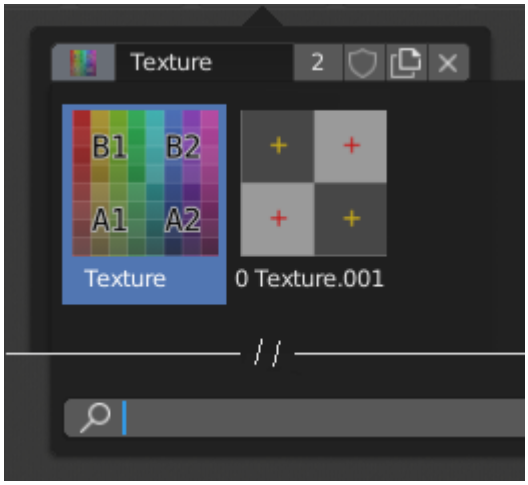
Sometimes there is a list of applied data-blocks (such as a list of materials used on the object).

See also

Data-blocks are discussed further in the Data System chapter.

Preview

Some data-block menus have large preview images in their drop-down instead of just icons and names.

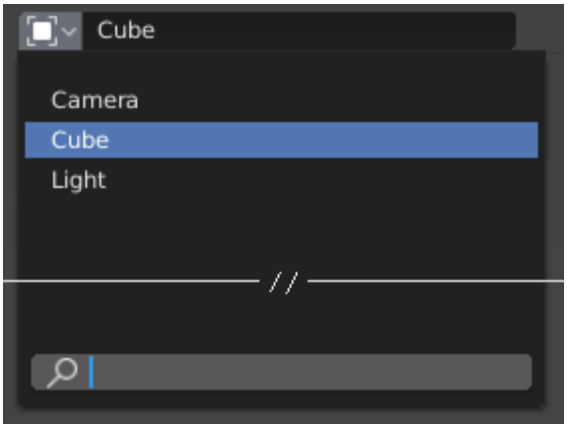


A Data-Block menu with preview.

Data ID

A Data ID field is similar to a Data Block Menu, but is only for selecting (and not for other features like creating new data or managing users).

It can show the following elements:



A Data ID field.

Type
: The icon on the left specifies the accepted data-block type.

Name
: The text field functions as a search field by matching elements in the list. Press `Tab` to auto-complete names up to the level where a

match is found. If more than one match exists, you have to continue typing. If you type an invalid name, the value will remain unchanged.

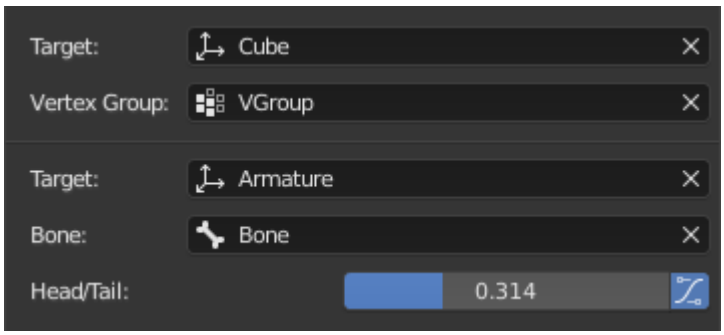
List
: Lets you select the data-block directly.

Eyedropper
: In some Data IDs there is an Eyedropper available through the pipette icon on the right side.

(Clear Button String)
: Click the button on the right to clear the reference.

ID Sub-Data

Related types of ID sub-data may become available to select, depending on the data-block type and its intended usage.



Sub-data Example.

Vertex Group
: If the selected Object in the *Target* field is a mesh or a lattice, an additional field may be displayed to select one of its vertex groups.

Bone
: If the selected Object in the *Target* field is an armature, an additional field may be displayed to select one of its bones.

Head/Tail

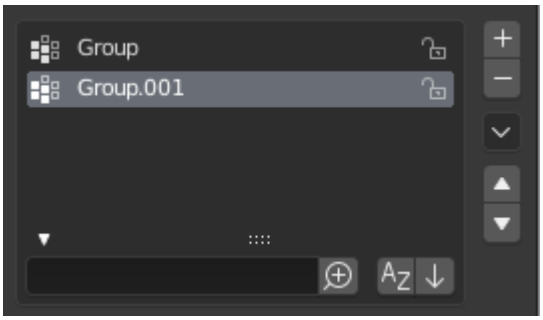
Once a bone is selected, a numeric field may become available for specifying a point along the bone. A value of **0.0** corresponds to the bone's head, while a value of **1.0** corresponds to its tail. Any values between these will result in linear interpolation (so a value of **0.5** matches the bone's center).

Use B-Bone Shape

If the bone is a bendy bone, clicking on this button will make the point follow the curvature of the B-spline between head and tail, rather than simply going in a straight line.

List View

This control is useful for managing lists of items. In addition to the main list, there is a Filtering panel on the bottom (hidden by default) and modification buttons on the right.



List view with expanded Filtering Options panel.

Select
: To select an item, click `LMB` on it.

Rename
: By double-clicking on an item, you can edit its name via a text field.
This can also be achieved by clicking it with `Ctrl-LMB`.

Resize
: The list view can be resized to show more or fewer items. Hover the mouse over the handle (::::), then click and drag to expand or shrink the list.

Filter
: Click the *Show filtering options* button (triangle on bottom left) to show or hide the filter option panel.

  Search `Ctrl-F`
  : Filters the list to only show items containing a certain term.

  Invert `<->`
  : Toggle between including items that match the search term and those that do not contain the search term.

  Sort by Name
  : This button switches between alphabetical and non-alphabetical ordering.

Reverse
: Sort objects in ascending or descending order. This also applies to alphabetical sorting, if selected.

On the right of the list view are list modification buttons:

Add +
: Adds a new item.

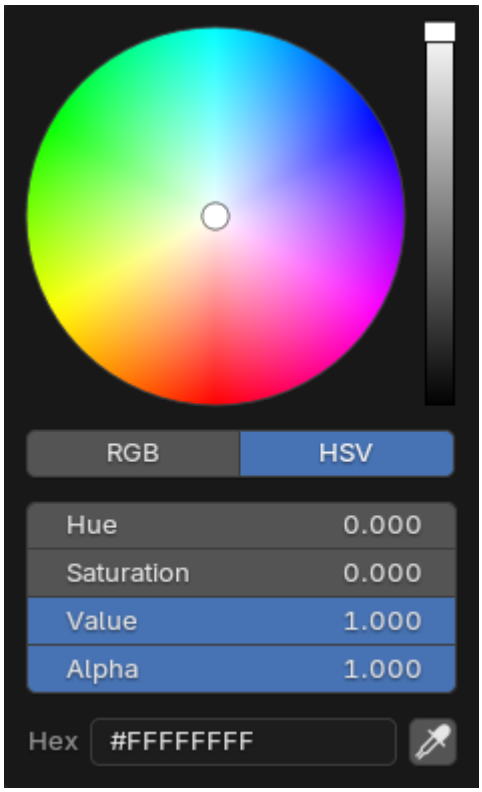
Remove −
: Removes the selected item.

Specials ∨
: A Specials menu with operators to edit list entries.

Move (up/down arrow icon)
: Moves the selected item up/down one position.

Color Picker

The color picker is a pop-up that lets you define a color value. Holding `Ctrl` while dragging snaps the hue to make it quick to select primary colors.



Circle HSV.

Color Picker
: Lets you pick the first and second color component. The shape can be changed; see Types.

Value/Lightness
: The slider with a gradient in the background defines the value/lightness of the color mixing. Fine control can be inputted with `Wheel`.

Color Model
: Selects the Color Model for the number value fields.

RGB::
: Create the final color by mixing red, green, and blue colors.

HSV/HSL::
: Create the final color by adjusting hue, saturation, and value/lightness.

Note

In Blender, the RGB and HSV/HSL values are in Scene Linear color space, and are therefore not Gamma corrected. On the contrary, *Hex* are automatically Gamma corrected for the sRGB Color Space. For more information, see Color Management.

Color Values
: Blender uses values from 0 to 1.0 to express the color mixing for RGB and HSV/HSL colors.

  For color inputs with an Alpha Channel, another slider is added.

Hex
: The hexadecimal (hex) equivalent value to the mixed color. Shorthand hex colors are can be typed in, e.g. dark yellow `FFCC00` can be written as `FC0`.

Eyedropper (pipette icon)
: Samples a color from inside the Blender window using the Eyedropper. Note, colors sampled from the eyedropper are in linear color space and do not account for view transform adjustments.

Shortcuts

- `Ctrl-LMB` (drag) snaps the hue to 30° intervals.
- `Shift-LMB` (drag) precision motion.
- `Wheel` adjust the value/lightness.
- `Backspace` reset the value to the default value.

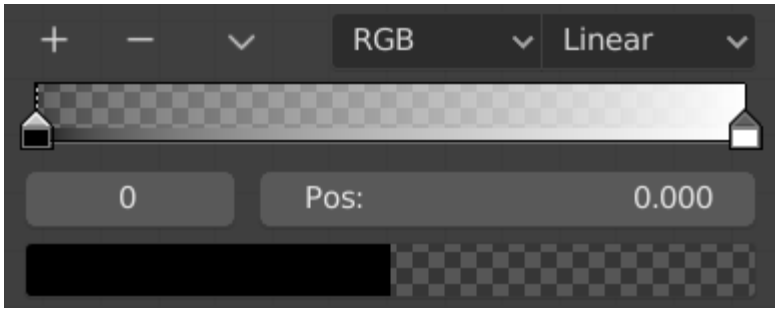
Types

The default color picker type can be selected in the Preferences, see: Interface.

Color Picker types.

Color Ramp Widget

Color Ramps specify a color gradient based on color stops. Each stop has a position and a color. The gradient is then calculated as the interpolation between these stops using the chosen interpolation method.



Color ramp.

Controls

Add +
: Adds a new stop between the selected stop and the one before it.

Delete –
: Deletes the selected color stop.

Specials v
: Contains more operators for the color ramp.

  Flip Color Ramp
  : Flips the gradient, mirroring the positions of the stops.

  Distribute Stops from Left
  : Distribute the stops so that every step has the same space to the right. This is mostly useful when used with Constant interpolation mode.

Distribute Stops Evenly
: Distribute the stops so that all neighbors have the same space between them.

Eyedropper (pipette icon) `E`
: An Eyedropper to sample a color or gradient from the interface to be used in the color ramp.

Reset Color Ramp
: Resets the color ramp to its default state.

Color Mode
: Selection of the Color Model used for interpolation.

RGB:: Blends color by mixing each color channel and combining.

HSV/HSL:: Blends colors by first converting to HSV or HSL, mixing, then combining again. This has the advantage of maintaining saturation between different hues, where RGB would de-saturate. This allows for a richer gradient.

Color Interpolation
: The interpolation method to use across the ramp.

RGB

B-Spline:: Uses a B-spline interpolation for the color stops.

Cardinal:: Uses a cardinal interpolation for the color stops.

Linear:: Uses a linear interpolation for the color stops.

Ease:: Uses an ease interpolation for the color stops.

Constant:: Uses a constant interpolation for the color stops.

HSV/HSL

Clockwise:: Clockwise interpolation around the HSV/HSL wheel.

Counter-Clockwise::
: Counterclockwise around the HSV/HSL wheel.

Near::
: Nearest route around the wheel.

Far::
: Furthest route around the wheel.



HSV and HSL interpolation options.

Active Color Stop
: Index of the active color stop (shown as a dashed line). Offers an alternative way of selecting a stop in case it's so close to others that it's hard to select it directly.

Position
: This slider controls the position of the selected color stop in the range.

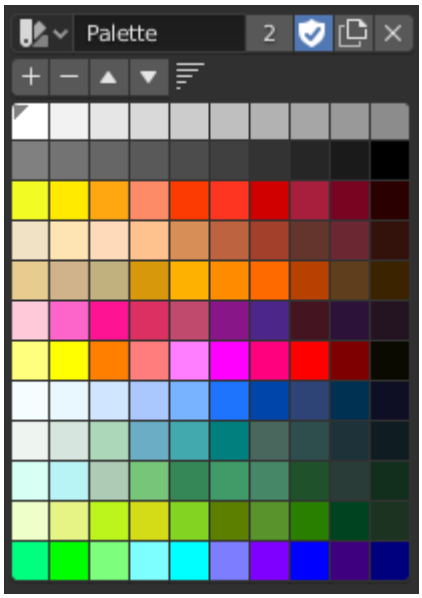
Color
: A color field where you can specify the color and alpha of the selected stop.

Shortcuts

- `LMB` (drag) moves color stops.
- `Ctrl-LMB` (click) adds a new color stop.

Color Palette

Color Palettes are a way of storing a brush's color so that it can be used at a later time. This is useful when working with several colors at once.



Color Palette.

Palette
: A Data-Block Menu to select a palette.

New +
: Adds the current brush's primary *Color* to the palette.

Delete –
: Removes the currently selected color from the palette.

Move (up/down arrow icon)
: Moves the selected color up/down one position.

Sort
: Sort Colors by Hue, Saturation, Value, Luminance.

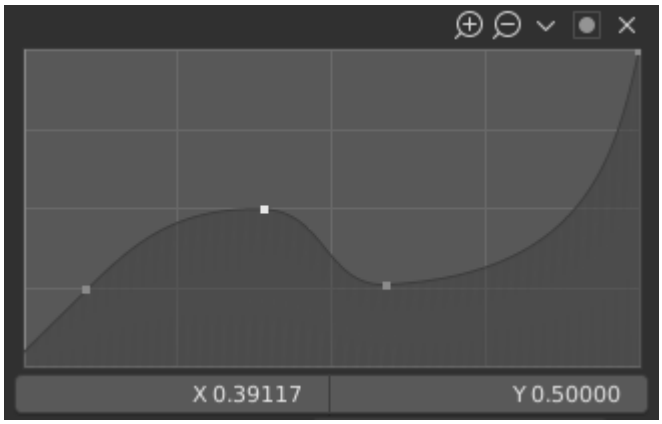
Color List
: Each color that belongs to the palette is presented in a list. Clicking on a color will change the brush's primary *Color* to that color.

Shortcuts

- `Ctrl-LMB` open the color picker to change color. See Color Picker.
- `Backspace` reset the value.

Curve Widget

The *Curve Widget* allows to intuitively map a range of input values to a set of output values by adjusting a curve, where the X axis represents the input and the Y axis the output.



Curve widget.

Control Points

Like all curves in Blender, the curve of the *Curve Widget* is controlled using *control points*.

By default, there are two control points: one at (0.0, 0.0) and one at (1.0, 1.0), meaning the input is mapped directly to the output (unchanged).

Move
: Simply click and drag it around.

Add
: Click anywhere on the curve where there is not already a control point.

Remove
: Select it and click the button at the bottom right.

Controls

Above the curve graph is a row of controls. These are:

Zoom In (plus magnifying glass icon)
: Zoom into the center of the graph to show more details and provide more accurate control. To navigate around the curve while zoomed in, click and drag in an empty part of the graph.

Zoom Out (minus magnifying glass icon)
: Zoom out of the graph to show fewer details and view the graph as a whole. You cannot zoom out further than the clipping region (see *Clipping* below).

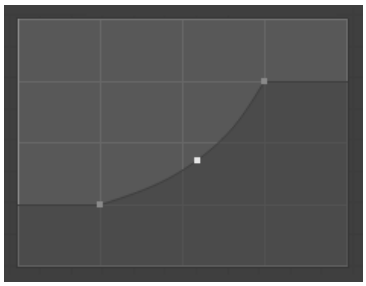
Specials v
: A Specials menu with operators to edit control points or to set properties.

  Reset View
  : Resets the view of the curve.

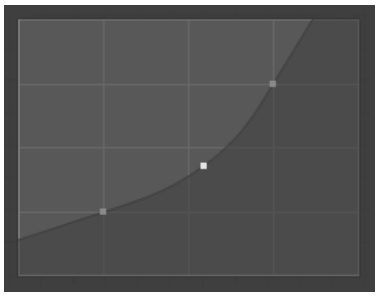
  Extend Options
  : Controls how the curve is extended before the first control point and after the last control point.

    Extend Horizontal
    : Causes the curve to stay horizontal before the first point and after the last point.

    Extend Extrapolated
    : Causes the curve to extrapolate before the first point and after the last point, based on the shape of the curve.



Extend Horizontal.



Extend Extrapolated.

Reset Curve

Resets the curve to default (removes all points added to the curve).

Clipping Options (dot icon)

Use Clipping

Forces curve points to stay between specified values.

Min X/Y and Max X/Y

Set the minimum and maximum bounds of the curve points.

Below the curve are options for the selected control point:

Handle Type

Controls how the control points affect the curve shape. It determines the interpolation of the curve segment at the selected control point.

Vector Handle:: Vector handles create straight lines and sharp corners.

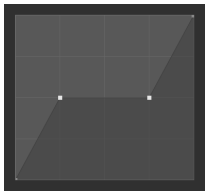
Auto Handle:: Automatic handles that create smooth curves.

Auto-Clamped Handle:: Automatic handles that create smooth curves while also preventing overshoot.

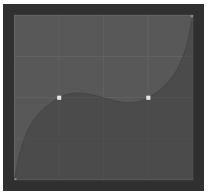
Free Handle:: The handles can be moved completely independently, and thus can result in a sharp change of direction.

Aligned Free Handles::

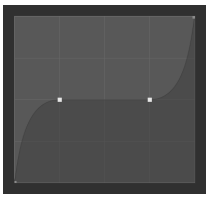
The two handles of the curve point are locked together to always point in exactly opposite directions. This results in a curve that is always smooth at the control point.



Vector Handles.



Auto Handles.



Auto Clamped Handles.

X, Y
: The coordinates of the selected control point.

Delete
: Remove the selected control point. The first and last points cannot be deleted.

Copy/Paste `Ctrl-C`, `Ctrl-V`
: The whole curve can be copied from one Curve Widget to another by hovering over the curve graph and pressing `Ctrl-C`, `Ctrl-V`.

Tools & Operators

- Tool System
- Operators
- Undo & Redo
- Annotations
- Selecting

Tool System

Tools are accessed from the Toolbar.

This is a general introduction to tools. Individual tools have their own documentation.

There can only be one active tool per Workspace and mode. This tool is remembered: if you're in Edit Mode and have the Extrude tool selected, then switch to Object Mode (which has no Extrude tool) and back to Edit Mode, the Extrude tool will still be active.

Most tools are controlled using just LMB, though some also have modifier keys (shown in the Status Bar while using the tool). This can all be customized in the Keymap Preferences.

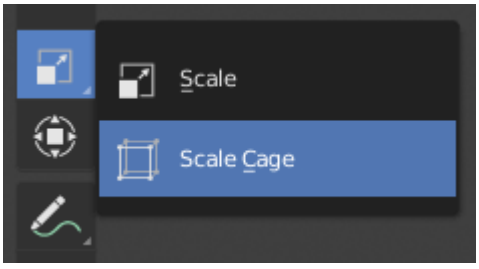
Some tools define gizmos (*Shear* and *Spin* for example) to help control them.

Toolbar

Reference

Shortcut:: T

The Toolbar contains buttons for the various tools. Buttons with a small triangle in their bottom right corner are tool groups which can be opened by holding LMB on them for a moment (or dragging LMB to open them instantly).



Expanded tool group.

Hovering your cursor over a tool for a short time will show its name, while hovering

longer will show the full tooltip.

Resizing the Toolbar horizontally will display the icons with two columns. Expanding it further will display the icon and its text.

Pop-Up Toolbar

Reference

Shortcut:: `Shift-Spacebar`

Pressing `Shift-Spacebar` will pop up a small toolbar right at your cursor for faster access. The shortcuts for selecting the tools are displayed on the right.

Alternatively, you can map this action to `Spacebar` in the Keymap Preferences. Then you'll be able use `Spacebar` like a modifier key (similar to holding `Ctrl` or `Shift`). For example, you can press `Spacebar T` for Transform, `Spacebar D` for Annotate, `Spacebar M` for Measure and so on. See Spacebar Action.

Quick Favorites

Reference

Shortcut:: `Q`

The Quick Favorites menu gathers your favorite tools. Any tool or menu item can be added to this pop-up menu via its context menu.

Changing Tools

If you have *Alt Click Tool Prompt* enabled in the Keymap Preferences, tapping `Alt` will display a tool prompt in the Status Bar. You can then press

a key to select the corresponding tool, or tap `Alt` again to cancel the prompt.

Fallback Tool

The fallback tool is the one that's selected by default (so the one at the top of the Toolbar). You can change it by either holding `LMB` on the toolbar button or pressing `Alt-W` to get a pie menu.

Cycling Tools

If you bind a key to a tool which is part of a group, you can enable the *Cycle* option in the keymap editor. Successive presses will then cycle through the tools in that group.

This is enabled by default for the selection tools in the 3D Viewport, for example: pressing `W` will cycle between Select Box, Select Circle and so on.

Properties

Tools can have their own settings, which are available from multiple places:

- The *Tool* ▸ *Active Tool* panel in the Sidebar `N`.
- The *Active Tool* tab in the Properties editor.
- The *Tool Settings* region below the area header.

Operators

Operators execute an action the moment they're activated, which makes them different from tools (which require some sort of input). Operators can be started from Operator Buttons, Popup Menus, or Menu Search. Examples of operators include adding a new object, deleting it, or setting its shading to smooth.

Operator Properties

Most operators have properties that can be adjusted to refine their result. First run the operator (which will use its default settings), then adjust the properties in the Adjust Last Operation region.

Modal Operators

Modal operators exist as a concept in between Tools and regular operators. They require some sort of interactive input.

The action of a modal operator can be confirmed using `LMB` or `Return`. To cancel a modal operator use `RMB` or `Esc`.

Slider Operators

Slider operators are used to interactively adjust a percentage value in the editor's Header.

You can adjust the percentage by dragging the slider left or right. This can be made coarser (snapping in 10% increments) by holding `Ctrl` and more precise by holding `Shift`. For some sliders, you can toggle "overshoot" with `E`, which lets you go beyond the 0-100% range.

Searching for Operators

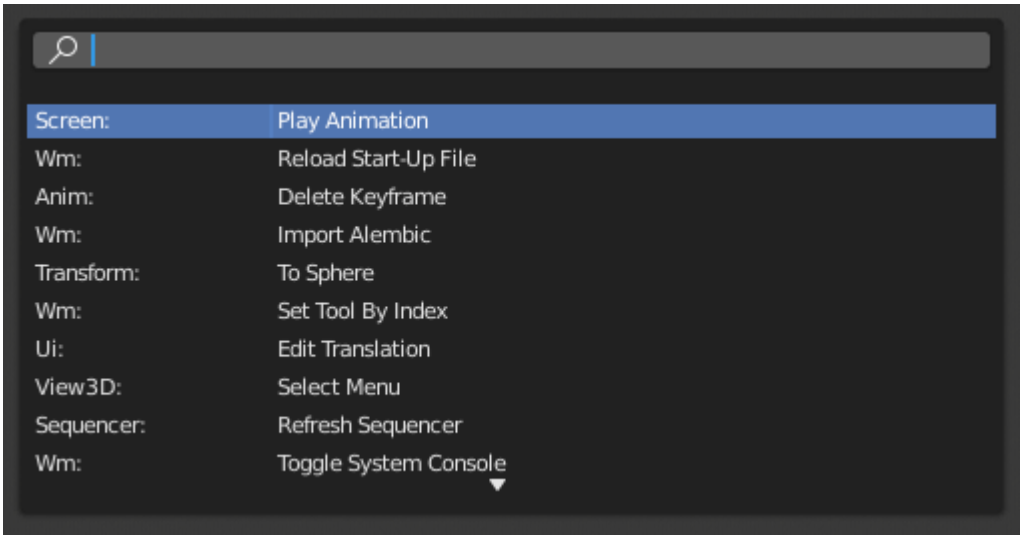
Menu Search

Reference

Mode:: All Modes
Menu:: *Edit ▸ Menu Search*
Shortcut:: `F3`

The Menu Search pop-up lets you search Blender's interface for a certain operator and execute it. First narrow down the list by typing (part of) the operator's name, then either click the operator with `LMB`, or navigate to it with `Down` and `Up` and activate it with `Return`.

Apart from the operator names, the pop-up also shows the menus where they're located.



The Menu Search pop-up.

See also

The Spacebar Action option in the Preferences.

Operator Search

Reference

Mode:: All Modes
Menu:: *Edit ▸ Operator Search*

When Developer Extras are activated, the Operator Search can be accessed from the Edit menu in the Topbar. This menu searches all Operators within Blender, even if they are not exposed in a menu. This is useful for Python developers for testing purposes. Blender might also include a few advanced operators that are not exposed in a menu and can only be accessed via this search menu.

See also

The User Preferences has an option to change how the search results are scored.

Undo & Redo

The tools listed below will let you roll back an accidental action, redo your last action, or let you choose to recover to a specific point, by picking from a list of recent actions recorded by Blender.

Undo

Reference

Mode::	All Modes
Menu::	*Edit* ‣ *Undo*
Shortcut::	`Ctrl-Z`

If you want to undo your last action, just press `Ctrl-Z`.

See also

Memory & Limits Preferences to change undo settings.

Redo

Reference

Mode::	All Modes
Menu::	*Edit* ‣ *Redo*
Shortcut::	`Shift-Ctrl-Z`

To roll back the Undo action, press `Shift-Ctrl-Z`.

Adjust Last Operation

Reference

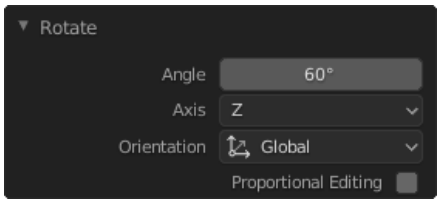
Mode::	All Modes
Menu::	*Edit ▸ Adjust Last Operation…*
Shortcut::	`F9`

You can tweak the parameters of an operator after running it. In editors that support it, there is a "head-up display" panel in the bottom left based on the last performed operation. Alternatively, you can create a pop-up with `F9` which does the same thing.

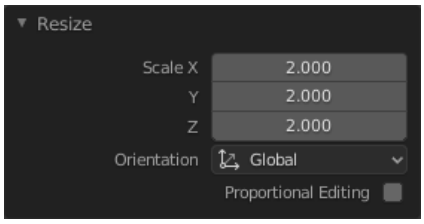
For example, if your last operation was a rotation in *Object Mode*, Blender will show you the last value changed for the angle (see Fig. Rotation (Object Mode, 60 degrees). left), where you can change your action back completely by typing `Numpad0` in the *Angle* Field. There are other useful options, based on the operator, and you cannot only Undo actions, but change them completely using the available options.

If you are in *Edit Mode*, Blender will also change its contents based on your last action taken. In the second example (on the right), the last operation was a Move in Object Mode; but a *Scale* on a Face in Edit Mode, and, as you can see, the contents of *Adjust Last Operation* are different, because of the mode (Edit Mode) (See Fig. Scale (Edit Mode, Resize face). right).

Adjust Last Operation.



Rotation (Object Mode, 60 degrees).



Scale (Edit Mode, Resize face).

Tip

Some operations produce particularly useful results by using *Adjust Last Operation*. For example, adding a Circle in the 3D Viewport; if you reduce the *Vertices* to three, you get a perfect equilateral triangle.

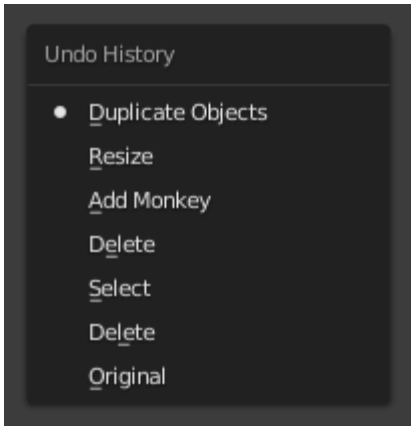
Tip

The *Adjust Last Operation* region can be hidden by *View ▸ Adjust Last Operation*.

Undo History

Reference

Mode:: All Modes
Menu:: *Edit ▸ Undo History*



The Undo History menu.

There is also an Undo History of the last actions taken, recorded by Blender.

The top of the list corresponds to the most recent actions. A small icon of a dot next to one of the entries indicates the current status. Rolling back actions using the *Undo History* feature will take you back to the action you choose. Much like how you can alternate between going backward in time with *Undo* and then forward with *Redo*, you can hop around on the Undo timeline as much as you want as long as you do not make a new change. Once you do make a new change, the Undo History is truncated at that point. Selecting one of the entries in the list takes the current status to that position.

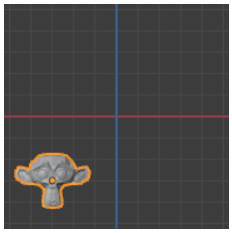
Repeat Last

Reference

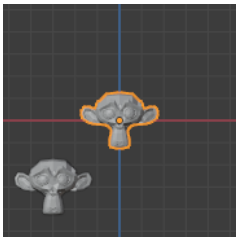
Mode::	All Modes
Panel::	*Edit ▸ Repeat Last*
Shortcut::	`Shift-R`

The Repeat Last feature will repeat your last action when you press `Shift-R`.

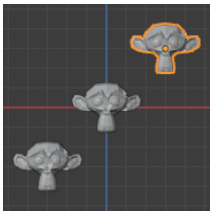
In the example images below, we duplicated a *Monkey* mesh and moved it a bit. Using repeat `Shift-R`, the *Monkey* was duplicated and moved a second time.



Suzanne.



After a `Shift-D` and move.



After a `Shift-R`.

Repeat History

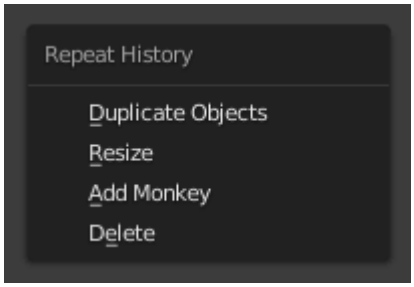
Reference

Mode::	All Modes
Menu::	*Edit ▸ Repeat History…*

The *Repeat History* feature will present you a list of the last repeated actions, and you can choose the actions you want to repeat. It works in the same way as the Undo History, explained above, but the list contains only repeated actions.

Important

When you quit Blender, the complete list of user actions will be lost, even if you save your file before quitting.



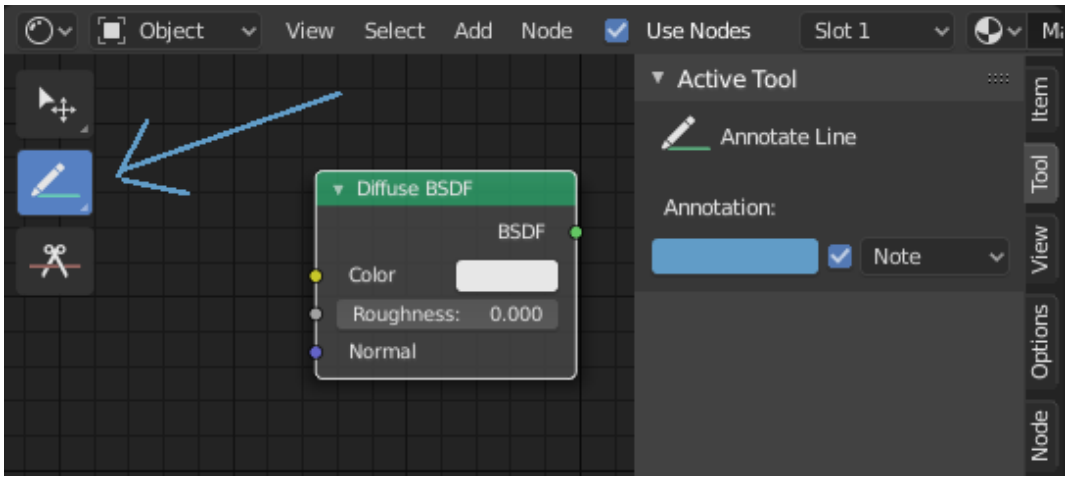
The Repeat History menu.

See also

Troubleshooting section on Recovering your lost work.

Annotations

The annotation tool is available in multiple editors. It can be used to add notes to e.g. 3D objects or node setups. The arrow in the screenshot below is an annotation.



Annotations tool in a node editor.

Annotation Tools

The annotation tool can be activated in the Toolbar and has the following sub-tools:

Annotate
: Draw free-hand strokes in the main area.

Annotate Line
: Click and drag to create a line. Optionally, you can select the arrow style for the start and end of the line.

Annotate Polygon
: Click multiple times to create multiple connected lines, then press `Return` or `Esc` to confirm.

Annotate Eraser
: Click and drag to remove lines. The eraser has a *Radius* setting found in *Tool Settings*.

Tool Settings

Common

Color
: Adjust the color of existing and new strokes.

Annotation Layer
: A pop-over menu, showing the name of the current layer, to access the Annotation Layers.

Placement
: Determines where the annotations are drawn.

3D Cursor:: Only available in the 3D Viewport. The new annotations become part of the 3D scene; they're drawn on an imaginary plane that goes through the 3D Cursor and is aligned to your view.

Surface:: Only available in the 3D Viewport. The new annotations become part of the 3D scene; they're drawn onto the surface of the object under the mouse. If there is no surface, you get the same behavior as *3D Cursor*.

Image:: Only available in 2D editors such as the Image Editor. The annotations become part of the 2D space, meaning their position and size change as you pan and zoom in the editor.

View:: The new annotations are 2D and get stuck to the screen. They keep the same position, rotation and size no matter how you pan, orbit or zoom in the editor.

Only End Points *Surface Placement*

Only use the first and last parts of the stroke for snapping.

Project Onto Selected *Surface Placement*
: Only project the strokes onto selected objects.

Stabilize Stroke
: Helps to reduce jitter of the strokes while drawing by delaying and correcting the location of points.

  Radius
  : Minimum distance from the last point before the stroke continues.

  Factor
  : A smooth factor, where higher values result in smoother strokes but the drawing sensation feels like as if you were pulling the stroke.

Annotate Line

Style Start, End
: The decoration to use at the beginning or end of the line segment. This can be used for example to create arrows to point out specific details in a scene.

Annotation Layers

When the annotation tool is enabled, the settings for managing multiple layers can be found in the *Sidebar* ▸ *View* ▸ *Annotations* panel.

Opacity
: Adjusts the opacity of existing and new strokes.

Thickness
: Adjusts the thickness of existing and new strokes.

Onion Skin

Shows a ghosted image of strokes made in frames before and after the current frame. Onion skinning only works in the 3D Viewport and Sequencer. See the Grease Pencil documentation for an explanation of Onion Skinning.

Before/After
: Color to use before and after the current frame on ghost frames. The number defines how many frames to show before and after the current frame.

Selecting

By default, Blender uses `LMB` to select items. This can be changed to `RMB` in the Preferences.

Blender has several selection tools that can be used across the different editors.

Note

Some editors deviate from the keyboard shortcuts shown below. For example, most editors use `Shift-LMB` to add a single item to the selection, but the Outliner uses `Ctrl-LMB`. Similarly, most editors use `Ctrl-RMB` for performing a *Lasso Select*, but node editors use `Ctrl-Alt-LMB`.

Most selection tools come in two variants, where one variant is available in the Toolbar and the other in the *Select* menu. While the variants' names are almost identical (such as *Select Box* in the Toolbar versus *Box Select* in the menu), the way they work is a bit different. New users coming from other applications will find the Toolbar variants to be the most familiar.

Toolbar Selection Tools

All the Toolbar selection tools behave the same when clicking an item: they select it (and deselect any previously selected items). If you hold `Shift` while clicking, the item will be added to the selection (if it's not selected) or removed from the selection (if it is selected).

What makes the tools different is what happens when you drag.

Tweak

Reference

Tool:: *Toolbar ▸ Tweak*
Shortcut:: `W`

Dragging an item will move it around.

Select Box

Reference

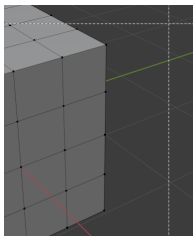
Tool:: *Toolbar ▸ Select Box*
Shortcut:: `W`

Dragging will create a rectangle, and select all the items that are partially or completely inside it once you release. (Any other items will be deselected.)

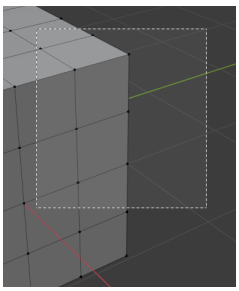
Holding `Shift` while dragging will add the items to the selection. Holding `Ctrl` will remove them.

While dragging, you can additionally hold `Spacebar` to move the rectangle around with the mouse.

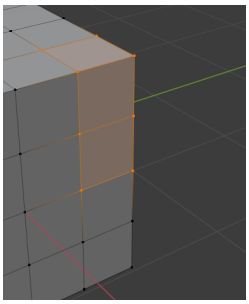
Select Box example (Edit Mode).



Start.



Selecting vertices.



Complete.

Select Circle

Reference

Tool:: *Toolbar ▸ Select Circle*
Shortcut:: `W`

Dragging will select all the items which the circle passed over. Items which you didn't pass over will be deselected.

Holding `Shift` while dragging will add the items to the selection. Holding `Ctrl` will remove them.

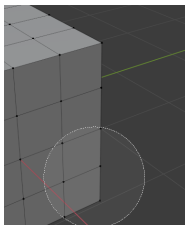
You can change the radius of the circle in the tool settings (which can be found in the area header, the Tool tab of the Sidebar `N`, or the Active Tool tab of the Properties editor).

Note

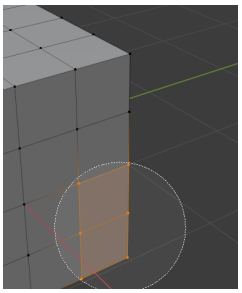
In Object Mode: unlike *Select Box*, which selects objects as soon as the box covers any part of their geometry, *Select Circle* only selects objects if the circle passes over their origin point. The origin is shown as an orange dot for selected objects but is invisible for unselected ones, unless "Origins (All)" is enabled in the Viewport Overlays.

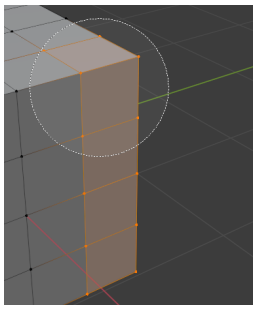
This difference in behavior does not apply to the other modes (like Edit Mode and Pose Mode).

Select Circle example (Edit Mode).



Start.





Complete.

Selecting vertices.

Select Lasso

Reference

Tool:: *Toolbar ▸ Select Lasso*
Shortcut:: `W`

Dragging will create a freeform shape, and select all the items inside it once you release. (Any other items will be deselected.)

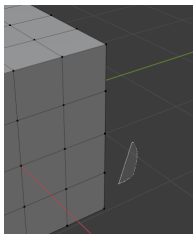
Holding `Shift` while dragging will add the items to the selection. Holding `Ctrl` will remove them.

While dragging, you can additionally hold `Spacebar` to move the shape around with the mouse.

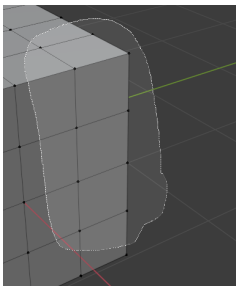
Note

Select Lasso behaves the same as *Select Circle* in that it only looks at origin points in Object Mode.

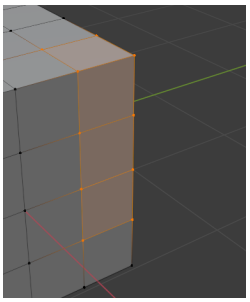
Select Lasso example (Edit Mode).



Start.



Selecting vertices.



Complete.

Selection Modes

Reference

Tool::	Select Tools
Panel::	*Tool Settings* ▸ *Mode*

Each of the Toolbar selection tools has a mode to configure how it interacts with existing selections. Note that not every tool supports all of these modes.

Set
: Sets a new selection (the previous selection is discarded). This is the default.

Extend
: Adds newly selected items to the existing selection.

Subtract
: Removes newly selected items from the existing selection.

Invert `Ctrl-I`
: Inverts the selection (unselected items become selected and vice versa).

Intersect
: Selects items that intersect with the existing selection.

Menu Selection Tools

These tools are variants of the previously described ones. They're available in the menu rather than the Toolbar and work slightly differently.

Box Select

Reference

Menu:: *Select ▸ Box Select*
Shortcut:: `B`

To use this tool, you first activate the menu item or keyboard shortcut and then drag a box as usual. Unlike *Select Box*, the default behavior here is to add the items inside the box to the selection. (The ones outside the box are not deselected.)

To remove the items inside the box from the selection, hold `Shift`, or drag with `MMB` instead.

While dragging, you can additionally hold `Spacebar` to move the box around with the mouse.

Circle Select

Reference

Menu:: *Select ▸ Circle Select*
Shortcut:: `C`

To use this tool, you first activate the menu item or keyboard shortcut and then drag a circle around as usual. Unlike *Select Circle*, the default behavior here is to add the items inside the circle to the selection. (The ones outside the circle are not deselected.)

To remove the items inside the circle from the selection, hold `Shift`, or drag with `MMB` instead.

You can change the radius of the circle by scrolling with the `Wheel` or using the `NumpadPlus` and `NumpadMinus` keys.

Once activated, *Circle Select* stays active: you can release the mouse button and start dragging somewhere else without having to press `C` again. At the same time, however, it blocks all other parts of Blender while it's active. To deactivate the tool again, press `RMB`, `Return`, or `Esc`.

Lasso Select

Reference

Menu::	*Select ▸ Lasso Select*
Shortcut::	`Ctrl–RMB`

To use this tool, you first activate the menu item and drag a freeform shape around the item(s) you want to select with `LMB`. The menu lets you choose whether to set, extend or reduce the selection.

Alternatively, you can immediately start dragging with `Ctrl–RMB`. Unlike *Select Lasso*, the default behavior then is to add the items inside the lasso to the selection. (The ones outside the lasso are not deselected.)

To remove the items inside the lasso from the selection, drag with `Shift–Ctrl–RMB` instead.

While dragging, you can additionally hold `Spacebar` to move the lasso around with the mouse.

Nodes

- Introduction
  - Editor Interface
  - Navigating
  - Adding Nodes
- Node Parts
  - Title
  - Sockets
  - Properties
- Selecting
- Arranging Nodes
  - Snapping
  - Auto-Offset
- Editing
  - Transform
  - Connecting Sockets
  - Disconnecting Sockets
  - Copy/Paste
  - Duplicate
  - Duplicate Linked
  - Delete
  - Mute
  - Show/Hide
  - Layers
  - Connect to Output
- Sidebar
  - Node
  - Tool
  - View
- Node Groups
  - Properties
  - Make Group
  - Insert Into Group
  - Edit Group

    - Ungroup
    - Reusing Node Groups
- Frame Node
    - Properties
    - Editing
- Reroute Node
    - Properties

Introduction

Blender contains different node-based editors with different purposes, so this section only explains how to work with nodes in general. The list below shows the different types of nodes and where they're documented.



Example of a node editor.

Icon	Name	Description
	Geometry Nodes	Used for procedural modeling.
	Shader Nodes	Used to create materials for objects.
	Composite Nodes	Used to edit rendered images.

Icon	Name	Description
	Texture Nodes	Used to create custom textures.

Editor Interface

Header

The *Header* contains various menus, buttons and options, partially based on the current node tree type.



Common node editor header options.

View
: This menu changes your view of the editor.

Select
: This menu allows you to select a node or groups of nodes.

Add
: This menu allows you to add nodes.

Node
: This menu allows you to do things with selected nodes.

Use Nodes
: Tells the render engine to use the node tree when computing the material color or rendering the final image, or not. If not, the tree is ignored. For materials, this is mostly a legacy option, because in the past materials could not be created with node trees.

Pin

When enabled, the editor will retain the material or texture, even when the user selects a different object. A node tree can then be edited independent of the object selection in the 3D Viewport.

Parent Node Tree
: Leaves the current node group and returns to the parent node group/tree.

Snapping
: Change options for snapping node positions to achieve a cleaner node tree layout. See Arranging Nodes.

Overlays

Overlays are information that is displayed on top of the nodes and node trees. There is a toggle to show or hide all overlays for the node editor next to the overlay popover.

Wire Colors
: Color node links based on their connected sockets.

Reroute Auto Labels
: Label Reroute Nodes based on the label of connected reroute nodes.

Context Path
: Display breadcrumbs in the upper left corner indicating the hierarchy location of the node tree/group that's currently being displayed.

Annotations
: Displays Annotations in the preview region.

Previews
: Display each node's Preview if the node's preview is also toggled.

Timings
: Display each node's last execution time. This option is only available for compositing and geometry nodes.

  In the context of geometry nodes, see Node Timings Overlay.

Toolbar

The *Toolbar* contains a set of tools that can be used in the node editor.

Sidebar

The Sidebar region contains properties for the currently selected node as well as node editor-specific settings.

Navigating

Navigating the node editors is done with the use of both mouse movement and keyboard shortcuts.

Pan `MMB`
: Move the view up, down, left and right.

Zoom `Ctrl-MMB`, `Wheel`
: Move the camera forwards and backwards.

Frame Selected `NumpadPeriod`
: Adjusts the zooms to fit only the selected nodes in the view.

Frame All `Home`
: Adjusts the zoom to fit all nodes in the view.

Adding Nodes

Reference

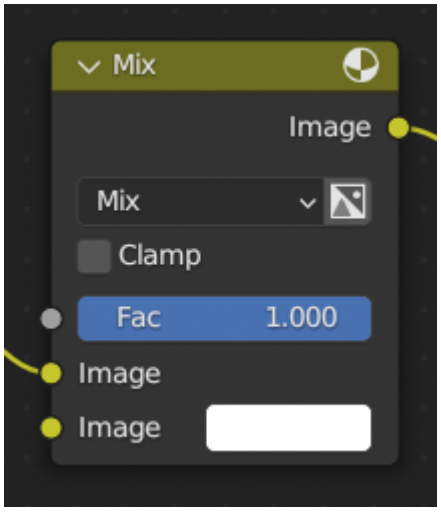
Menu::	*Add*
Shortcut::	`Shift-A`

Nodes are added via the *Add* menu in the editor's header or using a keyboard shortcut.

Nodes can also be added by dragging a connection from an existing node's input or output socket and dropping the connection above an empty space instead of connecting to another socket. This action will open a search menu with a list of compatible nodes and their sockets that can be added and connected to the existing node. Confirming the menu selection will add the node which can then be moved and placed.

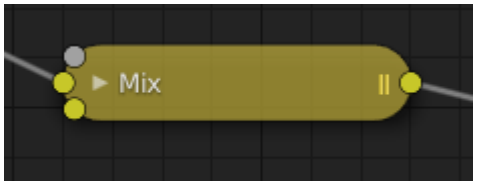
Node Parts

All nodes in Blender are based on a similar construction. This applies to any type of node. These parts include the title, sockets, properties and more.



Title

The title shows the name/type of the node; it can be overridden by changing the node's Label. On the left side of the title is the *collapse* toggle which can be used to collapse the node. This can also be done with `H`.



How a node appears when collapsed.

Preview

Previews are an overlay that shows a small image above the node displaying the node result. Not all nodes support previews, but the ones that

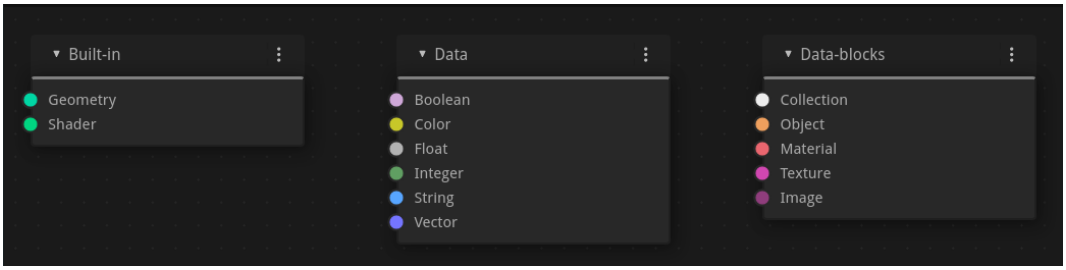
do can be toggled using the / icons in the top right-hand corner of the node next to the title.

Previews can be disabled for the whole node tree by using Previews overlay toggle.

Sockets

Sockets are input and output values for the node. They appear as little colored circles on either side of the node. Unused sockets can be hidden with `Ctrl-H`.

Each socket is color-coded depending on what type of data it handles.



Built-in

Shader (bright green)
: Used for shaders in Cycles and EEVEE.

Geometry (turquoise)
: Used in Geometry Nodes.

Data

Boolean (light pink)
: Used to pass a true or false value.

Color (yellow)
: Indicates that the socket accepts/produces color information. The colors may or may not have an alpha component depending on the node tree

type.

Float (gray)
: Indicates that the socket accepts/produces floating-point numbers. It can either be a single value or a so-called “value map”. (You can think of a value map as a grayscale image where the brightness of a pixel represents its value.) If a single value is used as an input for a “value map” socket, all points of the map are set to this same value.

Integer (lime green)
: Used to pass an integer value (a number without a fractional component).

String (light blue)
: Used to pass a text value.

Vector (dark blue)
: Indicates vector, coordinate and normal information.

Rotation (pink)
: Indicates a rotation/quaternion.

Matrix (dark pink)
: Indicates a 4×4 matrix of float values, it is often used to represent a Transformation Matrix.

Data-Blocks

Collection (white)
: Used to pass a collection data-block.

Object (orange)
: Used to pass an object data-block.

Material (salmon)
: Used to pass a material data-block.

Texture (pink)
: Used to pass a texture data-block.

Image (apricot)
: Used to pass an image data-block.

Inputs

The inputs are located on the bottom left side of the node, and provide the data the node needs to perform its function. Each input socket, except for the green shader input, when disconnected, has a default value which can be edited via a color, numeric, or vector interface input. In the screenshot of the node above, the second color option is set by a color interface input.

Some nodes have special sockets that can accept multiple inputs. These sockets will have an ellipsis shape rather than a circle to indicate their special behavior.

Outputs

The outputs are located on the top right side of the node, and can be connected to the input of nodes further down the node tree.

Conversion

Some socket types can be converted to others either implicitly or explicitly. Implicit conversion happens automatically without the need of a conversion node. For example, Float sockets and Color sockets can be linked to each other.

Once a socket conversion is made, data may be lost and cannot be retrieved later down the node tree. Implicit socket conversion can sometimes change the data units as well. When plugging a *Value* input node into an angle socket, it'll default to use radians regardless of the scene's Units. This happens because the Value node has no unit while the angle input does.

Valid conversions:

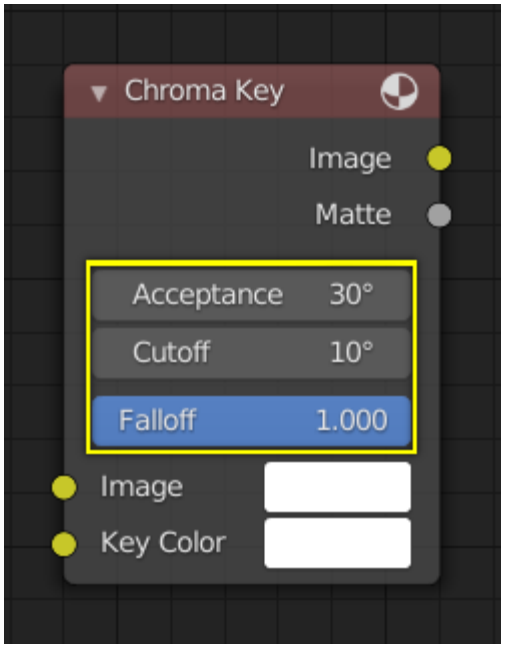
- Between color and vector – mapping between color channels and vector components.

- Between color and float – the color data is converted to its grayscale equivalent.
- Color/float/vector to Shader – implicitly converts to color and gives the result of using an Emission node.
- Between float and integer – integers simply become floats, floats are truncated.
- Between float and vector – when a float becomes a vector the value is used for each component. When a vector becomes a float the average of the components is taken.
- Between float and boolean – values greater than 0 are true, true maps to 1, and false maps to 0.
- Between rotations and matrices.

Explicit conversion requires the use of a conversion node such as the Shader To RGB node or the RGB to BW Node node. The Math Node node also contains some functions to convert between degrees and radians.

Properties

Many nodes have settings which can affect the way they interact with inputs and outputs. Node settings are located below the outputs and above any inputs.



An example of the controls on the Chroma Key node.

Selecting

All `A`
: Selects all nodes.

None `Alt-A`
: Deselects all nodes.

Invert `Ctrl-I`
: Inverts the current selection.

Box Select `B`
: See Box Select.

Circle Select
: See Circle Select.

Lasso Select
: See Lasso Select.

Select Linked From `L`
: Expand the selection to nodes which are linked to the inputs of the currently selected nodes.

Select Linked To `Shift-L`
: Expand the selection to nodes which are linked to the outputs of the currently selected nodes.

Select Grouped `Shift-G`
: Selects nodes that have similar properties as the active node.

  Type
  : The node type, e.g. all Math nodes.

  Color

The node color. (Nodes can be given a custom color to visually organize them in the editor; this is not related to any color information they might consume or produce as part of their function. The color can be set in the Sidebar.)

Prefix, Suffix
: Matches the name property from start/end of the text.

Activate Same Type Previous/Next `Shift-]`/`Shift-[`
: Finds the previous/next node of same type, activates it, and ensures it's visible.

Find Node `Ctrl-F`
: Shows a search pop-up for finding a node by name.

Select Multiple `Shift-LMB`
: Add/remove a node to/from the selection.

Arranging Nodes

Snapping

The snapping options can be found on the rightmost side of the node editor's header.

Snap `Shift-Tab`
: Toggle snapping on or off. You can also do this temporarily by holding `Ctrl` after starting to drag one or more nodes around.

Snap Node Element `Shift-Ctrl-Tab`
: What to snap the selected nodes to:

Grid::	Snap to the grid in the background.
Node X::	Snap to the X coordinate of another node's vertical border.
Node Y::	Snap to the Y coordinate of another node's horizontal border.
Node X/Y::	Combination of the above.

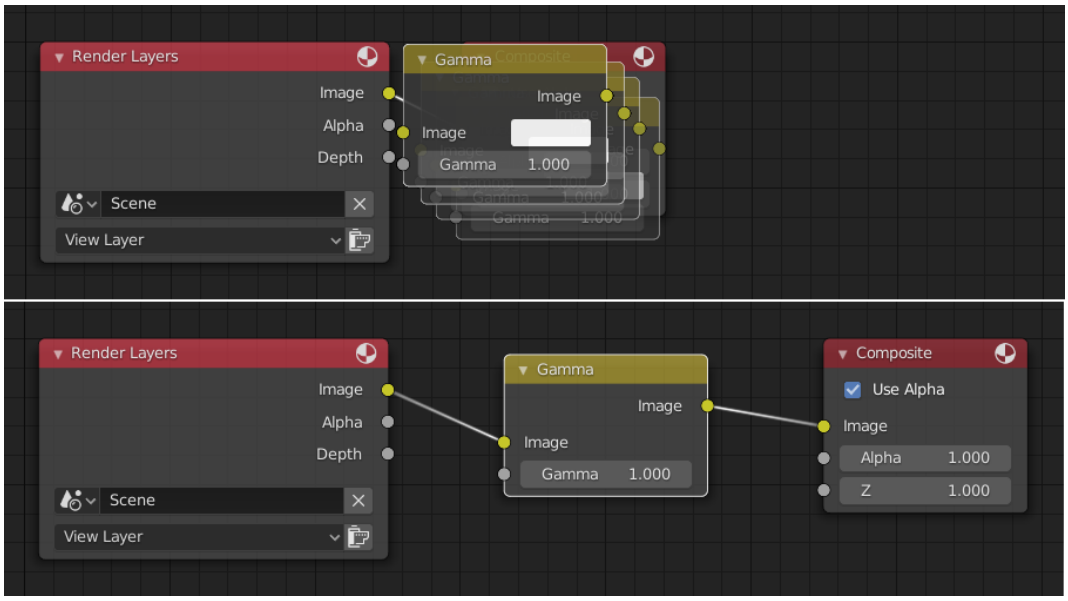
Snap Target
: Which part of the selected nodes to snap:

Closest::	Snap closest point onto target.
Center::	Snap center of selected nodes onto target.
Median::	Snap median of selected nodes onto target.
Active::	Snap active node onto target.

Auto-Offset

When you drop a node with at least one input and one output socket onto an existing connection between two nodes, *Auto-offset* will, depending on the direction setting, automatically move the left or right node away to make

room for the new node. *Auto-offset* is a feature that helps organizing node layouts interactively without interrupting the user workflow.



Auto-offset is enabled by default, but it can be disabled in the Preferences.

You can toggle the offset direction while you are moving the node by pressing `T`.

The offset margin can be changed using the *Auto-offset Margin* setting in the Editing section of the Preferences.

See also

Example Video:

Auto-Offset. A workflow enhancement for Blender's node editors.

Editing

Transform

Reference

Menu::	*Node* ▸ *Move*, *Rotate*, *Resize*
Shortcut::	`G`, `R`, `S`

You can move the selected node(s) by simply clicking and dragging any empty part on them. Alternatively, you can press `G`, move the mouse, and click `LMB` to confirm.

Dragging a node on top of an existing link will intelligently insert the selected node into the link path. This generally works by using the first socket that matches the link type. The automatic node attachment feature can be toggled with `Alt`. When a node is automatically attached the surrounding nodes will be shifted to the right or left depending on the `T` toggle; see Auto-Offset for more information on this feature.

In general it is recommended to arrange your nodes within the view such that the data flows from left to right, top to bottom.

The width of a node can be changed by dragging its left or right border.

Rotating (`R`) and scaling (`S`) only apply when you have multiple nodes selected and only affect the nodes' positions.

Connecting Sockets

`LMB`-click on a socket and drag. YConnect to Outputou will see a line coming out of it: this is called a *link*. Keep dragging and connect the link to an input socket of another node, then release the `LMB`.

While multiple links can route out of an output socket, typically a single link can be attached to an input socket, that is unless the input is a multi-socket input with looks like a pill shaped socket.

To swap multiple links of a similar type, press and hold `Alt` while moving a link. This feature also works when adding a new link into a pre-existing socket.

To reposition the outgoing links of a node, rather than adding a new one, hold `Ctrl` while dragging from an output socket. This works for single as well as for multiple outgoing links.

Nodes that have no connections can be inserted on a link by just move the node over the link and release when the link is highlighted.

Make Links `F`
: Select multiple nodes with open sockets, then use the Make Links to create links between them. Use Make Links again if there are other nodes which can be connected.

Make and Replace Links `Shift-F`
: *Make and Replace Links* works similarly to *Make Links*, but it will replace existing links if any exist.

Disconnecting Sockets

Interactively

Drag the link away from its input socket and let it go, keeping it unconnected.

Mute Links

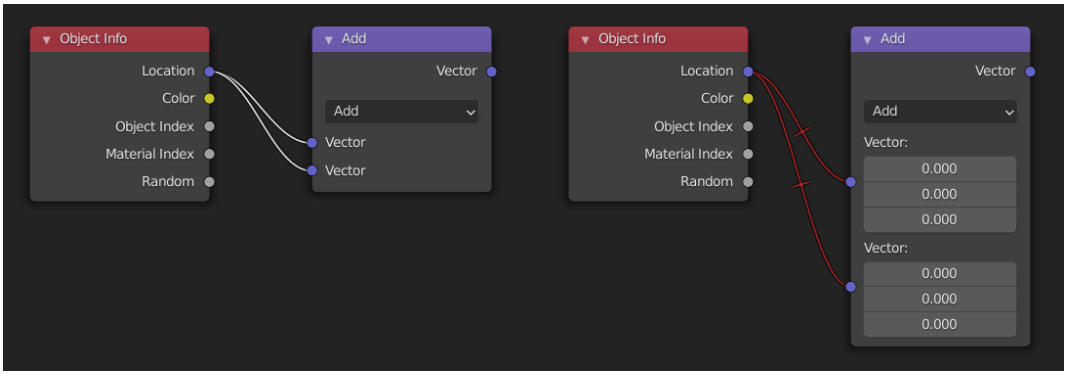
Reference

Menu:: *Node* ▸ *Mute Links*
Shortcut::

`Ctrl-Alt-RMB`

Activate the menu item or hold the key combination, then draw a line across one or more links to mute/unmute them. A muted link acts as though it's no longer there; this also means the input fields for specifying fixed values become visible again.

When muting links on the input side of a reroute node, the links on its output side will be muted too.



Cut Links

Reference

Menu:: *Node ▸ Cut Links*
Shortcut:: `Ctrl-RMB`

Activate the menu item or hold the key combination, then draw a line across one or more links to delete them.

Note

The key combination is normally reserved for Lasso Select. In node editors, lasso selection is instead performed with `Ctrl-Alt-LMB`.

Detach Links `Alt-LMB` drag

Use Detach Links to cut all the links attached to the selected nodes and move the nodes to a new location.

Copy/Paste

Reference

Menu::	*Node* ▸ *Copy*, *Node* ▸ *Paste*
Shortcut::	`Ctrl-C`, `Ctrl-V`

Not only the selected nodes but also the connections between them are copied to the clipboard.

Note

The pasted node will be placed in the *same* position as when it was copied. Use the same cautions as when duplicating.

Duplicate

Reference

Menu::	*Node* ▸ *Duplicate*
Shortcut::	`Shift-D`

Select one or more nodes, activate the menu item or press the key combination, then move the mouse to a new location and click `LMB` (or press `Return`) to place the duplicated node(s).

Note

When you duplicate a node, the new node will be positioned *exactly* on top of the node that was duplicated. If you leave it there (and it is quite easy to do so), you can **not** easily tell that there are *two* nodes there!

When in doubt, select a node and move it slightly to see if something is hidden underneath.

Duplicate Linked

Reference

Menu::	*Node ▸ Duplicate Linked*
Shortcut::	`Alt-D`

Duplicate selected nodes, but not their node trees (in the case of group nodes), and move them.

Delete

Delete `X`, `Delete`
: Deletes the selected node(s).

Delete with Reconnect `Ctrl-X`
: Deletes the selected node(s), then creates new links connecting their former input nodes to their former output nodes.

Mute

Reference

Menu::	*Node ▸ Toggle Node Mute*
Shortcut::	`M`

Muting a node removes its contribution to the node tree, and makes all links pass through it without change. Links will appear red as an indicator of passing through the muted node.

Tip

Individual node links can be muted with Mute Links.

Show/Hide

Hide `H`
: Collapses the node so only the node header is visible. This can also be toggled by clicking the triangle on the left of the node header.

Toggle Node Preview `Shift-H`
: Shows/Hides a preview region on the node that displays the frame after that node's operation has been applied. This can also be toggled by clicking the material ball icon in the node header.

Note

This operator are only available in the Compositor.

Toggle Hidden Node Sockets `Ctrl-H`
: Collapses/Expands any input or output sockets that have no other nodes connected to them.

Toggle Node Options
: Shows/Hides all node properties.

Collapse and Hide Unused Sockets
: Applies both the *Toggle Hidden Node Sockets* and *Hide* operations.

Layers

Read Render Layers `Ctrl-R`
: Reads all the current scene's render layers from cache, as needed. This can be used to save RAM while rendering because the render layers do

not have to be saved in RAM. And also for recovering some information from a failed render. For this to work, Cache Result must be enabled.

Note

This operator are only available in the Compositor.

Connect to Output

Reference

Shortcut:: `Shift-Alt-LMB`

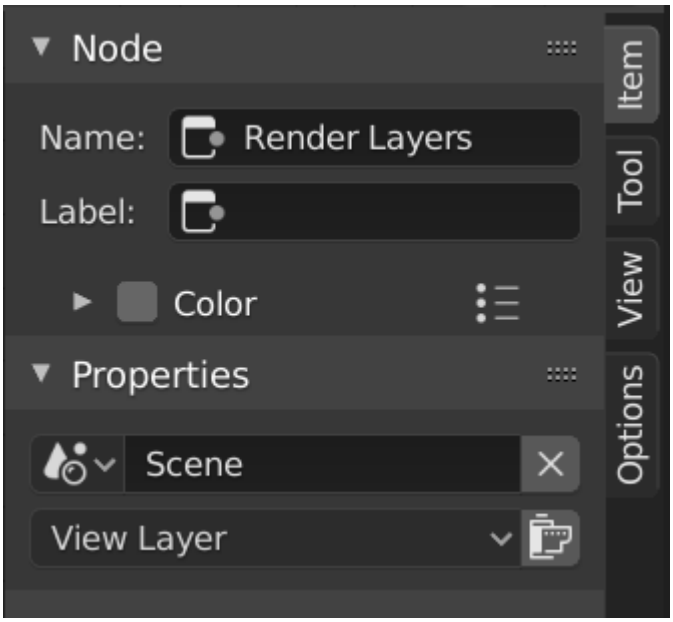
Connect the output of the selected node to the final output of the node tree (Composite in Compositor, Material Output or World Output in Shader, the final Group Output in Geometry Nodes, Output in Texture Nodes), or, if the node is inside a group, to the Group Output.

Sidebar

Node

Reference

Panel:: *Sidebar ▸ Node*



Node tab with a compositing Render Layers node selected.

Node

Name
: A unique node identifier inside this node tree.

Label
: Nodes can be given a title by modifying the text field.

Warning Propagation

Controls which warnings in this node will be propagated to the parent node group or modifier. This only exists for Geometry Nodes.

Color

By default, the node's background color is defined by the user theme. This color can be overridden by selecting a custom color in this panel. Custom node colors can be used to provide a visual cue to help distinguish some nodes from others. The button to the right of the checkbox lets you save colors as presets for reusing later on (much like a palette).

Color
: Color of the node background.

Node Color Specials
: This menu contains Operators for working with nodes with custom colors.

Copy Color
: Copies the color of the Active node and applies it to all selected nodes.

Properties

The properties that are shown depend on the type of node selected, e.g. a Mix node has different properties than a Mask node.

Tool

Reference

Panel:: *Sidebar region ▸ Tool*

Active Tool

The info in this panel changes with the selected tool.

View

Reference

Panel:: *Sidebar region ▸ View*

Annotations

You can select the Annotate tool in the Toolbar to make annotations in the node editor. See Annotate Tool for more info.

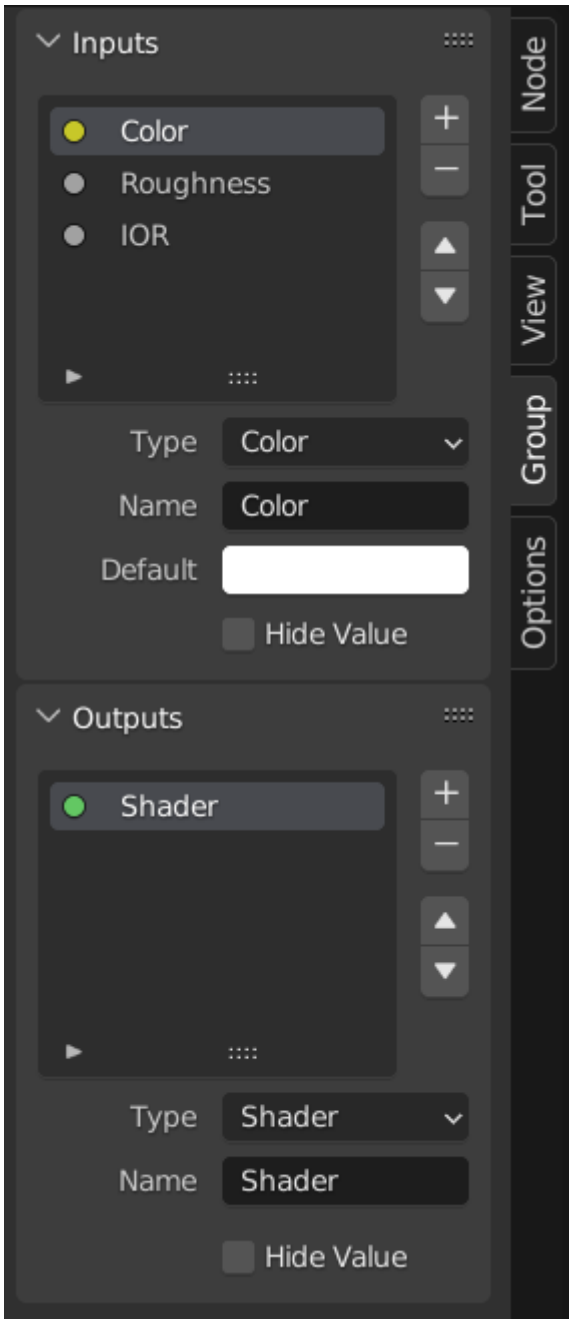
Node Groups

Grouping nodes can simplify a node tree by hiding away complexity and reusing repetitive parts.

Conceptually, node groups allow you to treat a *set* of nodes as though it were just one node. They're similar to functions in programming: they can be reused (even in different node trees) and can be customized by changing their "parameters."

As an example, say you created a "Wood" material that you would like to have in different colors. One way of doing this would be to duplicate the entire material for every color, but if you did that, you'd have to go over all those copies again if you later wanted to change the density of the grain lines. Instead, it would be better to move the nodes that define the wood look into a node group. Each material can then reuse this node group and just supply it with a color. If you then later want to change the grain line density, you only have to do it once inside the node group, rather than for every material.

Node groups can be nested (that is, node groups can contain other node groups).



Example of a node group.

Note

Recursive node groups are prohibited for all the current node systems to prevent infinite recursion. A node group can never contain itself (or another group that contains it).

Tip

Like all data-blocks, node groups with names that start with `.` are normally hidden from lists and menus and can only be accessed through search. This can be useful for node asset authors to hide their internal sub-groups from the final user.

When a node group is created, new *Group Input* and *Group Output* nodes are generated to represent the data flow into and out of the group. Furthermore connections to input sockets coming from unselected nodes will become attached to new sockets on the *Group Input* node. Similarly, outgoing connections to input sockets of unselected nodes will become attached to the new *Group Output* node.

If you want to pass an additional parameter into the group, a socket must be added to the *Group Input* node. To do this, drag a connection from the hollow socket on the right side of the *Group Input* node to the desired input socket on the node requiring an input. The process is similar for the *Group Output* regarding data you want to be made available outside the group.

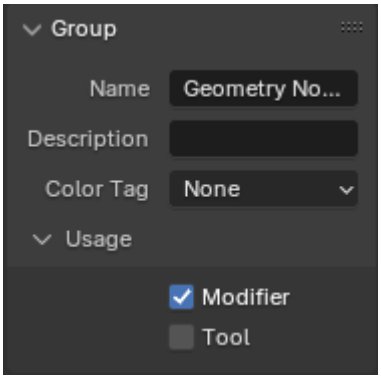
Properties

Group

Reference

Panel:: *Sidebar ▸ Group ▸ Group*

This panel contains properties that relate the group node such as it's name and look.



The *Group* panel.

Name
: The name of node as displayed in the Title.

Description
: The message displayed when hovering over the Title or in add menus.

Color Tag
: Color tag of the node group which influences the header color.

Node Width
: The width for newly created group nodes.

(Set Default Node Width)
: Set the width based on the parent group node in the current context

Usage *Geometry Nodes*

This panel is only visible in the Geometry Node Editor.

Modifier
: The node group is used as a Geometry Nodes Modifier.

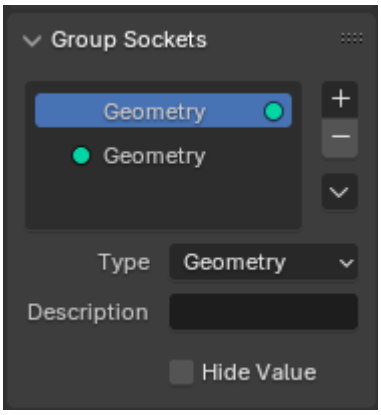
Tool
: The node group is used as a Node-Based Tools.

Group Sockets

Reference

Panel:: *Sidebar ▸ Group ▸ Group Sockets*

This panel is used to add, remove, reorder, and edit the sockets of the group's input and output.



The *Group Sockets* panel.

Socket List
: A List View of all inputs, outputs and panels.

  Here you can name the socket which is displayed in the node's interface.

Description
: The message displayed when hovering over socket properties.

Closed by Default *Panels*
: Panel is closed by default on new nodes.

Default
: The value to use when nothing is connected to the socket.

Min, Max
: The minimum and maximum value for the UI button shown in the node interface. Note, this is not a minimum or maximum for the data that can pass through the node. If a socket passes a higher value than the maximum, it will still pass into the node unchanged.

Geometry Nodes

Default Input
: Input to use when the socket is unconnected. Requires *Hide Value* to be enabled.

Hide Value
: Hide the socket value even when the socket is not connected.

Hide in Modifier
: Don't show the input value in the geometry nodes modifier interface. This allows the input to be used in the context of a node group but not

as a modifier input.

This option is only available for geometry nodes and only for input sockets.

Single Value
: Only allow single value inputs rather than Fields.

Make Group

Reference

Menu::	*Node ▸ Make Group*
Shortcut::	`Ctrl-G`

To create a node group, select the nodes you want to include, then press `Ctrl-G` or click *Group ▸ Make Group*. A node group will have a green title bar. All selected nodes will now be contained within the node group. Default naming for the node group is “NodeGroup”, “NodeGroup.001” etc. There is a name field in the node group you can click into to change the name of the group. Change the name of the node group to something meaningful.

When appending node groups from one blend-file to another, Blender does not make a distinction between material node groups or composite node groups. So it is recommended to use some naming convention that will allow you to distinguish between the two types.

Tip

The “Add” menu of each node editor contains an “Output” category with node types such as “Material Output.” These node types should not be confused with the “Group Output” node found in node groups, and should not be used in node groups either (only in the top-level node tree).

Insert Into Group

Reference

Menu:: *Node ▸ Insert Into Group*

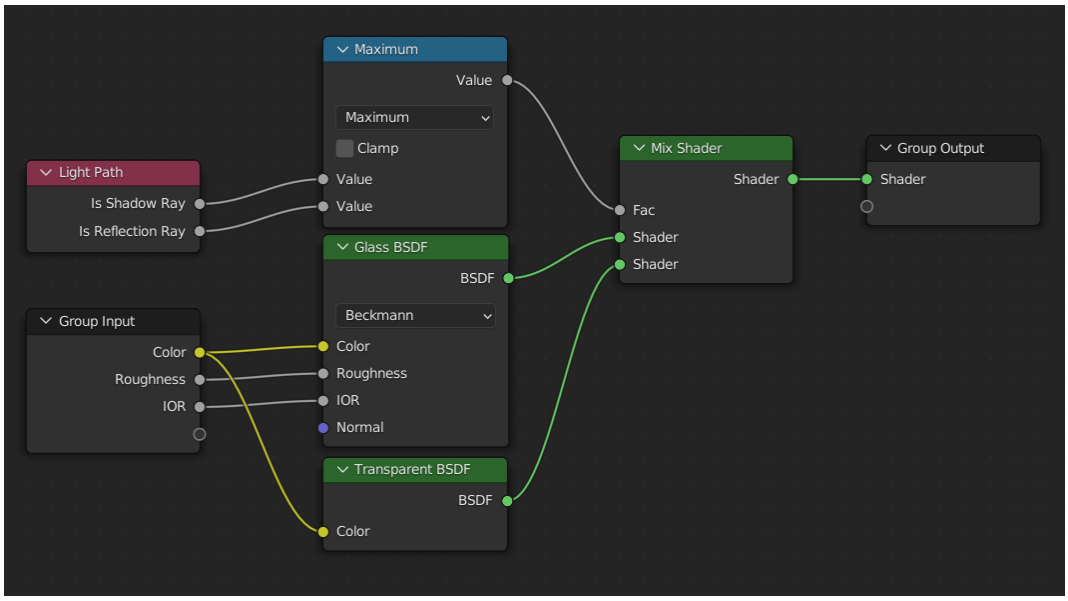
Moves the selected nodes into the active group node. To use, select a set of nodes, ending with the destination group node, then, running the operation will move those nodes into that group. The moved nodes are collected into a group of their own to preserve their connection context, having their own group input and output nodes. The group's existing input and output nodes are updated with new sockets, if any, from the new nodes. The node group must be edited to contain a single *Group Input* and a single *Group Output* node.

Edit Group

Reference

Menu:: *Node ▸ Edit Group*
Header:: *Go to Parent Node Tree*
Shortcut:: `Tab`, `Ctrl-Tab`

With a node group selected, press `Tab` to move into it and see its content. Press `Tab` again (or `Ctrl-Tab`) to leave the group and go back to its parent, which could be the top-level node tree or another node group. You can refer to the breadcrumbs in the top left corner of the node editor to see where you are in the hierarchy.



Example of an expanded node group.

Ungroup

Reference

Menu:: *Node ▸ Ungroup*
Shortcut:: `Ctrl-Alt-G`

Removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can link internal nodes to other nodes in your workspace.

Separate `P`
: Separate selected nodes from the node group.

Copy
: Copy to parent node tree, keep group intact.

Move
: Move to parent node tree, remove from group.

Reusing Node Groups

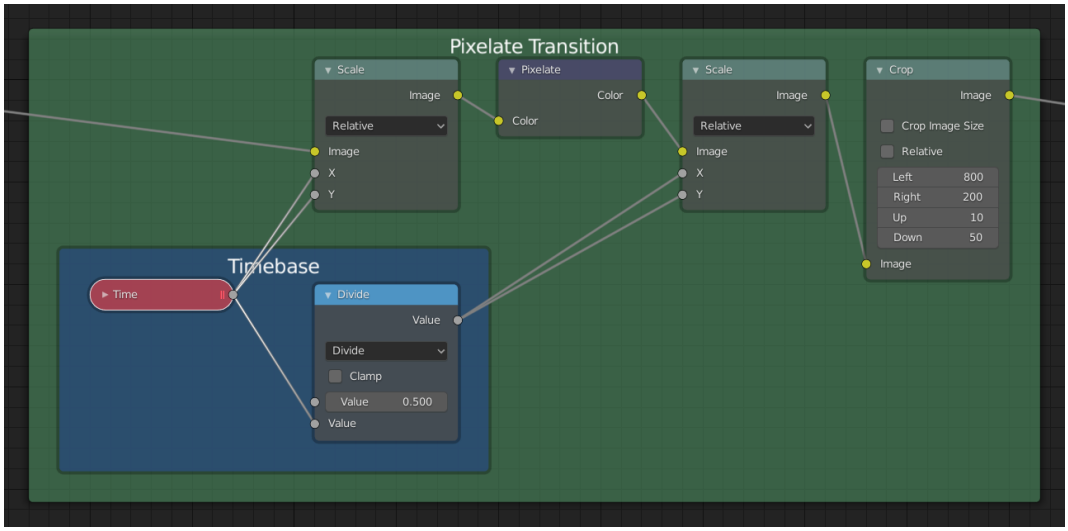
Reference

Menu::	*Add ▸ Group*
Shortcut::	`Shift-A`

Existing node groups can be placed again after they're initially defined, be it in the same node tree or a different one. It's also possible to import node groups from a different blend-file using *File ▸ Link/Append*.

Frame Node

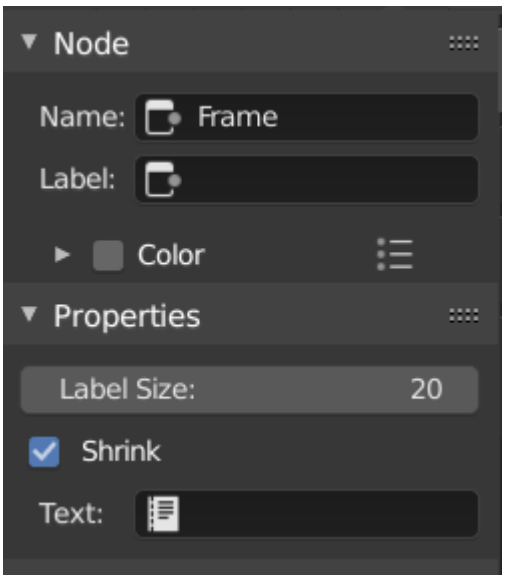
The Frame node is useful for organizing nodes by collecting related nodes together in a common area. Frames are useful when a node setup becomes large and confusing yet the re-usability of a Node Group is not required.



Properties

Label Size

Font size of the label. For example, for subordinate frames to have smaller titles.



Shrink

Once a node is placed in the Frame, the Frame shrinks around it so as to remove wasted space. At this point it is no longer possible to select the edge of the Frame to resize it, instead resizing occurs automatically when nodes within the Frame are

rearranged. This behavior can be changed by disabling this option.

Text
: When you need to display more comprehensive text, frame nodes can display the contents of a text data-block. This is read-only, so you will need to use the *Text Editor* to modify the contents.

Editing

Join in New Frame

Reference

Menu:: *Node ▸ Join in new Frame*
Shortcut:: `Ctrl-J`

Make a new frame including the selected nodes.

Add to Frame

Reference

Shortcut:: `Ctrl-P`

Once a frame node is placed, nodes can be added by dropping them onto the frame or by selecting the node(s) then the frame and using `Ctrl-P`. This can be thought of as Parenting the selection to the frame.

Remove from Frame

Reference

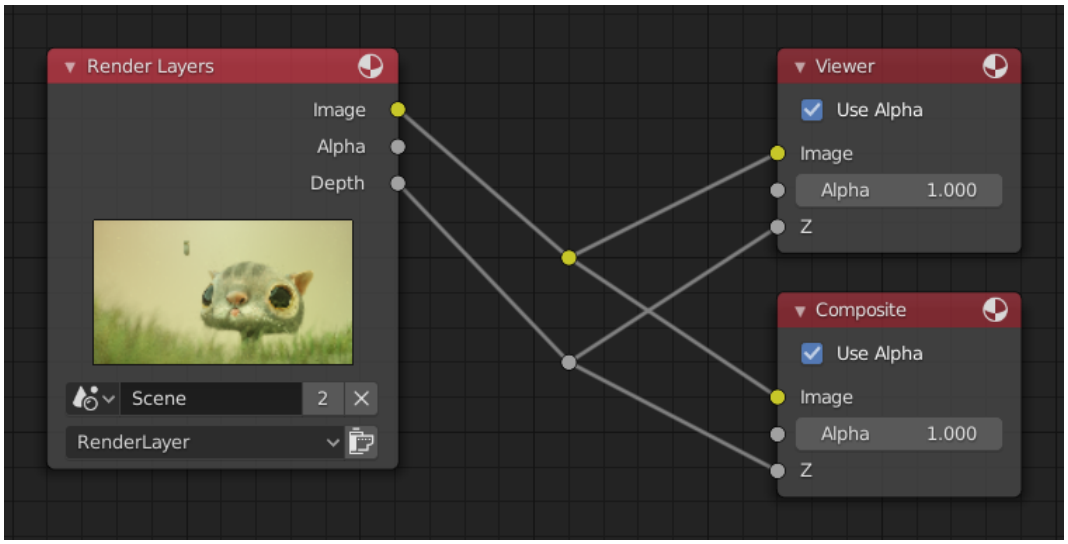
Menu:: *Node ▸ Remove from Frame*
Shortcut:: `Alt-P`

To remove nodes from a frame, select and use `Alt-P`. This can be thought of as unparenting the selection from the frame.

Reroute Node

A node used primarily for organization. Reroute looks and behaves much like a socket on other nodes in that it supports one input connection while allowing multiple output connections.

To quickly add a Reroute node into an existing connection, hold `Shift` and `RMB` while sweeping across the link to add a *Reroute node*.



Properties

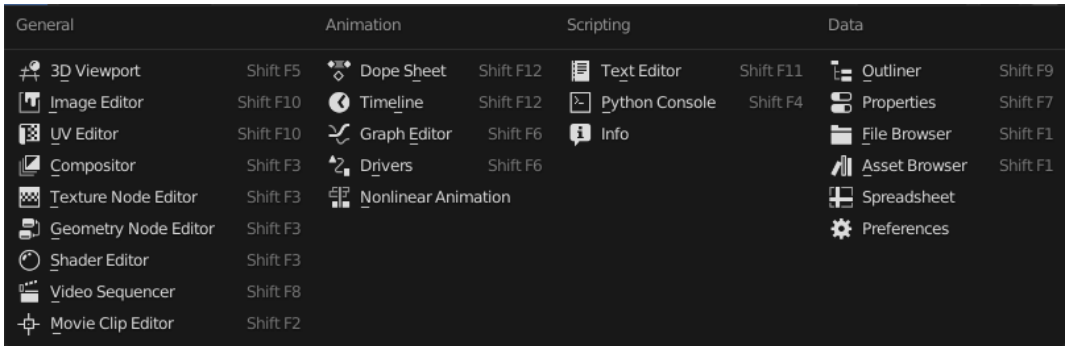
Input
: Input value used for unconnected sockets.

Editors

Blender provides a number of different editors for displaying and modifying different aspects of data. An Editor is contained inside an Area which determines its size and placement within the Blender window. Every area may contain any type of editor.

The *Editor Type* selector, the first button at the left side of a header, allows you to change the Editor in that area. It is also possible to open the same Editor type in different areas at the same time.

See User Interface for documentation on the general interface.



The Editor Type selector.

General

- 3D Viewport
- Image Editor
- UV Editor
- Compositor
- Texture Nodes
- Geometry Node Editor
- Shader Editor
- Video Sequencer
- Movie Clip Editor

Animation

- Dope Sheet
- Timeline
- Graph Editor
- Drivers Editor
- Nonlinear Animation

Scripting

- Text Editor
- Python Console
- Info Editor

Data

- Outliner
- Properties
- File Browser
- Asset Browser
- Spreadsheet
- Preferences

3D Viewport

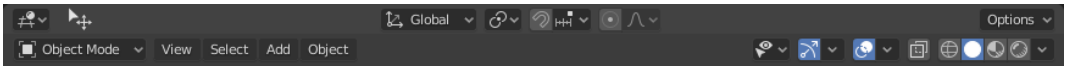
- Introduction
  - Header Region
  - Toolbar Region
  - Sidebar Region
  - Asset Shelf Region
- Startup Scene
  - Elements
- Object Modes
  - Object Mode List
  - Switching Objects
  - Multi-Object Editing
- Navigating
  - Introduction
  - Navigation
  - Fly/Walk Navigation
  - Aligning
  - Perspective/Orthographic
  - Local View
  - Camera View
  - Viewpoint
  - View Regions
  - Contextual Views
- 3D Cursor
  - Placement
- Selecting
  - Object Mode
  - Edit Mode
  - Pose Mode
  - Particle Edit Mode
- Controls
  - Transform Orientation
  - Pivot Point
  - Snapping

    - Proportional Editing
- Display
    - Object Type Visibility
    - Viewport Gizmos
    - Viewport Overlays
    - Viewport Shading
- Toolbar
    - Object Mode
    - Edit Mode
    - Paint Modes
    - Grease Pencil
- Sidebar
    - Item
    - Tool
    - View
- Viewport Render
    - Settings
    - Rendering

Introduction

The 3D Viewport is used to interact with the 3D scene for a variety of purposes, such as modeling, animating, texture painting, etc.

Header Region



Object Mode header.

The header contains various menus and controls based on the current mode. Its items are split into three groups:

Mode & Menus

Mode `Ctrl-Tab`
: The 3D Viewport has several modes used for editing different kinds of data. For example, the default Object Mode would let you place a character in the scene, while Pose Mode would allow you to pose it.

  The shortcut `Ctrl-Tab` brings up a pie menu for quick mode switching. If you have an Armature selected, it'll instead switch between Object Mode and Pose Mode.

  Pressing `Tab` will switch between Object Mode and Edit Mode for objects that support it.

View
: This menu offers tools for navigating in 3D space.

The other menus depend on the current mode, Object Mode menus listed below:

Select
: Contains tools for selecting objects.

Add `Shift-A`
: Contains a list of different objects types that can be added to the scene.

Object
: Contains tools for operating on objects, such as duplicating them. A subset of these tools can also be accessed by rightclicking in the 3D Viewport.

Transform Controls

Transform Orientation `Comma`
: Used to change the Transform Orientation, which affects the rotation of the transform gizmo.

Pivot Point `Period`
: Used to change the Pivot Point, which affects the location of the transform gizmo.

Snapping `Shift-Tab`
: Offers options for snapping items to others that are nearby. You can hold `Ctrl` to toggle snapping on/off temporarily (as long as the key is held).

Proportional Editing `O`
: Used to smoothly transform unselected items that are near the selected ones. See Proportional Editing.

Display & Shading

Object Type Visibility
: Change which types of objects are visible/selectable in the 3D Viewport. See Object Type Visibility.

Viewport Gizmos
: Change how gizmos are displayed in the 3D Viewport.

Viewport Overlays
: Change how overlays are displayed in the 3D Viewport.

X-Ray `Alt-Z`
: Make the whole scene transparent, allowing you to see and select items that would otherwise be occluded. This is a shortcut to the X-Ray option inside the Viewport Shading popover.

  In Pose Mode, this same button controls a different setting with its own separate on/off state. Rather than making the scene transparent, it shows the armature in front of any geometry.

Viewport Shading
: Change the shading of the 3D Viewport.

Toolbar Region

The Toolbar contains tools depending on the current mode (for example, modeling tools in *Edit Mode*, brush tools in *Sculpt Mode*…).

See Tools for more information.

Sidebar Region

The Sidebar region contains properties of the active object and tool, as well as of the viewport itself.

See Sidebar for more information.

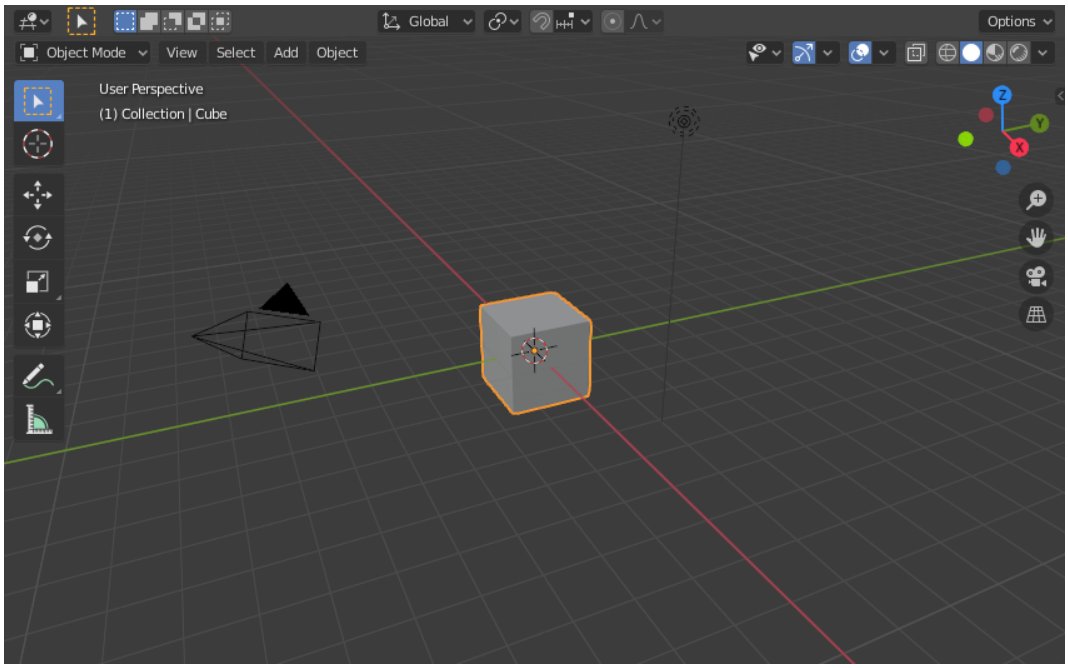
Asset Shelf Region

Depending on the current mode, the asset shelf may be available, providing quick access to assets for this specific mode (for example pose assets in *Pose Mode*, brush assets in *Sculpt Mode*).

See Asset Shelf for more information.

Startup Scene

After closing the splash screen, the startup scene is displayed in the 3D Viewport (if no other blend-file was loaded). This startup scene can be customized.



The startup scene.

Elements

Cube
: The gray cube in the center of the scene is a mesh object. Its orange outline indicates that it's selected. The orange dot in the center is its Origin, which indicates its precise location.

Light
: The set of concentric black circles is a light source illuminating the cube.

Camera
: The pyramid with a big triangle above it is the camera, which is used as the point of view for rendering.

3D Cursor
: The 3D cursor, a cross with a red-and-white circle, determines where newly added objects are placed and can also serve as a transformation pivot point.

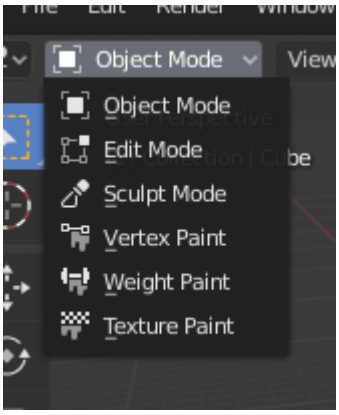
Grid Floor
: The gray lines forming a floor mark the zero height of the world. The red and green lines are the axes of the world coordinate system. They meet at the world origin, which is also where the origin of the *Cube* is located. The Grid Floor settings are in the Viewport Overlays popover.

Text Info

The top left corner of the viewport shows various bits of information – see Viewport Overlays for details.

Object Modes

Modes allow editing different aspects of objects. While Object Mode allows you to position/rotate/scale them, Edit Mode allows changing their geometry, Pose Mode allows posing them, and so on.



The Mode select menu.

You can change the current mode using the *Mode* selector in the 3D Viewport header. Which modes are available depends on the object's type. The complete list is shown below.

Apart from using the selector, you can also press `Ctrl-Tab` to bring up a pie menu around the cursor for faster access. (If the selected object is an Armature, this shortcut will instead switch between Object Mode and Pose Mode.)

Pressing `Tab` will toggle Edit Mode for objects that support it.

Modes can affect many things in Blender:

- Each mode changes the header and Toolbar to show its own unique set of menus and tools. This also means it affects the available keyboard shortcuts.
- Modes can completely change the look of the viewport. For example, Weight Paint mode will shade the object to show its vertex weights, which are not normally visible.
- Modes can affect other editors. For example, the UV Editor can only be used if the 3D Viewport is in Edit Mode. In the Properties editor, too, certain buttons and panels can only be used in certain modes.

Object Mode List

Blender's Modes

Icon	Name	Details
	Object Mode	The default mode, available for all object types. Allows editing position, rotation and scale, duplicating objects, and so on.
	Edit Mode	A mode for editing an object's shape (vertices/edges/faces for meshes, control points for curves/surfaces, points/strokes for Grease Pencil, etc.).
	Sculpt Mode	Provides an alternative toolset for editing an object's shape (only for meshes).
	Vertex Paint Mode	A mesh-only mode that allows you to set your mesh's vertex colors (i.e. to "paint" them).
	Weight Paint Mode	A mesh-only mode, dedicated to vertex group weighting.
	Texture Paint Mode	A mesh-only mode that allows you to paint a texture directly on the model, in the 3D Viewport.
	Particle Edit Mode	A mesh-only mode dedicated to particle systems, useful for editable systems (hair).

Icon	Name	Details
	Pose Mode	An armature-only mode, dedicated to posing.
	Draw Mode	A Grease Pencil-only mode, dedicated to creating Grease Pencil strokes.
	Sculpt Mode (Grease Pencil)	A Grease Pencil-only mode, used to deform and shape existing strokes more organically.
	Edit Mode (Grease Pencil)	A Grease Pencil-only mode, dedicated to modifying individual strokes and points of Grease Pencil objects.
	Vertex Paint Mode (Grease Pencil)	A Grease Pencil-only mode, dedicated to adding color the vertices of strokes directly.
	Weight Paint Mode (Grease Pencil)	A Grease Pencil-only mode, dedicated to assigning vertex weights to stroke.

Note

The cursor becomes a brush in Paint and Sculpt Modes.

We will not go into any more detail on mode usages here, because they are dealt with in their own sections.

Hint

If you are reading this manual and some button or menu option is referenced that does not appear on your screen, it may be that you are not in the proper mode for that option to be valid.

Switching Objects

Reference

Mode::	All Modes
Shortcut::	`Alt-Q`

If you enter a mode such as Weight Paint for an object and then select another object, Blender will typically switch back to Object Mode. This means that, if you want to weight paint the other object too, you have to enter the mode a second time.

There is a way of avoiding this, however. Once you enter a mode, the Outliner will show a dot next to other objects that also support it. By clicking such a dot, you can switch over to another object without leaving the mode.

Alternatively, you can hover over the other object in the 3D Viewport and press `Alt-Q`.

See also

Lock Object Modes for preventing *accidental* mode changes.

Multi-Object Editing

Edit Mode and Pose Mode let you work with multiple objects even more easily than described above, as they can have multiple objects in the mode at the same time.

There are two ways of accomplishing this:

- If you're not yet in the mode, you can simply select all the objects and enter it.
- If you're already in the mode, you can bring other objects into it by clicking `Ctrl–LMB` on the dot in the Outliner. Removing objects from the mode works in the same way.

Some points of note:

- The Properties editor will only ever show the details (shape keys, UV maps…) of the active object, not of all the selected ones.
- Selecting any element from an object will make it the active one.
- There are limits to the edits you can make. For example, you can't create an edge that connects vertices from different objects.

Navigating

- Introduction
  - Navigation Gizmo
- Navigation
  - Orbit
  - Roll
  - Pan
  - Zoom In/Out
  - Zoom Region
  - Dolly View
  - Frame All
  - Frame Selected
- Fly/Walk Navigation
  - Walk Navigation
  - Fly Navigation
- Aligning
- Perspective/Orthographic
  - Options
- Local View
  - Toggle Local View
- Camera View
  - Viewing the Active Camera
  - Setting the Active Camera
  - Frame Camera Bounds
  - Zoom Camera 1:1
  - Camera Positioning
- Viewpoint
- View Regions
  - Clipping Region
  - Render Region
- Contextual Views
  - Quad View

Introduction

To be able to work in the three-dimensional space that Blender uses, you must be able to change your viewpoint as well as the viewing direction of the scene. While we will describe the *3D Viewport* editor, most of the other editors have similar functions. For example, it is possible to pan and zoom in the Image editor.

Tip

Some navigation tools require a middle mouse button or numpad. If you don't have one of these, see the Keyboard and Mouse page of the manual to learn how to work around this.

Navigation Gizmo

The navigation gizmo can be found in the top right of the editor.

Navigation Gizmo (left) and Navigation Gizmo in camera view (right).

Navigation

Orbit

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Orbit*
Shortcut::	`MMB`, `Numpad2`, `Numpad4`, `Numpad6`, `Numpad8`.

Rotate the view around the point of interest by clicking and dragging `MMB` on the viewport's area.

`RMB` cancels the orbit operation.

The `Alt` key has several effects on orbiting:

- Clicking a point with `Alt-MMB` will make it the point of interest: it becomes the central point which the view orbits around.
- Holding `Alt` and then dragging with `MMB` in a certain direction will align the view to an axis and make it orthographic.
- Dragging with `MMB` and then holding `Alt` will perform an orbit while also snapping to the world axes, as well as the diagonals between them.

To change the viewing angle in discrete steps, use `Numpad8` and `Numpad2` to go up and down, or `Numpad4` and `Numpad6` for left and right. You can also press `Numpad9` to switch to the opposite side of the view (rotates the camera 180° around the Z axis).

See also

- Orbit Style Preference

- Auto-Perspective Preference

Roll

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Roll*
Shortcut::	`Shift-Numpad4`, `Shift-Numpad6`

Rotate the viewport camera around its viewing direction in 15° discrete steps by default. See the rotation angle preference to configure.

To reset the roll, you can first align the view to the global X axis using `Numpad3`, then orbit to get back to the regular perspective view.

`RMB` cancels the roll operation.

Pan

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Pan*
Shortcut::	`Shift-MMB`, `Ctrl-Numpad2`, `Ctrl-Numpad4`, `Ctrl-Numpad6`, `Ctrl-Numpad8`

Moves the view up, down, left and right. To pan the view, hold down `Shift` and drag `MMB` in the 3D Viewport. For discrete steps, use the hotkeys `Ctrl-Numpad8`, `Ctrl-Numpad2`, `Ctrl-Numpad4` and `Ctrl-Numpad6` as with orbiting.

`RMB` cancels the orbit operation.

Zoom In/Out

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Zoom In/Out*
Shortcut::	`Ctrl-MMB`, `Wheel`, `NumpadPlus`, `NumpadMinus`

Moves the view closer to, or further away from, the point of interest. You can zoom in and out by rolling the `Wheel` or dragging with `Ctrl-MMB`. To zoom with discrete steps, use the hotkeys `NumpadPlus` and `NumpadMinus`.

Hint

If you get lost in 3D space (which is not uncommon), Frame All and Frame Selected can be used to show the contents of your scene.

`RMB` cancels the zoom operation.

Zoom Region

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Zoom Region…*
Shortcut::	`Shift-B`

The *Zoom Region* tool allows you to specify a rectangular region by dragging with `LMB`. The view will then zoom in on this region.

You can also drag with `MMB` to zoom out instead.

Dolly View

Reference

Mode:: All modes
Menu:: *View ▸ Navigation ▸ Dolly View…*
Shortcut:: `Shift-Ctrl-MMB`

In most cases it's sufficient to zoom the view to get a closer look at something. However, zooming only gets you up to the point of interest and no further. If you hit this point where zooming no longer works, you can instead Dolly by holding `Shift-Ctrl` and dragging up or down with `MMB`. This will move the point of interest (and the view along with it).

`RMB` cancels the dolly operation.

Hint

Dolly changes orthographic views to a perspective projection.

This is done because dolly doesn't work well with an orthographic projection because *moving* forwards/backwards with an orthographic projection doesn't have the effect of zooming.

Frame All

Reference

Mode:: All modes
Menu:: *View ▸ Frame All*
Shortcut:: `Home`

Changes the view so that you can see all objects.

Frame Selected

Reference

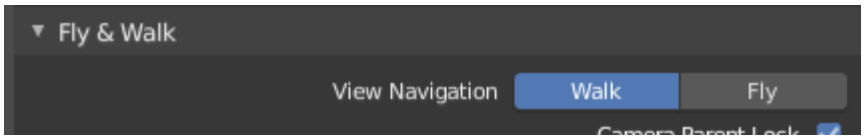
Mode::	All modes
Menu::	*View ▸ Frame Selected*
Shortcut::	`NumpadPeriod`

Changes the view so that you can see the selected object(s).

Fly/Walk Navigation

The standard navigation controls are sometimes limiting, especially for large environments such as architectural models. In these cases, it may be preferable to use first person controls instead, where you can look around while "standing" in one place rather than orbiting around a central viewpoint.

Blender offers two such alternative navigation methods: Flying and Walking. You can initiate either method from the *View ▸ Navigation* menu. You can also initiate your preferred one (configured in the Preferences) by pressing `Shift-AccentGrave`.



View Navigation.

Common use cases for Fly/Walk include:

Navigating
: This can be a quick way to navigate a large scene.

Positioning a camera
: When activated from a camera view `Numpad0`, the camera will move along with you.

Recording camera movement
: You can record the path you take by entering a camera view, enabling Auto Keying in the Timeline, starting animation playback, and finally activating Fly/Walk navigation. The path will be recorded as camera keyframes which can then be used for rendering.

Animation playback can't be controlled while Fly/Walk navigation is active, so when you're done recording, you first need to exit the navigation with `LMB` before you can stop playback.

Walk Navigation

Reference

Mode::	All modes
Menu::	*View* ▸ *Navigation* ▸ *Walk Navigation*

This navigation method behaves like a typical first person game. It works with a combination of keyboard keys and mouse movement.

Usage

Move the mouse in the direction you want to look and use the keys listed below to walk around the scene.

When you are happy with the new view, press `LMB` to confirm. In case you want to go back to where you started, press `Esc` or `RMB`.

All these keys are also listed in the Status Bar while navigating. Settings like mouse sensitivity and default speed can be adjusted in the Preferences.

`W/Up`	Move forward.
`S/Down`	Move backward.
`A/Left`	Strafe left.
`D/Right`	Strafe right.

`E`	Move up (global) – only available if *Gravity* is off.
`Q`	Move down (global) – only available if *Gravity* is off.
`R`	Move up (local) – only available if *Gravity* is off.
`F`	Move down (local) – only available if *Gravity* is off.
`Spacebar`	Teleport to the location at the crosshair (offset by the *Camera Height* value set in the Preferences).
`WheelUp/NumpadPlus`	Increase the movement speed.
`WheelDown/NumpadMinus`	Decrease the movement speed.
`Shift`	Speed up the movement temporarily.
`Alt`	Slow down the movement temporarily.
`V`	Jump – only available if *Gravity* is on.
`Tab`	Toggle *Gravity*.

`Z`	Correct the Z axis of the view (smoothly roll it to ensure it's upright, not tilted to a side).
`Period`	Increases the jump height.
`Comma`	Decreases the jump height.

Fly Navigation

Reference

Mode:: All modes
Menu:: *View ▸ Navigation ▸ Fly Navigation*

On activation, the cursor is centered inside a rectangle that defines a safe zone. When the cursor is outside this zone, the view will rotate/pan.

Usage

Move the mouse outside the safe zone in the direction you want to look.

Click `LMB` or press `Spacebar` to keep the current view and exit Fly navigation. In case you want to go back to where you started, press `Esc` or `RMB`.

`W/Up`	Accelerate forward.
`S/Down`	Accelerate backward.
`A/Left`	Accelerate left.

D/Right	Accelerate right.
E	Accelerate upward.
Q	Accelerate downward.
MMB	Drag to pan the view. Flying will pause while you're doing this.
WheelUp/NumpadPlus	Increase the acceleration in the direction of motion. If there is no motion, start accelerating forward.
WheelDown/NumpadMinus	Decrease the acceleration in the direction of motion. If there is no motion, start accelerating backward.
Alt	Slow down as long as the key is held, until the view eventually comes to a standstill.
Ctrl	Disable rotation – while held, the view rotation doesn't influence the flight direction. This allows you to fly past an object, keeping it centered in the view even as you fly away from it.
X	Toggle X axis correction. If enabled, the view will smoothly pitch to look at the horizon when the cursor is in the safe zone.

Z	Toggle Z axis correction. If enabled, the view will smoothly roll to an upright orientation.

Aligning

Reference

Menu:: *View ▸ Align View*

These options allow you to align and orient the view.

Align View to Active
: Aligns the view to a certain local axis of the active object, bone, or (in Edit Mode) the normal of the active face. The view also becomes orthographic.

  To return to the regular (untilted) perspective view, you can first press `Numpad3` to align to the global X axis, then orbit with `MMB`.

Align Active Camera to View `Ctrl-Alt-Numpad0`
: Moves and rotates the active camera so it matches the current viewpoint.

Align Active Camera to Selected
: Moves the active camera (without changing its orientation) so that its view frames the selected objects.

Center Cursor and Frame All `Shift-C`
: Moves the 3D Cursor back to the world origin and changes the view so that you can see everything in your scene.

Center View to Cursor
: Centers the view on the 3D Cursor.

View Lock to Active
: Centers the view on the active object and makes it the point of interest. The view will continue orbiting around the object even if you pan to a

different location. In addition, it will follow the object if it moves.

View Lock Clear

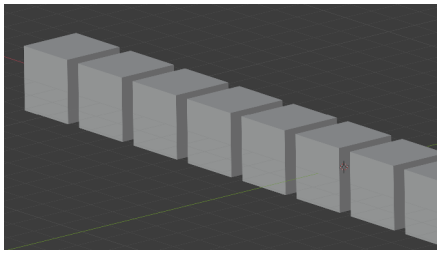
Returns the view to how it was before using *View Lock to Active*.

Perspective/Orthographic

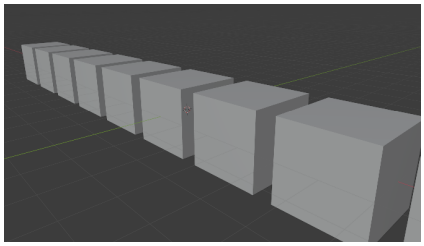
Reference

Mode::	All modes
Menu::	*View ▸ Perspective/Orthographic*
Shortcut::	`Numpad5`

This operator changes the projection of the viewport camera. Each 3D Viewport supports two different types of projection. These are demonstrated in the Fig. below.



Orthographic projection.



Perspective projection.

Our eyes are used to perspective viewing where distant objects appear smaller. Orthographic projection often seems a bit odd at first, because objects stay the same size regardless of their distance. It is like viewing the scene from an infinitely distant point. Nevertheless, orthographic viewing can be very useful, because it provides a more “technical” insight into the scene, making it easier to model and judge proportions.

Options

To toggle between the two projections for the 3D Viewport, select *View ▸ Perspective/Orthographic* or use the shortcut `Numpad5`. Changing the projection for a 3D Viewport does not affect the way the scene will be

rendered. Rendering is in perspective by default. If you need to create an orthographic rendering, select the camera, go to the Camera tab in the Properties editor, and set the Type in the Lens panel to *Orthographic*.

See also

- Auto-Perspective Preference
- Camera Lens Type
- Camera Projections

Local View

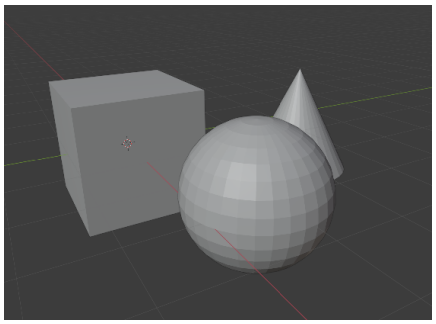
Toggle Local View

Reference

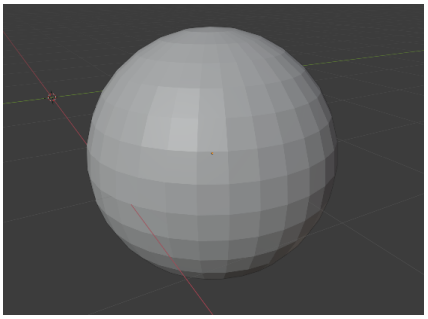
Mode::	All modes
Menu::	*View* ▸ *Local View* ▸ *Toggle Local View*
Shortcut::	`NumpadSlash`, `Slash`

Global view shows all 3D objects in the scene. Local view isolates the selected object(s) so that they are the only ones visible in the viewport. This is useful for working on objects that are obscured by others, or to speed up the viewport performance in heavy scenes. Local view is contextual, meaning that it can be set per 3D Viewport.

You can toggle between *Global* and *Local View* by selecting the option from the View menu or using the shortcut `NumpadSlash`.



Global View.



Local View.

Note

In local view, the 3D cursor is not locked to the scene. Instead, each view has an independent cursor location.

Tip

Accidentally pressing `NumpadSlash` can happen rather often if you are new to Blender, so if a bunch of the objects in your scene seem to have mysteriously vanished, try pressing `NumpadSlash` again.

Remove from Local View

Reference

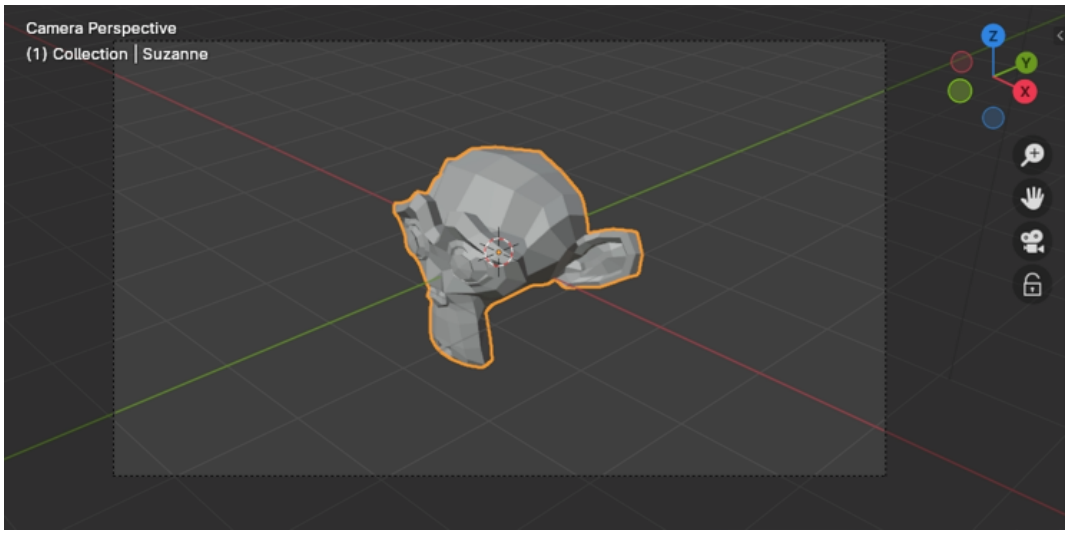
Mode::	All modes
Menu::	*View ▸ Local View ▸ Remove from Local View*
Shortcut::	`Alt-NumpadSlash`, `Alt-Slash`

Objects can be removed from Local View by selecting them and using the *Remove from Local View* operator. This will move them back to the global view. If the last remaining object is removed, the local view will be left empty and you will have to exit it to see any objects.

Hint

This is useful when working with objects in dense scenes where painstakingly selecting objects to include in the local view isn't practical, especially when they intersect or are obscured by objects you don't want to include. In this case it's simpler to select many objects in a region and enter local view, then remove the ones you don't need.

Camera View



Demonstration of camera view.

The Camera view shows the current scene from the active camera's viewpoint.

The Camera view can be used to virtually compose shots and preview how the scene will look when rendered. The rendered image will contain everything within the dashed frame.

See also

Camera Settings for details on how camera settings are used for display and rendering.

Hint

While in camera view, you can select the camera by clicking the dashed frame (assuming the camera object isn't hidden).

Viewing the Active Camera

Reference

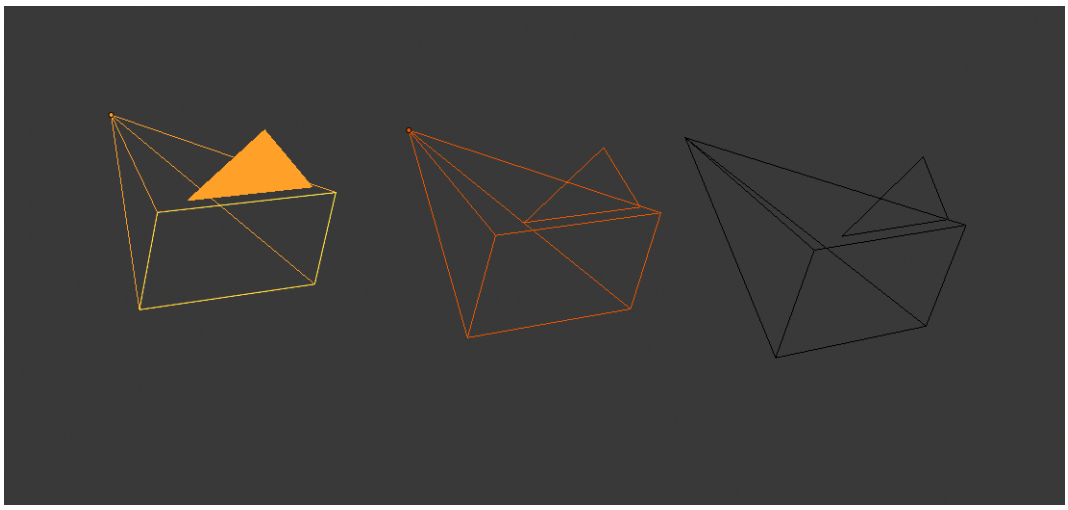
Mode::	All Modes
Menu::	*View* ▸ *Cameras* ▸ *Active Camera*, *View* ▸ *Viewpoint* ▸ *Camera*
Shortcut::	`Numpad0`

This switches the view to the active camera.

Setting the Active Camera

Reference

Mode::	Object Mode
Menu::	*View* ▸ *Cameras* ▸ *Set Active Object as Camera*
Shortcut::	`Ctrl-Numpad0`



Active camera (left) displayed with a solid triangle above it.

This sets the current active object as the active camera and switches to the camera view.

The active camera is the one that will be used for rendering, and which you'll look through when choosing camera view.

Another way of setting the active camera is through the *Scene* tab of the Properties.

Note

The active camera is normally defined on the scene level, so that it's the same across all 3D Viewports. However, it's also possible to make a camera the active one within one Viewport only. See Local Camera.

Animated Camera Switching

While a scene contains only one camera by default, it's possible to have multiple. You can then bind the cameras to specific time points in your animation to create jump cuts showing different viewpoints. See Animating Cameras.

Frame Camera Bounds

Reference

Mode::	All Modes
Menu::	*View* ▸ *Cameras* ▸ *Frame Camera Bounds*
Shortcut::	`Home`

Centers the camera view inside the 3D Viewport's screen area and resizes the view to fit within the area's bounds.

Zoom Camera 1:1

Reference

Mode::	All Modes
Menu::	*View ▸ Navigation ▸ Zoom Camera 1:1*

Zooms the view so that the camera frame has the exact same size as the output resolution. This allows you to preview exactly how large objects will be in the rendered image/animation.

Camera Positioning

There are several different ways to position the camera in your scene. Some of them are explained below.

Hint

The active “camera” might be any kind of object, meaning these actions can also be used to position and aim a light for example.

Align Active Camera to View

Reference

Mode::	Object Mode
Menu::	*View ▸ Align View ▸ Align Active Camera to View*
Shortcut::	`Ctrl-Alt-Numpad0`

Moves and rotates the camera so it perfectly matches your current viewport view.

Camera Navigation

By enabling Lock Camera to View in *Sidebar ▸ View* and switching to camera view or toggle the lock navigation gizmo button when in camera view, the camera will become “glued” to the view and follow it around as you navigate.

See also

Fly/Walk Navigation for first person navigation that moves the active camera too.

Roll, Pan, Dolly, and Track

To perform these camera moves, the camera must first be selected so transform operations apply to it. The following actions also assume that you are in camera view. Having done so, you can now manipulate the camera using the same tools that are used to transform any object:

Roll
: Press `R` to enter object rotation mode. The default will be to rotate the camera along its local Z axis (the axis orthogonal to the camera view), which is the definition of a camera "roll".

Vertical Pan or Pitch
: This is just a rotation along the local X axis. Press `R` to enter object rotation mode, then `X` twice. (The first press selects the *global* axis, the second the *local* axis. This works with any axis; see Axis Locking).

Horizontal Pan or Yaw
: This corresponds to a rotation around the camera's local Y axis. Press `R`, then `Y` twice.

Dolly
: To dolly the camera, press `G` then `MMB` (or `Z` twice).

Sideways Tracking
: Press `G` and move the mouse (you can use `X` or `Y` twice to get purely horizontal or vertical tracking).

Viewpoint

The menu *View ▸ Viewpoint* lets you align the viewing direction to a specific axis. This can also be done using the Navigation Gizmo or the following hotkeys:

Top:: `Numpad7`
Front:: `Numpad1`
Right:: `Numpad3`
Bottom:: `Ctrl-Numpad7`
Back:: `Ctrl-Numpad1`
Left:: `Ctrl-Numpad3`

The above hotkeys align the view to a global (world) axis. You can also align to a local axis of the selected item by additionally holding `Shift`. This way, you can for example view any mesh face head-on, no matter how it's oriented. (To get out of this local viewpoint, simply align to a global axis again.)

The view can also be aligned by holding `Alt-MMB` and dragging the mouse in a certain direction.

View Regions

Clipping Region

Reference

Mode::	All modes
Menu::	*View* ▸ *View Regions* ▸ *Clipping Region…*
Shortcut::	`Alt-B`

Allows you to define a clipping region to limit the 3D Viewport display to a portion of 3D space. It can assist in the process of working with complex models and scenes.

Once activated, you have to draw a rectangle with the mouse. It becomes a clipping volume of four planes:

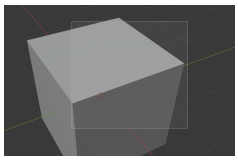
- A right-angled parallelepiped (of infinite length) if your view is orthographic.
- A rectangular-based pyramid (of infinite height) if your view is in perspective.

Once clipping is used, you will only see what's inside the volume you defined. Tools such as paint, sculpt, selection, transform snapping, etc. will also ignore geometry outside the clipping bounds.

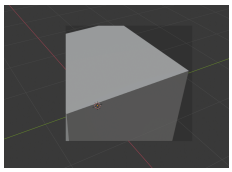
To delete this clipping, press `Alt-B` again.

Example

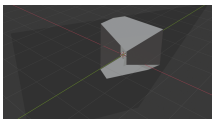
Region/Volume clipping.



Selecting a region.



Region selected.



View rotated.

The *Region/Volume clipping* image shows an example of using the clipping tool with a cube. Start by activating the tool with `Alt-B`. This will generate a dashed cross-hair cursor. Click with the `LMB` and drag out a rectangular region. Now clipping is applied against that region in 3D space. Use the `MMB` to rotate the view and you will see that only what is inside the clipping volume is visible. All the editing tools still function as normal, but only within the clipping volume.

The dark gray area is the clipping volume itself. Once clipping is deactivated with another `Alt-B`, all of 3D space will become visible again.

Render Region

Reference

Mode::	All modes
Menu::	*View* ▸ *View Regions* ▸ *Render Region…* *View* ▸ *View Regions* ▸ *Clear Render Region*
Shortcut::	Mark: `Ctrl-B` Clear: `Ctrl-Alt-B`

Allows you to limit rendering to a 2D rectangular area. If you're busy tweaking just a small part of the scene, it can be quite wasteful to have the whole viewport in Rendered shading mode or make full-frame renders, so this feature lets you save time.

You can define Render Regions in two different contexts:

Contextual Views

By default, the 3D Viewport only shows the scene from one viewpoint. By using Quad Views, you can see it from multiple viewpoints at the same time, which gives more context about the changes you're making.

Quad View

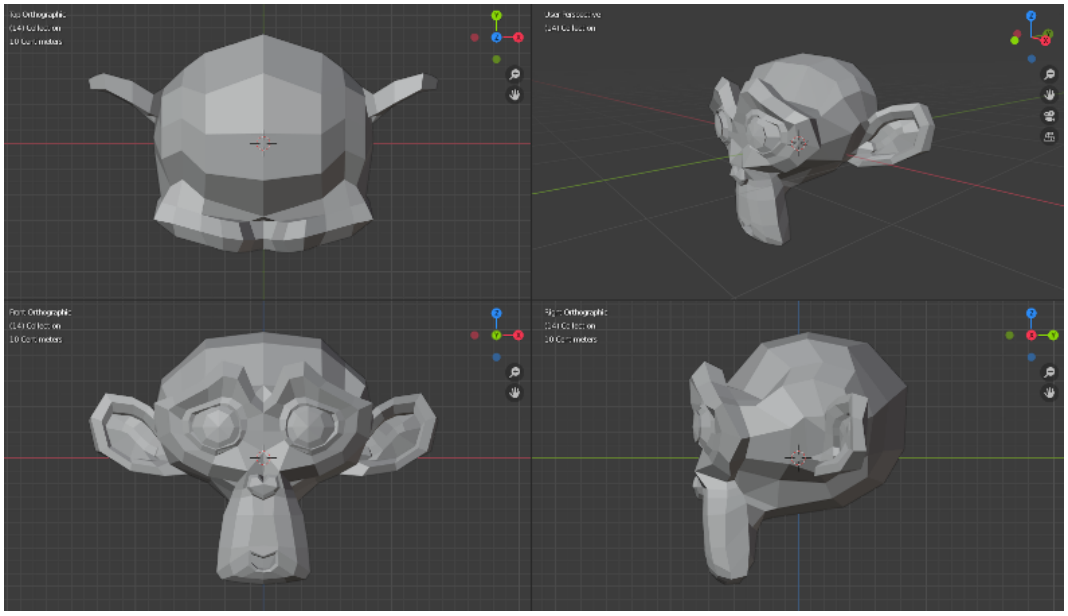
Reference

Mode::	All modes
Menu::	*View* ‣ *Area* ‣ *Toggle Quad View*
Shortcut::	`Ctrl-Alt-Q`

Toggling Quad View will split the 3D Viewport into four views: three orthographic side views and one user perspective view.

Note

Quad View is different from splitting the area and aligning the views manually. In Quad View, the four views are still part of a single 3D Viewport, so that they share the same display options.



Quad View.

Options

Reference

Mode:: All modes
Menu:: *Sidebar ▸ View ▸ Quad View*

Lock Rotation
: When disabled, makes it possible to orbit in the orthographic views as well (turning them into perspective views instead).

Sync Zoom/Pan
: Syncs the view position between side views. (Requires *Lock Rotation* to be enabled.)

Clip Contents
: Clip objects based on what is visible in the other side views.

3D Cursor

The 3D Cursor is a point in space that has both a location and a rotation. It's used for a number of purposes. For example, it defines where newly added objects are placed, and can also be used to manually position and orient the transform gizmo (see Pivot Point and Transform Orientation). Some tools, such as Bend, also use the Cursor.

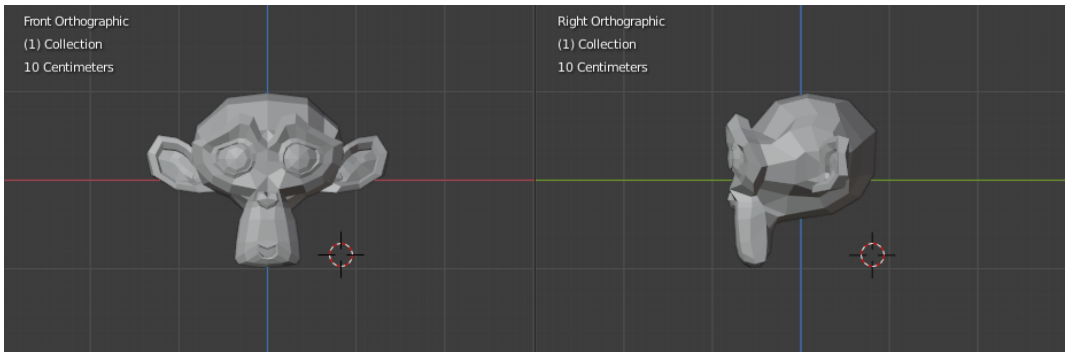
Placement

There are a few methods to position the 3D Cursor.

Direct Placement with the Mouse

Reference

Mode::	Object, Edit, and Pose Mode
Tool::	Cursor
Shortcut::	`Shift-RMB`



Positioning the 3D Cursor with two orthogonal views.

The Cursor tool offers the most flexibility. Simply select it in the Toolbar and click a point in the scene with `LMB` to place the 3D Cursor there. In the tool settings, you can choose how it should be oriented: by default, it

matches the view orientation, but you can also make it match the surface normal of a piece of geometry, or the transform orientation.

Alternatively, you can press `Shift-RMB` with any tool selected. In this case, the 3D Cursor will always be aligned to the view orientation.

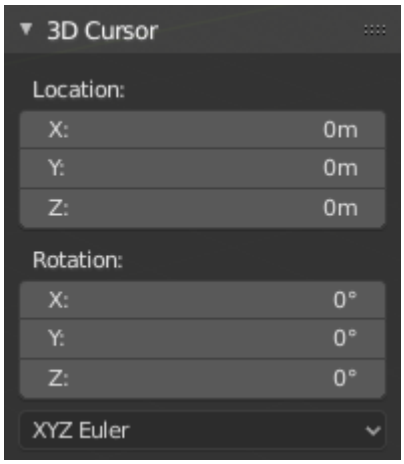
For accuracy you should use two perpendicular orthogonal 3D Viewports, i.e. any combination of top `Numpad7`, front `Numpad1` and side `Numpad3`. That way you can control the positioning along two axes in one view and determine the depth in the other.

By default, the depth of the geometry under the cursor is used. This can be disabled using the *Cursor Surface Project* toggle in the Preferences.

Sidebar

Reference

Mode:: All Modes
Panel:: *Sidebar region ▸ View ▸ 3D Cursor*



The 3D Cursor panel of the Sidebar region.

The 3D Cursor can also be positioned and oriented by editing the respective values in the Sidebar.

Snapping

Reference

Mode::	Object, Edit, and Pose Mode
Menu::	*Object*/*Mesh*/… ▸ *Snap* ▸ *Cursor to …*
Shortcut::	`Shift-S`

One more way of positioning the 3D Cursor is through the Snap menu, which allows you to move the Cursor to the origin of the selected object for example.

Selecting

This page discusses selection tools that are specific to the 3D Viewport. The generic selection tools are described in the Interface section.

The 3D Viewport has two keys that affect selection:

Select by Origin `Ctrl`
: Selects objects by their origin rather than their geometry.

Selection Menu `Alt`
: Shows a menu in case there are multiple objects under the mouse cursor, making it easier to select the one you want.

These keys can be combined to get a selection menu based on object origins.

The mode-specific selection pages are listed below.

Object Mode

- Object Mode

Edit Mode

- Mesh Edit Mode
- Curve Edit Mode
- Surface Edit Mode
- Metaball Edit Mode
- Text Edit Mode
- Grease Pencil Edit Mode
- Bone Edit Mode
- Lattice Edit Mode

Pose Mode

- Pose Mode

Particle Edit Mode

- Particle Edit Mode

Controls

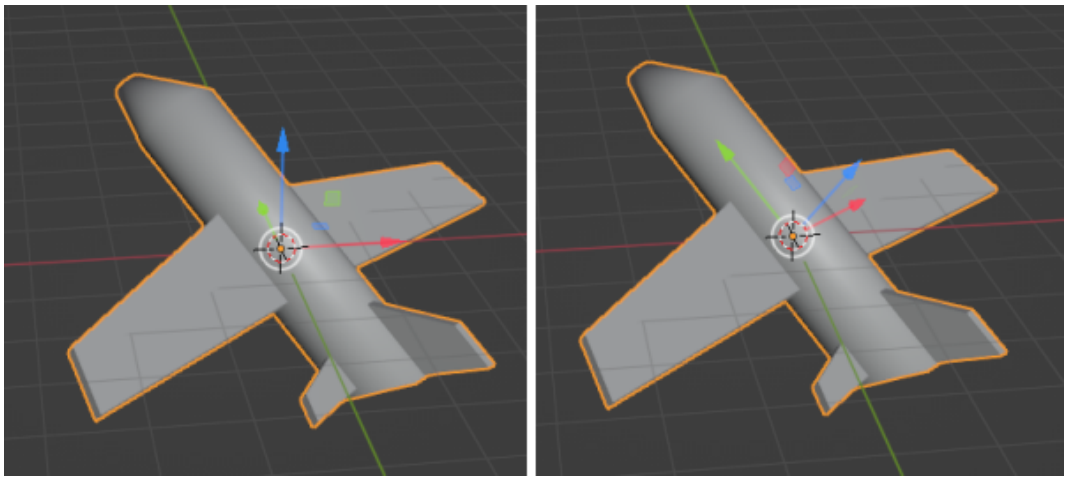
- Transform Orientation
  - Orientations
- Pivot Point
  - Pivot Types
- Snapping
  - Snap Base
  - Snap Target
  - Snap Target for Individual Elements
  - Target Selection
  - Affect
  - Rotation Increment
- Proportional Editing
  - Controls
  - Object Mode
  - Edit Mode
  - Example

Transform Orientation

Reference

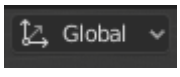
Mode::	Object and Edit Modes
Panel::	*Header ▸ Transform Orientation*
Shortcut::	`Comma`

The *Transform Orientation* determines the orientation of the Object Gizmo. Changing this orientation can make it easier to perform transformations in the direction you want.



With the default Global transform orientation (left) it's tricky to move the plane in the direction it's facing, but with Local (right) it's easy.

The Transform Orientation can be changed using a selector in the 3D Viewport's header:



Transform Orientation selector.

The orientation can also be changed temporarily while performing a hotkey-based transformation with axis locking. For example, if you first press `G` to start moving an object, then `X` to lock to the orientation's X axis, and finally `X` a second time, you'll get a lock to an alternative orientation: the Local orientation if it was Global previously, and the Global orientation otherwise.

In addition to the builtin orientations, you can also define your own (see Custom Orientations below).

Orientations

Global
: Align the transformation axes to world space. The world axes are shown by the Navigation Gizmo in the top right corner of the viewport, as well as the *Grid Floor*.

Local
: Align the transformation axes to the active object's orientation.

Normal
: In Edit Mode, orient the transformation axes so that the Z axis of the gizmo matches the average Normal of the selected elements.

  In Object Mode, this is equivalent to *Local* orientation.

Gimbal
: Orient the transformation axes to visualize the workings of the object's Rotation Mode. This is specifically useful for the Euler modes, where the object is rotated one axis at a time: the rotation axes don't stay perpendicular to each other and might even overlap, a phenomenon known as gimbal lock that complicates animation.

View
: Align the transformation axes to the view (meaning they change as you orbit around):

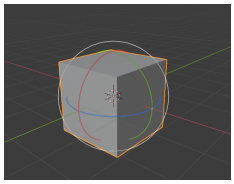
- X: Left/Right
- Y: Up/Down
- Z: Towards/Away from the screen

Cursor
: Align the transformation axes to the 3D Cursor.

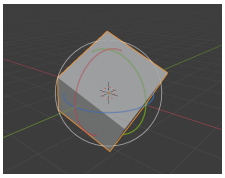
Parent
: Align the transformation axes to the Parent.

Examples

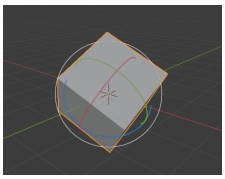
Cube with the rotation gizmo active in multiple transform orientations.



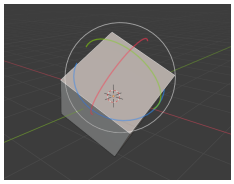
Default cube with Global transform orientation selected.



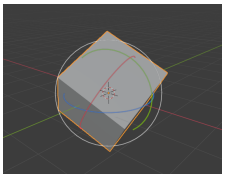
Rotated cube with Global orientation, gizmo has not changed.



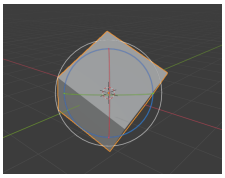
Local orientation, gizmo matches the object's rotation.



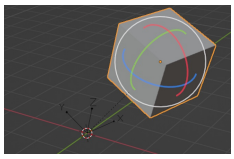
Normal orientation, in Edit Mode.



Gimbal transform orientation.



View transform orientation.



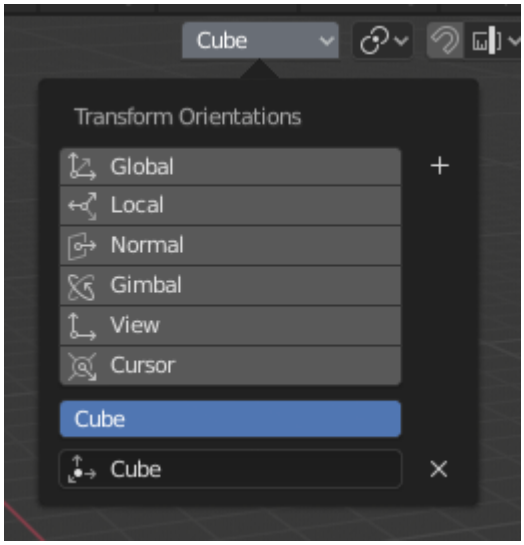
Parent transform orientation. Cube parented to rotated empty.

Custom Orientations

Reference

Mode::	Object and Edit Modes
Panel::	*Header ‣ Transform Orientation*

You can define custom transform orientations using objects or mesh elements. Custom orientations defined from an object use the *Local* orientation of that object, whereas those defined from mesh elements (vertices, edges, faces) use the average *Normal* orientation of those elements.



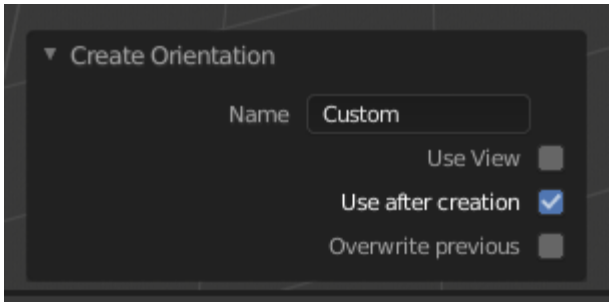
Transform Orientation panel.

The *Transform Orientation* panel, found in the header of the 3D Viewport, can be used to select, add, remove, and rename transform orientations.

The default name for these orientations is derived from the selection. If it's an object it will take that object's name, if it's an edge it will be titled "Edge", and so on.

Create Orientation

To create a custom orientation, select an object or mesh element(s) and click the "+" button in the *Transform Orientation* panel.



Create Orientation Adjust Last Operation panel.

Right after creating the orientation, the *Create Orientation* Adjust Last Operation panel gives a few options:

Name
: Text field for naming the new orientation.

Use View
: The new orientation will be aligned to the view space.

Use After Creation
: The new orientation stays selected.

Overwrite Previous
: If the new orientation is given an existing name, a suffix will be added to it's name to avoid overwriting the existing orientation, unless *Overwrite Previous* is checked, in which case it will be overwritten.

Delete Orientation

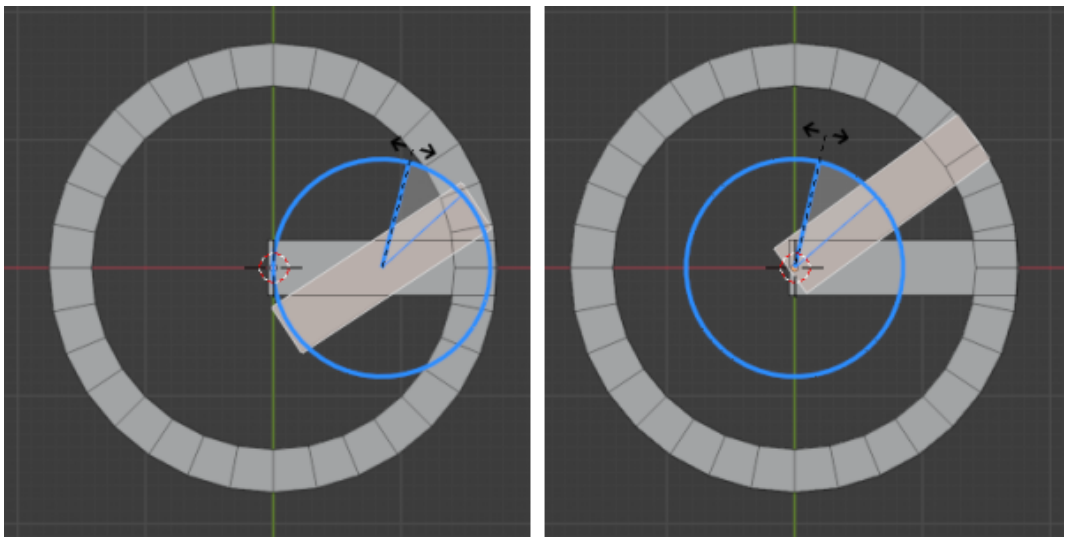
To delete a custom orientation, simply select it and click the × button.

Pivot Point

Reference

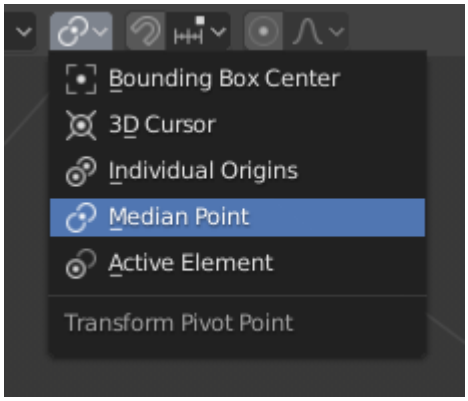
Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point*
Shortcut::	`Period`

The *Pivot Point* determines the location of the Object Gizmo. Changing this location can make it easier to perform transformations around the point you want.



With the default “Median Point” pivot point (left) it’s tricky to bring the second wheel spoke into place, but with “3D Cursor” (right) it’s easy.

The Pivot Point can be changed using a selector in the 3D Viewport’s header:



Pivot Types

- Bounding Box Center
  - In Object Mode
  - In Edit Mode
- 3D Cursor
  - Example
- Individual Origins
  - In Object Mode
  - In Edit Mode
- Median Point
  - In Object Mode
  - In Edit Mode
- Active Element
  - In Object Mode
  - In Edit Mode

Bounding Box Center

Reference

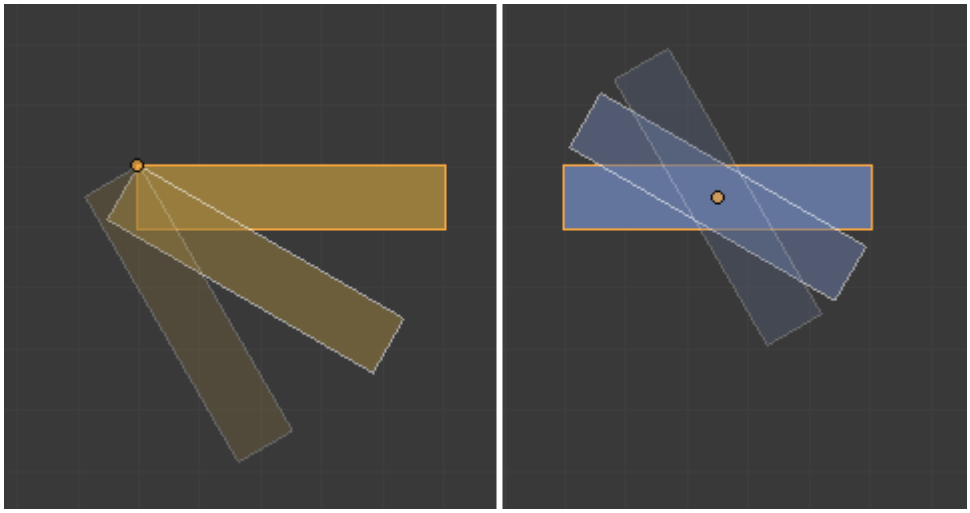
Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point* ▸ *Bounding Box Center*
Shortcut::	`Period`

In this mode, the pivot point lies at the center of the bounding box, which is a box that's wrapped as tightly as possible around the selection while still being aligned to the world axes.

In Object Mode

The pivot point becomes the center of the bounding box *around the selected objects' origin points*, not their geometry.

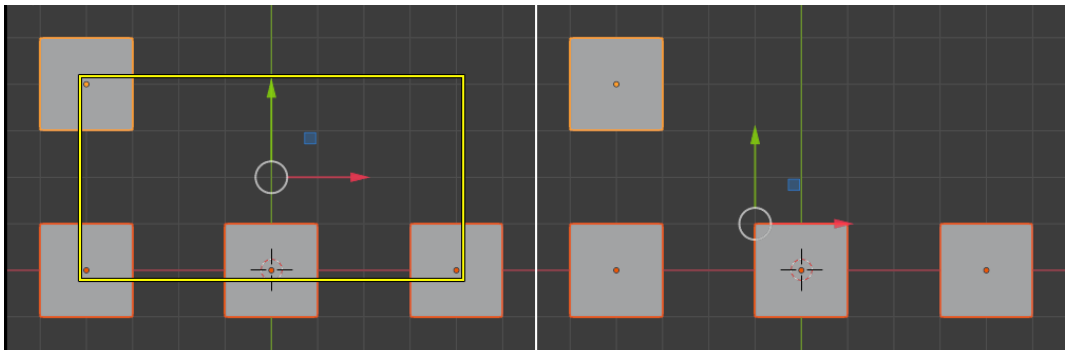
This means that, if you have a single object selected, the pivot point is the same as the object's origin point – which can be customized and doesn't have to be in the center. In the example below, the orange rectangle has it in a corner instead.



Single object rotation.

If you have multiple objects selected, the pivot point becomes the center of an imaginary box around their origins.

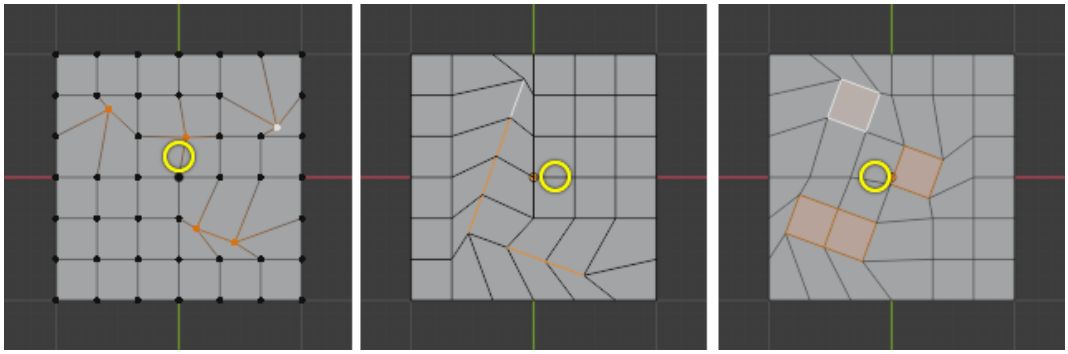
The image below shows the difference between Bounding Box Center and Median Point. The latter calculates the average position of the origins, meaning that the pivot point shifts towards the area with the most objects.



Difference between “Bounding Box Center” (left) and “Median Point” (right).

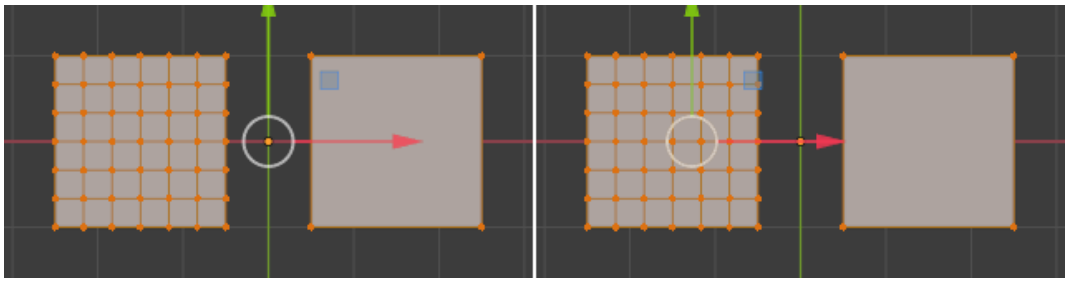
In Edit Mode

The pivot point becomes the center of the bounding box around the selected mesh elements.



The effects of rotation in different mesh selection modes. The pivot point is shown by a yellow circle.

Median Point may again give a different result.



Difference between “Bounding Box Center” (left) and “Median Point” (right).

3D Cursor

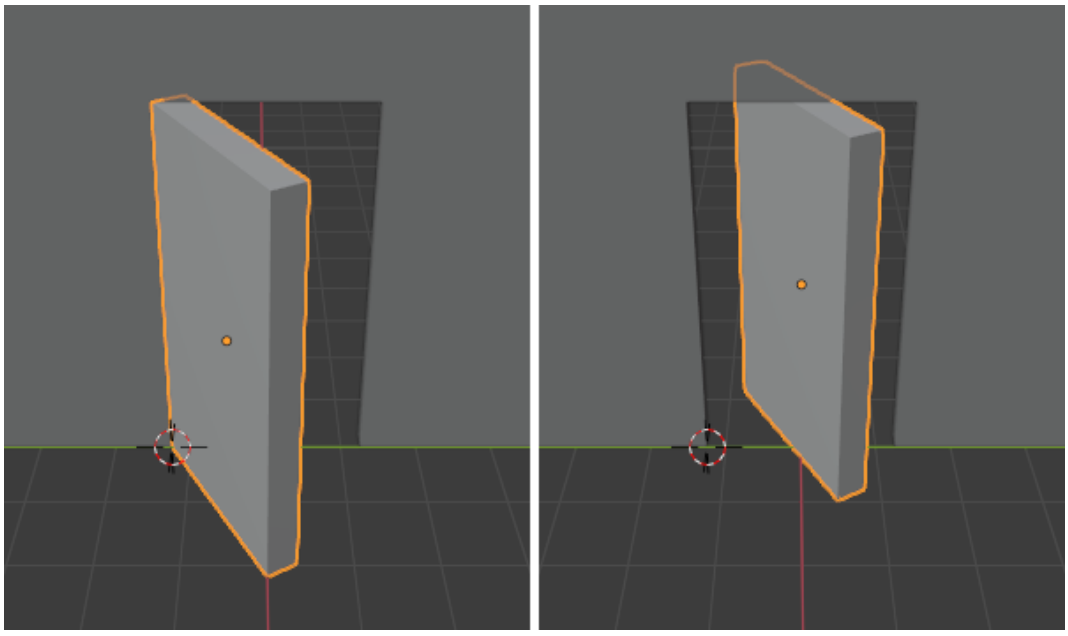
Reference

Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point* ▸ *3D Cursor*
Shortcut::	`Period`

Places the pivot point at the location of the 3D Cursor.

Example

The image below shows the difference between rotating an object around the 3D Cursor (left) and rotating it around the Median Point (right).



Rotation around the 3D Cursor compared to the Median Point.

Individual Origins

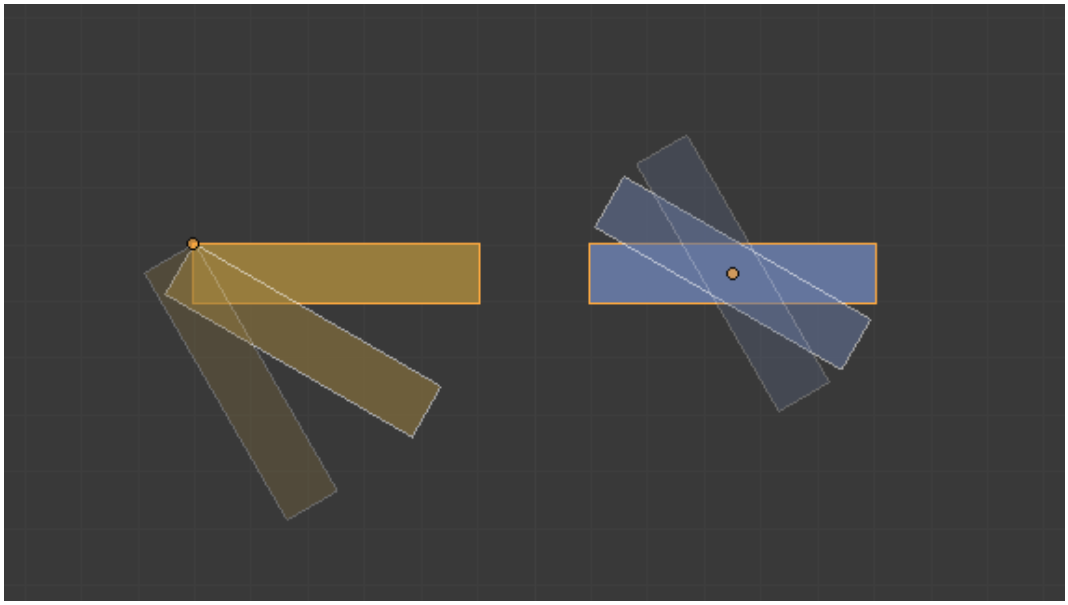
Reference

Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point* ▸ *Individual Origins*
Shortcut::	`Period`

While the other pivot point modes transform the whole selection around one point, Individual Origins transforms each item around itself.

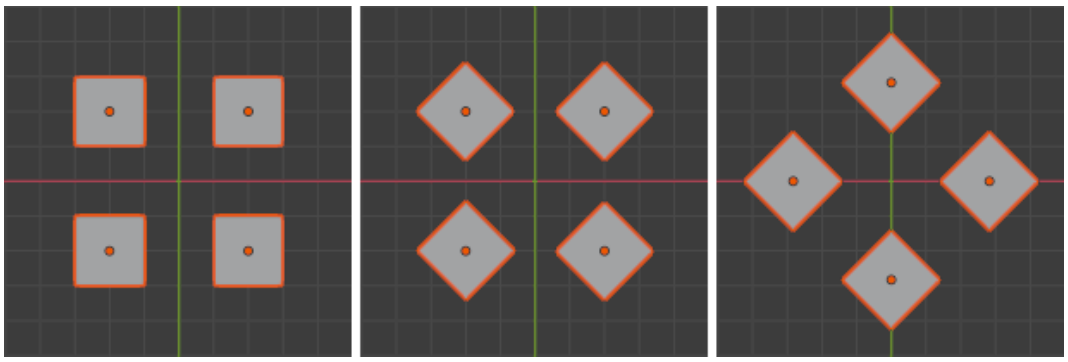
In Object Mode

Each object gets transformed around its origin, which is a point that can be chosen freely and doesn't have to be in the center. In the example below, the orange rectangle has it in a corner instead.

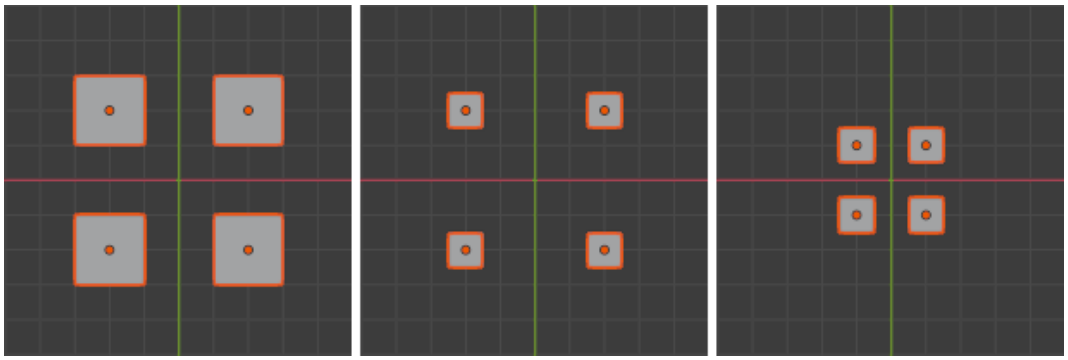


Rotation around individual origins.

The images below compare Individual Origins to Median Point.



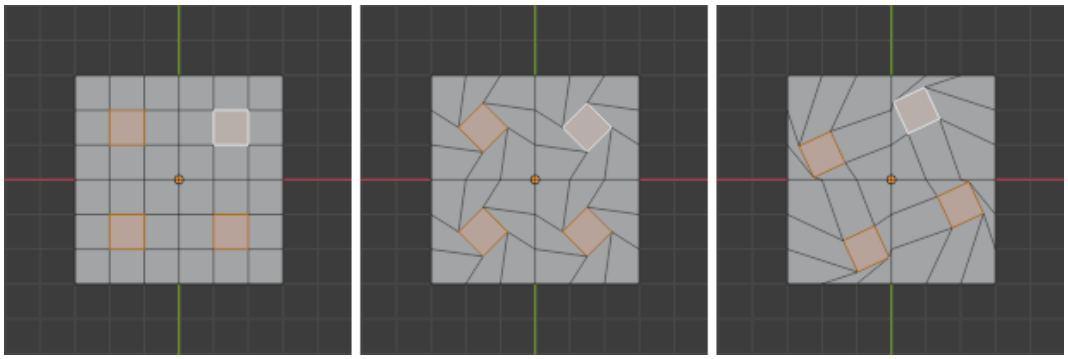
Starting situation, rotation around Individual Origins, rotation around Median Point.



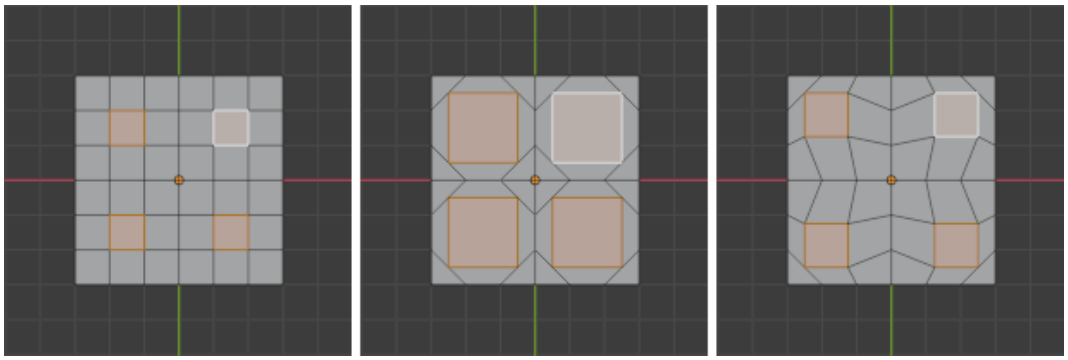
Starting situation, scaling using Individual Origins, scaling using Median Point.

In Edit Mode

Each selected element is transformed around its own centerpoint.



Starting situation, rotation around Individual Origins, rotation around Median Point.



Starting situation, scaling using Individual Origins, scaling using Median Point.

When you transform adjacent faces or edges, they are treated as a single element (meaning they don’t become disconnected).

Median Point

Reference

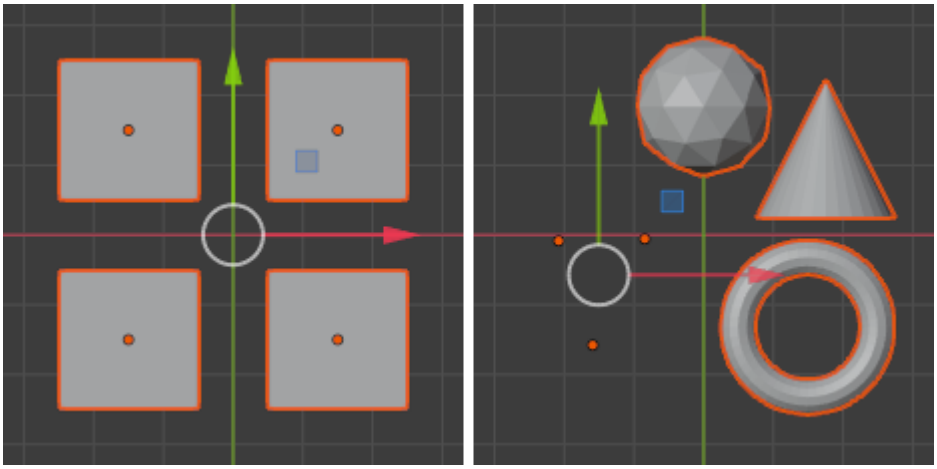
Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point* ▸ *Median Point*
Shortcut::	`Period`

Places the pivot point at the averaged-out position of the selected items.

In Object Mode

In Object Mode, the Median Point is the averaged-out position of the origins of the selected objects. The shape and size of the objects is not taken into account.

Origins can be chosen freely and can even lie outside their object's geometry, so that the Median Point is not always what you might expect.

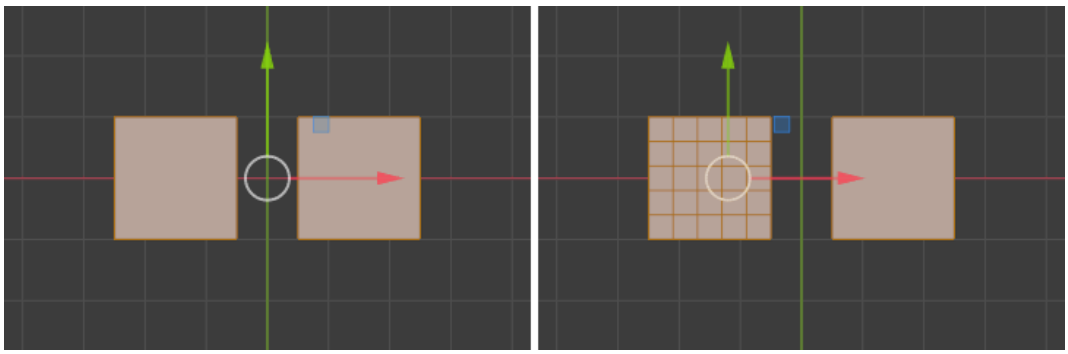


Median points in Object Mode.

In Edit Mode

In Edit Mode, the Median Point is the averaged-out position of the selected vertices. This means that the pivot point will shift towards the area with the densest geometry.

In the example below, the pivot point lies perfectly in the middle if both cubes have the same number of vertices, but heavily leans towards the side if one cube is subdivided – even though both cubes still have the same size.



Median points in Edit Mode.

Active Element

Reference

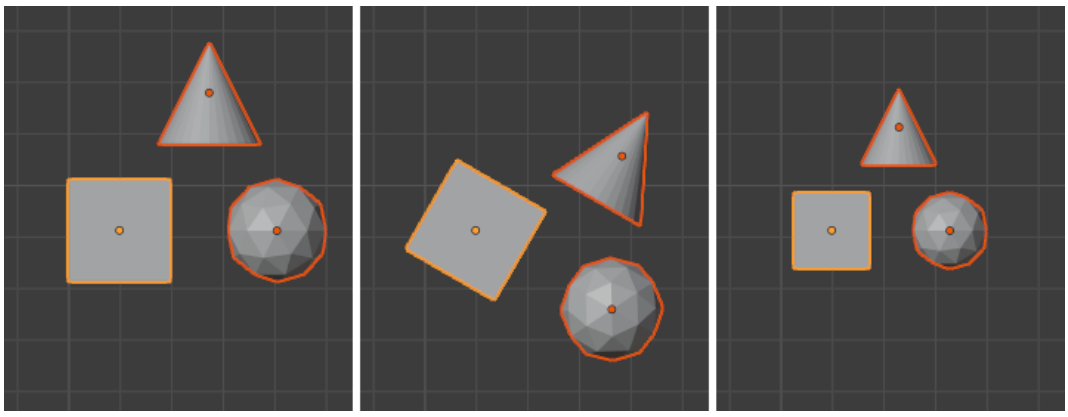
Mode::	Object Mode and Edit Mode
Header::	 *Pivot Point* ▸ *Active Element*
Shortcut::	`Period`

Places the pivot point at the active element, which is the element that was selected most recently.

In Object Mode

When in Object Mode, rotation and scaling happen around the origin of the active object, which is the element with a lighter outline than the others.

The effect of the pivot point is shown in the image below, where the active object (the cube) remains in the same location while the others move.

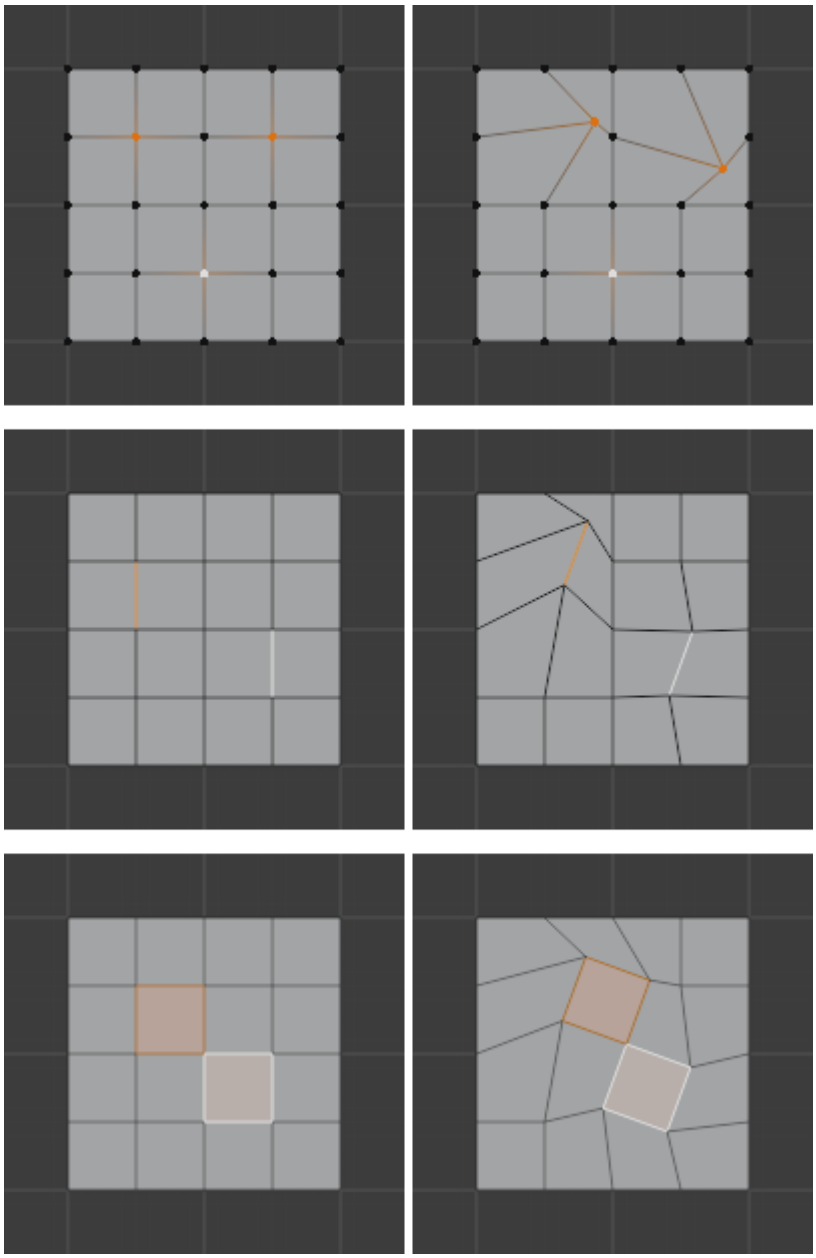


Starting point, rotation, and scaling.

In Edit Mode

When in Edit Mode, rotation and scaling happen around the centerpoint of the active element, which is the element with a white outline. That centerpoint remains in the same location while everything else is transformed around it.

The image below illustrates rotation around the active vertex, edge, and face. Each time, the active element rotates in place, while the others “orbit” around it. In the case of vertices, the active vertex is not changed at all, as a vertex on its own is just a point that has no concept of rotation.



Left column: starting situation, right column: after rotation.

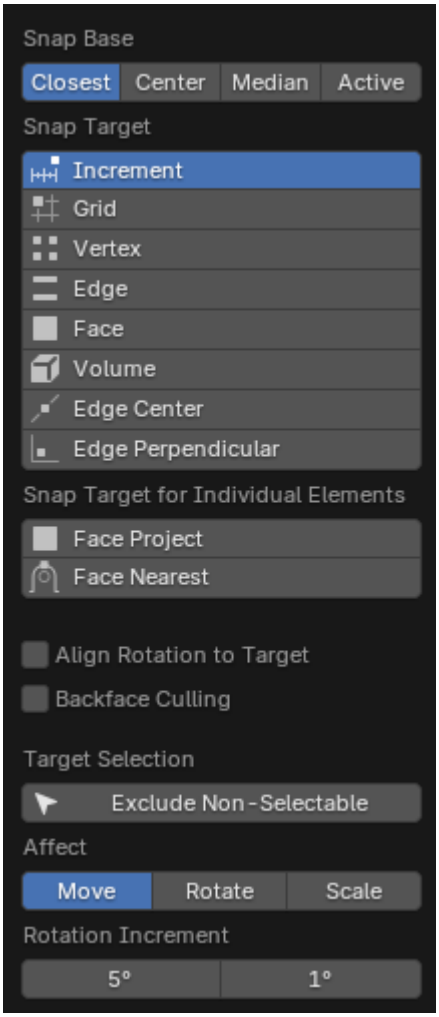
Snapping

Reference

Mode:: Object, Edit, and Pose Mode
Header:: *Snap* (

)
Shortcut:: `Shift-Tab`

Snapping lets you easily align objects and mesh elements to others. It can be toggled by clicking the magnet icon in the 3D Viewport's header, or more temporarily by holding `Ctrl`.



Snap menu.

See also

Transform operators have their own snapping operators, see Transform Modal Map

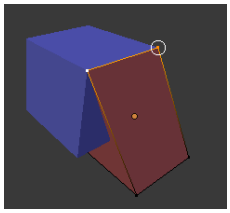
Snap Base

Reference

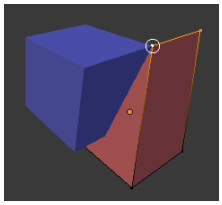
Mode:: Object, Edit, and Pose Mode
Header:: *Snapping* ▸ *Snap Base*
Shortcut:: `Shift-Ctrl-Tab`

Determines which point in the geometry is the snap base that will snap to the target.

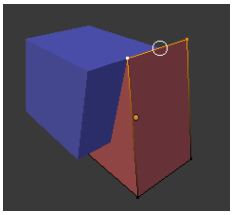
Active:: Snaps using the origin (in Object Mode) or center (in Edit Mode) of the active element.
Median:: Snaps using the median of the selection.
Center:: Snaps using the current transformation center (another word for the pivot point). This option is especially useful in combination with the 3D Cursor for choosing the snapping point completely manually.
Closest:: Snaps using the vertex that's closest to the target.



Closest.



Active.



Median.

Snap Target

Reference

Mode:: Object, Edit, and Pose Mode
Header:: *Snapping* ▸ *Snap Target*

Shortcut:: `Shift-Ctrl-Tab`

Determines the target which the selection will be snapped to.

Increment:: Snaps to grid points. When in Orthographic view, the snapping increment changes depending on the zoom level.
This option snaps to an imaginary grid that starts at the selection's original location and has the same resolution as the viewport grid. In other words, it lets you move the selection in "increments" of the grid cell size.

Grid:: Snaps to the grid that's displayed in the viewport.

Vertex:: Snaps to the vertex that's closest to the mouse cursor.

Edge:: Snaps to the edge that's closest to the mouse cursor.

Volume:: Snaps the selection to a depth that's centered *inside* the object under the cursor. This is useful for positioning an Armature bone so it's centered inside a character's arm, for example; the other snapping options would place it on the arm's surface instead. While Blender also has Volume objects, this option is not related to those.

Edge Center:: Snaps to the centerpoint of the edge that's closest to the mouse cursor.

Edge Perpendicular:: Snaps to a specific point on the edge so that the line from the selection's original location (indicated by a white cross) to its new location is perpendicular to that edge.

Tip

Multiple snapping modes can be enabled at once using `Shift-LMB`.

Snap Target for Individual Elements

Reference

Mode::	Object, Edit, and Pose Mode
Header::	*Snapping* ▸ *Snap Target for Individual Elements*
Shortcut::	`Shift-Ctrl-Tab`

Type of element for individual transformed elements to snap to.

Face Project:: Snaps to the face that's under the mouse cursor. This can be used for bending a flat sheet so it snugly fits against a curved surface, for example.
This works similar to the Shrinkwrap Modifier.

Face Nearest:: Individually snaps each object (in Object Mode) or vertex (in Edit Mode) to the face that's closest to its new location. This makes it possible to snap to occluded geometry.

Target Selection

Sets more detailed snapping options. The available options depend on the mode (Object/Edit) as well as the Snap Target.

Include Active *Edit Mode*
: Snap to other mesh elements of the active object.

  This checkbox is ignored if Proportional Editing is enabled.

Include Edited *Edit Mode*
: Snap to other objects that are also in Edit Mode.

Include Non-Edited *Edit Mode*
: Snap to other objects that are not in Edit Mode.

Exclude Non-Selectable

Snap only to objects that are selectable.

Align Rotation to Target
: Rotates the selection so that its Z axis gets aligned to the normal of the target.

Backface Culling
: Exclude back-facing geometry from snapping.

Snap to Same Target *Face Nearest*
: Snap only to the object which the selection was nearest to before starting the transformation.

Face Nearest Steps *Face Nearest Edit Mode*
: Breaks the overall transformation into multiple steps, performing a snap each time. This can give better results in certain cases.

Snap Peel Object *Volume*
: If the target object is composed of several disconnected mesh islands that intersect each other, “Snap To Volume” will normally snap to the island which the mouse is hovering over, ignoring the other islands. By enabling “Snap Peel Object,” you can instead treat the target object as one connected whole.

Affect

Specifies which transformations are affected by snapping. By default, snapping only happens while moving something, but you can also enable it for rotating and scaling.

Rotation Increment

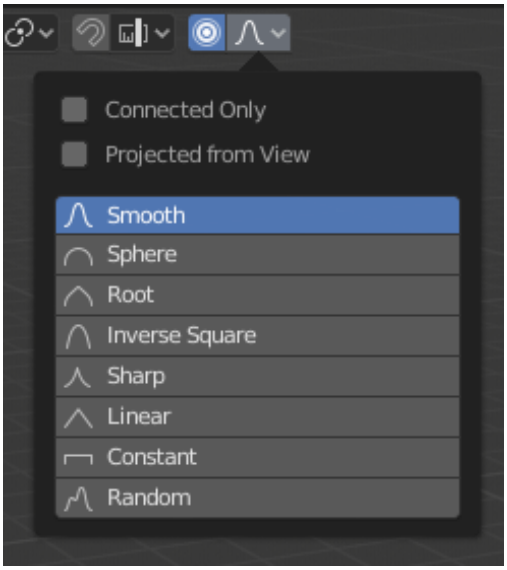
Angle used in incremental snapping for the rotation operator. The second value is the *Rotation Precision Increment*, used for finer transformations and activated by default with the `Shift` key.

Proportional Editing

Reference

Mode:: Object and Edit Mode
Header::  *Proportional Editing*
Shortcut:: O

Proportional Editing is a way of transforming selected elements while also affecting the nearby unselected elements. The farther away an unselected element is, the less it will be affected (hence the "proportional"). This feature is very useful for smoothly deforming dense meshes.



Proportional Editing popover.

Note

Blender also has a Sculpting workflow that contains brushes and tools for proportionally editing a mesh without seeing the individual vertices.

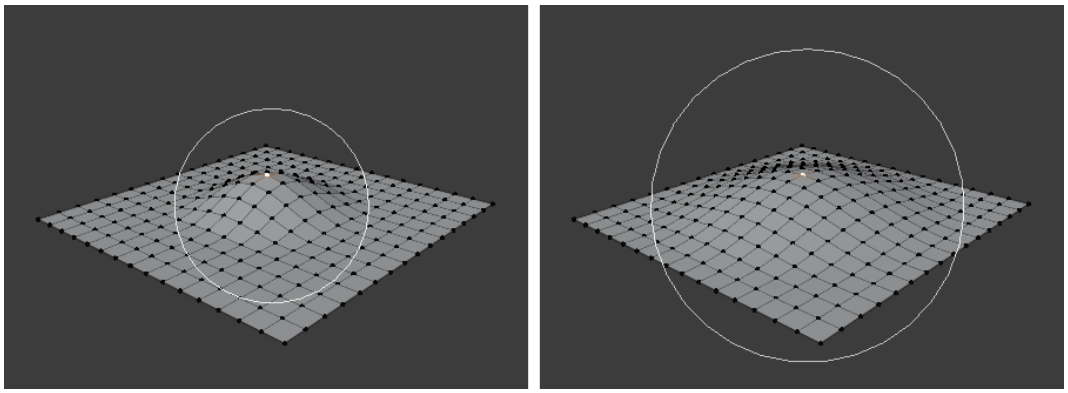
Controls

Disable O
: Proportional Editing is off, only selected vertices will be affected.

Enable O
: Vertices other than the selected vertex are affected, within a defined radius.

Proportional Size

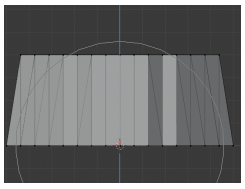
You can increase or decrease the radius of the tool's influence during a transform operation from the Proportional Editing popover or with `WheelUp`/`WheelDown`, or `PageUp`/`PageDown` respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly.



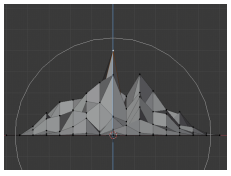
Influence circle.

Falloff

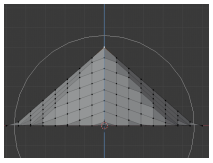
While editing, you can change the curve profile by either clicking the *Falloff* icon in the header or pressing `Shift-O` to get a pie menu.



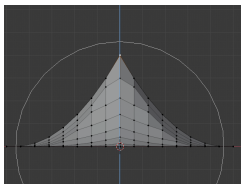
Constant, No Falloff.



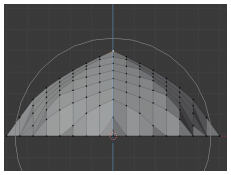
Random Falloff.



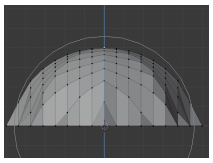
Linear Falloff.



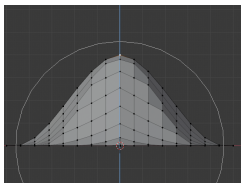
Sharp Falloff.



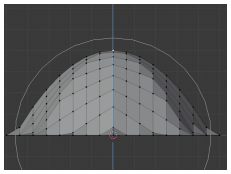
Root Falloff.



Sphere Falloff.



Smooth Falloff.

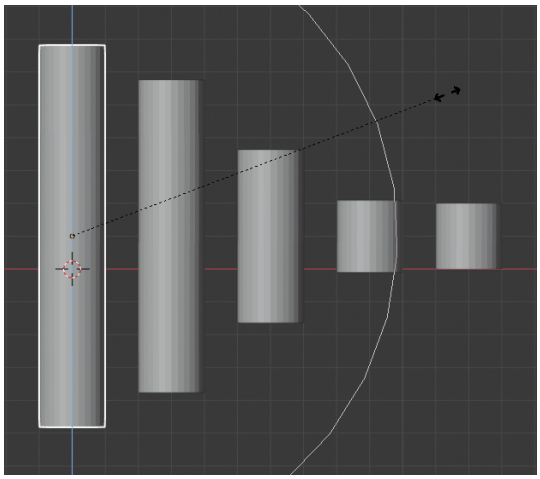


Inverse Square Falloff.

Object Mode

Proportional Editing is typically used in *Edit Mode*, but it can also be used in *Object Mode*. The tool then works on entire objects rather than individual mesh components.

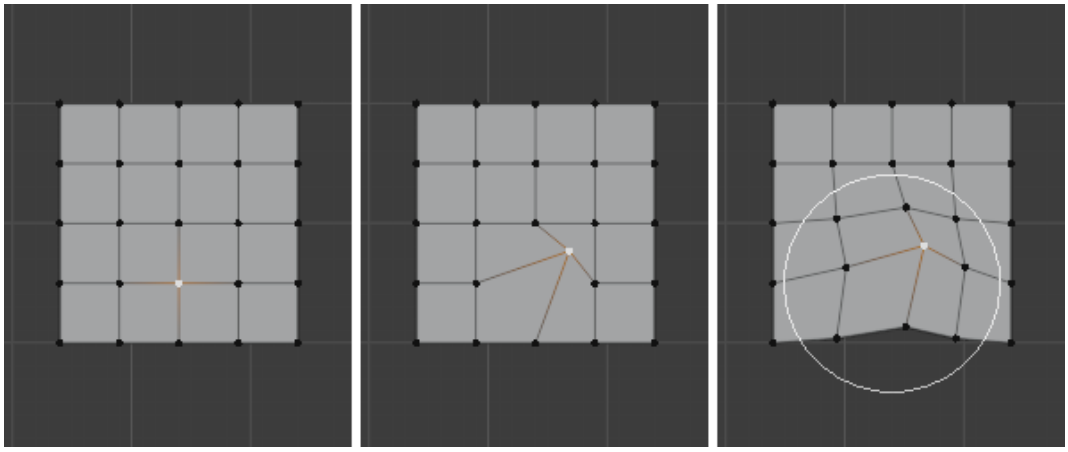
In the image below, the leftmost cylinder is being scaled up vertically, which also affects the cylinders near it.



Proportional Editing in Object Mode.

Edit Mode

When working with dense geometry, it can become difficult to make subtle adjustments without causing visible lumps and creases in the model's surface. When you face situations like this, Proportional Editing can help.



Proportional Editing in Edit Mode.

Options

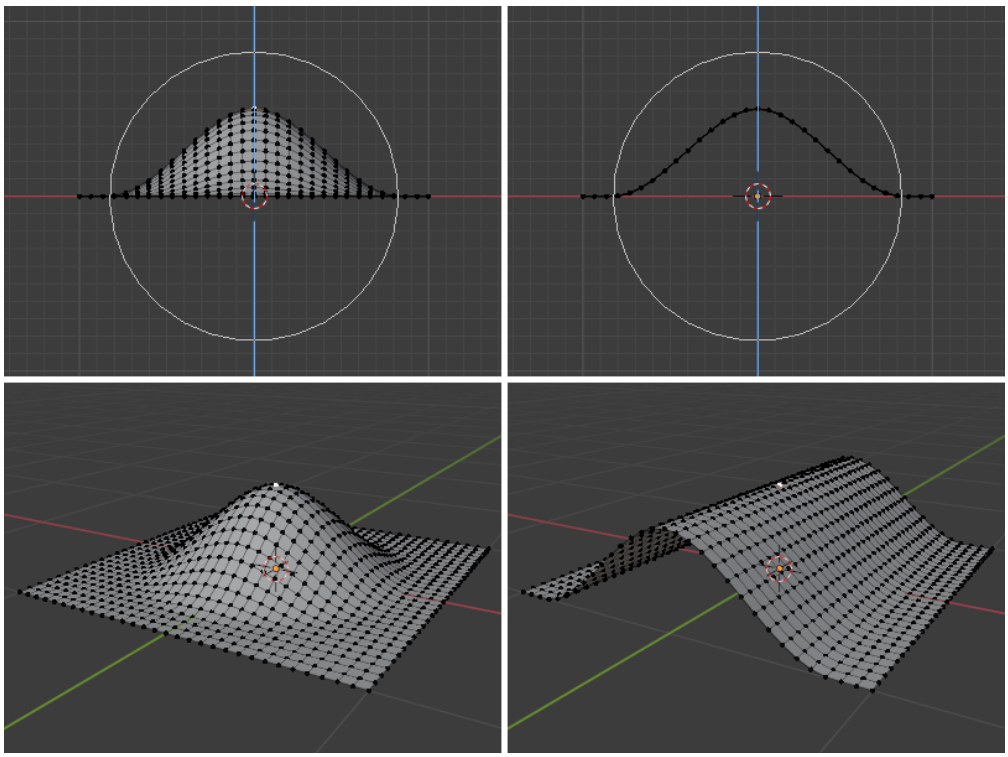
Connected Only `Alt-O`

Rather than using a radius only, the proportional falloff spreads via connected geometry. This means that you can proportionally edit the

vertices in a finger of a hand without affecting the other fingers. While the other vertices are physically close (in 3D space), they are far away following the topological edge connections of the mesh. The icon will have a blue center when *Connected* is active. This mode is only available in *Edit Mode*.

Projected from View

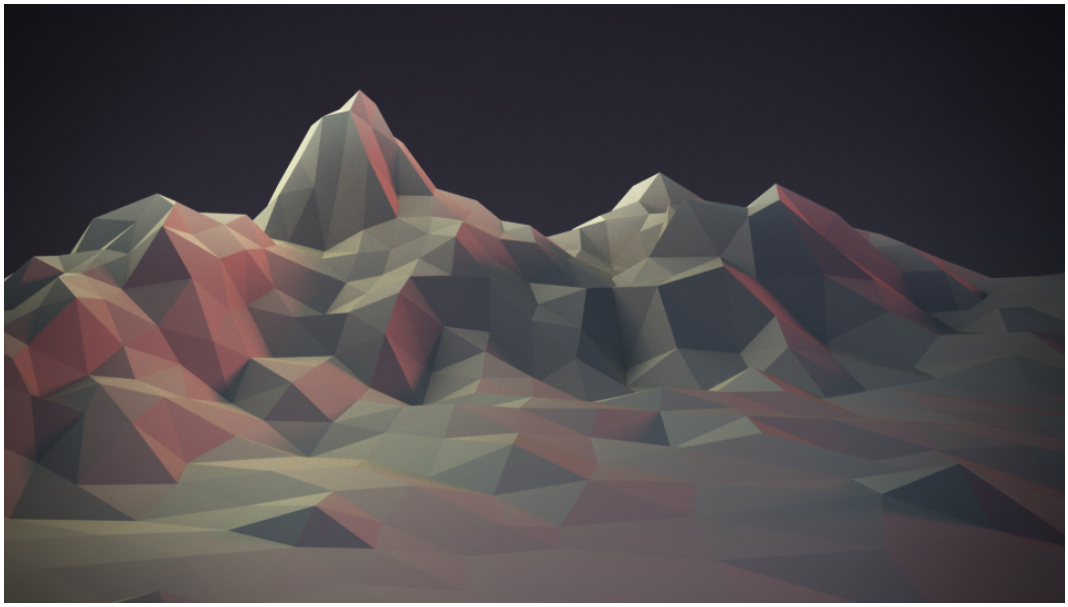
Depth along the view is ignored when applying the radius.



The difference between having “Projected from View” disabled (left) and enabled (right).

Example

The image below shows the final render of a low-poly landscape obtained by moving up the vertices of a triangulated grid with *Proportional Editing* enabled.



A landscape obtained via Proportional Editing.

Display

- Object Type Visibility
- Viewport Gizmos
  - Viewport Gizmos
  - Object Gizmos
  - Empty
  - Light
  - Camera
- Viewport Overlays
  - General
  - Mesh Edit Mode Overlays
  - Sculpt Mode Overlays
  - Vertex Paint Overlays
  - Weight Paint Overlays
  - Texture Paint Overlays
  - Pose Mode Overlays
  - Grease Pencil
- Viewport Shading
  - Wireframe
  - Solid
  - Material Preview
  - Rendered

Object Type Visibility

Reference

Mode:: All Modes
Header::  *View Object Types*

This popover lets you control the visibility and selectability of the various object types. For example, you can hide all the Lights in the scene with one click.

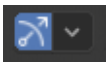
The settings only apply to the current 3D Viewport. Object types marked as unselectable can still be selected in other viewports and in the Outliner, for example.

See also

Outliner Restriction Columns for making individual objects invisible or unselectable across all viewports.

Viewport Gizmos

Reference

Mode:: All Modes
Header::  *Gizmos*

Clicking the icon toggles all gizmos in the 3D Viewport. The drop-down button displays a popover with more detailed settings, which are described below.

Viewport Gizmos

Navigate
: Enable/disable the navigation gizmo.

Active Tool
: Enable/disable the gizmo of the active tool.

Active Object
: Enable/disable the Object Gizmos for the active element (see below).

Object Gizmos

Object Gizmos allow mouse-controlled translation, rotation and scaling in the 3D Viewport. While they're called "object" gizmos in the popover, they also apply to other transformable elements such as mesh vertices.

There is a separate gizmo for each operation. Each gizmo can be used separately or in combination with the others.

A gizmo always has three color-coded axes: X (red), Y (green), and Z (blue). You can drag an axis with `LMB` to transform along it. The Move and Scale gizmos additionally have small colored squares for transforming along two axes in one go.

Various modifier keys can be used:

- Holding `Ctrl` at any time will toggle snapping and also make rotation and scaling work in coarse increments.
- Holding `Shift` *after* pressing `LMB` will do the opposite of the above, "slowing down" the transformation relative to mouse movement to allow finer adjustments.
- Holding `Shift` *before* pressing `LMB` will perform the transformation in the plane that's perpendicular to the clicked axis. See Plane Locking.

The Gizmos popover has the following settings for object gizmos:

Orientation
: The orientation to use for the gizmo. *Default* means to use the viewport's Transform Orientation. The other options override it.

Move
: Show the gizmo to control the location. Dragging the small white circle allows free movement in the viewing plane.

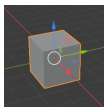
Rotate
: Show the gizmo to control the rotation. Dragging the large white circle allows rotation around the viewing direction. Dragging the translucent white disc within that circle (only visible when hovering over the gizmo) allows trackball rotation.

Scale
: Show the gizmo to control the scale. Dragging the area between the small and large white circles scales along all three axes.

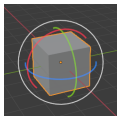
The latter three options are also available in a pie menu if you have the *Grave Accent / Tilde Action* in the Keymap Preferences set to *Gizmos*.

Note

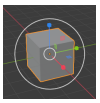
If you're using a tool that's tied to a particular gizmo setup (the *Move*, *Rotate*, *Scale* and *Transform* tools), the Move/Rotate/Scale checkboxes won't have any effect.



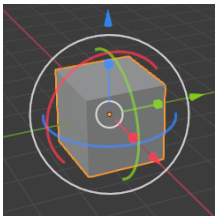
Move.



Rotate.



Scale.



Combination.

See also

The Gizmo Preferences.

Empty

Gizmo settings for empties.

Image
: Show the gizmo to adjust the image size and position.

Force Field
: Show the gizmo to adjust the force field.

Light

Gizmo settings for lights.

Size
: Show the gizmo to adjust the Spot Size of spotlights.

Look At
: Show the gizmo to adjust the direction of lights.

Camera

Gizmo settings for cameras.

Lens
: Show the gizmo to adjust the focal length (for Perspective cameras) or orthographic scale (for Orthographic cameras).

Focus Distance
: Enable the gizmo for adjusting the focus distance. To see this gizmo, you need to enable the *Viewport Display ▸ Limits* checkbox in the camera's properties (green camera icon).

Viewport Overlays

Reference

Mode:: All Modes
Header::  *Overlays*

Clicking the icon toggles all overlays in the 3D Viewport.

Note

Cameras outline & passepartout are not considered Viewport overlays.

The drop-down button displays a popover with more detailed settings, which are described below.

Depending on the current object interaction mode, there may be a second button with yet more settings, which are also described here.

General

The following options are always present, independent of the current mode. Some of the overlays can be customized in the Viewport Preferences.

Guides

Grid
: Show grid in orthographic side view.

Floor
: Show the ground plane in perspective view.

Axes

Show the X, Y and/or Z axis lines.

Scale
: The distance between lines in the grid/floor.

Subdivisions
: The number of subdivisions between grid lines.

Text Info
: Show various bits of information in the top left corner of the viewport.

- **View Perspective** – Name of the View Perspective, such as "Top Orthographic" or "User Perspective."
- **Playback Frame Rate (FPS)** – Displays the Frames Per Second at which the animation is playing. By default, Blender goes through every single frame, which may result in an FPS that's lower than intended (and the animation playing slower than realtime); the FPS turns red in this case. You can change this behavior in the Playback popover of the Timeline.
- **Object Info** – Shows the current frame in parentheses, followed by the names of the selected Collection and the active object. When applicable, also shows the selected Shape Key and (in angle brackets) the Marker on the current frame. If the object has a keyframe on the current frame, the Object Info is displayed in yellow.
- **Grid Resolution** – When the view is aligned to a world axis (see Viewpoint), the Text Info additionally shows the smallest distance between two parallel grid lines.

Statistics
: Show information about the amount of objects and geometry. Note that the counters depend on the current selection. For example, selecting a mesh gives info on the number of vertices, edges, and faces, while selecting a light shows the number of lights in the scene.

- **Objects** – Number of the selected objects and the total count.
- **Geometry** – Displays information about the current scene depending on the mode and object type. This can be the number of vertices, faces, triangles, or bones.

Camera Guides
: Show Camera guides (Safe Areas & Composition Guides), only available in camera view.

HDRI Preview
: Show two spheres, one glossy and one diffuse, to preview the HDRI that's being used for world lighting. While HDRIs can be used in both the *Material Preview* and *Rendered* shading modes, the HDRI Preview overlay is only available in the former.

3D Cursor
: Show the 3D Cursor.

Annotations
: Show annotations.

Objects

Extras
: Show objects that don't have geometry (such as empties, cameras and lights).

Light Colors
: Shades the outline of light objects to the color the light produces.

Relationship Lines
: Show dashed lines indicating parent or constraint relationships.

Outline Selected
: Show an outline around selected objects.

Bones
: Show Bones.

Motion Paths
: Show the motion path overlay.

Origin

Show the origins of the selected objects.

Origin (All)
: Show the origins of all objects.

Bone Wireframe Opacity
: The maximum opacity used for bones drawn in the *Wireframe* shading mode (or in *Solid* shading mode with X-Ray active). This is helpful when it is necessary to reduce clutter and focus on the mesh rather than bones.

Geometry

Wireframe
: Display mesh edges. Similar to Wireframe Shading, but displays edges on top of existing shading. The value slider adjusts which edges to display: lower values hide edges on surfaces that are almost flat, while a value of 1 shows all edges.

  Opacity
  : The opacity of the displayed edges, from 0 (invisible) to 1 (fully opaque).

Fade Inactive Geometry
: In modes other than Object Mode, fade out objects that you're not working on. The slider controls how much they're faded out.

Face Orientation
: Show faces whose normal is pointing towards the camera in blue, and faces whose normal is pointing away from the camera in red. This lets you quickly check for faces that are oriented incorrectly: the outside surface of an object should typically be all blue.

Viewer Node

Visualizes Attributes connected to the Viewer Node.

Viewer Node
: Visualize the value of the attribute connected to the *Viewer Node* as a greyscale color.

  Color Opacity
  : Opacity of the attribute that is currently visualized.

Attribute Text
: Show attribute values as text in viewport.

Motion Tracking

Show the motion tracking overlay.

Camera Path
: Show the reconstructed camera path.

Marker Names
: Show the names for reconstructed track objects.

Tracks
: Change the display of the reconstructed tracks: plain axes, arrows and so on.

Size
: Change the display size of the reconstructed tracks.

Mesh Edit Mode Overlays

The following options are available when in Mesh Edit Mode.

Faces
: Highlight selected faces. Affects all selection modes.

Center
: Show face center points in solid shading modes. (They're always shown in wireframe shading mode.)

Only affects face selection mode.

Creases
: Display edges marked with a crease for the Subdivision Surface Modifier.

Sharp
: Display sharp edges, used with the Edge Split modifier.

Bevel
: Display weights created for the Bevel Modifier.

Seams
: Display the UV unwrapping seams.

Shading

Retopology
: This overlay is useful when you have a sculpted mesh with the desired shape and want to recreate it with better topology. It makes the edited mesh see-through (so that you can see the sculpted mesh underneath it) and optionally renders it in front of nearby geometry (so that you can see it underneath the sculpted mesh).

  Offset
  : Distance to "move the edited mesh towards the camera." Use this to display the mesh in front of other objects that would normally occlude it.

Vertex Groups Weights
: Visualize the weights of the active vertex group, much like in Weight Paint mode.

  Zero Weights
  : Display unreferenced and zero-weighted areas in black. This helps to identify areas with very low weights that have been painted onto.

    None:: Vertices are displayed in the usual way.

Active::	Vertices are shown in black if they have no weight in the active vertex group.
All::	Vertices are shown in black if they have no weight in any vertex group.

Mesh Analysis

Show the Mesh Analysis overlay.

Measurement

Show numerical measures of the selected elements. The Units can be set in the Scene properties.

Edge Length
: Show the length of selected edges.

Edge Angle
: Show the angle of selected edges between two faces.

Face Area
: Show the area of selected faces.

Face Angle
: Show the angle of selected face corners.

Tip

Geometry connected to the selection is shown while transforming, allowing you to move a vertex and see the connected edge lengths for example.

Note

These values respect the Transform Space in the Sidebar. Use *Global* if you want the object's scale to be applied to the measurements.

See also

The Measure tool for measuring arbitrary distances and angles.

Normals

- Display vertex normals
- Display face normals at vertices (split normals)
- Display face normals

Size
: The size to show the selected normals.

Constant Screen Size Normals
: Keep the size of normals constant in relation to the zoom level.

Developer

These overlays are only available if *Developer Extras* is enabled in the Interface Preferences.

Indices
: Display the indices of selected vertices, edges, and faces.

Freestyle

These settings apply to the Freestyle line art renderer.

Edge Marks
: Display Freestyle edge marks.

Face Marks

Display Freestyle face marks.

Sculpt Mode Overlays

Mask
: Show Masks as overlays on an object. The opacity of the overlay can be adjusted.

Face Sets
: Show Face Sets as overlays on an object. The opacity of the overlay can be adjusted.

Vertex Paint Overlays

Stencil Mask Opacity
: Does nothing. (Stencil masks are only available for texture painting.)

Show Wire
: Display mesh edges in white (unlike the *Wireframe* overlay which shows them in black).

Weight Paint Overlays

Opacity
: The opacity of the overlay.

Zero Weights
: Display unreferenced and zero-weighted areas in black. This helps to identify areas with very low weights that have been painted onto.

None::	Vertices are displayed in the usual way.
Active::	Vertices are shown in black if they have no weight in the active vertex group.
All::	Vertices are shown in black if they have no weight in any vertex group.

Show Weight Contours
: Show contour lines formed by points with the same interpolated weight.

  This visualizes weight variations too small to be seen from colors and can be useful for judging the smoothness and consistency of gradients, e.g. when using smoothing tools and brushes.

Show Wire
: Display mesh edges in white (unlike the *Wireframe* overlay which shows them in black).

Texture Paint Overlays

Stencil Mask Opacity
: Opacity of the stencil mask overlay.

Pose Mode Overlays

Fade Geometry
: Show the bones on top and face other geometry to the back. The opacity can be controlled with the slider. Only available in Pose Mode.

Grease Pencil

These overlays are available when a Grease Pencil object is selected.

Onion Skin
: Show ghosts of the keyframes before and after the current frame. If Multiframe is enabled, ghosts of the selected keyframes are shown instead. See Onion Skinning.

Fade Inactive Layers
: Decrease the opacity of all the layers in the object other than the active one. The opacity factor can be controlled with the slider.

Fade Inactive Objects

Cover all of the viewport except the active Grease Pencil object with a full color layer to improve visibility while drawing over complex scenes.

Fade Grease Pencil Objects
: Include or exclude Grease Pencil objects.

Edit Lines *Edit, Sculpt, Weight Paint, or Vertex Paint Modes* `Shift-Q`
: Shows a line between points on top of other geometry when editing strokes.

Only in Multiframe `Shift-Alt-Q`
: When Multiframe is enabled and keyframes other than the current frame are selected, strokes on those keyframes are displayed as just their edit lines – the strokes themselves are hidden. Note that this does not affect Onion Skinning.

Stroke Direction *Edit*
: Toggle the display of the selected strokes' start points (green) and end points (red) to visualize their direction.

Material Name *Edit*
: Show material name next to the selected strokes.

Vertex Paint

Vertex Paint Opacity *Vertex Paint*
: The opacity of the vertex color overlay in Vertex Paint Mode and Draw Mode. Note that in Draw Mode, vertex paint is only visible in the *Material Preview* and *Rendered* shading modes by default. To see it in *Solid* mode, you either need to use Vertex Paint Mode, or set the Color shading setting to *Attribute*.

Canvas

Display a grid over the Grease Pencil drawing plane.

Canvas Grid Opacity

The opacity of the grid.

Canvas X-Ray
: Objects are drawn behind the canvas grid.

Subdivisions
: The number of subdivisions between grid lines.

Grid Color
: The color of the grid lines.

Scale X/Y
: The horizontal/vertical size of the grid.

Offset X/Y
: The amount to shift the grid up/down and left/right.

Viewport Shading

Reference

Mode:: All Modes
Header::  *Viewport Shading*
Shortcut:: `Z` `Shift-Z`

Blender offers different shading modes for helping with different tasks. For example, Solid shading is well-suited for modeling, while Rendered is useful for setting up lighting.

The radio buttons let you change the shading mode, while the drop-down button opens a popover with additional options described below.

Pressing `Z` opens a pie menu for changing the shading mode. Pressing `Shift-Z` switches between the current shading mode and Wireframe.

Wire Color
: How wireframes are colored. This affects the Wireframe shading mode and overlay.

Theme:: Use the *Active Object*, *Wire*, or *Wire Edit* theme color based on the object's current state.
Object:: Use the color from the object's Viewport Display settings.
Random:: Each object gets displayed in a random color.

Wireframe

Only displays the edges (wireframes) of the objects in the scene.

Background
: How the background is displayed in the 3D Viewport.

Theme:: Use the background of the theme. This can be configured in the Themes Preferences under *3D Viewport ▸ Theme Space ▸ Gradient Colors*.

World:: Use the color from the World's Viewport Display options.

Viewport:: Select a custom color for the background of the 3D Viewport.

Options

X-Ray `Alt-Z`

Make objects transparent, allowing you to see and select items that would otherwise be occluded. The slider controls object opacity.

Outline

Draw an outline around objects. The color of the outline can be adjusted.

Solid

This mode utilizes the Workbench Render Engine to render the 3D Viewport. It shows solid geometry but uses simplified shading and lighting without the use of shader nodes. Solid mode is good for modeling and sculpting, and is really useful with the multitude of options to emphasize certain geometric features.

Lighting

How lights are computed.

Flat:: Do not calculate any lighting. The base color of the scene will be rendered.

Studio:: Use studio lights to light the objects. The studio lights can be configured in the preferences.
Studio lights can follow the camera or be fixed.
When fixed the angle of the lights can be adjusted.
World Space Lighting

Uses world space lighting so lights do not follow the view camera.

Rotation
: The rotation of the studio lights on the Z axis.

MatCap:: Use a material capture to light the objects in the scene. MatCaps can be flipped horizontally by clicking the Flip MatCap button.
Custom MatCaps can be loaded in the preferences.

Color
: The source to compute the color for objects in the viewport.

Material:: Use the color that can be set per material in the Viewport Display Material panel.

Object:: Use the color that can be set per object in the Viewport Display Object panel.

Attribute:: Display the active Color Attribute of an object. When an object has no active Color Attribute it will be rendered in the color set in the Viewport Display Object panel.

Single:: Render the whole scene using a single color. The color can be chosen.

Random:: A random color will be selected for every object in the scene.

Texture:: Show the texture from the active image texture node using the active UV map coordinates When an object has no active texture the object will be rendered with the settings in the Viewport Display Material panel.

Background
: How the background is displayed in the 3D Viewport.

Theme:: Use the background of the theme. This can be configured in the Themes Preferences under *3D Viewport ▸ Theme Space ▸ Gradient Colors*.

World:: Use the color from the World's Viewport Display options.

Viewport:: Select a custom color for the background of the 3D Viewport.

Options

Backface Culling

Use backface culling to hide backsides of faces.

X-Ray

Render the scene transparent. With the slider you can control how transparent the scene should appear.

Shadow

Renders a sharp shadow in the scene.

Darkness

Defines how dark the shadow should be rendered. This slider can be adjusted between 0 (shadow not visible) and 1 (shadow is black).

Light Direction

Controls the direction of the light source that casts the shadows.

Shadow Shift

Controls the Shadow termination angle. It can be used to limit self shadowing artifacts.

Shadow Focus

Controls the falloff near the edge of the shadow.

Cavity

Highlight ridges and valleys in the scene geometry.

Type

Method how to calculate the cavity.

World:: More precise but is slower to calculate.

Screen:: Fast but does not take the size of the ridges and valleys into account.

Both:: Both will use both methods.

Ridge
: Control the visibility of ridges.

Valley
: Control the visibility of valleys.

Depth of Field
: Use the Depth of Field settings of the active camera in the viewport. Only visible when looking through the camera.

  The settings are located on *Properties* ▸ *Camera* ▸ *Depth of Field* panel.

Outline
: Render the outline of objects in the viewport. The color of the outline can be adjusted.

Specular Highlighting
: Render specular highlights.

  Note

  Only available when Lighting is set to *Studio* lighting or when a MatCap has been selected that contains a specular pass.

Material Preview

Render the 3D Viewport with EEVEE and an HDRI environment. This mode is particularly suited for previewing materials and painting textures. You can select different lighting conditions to test your materials.

Note

The Material Preview shading mode is not available when the scene's render engine is set to Workbench.

Lighting

Scene Lights

Use the lights in the scene. When disabled (or when the scene contains no lights), a virtual light is used instead.

Scene World

Use the World of the scene. When disabled, a world will be constructed with the following options:

HDRI Environment

The environment map used to light the scene.

Rotation

The rotation of the environment on the Z axis.

World Space Lighting

Makes the lighting rotation fixed and not follow the camera.

Strength

Light intensity of the environment.

World Opacity

Opacity of the HDRI as a background image in the viewport.

Blur

Factor to unfocus the HDRI. Note that this does not change the diffusion of the lighting, only the appearance of the background.

Render Pass

Instead of the combined render, show a specific render pass. Useful to analyze and debug geometry, materials and lighting.

Compositor

Controls the enablement of the compositor in the viewport. If enabled, the scene compositor node tree will be evaluated using the GPU Compositor and the output will be displayed directly in the viewport.

Disabled::	Always disabled.
Camera::	Enabled only in camera view.
Always::	Always enabled regardless of the view.

Tip

Since it might be difficult to control the look of the viewport compositor due to changes in the aspect ratio, pan, and zoom of the viewport, one can limit the compositing space to the camera region by changing to Camera View, which should make the results better match the final render.

Rendered

Render the 3D Viewport using the scene's *Render Engine*, for interactive rendering. This gives you a preview of the final result, including scene lighting effects.

The options are the same as for *Material Preview*, except that the *Render Pass* selector will offer different passes if the scene uses the Cycles render engine.

Toolbar

The Toolbar contains a list of tools. Links to each mode's Toolbar are listed below.

Object Mode

- Object Mode

Edit Mode

- Mesh Edit Mode
- Curve Edit Mode
- Surface Edit Mode
- Metaball Edit Mode

Paint Modes

- Sculpt Mode
- Texture Paint Mode
- Vertex Paint Mode
- Weight Paint Mode

Grease Pencil

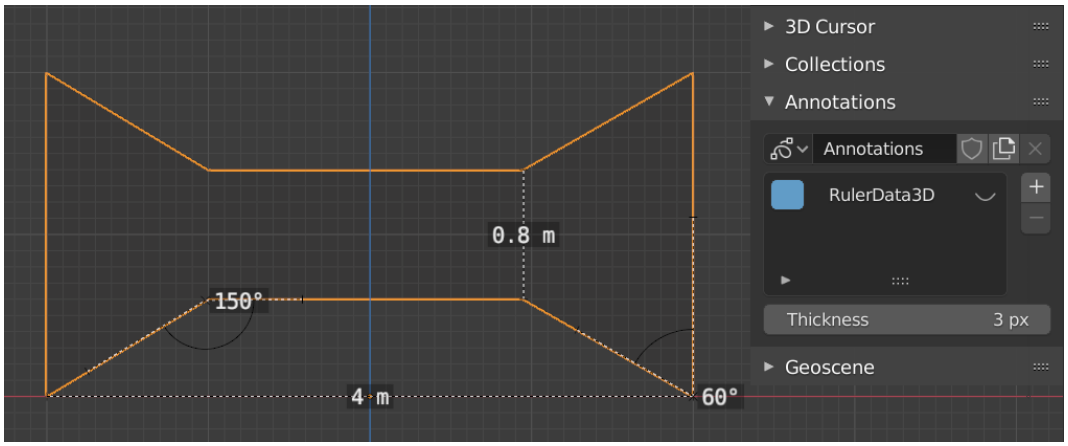
- Grease Pencil Edit
- Grease Pencil Draw
- Grease Pencil Sculpting
- Grease Pencil Weight Paint

Measure

Reference

Mode::	All Modes
Tool::	*Toolbar* ‣ *Measure*

The *Measure* tool is an interactive tool where you can drag lines in the scene to measure distances or angles. Snapping to geometry could be activated for better accuracy or to measure wall thickness. The *Measure* tool can be accessed from the Toolbar.



Examples of the Measure tool.

Usage

Here are some common steps for using the *Measure* tool:

1. Activate the *Measure* tool from the Toolbar.
2. Click and drag in the viewport to define the initial start and end point for the ruler. You can add multiple rulers in the viewport.
3. Drag either end of the ruler to move it.

- Holding `Ctrl` while moving enables snapping to edges and vertices.
- Holding `Shift` while moving lets you measure the distance between faces. This only works well with parallel faces, e.g. walls.

You can always navigate (pan, zoom, …) or change the view (orthogonal, perspective) in the viewport to have better access to the ruler.

4. Click on the midpoint of a created ruler to convert it to a protractor. The midpoint can then be dragged just like the endpoints.

5. A selected ruler can be deleted with `Delete` or `X`. To delete all measurements, delete the "RulerData3D" layer in the *Sidebar* ▸ *View* ▸ *Annotations* panel (see image above).

All measurements are hidden when another tool is selected. They are shown when the *Measure* tool is selected again. However, you can do editing operations while the ruler is active. For example, you can edit the rotation or scale of the selected object in the Sidebar.

Measurements do not appear in the Render output.

Unit settings and scale from the scene are used for displaying dimensions. Changing the unit system (metric, imperial), or the units of length (cm, m, …) or angle (degrees, radians) will update the measurements.

Tip

In Edit Mode only, there is also a *Measurement* group in the Viewport Overlays popover. Using the settings in this group, you can have the viewport automatically display measurements for selected edges and faces, without the need to manually create a ruler.

Add Cube

Reference

Mode::	Object Mode and Edit Mode
Tool::	*Toolbar* ▸ *Add Cube*

Interactively add a cube mesh object.

Usage

First define the base of the object by dragging with `LMB`. Next, release `LMB` and move the mouse to define the height of the object. Finally, click `LMB` to confirm the shape of the object.

You can use the following hotkeys to temporarily change a setting (for as long as the key is held):

`Ctrl`	Toggles snapping.
`Alt`	Toggles the *Origin* setting.
`Shift`	Toggles the *Aspect* setting.

Tool Settings

Depth
: The initial depth (from the screen into the scene) used when placing the object.

Surface:: Start placing on the surface under the mouse cursor. If there is no surface, this does the same as *Cursor Plane*.

Cursor Plane:: Start placing on a plane that goes through the 3D Cursor and is aligned according to the *Orientation* and *Plane Axis*.

Cursor View:: Start placing on a plane that goes through the 3D Cursor and is aligned to the view.

Orientation

The new object's orientation – a set of three axes, out of which *Plane Axis* chooses one.

Surface:: The object uses the normal orientation of the surface under the mouse cursor. If there is no surface, this does the same as *Default*.

Default:: The object uses the default Transform Orientation.

Snap To

The target to use while Snapping.

Geometry:: Snap to all types of geometry (vertices, edges, and faces).

Default:: Snap to the target defined in the global snapping options.

Plane Axis

Which of the three *Orientation* axes (X, Y or Z) is "up" for the object. The object's base will be perpendicular to this axis.

Auto Axis

Rather than using the *Orientation* axis indicated by *Plane Axis*, use the one that's closest to the viewport's viewing direction (when not hovering over a surface).

Base

Origin

How the base is defined.

Edge:: The base is defined from one corner to the opposing corner.

Center:: The base is defined from the centerpoint to a corner.

Aspect

Whether the base has a free or fixed aspect ratio.

Free:: The width and depth of the base can be chosen independently.

Fixed:: The width and depth of the base are forced to be equal.

Height

Origin

How the height is defined.

Edge:: The base becomes the bottom, after which you define the top.

Center:: The base becomes the center, after which you define the top.

Aspect

Whether the object's side has a free or fixed aspect ratio.

Free:: The height can be chosen independently of the base.

Fixed:: The height is forced to be equal to the largest side of the base.

Add Cone

Reference

Mode::	Object Mode and Edit Mode
Tool::	*Toolbar* ▸ *Add Cone*

Interactively add a cone mesh object.

Usage

First define the base of the object by dragging with `LMB`. Next, release `LMB` and move the mouse to define the height of the object. Finally, click `LMB` to confirm the shape of the object.

You can use the following hotkeys to temporarily change a setting (for as long as the key is held):

`Ctrl`	Toggles snapping.
`Alt`	Toggles the *Origin* setting.
`Shift`	Toggles the *Aspect* setting.

Tool Settings

Depth
: The initial depth (from the screen into the scene) used when placing the object.

Surface:: Start placing on the surface under the mouse cursor. If there is no surface, this does the same as *Cursor Plane*.

Cursor Plane:: Start placing on a plane that goes through the 3D Cursor and is aligned according to the *Orientation* and *Plane Axis*.

Cursor View:: Start placing on a plane that goes through the 3D Cursor and is aligned to the view.

Orientation

The new object's orientation – a set of three axes, out of which *Plane Axis* chooses one.

Surface:: The object uses the normal orientation of the surface under the mouse cursor. If there is no surface, this does the same as *Default*.

Default:: The object uses the default Transform Orientation.

Snap To

The target to use while Snapping.

Geometry:: Snap to all types of geometry (vertices, edges, and faces).

Default:: Snap to the target defined in the global snapping options.

Plane Axis

Which of the three *Orientation* axes (X, Y or Z) is "up" for the object. The object's base will be perpendicular to this axis.

Auto Axis

Rather than using the *Orientation* axis indicated by *Plane Axis*, use the one that's closest to the viewport's viewing direction (when not hovering over a surface).

Base

Origin

How the base is defined.

Edge:: The base is defined from one corner to the opposing corner.

Center:: The base is defined from the centerpoint to a corner.

Aspect

Whether the base has a free or fixed aspect ratio.

Free:: The width and depth of the base can be chosen independently.

Fixed:: The width and depth of the base are forced to be equal.

Height

Origin

How the height is defined.

Edge:: The base becomes the bottom, after which you define the top.

Center:: The base becomes the center, after which you define the top.

Aspect

Whether the side of the bounding box has a free or fixed aspect ratio.

Free:: The height can be chosen independently of the base.

Fixed:: The height is forced to be equal to the largest side of the base.

Vertices

The number of vertices in the base.

Base Fill Type

Set how the circle at the base will be filled.

Triangle Fan::	Fill with triangular faces which share a vertex in the middle.
N-gon::	Fill with a single N-gon.
Nothing::	Do not fill. Creates only the outer ring of vertices.

Add Cylinder

Reference

Mode::	Object Mode and Edit Mode
Tool::	*Toolbar* ▸ *Add Cylinder*

Interactively add a cylinder mesh object.

Usage

First define the base of the object by dragging with `LMB`. Next, release `LMB` and move the mouse to define the height of the object. Finally, click `LMB` to confirm the shape of the object.

You can use the following hotkeys to temporarily change a setting (for as long as the key is held):

`Ctrl`	Toggles snapping.
`Alt`	Toggles the *Origin* setting.
`Shift`	Toggles the *Aspect* setting.

Tool Settings

Depth
: The initial depth (from the screen into the scene) used when placing the object.

Surface:: Start placing on the surface under the mouse cursor. If there is no surface, this does the same as *Cursor Plane*.

Cursor Plane:: Start placing on a plane that goes through the 3D Cursor and is aligned according to the *Orientation* and *Plane Axis*.

Cursor View:: Start placing on a plane that goes through the 3D Cursor and is aligned to the view.

Orientation

The new object's orientation – a set of three axes, out of which *Plane Axis* chooses one.

Surface:: The object uses the normal orientation of the surface under the mouse cursor. If there is no surface, this does the same as *Default*.

Default:: The object uses the default Transform Orientation.

Snap To

The target to use while Snapping.

Geometry:: Snap to all types of geometry (vertices, edges, and faces).

Default:: Snap to the target defined in the global snapping options.

Plane Axis

Which of the three *Orientation* axes (X, Y or Z) is "up" for the object. The object's base will be perpendicular to this axis.

Auto Axis

Rather than using the *Orientation* axis indicated by *Plane Axis*, use the one that's closest to the viewport's viewing direction (when not hovering over a surface).

Base

Origin

How the base is defined.

Edge:: The base is defined from one corner to the opposing corner.

Center:: The base is defined from the centerpoint to a corner.

Aspect

Whether the base has a free or fixed aspect ratio.

Free:: The width and depth of the base can be chosen independently.

Fixed:: The width and depth of the base are forced to be equal.

Height

Origin

How the height is defined.

Edge:: The base becomes the bottom, after which you define the top.

Center:: The base becomes the center, after which you define the top.

Aspect

Whether the side of the bounding box has a free or fixed aspect ratio.

Free:: The height can be chosen independently of the base.

Fixed:: The height is forced to be equal to the largest side of the base.

Vertices

The number of vertices in the caps.

Cap Fill Type

Set how the caps will be filled.

Triangle Fan::	Fill with triangular faces which share a vertex in the middle.
N-gon::	Fill each ring with an N-gon.
Nothing::	Do not fill. Creates only the outer rings of vertices.

Add UV Sphere

Reference

Mode::	Object Mode and Edit Mode
Tool::	*Toolbar* ‣ *Add UV Sphere*

Interactively add a UV sphere mesh object.

Usage

First define the base of the object by dragging with `LMB`. Next, release `LMB` and move the mouse to define the height of the object. Finally, click `LMB` to confirm the shape of the object.

You can use the following hotkeys to temporarily change a setting (for as long as the key is held):

`Ctrl`	Toggles snapping.
`Alt`	Toggles the *Origin* setting.
`Shift`	Toggles the *Aspect* setting.

Tool Settings

Depth
: The initial depth (from the screen into the scene) used when placing the object.

Surface:: Start placing on the surface under the mouse cursor. If there is no surface, this does the same as *Cursor Plane*.

Cursor Plane:: Start placing on a plane that goes through the 3D Cursor and is aligned according to the *Orientation* and *Plane Axis*.

Cursor View:: Start placing on a plane that goes through the 3D Cursor and is aligned to the view.

Orientation

The new object's orientation – a set of three axes, out of which *Plane Axis* chooses one.

Surface:: The object uses the normal orientation of the surface under the mouse cursor. If there is no surface, this does the same as *Default*.

Default:: The object uses the default Transform Orientation.

Snap To

The target to use while Snapping.

Geometry:: Snap to all types of geometry (vertices, edges, and faces).

Default:: Snap to the target defined in the global snapping options.

Plane Axis

Which of the three *Orientation* axes (X, Y or Z) is "up" for the object. The object's base will be perpendicular to this axis.

Auto Axis

Rather than using the *Orientation* axis indicated by *Plane Axis*, use the one that's closest to the viewport's viewing direction (when not hovering over a surface).

Base

Origin

How the base is defined.

Edge::	The base is defined from one corner to the opposing corner.
Center::	The base is defined from the centerpoint to a corner.

Aspect

Whether the base has a free or fixed aspect ratio.

Free::	The width and depth of the base can be chosen independently.
Fixed::	The width and depth of the base are forced to be equal.

Height

Origin

How the height is defined.

Edge::	The base becomes the bottom, after which you define the top.
Center::	The base becomes the center, after which you define the top.

Aspect

Whether the side of the bounding box has a free or fixed aspect ratio.

Free::	The height can be chosen independently of the base.
Fixed::	The height is forced to be equal to the largest side of the base.

Segments

Number of vertical segments. Like the Earth's meridians, going pole to pole.

Rings

Number of horizontal segments. These are like the Earth's parallels.

Note

Rings are face loops and not edge loops, which would be one less.

Add Icosphere

Reference

Mode::	Object Mode and Edit Mode
Tool::	*Toolbar* ▸ *Add Icosphere*

Interactively add an Icosphere mesh object.

Usage

First define the base of the object by dragging with `LMB`. Next, release `LMB` and move the mouse to define the height of the object. Finally, click `LMB` to confirm the shape of the object.

You can use the following hotkeys to temporarily change a setting (for as long as the key is held):

`Ctrl`	Toggles snapping.
`Alt`	Toggles the *Origin* setting.
`Shift`	Toggles the *Aspect* setting.

Tool Settings

Depth
: The initial depth (from the screen into the scene) used when placing the object.

Surface:: Start placing on the surface under the mouse cursor. If there is no surface, this does the same as *Cursor Plane*.

Cursor Plane:: Start placing on a plane that goes through the 3D Cursor and is aligned according to the *Orientation* and *Plane Axis*.

Cursor View:: Start placing on a plane that goes through the 3D Cursor and is aligned to the view.

Orientation

The new object's orientation – a set of three axes, out of which *Plane Axis* chooses one.

Surface:: The object uses the normal orientation of the surface under the mouse cursor. If there is no surface, this does the same as *Default*.

Default:: The object uses the default Transform Orientation.

Snap To

The target to use while Snapping.

Geometry:: Snap to all types of geometry (vertices, edges, and faces).

Default:: Snap to the target defined in the global snapping options.

Plane Axis

Which of the three *Orientation* axes (X, Y or Z) is "up" for the object. The object's base will be perpendicular to this axis.

Auto Axis

Rather than using the *Orientation* axis indicated by *Plane Axis*, use the one that's closest to the viewport's viewing direction (when not hovering over a surface).

Base

Origin

How the base is defined.

Edge:: The base is defined from one corner to the opposing corner.

Center:: The base is defined from the centerpoint to a corner.

Aspect

Whether the base has a free or fixed aspect ratio.

Free:: The width and depth of the base can be chosen independently.

Fixed:: The width and depth of the base are forced to be equal.

Height

Origin

How the height is defined.

Edge:: The base becomes the bottom, after which you define the top.

Center:: The base becomes the center, after which you define the top.

Aspect

Whether the side of the bounding box has a free or fixed aspect ratio.

Free:: The height can be chosen independently of the base.

Fixed:: The height is forced to be equal to the largest side of the base.

Subdivisions

Influences how many vertices are used to define the sphere. At level 1 the icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Each increase in the number of subdivisions splits each triangular face into four.

Note

Subdividing an icosphere raises the vertex count very quickly even with few iterations (10 times creates 5,242,880 triangles). Adding such a dense mesh is a sure way to cause the program to crash.

Sidebar

Item

Shows Transform settings of the active object.

Tool

Shows settings of the active tool and Workspace.

View

View Panel

The *View* panel lets you change other settings regarding the 3D Viewport.

Focal Length
: Control the focal length of the 3D Viewport camera.

Clip Start/End
: Adjust the minimum and maximum distances for geometry to be visible. Geometry closer than *Start* or further away than *End* will not be shown.

Note

In Orthographic view, the viewport uses negative *End* instead of *Start*.

Warning

A large clipping range will allow you to see both near and far objects, but reduces the depth precision resulting in artifacts.

In some cases, a very large range may cause operations that depend on the depth buffer to become unreliable, although this depends on the

graphics card and drivers.

See Troubleshooting Depth Buffer Glitches for more information.

Local Camera
: Allow this 3D Viewport to have its own active camera, separate from the global active camera that's defined in the scene. The selector next to the checkbox lets you choose this camera.

Passepartout
: Show camera passepartout when in camera view.

Render Region
: Use the Render Region. Defining the region with `Ctrl-B` will automatically enable this option.

  Note that if you're viewing the scene through the active camera, this option has no effect – in this case, you instead need to use the checkbox *Output Properties ▸ Format ▸ Render Region* in the Properties editor. This will affect not just the viewport, but also the final render.

View Lock

Lock to Object
: Lets you select an object to become the point of interest of the viewpoint. The view will then orbit around, and zoom towards, that object. This option is not available when viewing the scene through the active camera.

Lock: To 3D Cursor
: Makes the 3D Cursor the point of interest of the viewpoint. This option is only available when *Lock to Object* is not active.

Lock: Camera to View
: When looking through a camera, the camera becomes "glued" to the view and will follow it around as you navigate. The camera frame will

be outlined with a red dashed line.

Hint

If the camera is parented to an object, you can choose to enable Camera Parent Lock in the camera's properties. This will cause viewport navigation to transform the camera's root parent rather than the camera itself.

3D Cursor

Location
: The location of the 3D Cursor.

Rotation
: The rotation of the 3D Cursor.

Rotation Mode
: The rotation mode of the 3D Cursor.

Collections

The *Collections* panel shows a list of collections and can be used to control their visibility. If a collection contains objects, there is a circle to the left of its name.

Local Collections
: Allows setting collection visibility per viewport rather than globally.

Hide in Viewport (eye icon)
: Shows or hides the collection.

You can also "isolate" a collection by clicking its name. This will show the collection as well as its ancestors and descendants, and hide all other collections.

Annotations

See Annotations for more information.

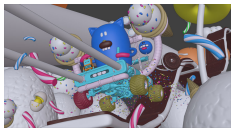
Viewport Render

Viewport rendering lets you create quick preview renders from the current viewpoint (rather than from the active camera, as would be the case with a regular render).

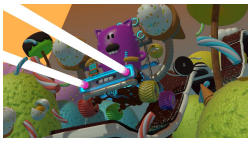
You can use *Viewport Render* to render both images and animations.

Below is a comparison between the Viewport render and a final render using the Cycles Renderer.

Model by © 2016 pokedstudio.com



Viewport render using Solid Mode.



Viewport render using Material Preview Mode.



Full render.

Note

Viewport rendering only works for the Workbench and EEVEE render engines. It's not supported for Cycles.

Tip

Disable overlays to get a render without "clutter" like rigs, empties and so on.

Settings

For the most part, *Viewport Render* uses the current viewport settings. Some settings are located in the properties of the render engine that is used to render the view.

Solid mode uses the render settings of Workbench; Material Preview mode uses the render settings of EEVEE.

Additionally, some output settings are used too:

- Resolution
- Aspect
- Output path
- File format

Rendering

Activating *Viewport Render* will render from the current active view. This means that if you are not in an active camera view, a virtual camera is used to match the current perspective. To get an image from the camera point of view, enter the active camera view with `Numpad0`.

As with a normal render, you can abort it with `Esc`.

Render a Still Image
: To render a still image, use *3D Viewport ▸ View ▸ Viewport Render Image*.

Render an Animation
: To render an animation, use *3D Viewport ▸ View ▸ Viewport Render Animation*.

Render Keyframes
: To render an animation, but only those frames that have a keyframe, use *3D Viewport ▸ View ▸ Viewport Render Keyframes*. This only renders

those frames for which the selected objects have an animation key. The other frames are still written to the output, but will simply repeat the last-rendered frame.

For example, when a six-frame animation is rendered, and the selected objects have a key on frames 3 and 5, the following frames will be output:

1. The 1st frame is always rendered.
2. The 1st frame is repeated because there is no key on this frame.
3. The 3rd frame is rendered.
4. The 3rd frame is repeated because there is no key on this frame.
5. The 5th frame is rendered.
6. The 5th frame is repeated because there is no key on this frame.

Tip

You can limit the viewport render to a particular region with Render Regions.

Image Editor

- Introduction
  - Toolbar
  - Header
  - Asset Shelf Region
  - Main View
- Navigating
  - Gizmos
  - View Menu
- Sidebar
  - Tool
  - Image
  - View
  - Scopes
- Overlays
  - Geometry
  - Image
- Image Settings
  - Source
  - Common Options
- Editing
  - New
  - Open
  - Open Cached Render
  - Replace
  - Reload
  - Edit Externally
  - Copy/Paste
  - Save
  - Save As
  - Save a Copy
  - Save All Images
  - Invert
  - Resize

- Transform
- Pack
- Unpack
- Extract Palette
- Generate Grease Pencil

Introduction

The Image Editor lets you create, view, and edit images, as well as see render results and intermediate Compositor output.

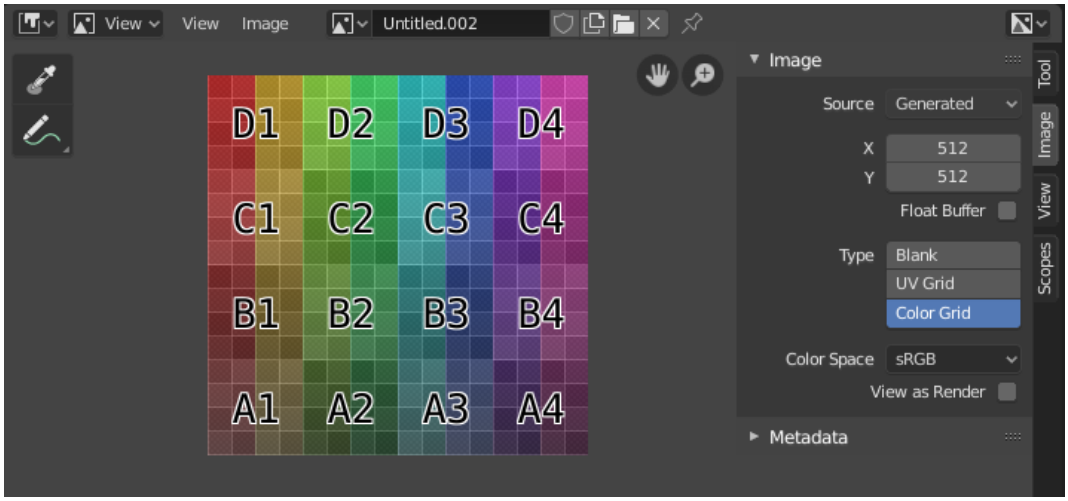


Image Editor with a test grid texture.

Toolbar

Sample
: Used to sample the color of one or more pixels in the image. As long as you hold `LMB`, the footer will show the following:

  - X and Y coordinates of the mouse cursor.
  - Color in RGBA.
  - Color in RGB after Color Management.
  - Color in HSV.
  - Luminance.

  Sample Size
  : The dimensions of the square used to sample underlying pixels. If larger than 1, the resulting sample is an average of all underlying pixels.

Annotate
: See Annotations for more information.

Header

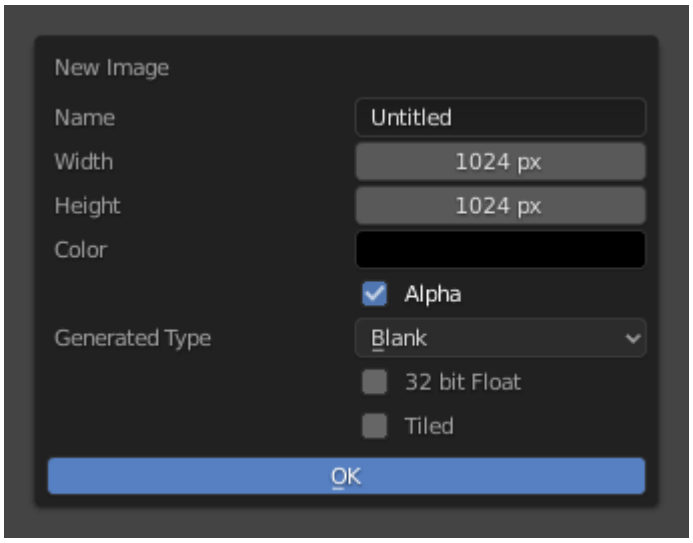
Mode
: **View::** Displays images.
Paint:: Texture Paint.
Mask:: Masking.

View
: Tools for controlling how the content is displayed in the editor. See Navigating.

Image
: Tools for opening and manipulating images. Shows an asterisk if the image has unsaved changes. See Editing.

Image
: A data-block menu used for selecting images. Once an image is selected, the Image tab appears in the Sidebar region.

  Apart from loading existing images, you can also create new ones:



The pop-over that's displayed when clicking "New Image" in the header.

The *Tiled* option creates an image with support for UDIMs. For the other options, see Generated Images.

In addition to images, the data-block selector includes the following items:

- Render Result: displays renders. When this item is selected, the *Slot*, *View Layer*, and *Render Pass* selectors become available (see below).
- Viewer Node: displays the image that's fed into the Viewer Node in the Compositor.

Image Pin

Prevents the Image Editor from automatically switching to the texture of the selected object. (This switching only happens if the 3D Viewport is in Texture Paint mode).

Slot

The render slot to view (and render to). You can create new renders without losing previous ones by selecting an empty slot before rendering. Afterwards, you compare them by pressing `J` and `Alt-J` to cycle forwards and backwards. Alternatively, you can use the number keys `1`, `2`, `3` etc. to select the slot with the corresponding number.

Slots can be renamed by double clicking their name in the Image panel in the Sidebar.

View Layer

The View Layer to display.

Render Pass

The Render Pass to display.

Viewport Gizmos

Lets you show/hide all gizmos using the toggle button, or specific gizmos using the drop-down arrow.

Navigate
: Enable/disable the gizmos used to pan or zoom the 2D viewport. See Navigation Gizmos for more information.

Display Channels
: Select which color channels are displayed.

Color & Alpha::	Enables transparency and shows a checkerboard behind the image.
Color::	Disables transparency.
Alpha::	Displays the alpha channel as a grayscale image. White areas are opaque, black areas are transparent.
Z-Buffer::	Displays the depth from the camera, from Clip Start to Clip End, as specified in the Camera settings.
Red, Green, Blue::	Single color channel visualized as a grayscale image.

Asset Shelf Region

Depending on the current mode, the asset shelf may be available, providing quick access to assets for this specific mode (for example brush assets in *Paint* mode).

See Asset Shelf for more information.

Main View

Holding `RMB` will sample the image just like the Sample tool, except it will always sample only one pixel.

Navigating

Panning can be done by dragging with `MMB`.

Zooming can be done using `Wheel` or `NumpadPlus`/`NumpadMinus`.

Gizmos

Next to the Sidebar region at the top, there are gizmos that allow panning and zooming more comfortably when e.g. no mouse wheel is available.

View Menu

Toolbar `T`
: Show or hide the Toolbar.

Sidebar `N`
: Show or hide the Sidebar.

Tool Settings
: Show or hide the settings for the currently selected tool.

Asset Shelf
: Toggle the visibility of the Asset Shelf.

Adjust Last Operation
: Displays a pop-up panel to alter properties of the last completed operation. See Adjust Last Operation.

Update Automatically
: Instantly update any other editors that are affected by changes in this Image Editor. When disabled, the other editors may display outdated information until they're manually refreshed (e.g. by orbiting for the 3D Viewport).

Show Metadata
: Displays metadata about the selected Render Result. See the Output tab's Metadata panel to change what metadata to include.

Display Texture Paint UVs
: Toggles UVs in Paint Mode. The object must be in Texture Paint Mode or Edit Mode for the UVs to be visible.

Zoom
: Menu with convenient zoom levels and operations. The zoom levels are calculated based on the images resolution compared to the screen resolution.

  - 12.5% (1:8) `Numpad8` zoom out to a factor of 12.5%.
  - 25% (1:4) `Numpad4` zoom out to a factor of 25%.
  - 50% (1:2) `Numpad2` zoom out to a factor of 50%.
  - 100% (1:1) `Numpad1` resets the zoom to 100%.
  - 200% (2:1) `Ctrl-Numpad2` zoom in to a factor of 200%.
  - 400% (4:1) `Ctrl-Numpad4` zoom in to a factor of 400%.
  - 800% (8:1) `Ctrl-Numpad8` zoom in to a factor of 800%.

  Zoom In/Out `Wheel`
  : Zooms the view in or out.

  Zoom to Fit `Shift-Home`
  : Like *Frame All*, but uses as much space in the editor as possible.

  Zoom Region `Shift-B`
  : Zoom in the view to the nearest item contained in the border.

Frame All `Home`
: Pans and zooms the view so that the image is centered and fully visible.

Center View to Cursor
: Pan the view so that the 2D cursor is at the center of the editor.

Render Region `Ctrl-B`
: Only available when viewing the Render Result. See Render Region.

Clear Render Region `Ctrl-Alt-B`
: Only available when viewing the Render Result. See Render Region.

Render Slot Cycle Next/Previous `J`/`Alt-J`
: Switch to the next/previous render slot (that contains a render).

Area
: Adjust the area the Image Editor is in.

Sidebar

Tool

Displays the settings of the active tool.

Image

Image

Tools for working with images. See Image Settings.

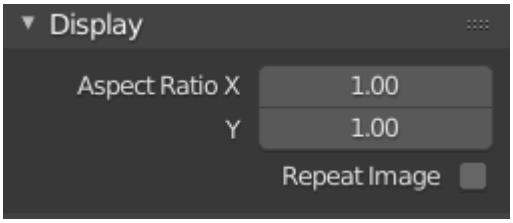
Metadata

Lists image metadata.

View

Display

You can set the editor's display options in this panel.



Display panel.

Aspect Ratio
: Display aspect for this image. Does not affect rendering.

Repeat Image
: Tile the image so it completely fills the editor.

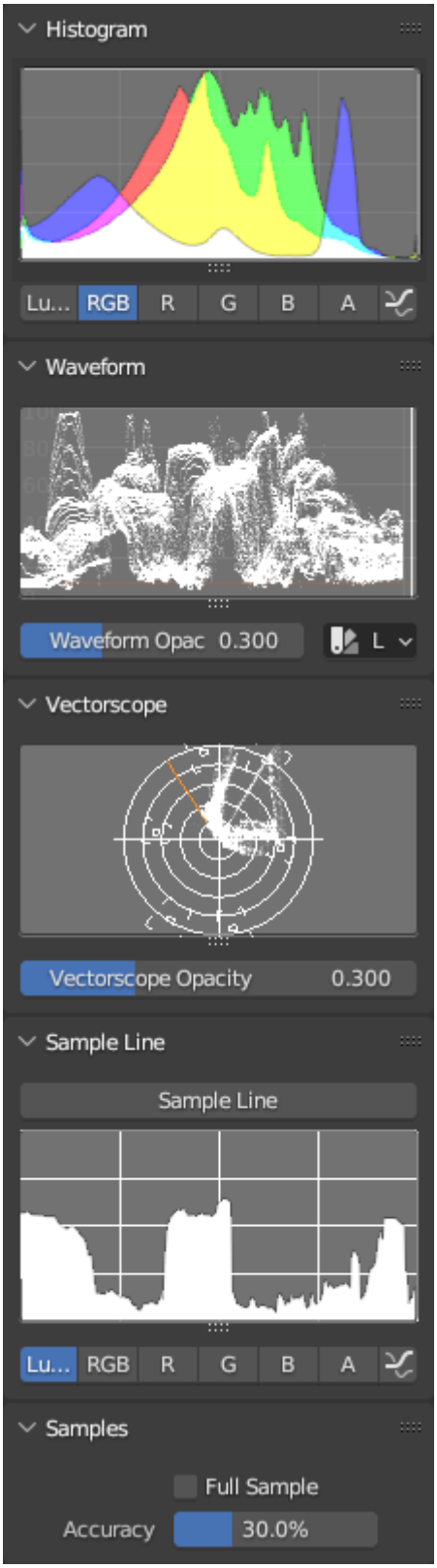
Annotations

Options for the annotation tool. See Annotations.

Scopes

Displays different kinds of statistical information about the colors in the image.

Note that the Scopes tab is not shown if the active object is in Edit Mode or Texture Paint Mode.



Scopes in the Image Editor.

Histogram

Displays a graph of the color distribution in the image. For each color value (such as Luminance) on the X axis, it shows the number of pixels with that value on the Y axis. A predominantly dark image would have the highest values toward the left side of the graph.

Use this mode to balance out the tonal range in an image. A well-balanced image should have a nice smooth distribution of color values.

You can drag `LMB` in the histogram to adjust its vertical zoom.

Luma
: Shows a luminosity histogram.

RGB
: Shows the RGB channels stacked on top of each other.

R/G/B/A
: Shows a single color channel.

Show Line
: Displays lines rather than filled shapes.

Waveform

Plots the color distribution for each vertical line of pixels in the image. The X axis of the Waveform corresponds to the X axis of the image, while the Y axis represents the range of a color component such as Luminance. The brighter a specific point is, the more pixels in that vertical line have that color value.

Waveform Opacity
: Opacity of the points.

Waveform Mode
: Luma
  : Show a single Waveform plotting the luminosity distribution.

YCbCr
: Show the Y, Cb and Cr Waveforms side by side.

Parade
: Show the R, G and B Waveforms side by side.

Red Green Blue
: Show the R, G and B Waveforms overlaid on top of each other.

Vectorscope

Shows the color distribution in a radial fashion. The angle represents the hue, while the distance from the center represents the saturation.

Vectorscope Opacity
: Opacity of the points.

Sample Line

The *Sample Line* scope is the same as the Histogram but allows you to get the sample data from a line.

Sample Line
: Used to draw a line to read the sample data from.

Samples

Full Sample
: Sample every pixel.

Accuracy
: Proportion of image pixels to sample if *Full Sample* is disabled.

Overlays

The Overlays pop-over configures the overlays that are displayed on top of images. In the header, there is a button to turn off all overlays for the Image Editor. This option also toggles the visibility of UDIM tile information.

The options that are visible in the pop-over depend on the Image Editor mode. The following overlay categories are available:

Geometry

Display Texture Paint UVs
: Display the active object's UVs. The Image Editor must be in Paint mode, and the active object must be in Texture Paint Mode or Edit Mode for the UVs to be visible.

Image

Show Metadata
: Displays metadata about the selected Render Result. See the Output tab's Metadata panel to change what metadata to include.

Image Settings

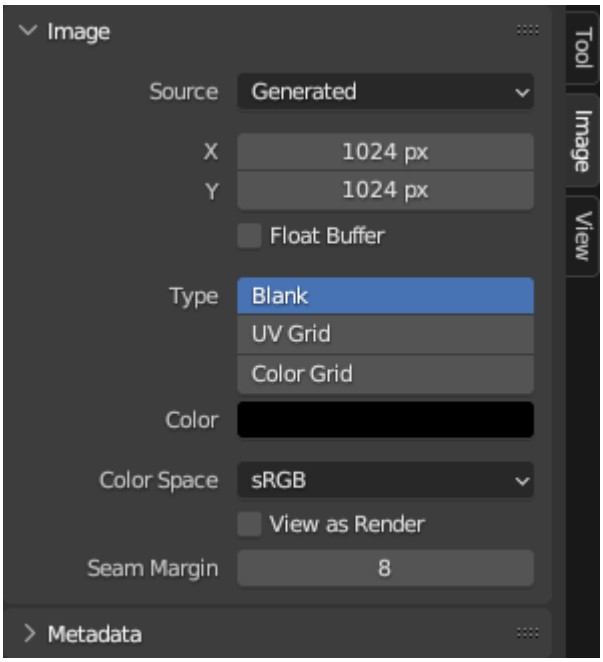


Image tab.

Source

Select the type of image to use. For images that come from files, see Supported Graphics Formats.

Single Image

A single, static image.

Image Sequence

An animation where each frame is stored in a separate file. See Opening an Image Sequence. For options, see Movie below.

Movie

A video file. Note that if you want to do motion tracking and video compositing rather than simply using the video as a texture, you should load it into the Movie Clip Editor instead.

Note

The options below are for preview purposes only; they don't affect the 3D Viewport or the render. For that, see the Image Texture Node.

Note

Blender plays all videos at the scene framerate, not their original framerate, meaning they'll be faster or slower than intended if these framerates don't match up. To work around this, see the *Offset* field of the Image Texture Node linked above.

Frames
: How many frames of the video to play. Past this point, the video will be paused (unless *Cyclic* is enabled).

  Match Movie Length
  : Sets the *Frames* to the number of frames in the video file.

Start
: Scene frame at which the video should start playing.

Offset
: Number of frames to offset the video to an earlier point in time. (Put differently: how many frames at the start of the video to skip.)

Cyclic
: Start over after the last frame to create a continuous loop.

Auto Refresh
: Play the video in the Image Editor when the scene animation is playing. (The mouse cursor should be in the Image Editor or the Timeline when starting playback for this to work.)

Deinterlace
: Apply deinterlacing to interlaced (analog) video.

Generated

Image generated by Blender.

X, Y
: The width and height of the image in pixels.

Float Buffer
: Creates a 32-bit image. This has a larger file size, but holds much more color information than the standard 8-bit image. For close-ups and large gradients, it may be better to use a 32-bit image.

Type

Blank::	Creates a blank image of a single specified color.
UV Grid::	Creates a checkerboard pattern with a colored cross (+) in each square.
Color Grid::	Creates a more complex colored grid with letters and numbers denoting locations. It could be used to check for stretching or distortion in the UV mapping.

Color
: The fill color when creating a *Blank* image.

Common Options

File
: Used for replacing or packing files.

Pack
: Embed the resource into the current blend-file. See Packed Data.

Path
: Path to the linked file.

Open
: Opens the File Browser to select a file from a drive.

Reload
: Reloads the file. Useful when it has been reworked in an external application.

Use Multi-View
: See Multi-View.

Color Space
: The Color Space the image file was saved in. This is used for converting the image to linear color (which is the color space Blender works with).

  Textures and final renders are often stored in sRGB, while OpenEXR images are stored in a linear color space. Some images such as normal, bump or stencil maps do not strictly contain "colors" and should never have a color conversion applied to them. For such images, the color space should be set to *Non-Color*.

  The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration.

Alpha
: How the image uses its Alpha Channel. This option is only available if the image format supports transparency.

Straight:: Store RGB and alpha channels separately with alpha acting as a mask, also known as unassociated alpha. Commonly used by image editing applications and file formats like PNG. This preserves colors in parts of the image with zero alpha.

Premultiplied:: Store RGB channels with alpha multiplied in, also known as associated alpha. The natural format for renders and used by file formats like OpenEXR. This can represent purely emissive effects like fire correctly, unlike straight alpha.

Channel Packed:: Different images are packed in the RGB and alpha channels, and they should not affect each other. Channel packing is commonly used by game engines to save memory.

None:: Ignore alpha channel from the file and make image fully opaque.

Half Float Precision

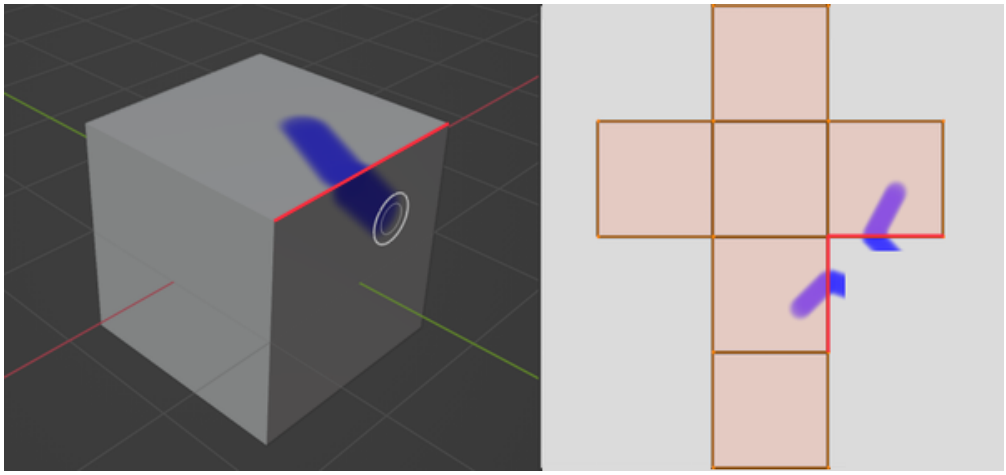
Load the image with a bit depth of only 16 bits per channel instead of 32, which saves memory.

View as Render

Apply the color management settings when displaying this image on the screen.

Seam Margin

The thickness of the margin around UV islands for texture painting to bleed into. This margin ensures that no unpainted pixels remain at the island border.



Painting a stroke across a seam in 3D space makes it extend past the UV island borders in the texture, until it gets cut off at the margin.

A higher value will result in a thicker margin, which can be useful if you intend to create mipmaps of the texture. However, this may also reduce painting performance.

Note

This setting only affects Sculpt Mode, where texture painting support is currently experimental. In Texture Paint Mode, a fixed margin is used instead.

Editing

New

Reference

Mode::	All Modes
Menu::	*Image* ▸ *New*
Shortcut::	`Alt-N`

Create a new Generated Image.

Open

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Open*
Shortcut::	`Alt-O`

Load an image from a file.

Open Cached Render

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Open Cached Render*
Shortcut::	`Ctrl-R`

Find the render cache file for the current scene and load it into the Render Result. This way, you can restore the last render from a previous Blender

session and continue working in the Compositor without having to render the scene again.

Note that Blender doesn't create these cache files by default. You have to enable Cache Result in the scene's Output options and then render it at least once.

Replace

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Replace*

Replace the current image by another.

Reload

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Reload*
Shortcut::	`Alt-R`

Reload the image from the file on drive.

Edit Externally

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Edit Externally*

Open the image in the *Image Editor* program specified in the File Paths Preferences.

Copy/Paste

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Copy/Paste*

Copy and pastes images to/from a computers operating system.

Currently, only PNG files are supported and Linux support requires Wayland.

Save

Reference

Mode::	All Modes
Menu::	*Image* ▸ *Save*
Shortcut::	`Alt-S`

Save the image to its current path.

Important

While animation renders are automatically saved, still renders are not. These have to be saved manually.

Save As

Reference

Mode:: All Modes
Menu:: *Image ▸ Save As*
Shortcut:: `Shift-Alt-S`

Save the image to a separate file of any type. The image output settings can be configured and are the same as the Render Output Properties.

Save a Copy

Reference

Mode:: All Modes
Menu:: *Image ▸ Save a Copy*

Save the file under a specified name, but keep the old one open in the Image editor.

Save All Images

Reference

Mode:: All Modes
Menu:: *Image ▸ Save All Images*

Save all modified images. Packed images will be repacked.

Invert

Reference

Mode:: All Modes
Menu:: *Image ▸ Invert*

Invert Image Colors

Invert the colors of an image.

Invert Red/Green/Blue/Alpha Channel
: Invert a single color channel.

Resize

Reference

Mode:: All Modes
Menu:: *Image ▸ Resize*

Adjust the image size in pixels.

Transform

Flip Horizontally
: Mirrors the image so the left side becomes the right side.

Flip Vertically
: Mirrors the image so the top becomes the bottom.

Rotate 90° Clockwise
: Rotates the image clockwise 90°.

Rotate 90° Counter-Clockwise
: Rotates the image counter-clockwise 90°.

Rotate 180°
: Rotates the image 180°.

Pack

Reference

Mode:: All Modes

Menu:: *Image ▸ Pack*

Pack the image into the blend-file. See Packed Data.

Unpack

Reference

Mode:: All Modes
Menu:: *Image ▸ Unpack*

Unpack the image to a drive.

Extract Palette

Reference

Mode:: All Modes
Menu:: *Image ▸ Extract Palette*

Extract a Color Palette from the image for use by painting tools.

Generate Grease Pencil

Reference

Mode:: All Modes
Menu:: *Image ▸ Generate Grease Pencil*

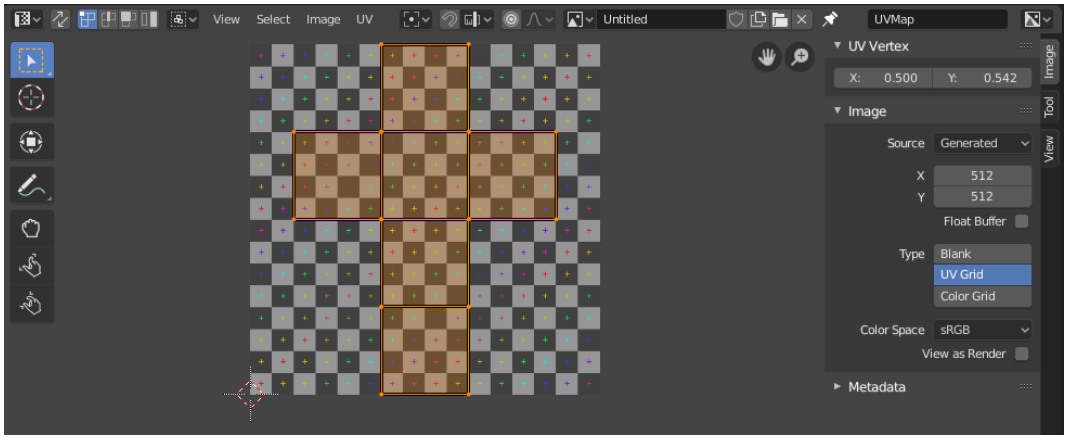
Create a Grease Pencil object using the currently selected image as a source.

UV Editor

- Introduction
  - UVs Explained
  - Interface
- Navigating
  - 2D Viewport
  - Gizmos
  - View Menu
  - 2D Cursor
- Overlays
  - Guides
  - UV Editing
  - Geometry
  - Image
- Sidebar
  - Image Tab
  - Tool Tab
  - View Tab
  - Scopes
- Selecting
  - Sync Selection
  - Selection Mode
  - Sticky Selection Mode
  - Select Menu
  - Shortest Path
  - Select Edge Loop
  - Select Edge Ring
- Snapping
  - Snap Target
  - Additional Options
  - Affect
  - Rotation Increment

Introduction

The UV Editor is used for editing UV maps, which describe how a 2D image should be mapped onto a 3D object.



UV Editor with a UV map and a test grid texture.

Image textures are typically needed when the desired look is hard to achieve with procedural textures, or if the texture is not uniform. For example, a car would only have scratches in a few places where they make sense, not in random places all over its body.

Blender offers a number of projections (Box, Sphere…) that automatically apply a 2D image to a 3D object, but these tend to only work for simple meshes. For more complex ones, you need to create a UV map instead. This is a flat area where each face of the 3D object is laid out on the 2D image, specifying which part of the image it should be textured with. This gives you complete control over the mapping process.

The name “UV” refers to the axes of the map: U for horizontal, V for vertical. These letters were chosen to avoid confusion with “X” and “Y”, which refer to axes in 3D space instead.

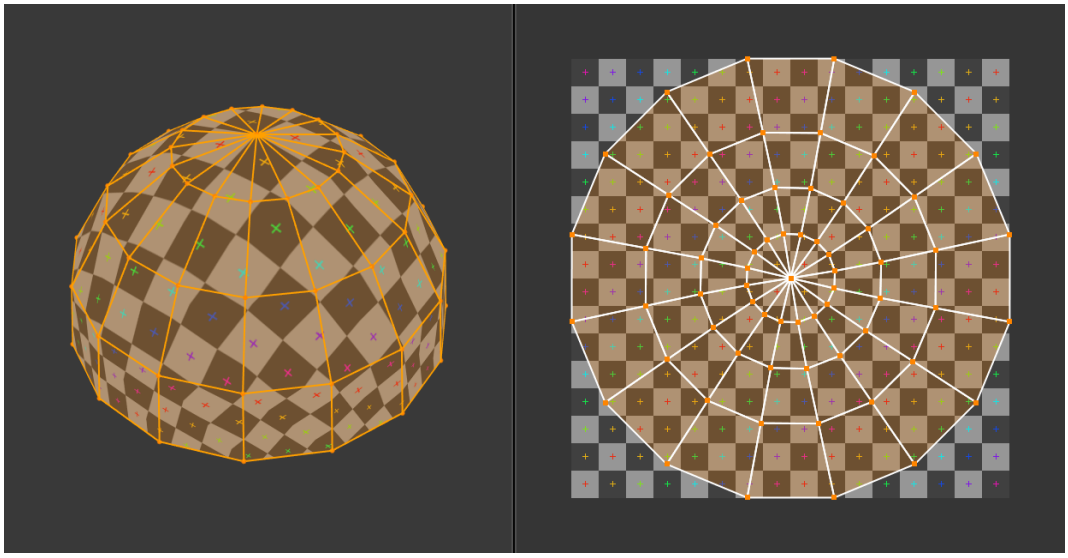
UVs Explained

The best analogy to understand UV mapping is cutting up a cardboard box. If you were to take a pair of scissors and cut along its edges, you would be able to spread it out flat on a tabletop. As you are looking down at the table, we could say that U is the left-right direction, and V is the up-down direction.

As a next step, you could put the spread-out box on top of a poster, cut the poster to match its shape, glue the poster to the box, and finally reassemble the box. You now have a 3D box textured with a 2D image.

A UV map describes how the box is cut up, and how it's laid out on the poster. You have complete freedom in how to do this: if you wanted to, you could cut each individual side of the box and position, rotate, scale, and even skew it on the poster independently of the other sides.

Example



3D space (XYZ) versus UV space.

In the above image, a dome in 3D space is flattened into a disc in UV space. Each 3D face is then textured with the part of the image it covers in the UV map.

The image also demonstrates a common problem in UV maps: distortion. Notice how, even though the checkered squares in the 2D texture are all the same size, they get different sizes when applied to the 3D dome (they're smaller at the base than at the top). This is because the faces in the UV map have different relative sizes than in 3D space, which is a result of the flattening process.

You'll typically want to minimize this distortion by manually guiding and tweaking the flattening, using seams for example. However, it's not always possible to eliminate it completely.

Interface

Header



UV Editor header.

The header contains several menus and options for working with UVs.

Sync Selection
: Synchronizes the selection between the UV Editor and the 3D Viewport. See Sync Selection for more details.

Selection Mode
: The UV element type to select. See Selection Mode for more details.

Sticky Selection Mode
: Which other vertices to select automatically. See Sticky Selection Mode for more details.

View
: Tools for controlling how the content is displayed in the editor. See Navigating.

Select

Tools for selecting UVs.

Image
: Tools for opening and manipulating images. See Editing.

UV
: Contains tools for Unwrapping Meshes and Editing UVs.

Pivot `Period`
: See Pivot Point.

Snap `Shift-Tab`
: See Snapping.

Proportional Editing `O`
: See Proportional Editing.

Image
: A data-block menu used for selecting images. When an image has been loaded or created in the UV Editor, the Image panel appears in the Sidebar region.

Image Pin
: When enabled the current image remains visible regardless of the object selection. This switching only happens if the 3D Viewport is in Edit Mode or Texture Paint Mode.

  This can be useful to enable when an image is used as a reference.

Show Gizmo
: Lets you show/hide all gizmos using the toggle button, or specific gizmos using the drop-down arrow.

  Navigate
  : Enable/disable the gizmos used to pan or zoom the 2D viewport. See Navigation Gizmos for more information.

Show Overlays

Lets you show/hide all overlays using the toggle button, or specific overlays using the drop-down arrow. See Overlays.

Active UV Map Layer

Select which UV map to use.

Display Channels

Select what color channels are displayed.

Color & Alpha::	Enables transparency and shows a checkerboard behind the image.
Color::	Displays the colored image, without alpha channel.
Alpha::	Displays the alpha channel as a grayscale image. White areas are opaque, black areas are transparent.
Z-Buffer::	Displays the depth from the camera, from Clip Start to Clip End, as specified in the Camera settings.
Red, Green, Blue::	Single color channel visualized as a grayscale image.

Navigating

2D Viewport

Panning can be done by dragging with `MMB`.

Zooming can be done using `Wheel` or `NumpadPlus`/`NumpadMinus`.

Gizmos

Next to the Sidebar region at the top, there are gizmos that allow panning and zooming more comfortably when e.g. no mouse wheel is available.

View Menu

Also see Navigating in the Image Editor.

Frame Selected `NumpadPeriod`
: Change the view so that all selected UV vertices are visible.

2D Cursor

Just like the 3D Viewport, the UV Editor has a Cursor that you can jump to (*View ▸ Center View to Cursor*). It can also serve as a pivot point and a snapping target.

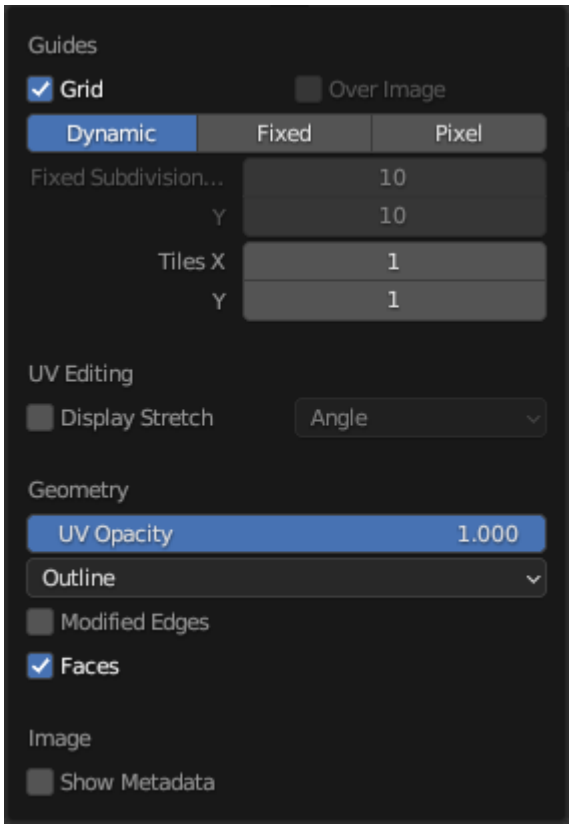
To change the Cursor's position, either press `LMB` with the Cursor tool selected, or `Shift-RMB` with any tool selected. You can also change the "Location X/Y" fields in the *View* tab of the Sidebar, in either relative coordinates (0 to 1) or pixel coordinates. In both cases, the lower left corner of the image serves as the origin (0, 0).

You can press `Shift-C` to move the Cursor to the center.

Overlays

In the header, there is a button to turn off all overlays for the UV Editor. This option also toggles the visibility of UDIM tile information.

The drop-down button opens a pop-over with more detailed settings. The following categories are available:



The Overlays pop-over.

Guides

Grid
: Show the grid.

Over Image
: Show the grid on top of the image rather than behind it.

Grid Shape Source
: How the row and column counts are determined.

Dynamic:: The grid starts at 8×8 cells that are automatically subdivided further as you zoom in.
Fixed:: The row and column counts are fixed and can be configured manually.
Pixel:: Each grid cell matches one image pixel.

Fixed Subdivisions X, Y
: Number of columns/rows in the grid.

Tiles X, Y
: The number of UDIM tile grids to display in each cardinal direction.

UV Editing

Display Stretch
: Show how much of a shape difference there is between UV space and 3D space. Blue means low distortion, red means high. You can choose whether to display the distortion based on *Angle* or *Area*.

Geometry

UV Opacity
: Opacity of edges and faces.

Display As
: Control how edges are shown.

Outline:: Display edges in gray with a black outline.
Dash:: Display edges as dashed black-gray lines.
Black:: Display edges in black.
White:: Display edges in white.

Modified Edges
: Additionally show the edges as they look after applying modifiers (in gray).

Faces

Display faces over the image.

Image

Show Metadata
: Display metadata about the selected Render Result. See the Output tab's Metadata panel to change what metadata to include.

Sidebar

Image Tab

UV Vertex

The averaged-out position of the selected UV vertices.

Image

See Image Settings.

UDIM Tiles

See UDIM Tiles.

Tool Tab

Shows the settings for the active tool.

View Tab

Display

You can set the editor's display options in this panel.

Aspect Ratio X, Y
: Display aspect for this image. Does not affect rendering.

Repeat Image
: Tile the image so it completely fills the editor.

Pixel Coordinates

Use pixel coordinates rather than relative coordinates (0 to 1) for the UV Vertex and 2D Cursor Location fields.

2D Cursor

Location X, Y
: View and change the location of the 2D Cursor.

Annotations

Options for the Annotate tool.

Scopes

See Scopes in the Image Editor.

Selecting

Much like the 3D Viewport, the UV Editor has selection mode buttons in the header, as well as a *Select* menu.

Sync Selection

If turned off (the default), the UV Editor only shows the faces that are selected in the 3D Viewport. Selecting an item in one editor does not automatically select it in the other. If one 3D vertex/edge corresponds to multiple UV vertices/edges, you can select each of those individually.

If turned on, the UV Editor always shows all faces. Selecting an item in one editor also selects it in the other. If one 3D vertex/edge corresponds to multiple UV vertices/edges, you can't select those individually (you can only select all of them).

Selection Mode

Vertex::	`1` Select vertices.
Edge::	`2` Select edges.
Face::	`3` Select faces.
Island::	`4` Select contiguous groups of faces. Only available if *Sync Selection* is disabled.

If *Sync Selection* is enabled, you can hold `Shift` while clicking a selection mode to activate multiple ones at the same time, or `Ctrl` to expand/contract the selection.

See also

Mesh Selection

Sticky Selection Mode

Options for automatically selecting additional UV vertices. Only available if *Sync Selection* is disabled.

Disabled
: Each UV vertex can be selected independently of the others.

Shared Location
: Automatically select UV vertices that correspond to the same mesh vertex and have the same UV coordinates. This is the default and gives the illusion that multiple faces in a UV map can share the same vertex; in reality, they have separate vertices that overlap.

Shared Vertex
: Automatically select UV vertices that correspond to the same mesh vertex, even if they have different UV coordinates. This is also the behavior when *Sync Selection* is enabled.

Select Menu

All `A`
: Selects all UV elements.

None `Alt-A`
: Deselects all UV elements.

Invert `Ctrl-I`
: Inverts the current selection.

Box Select `B`
: See Box Select.

Box Select Pinned `Ctrl-B`
: Like *Box Select*, but only selects pinned UV vertices.

Circle Select

See Circle Select.

Lasso Select
See Lasso Select.

More/Less `Ctrl-NumpadPlus`, `Ctrl-NumpadMinus`
Expands/contracts the selection to/from the adjacent elements.

Select Similar `Shift-G`
Selects UV elements that are similar to the active one in some way. The Adjust Last Operation panel provides several options:

Type
The property to compare. Which properties are available depends on the Selection Mode.

Vertex Selection Mode

Pinned::	Selects vertices with the same pinned state.

Edge Selection Mode

Length::	Selects edges with a similar length in the UV map.
Length 3D::	Selects edges with a similar length in the 3D mesh.
Pinned::	Selects edges with the same pinned state.

Face Selection Mode

Area::	Selects faces with a similar area in the UV map.
Area 3D::	Selects faces with a similar area in the 3D mesh.
Material::	Selects faces that have the same Material.
Object::	Selects faces that belong to the same object. This is useful when multiple objects are in Edit mode at once.
Polygon Sides::	Selects faces with a similar number of edges.

Winding:: Select faces that have the same orientation (facing upwards or downwards in the UV map).

Island Selection Mode

Area:: Selects islands with a similar area in the UV map.

Area 3D:: Selects islands with a similar area in the 3D mesh.

Amount of Faces in Island:: Selects islands with a similar number of faces.

Compare

The comparison operator.

Equal:: Select elements whose value is equal.

Greater:: Select elements whose value is greater or equal.

Less:: Select elements whose value is less or equal.

Threshold

Tolerance for values that are almost, but not quite the same. A higher threshold will select more elements.

Select Linked

Linked `Ctrl-L`

Selects all elements that are connected to the currently selected ones.

Shortest Path

Selects the path between two selected elements. (See below)

Select Pinned `Shift-P`

Selects all pinned UVs.

Select Split `Y`

“Detaches” the selected faces so they can be moved elsewhere without affecting their neighbors.

Hint

Unlike Split Selection for meshes, which physically disconnects faces, this is a pure selection operator. In UV space, the faces were never connected to begin with; it only seemed that way because *Sticky Selection* automatically selected the vertices of the neighboring faces. *Select Split* deselects those vertices again.

As an alternative to *Select Split*, you can set the *Sticky Selection Mode* to *Disabled*.

Select Overlap
: Selects all UV faces that overlap each other.

Shortest Path

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Linked* ▸ *Shortest Path*
Shortcut::	`Ctrl-LMB`

Selects all the UV elements along the shortest path between two elements: the two selected elements when activated using the menu, or the active one and the clicked one when activated using the shortcut.

Face Stepping
: For vertices: allows the path to step across faces, following their diagonal rather than their edges.

  For edges: selects disconnected edges that are perpendicular to the path (edge ring), rather than connected edges along the path (edge loop).

  For faces: allows the path to go through faces that only share a vertex, rather than an edge.

Topology Distance
: Calculates the distance by simply counting edges rather than measuring their lengths.

Fill Region `Shift-Ctrl-LMB`
: Selects all shortest paths (rather than just one).

Dashed Line Options
: Allows to only select elements at regular intervals, creating a “dashed line” rather than a continuous one.

  Deselected
  : The number of deselected elements in the repetitive sequence.

  Selected
  : The number of selected elements in the repetitive sequence.

  Offset
  : The number of elements to offset the sequence by.

See also

Mesh edit Select Shortest Path.

Select Edge Loop

Reference

Mode::	Edit Mode
Shortcut::	`Alt-LMB`, or `Shift-Alt-LMB` for extending the existing selection.

Holding `Alt` while clicking an edge selects that edge and then expands the selection as far as possible in the two directions parallel to it. (While this of

course works for selecting edge “loops” that go all the way around a mesh, it also works if there’s no loop.)

You can additionally hold `Shift` to extend the current selection rather than replacing it.

See also

Mesh edit Select Edge Loops.

Select Edge Ring

Reference

Mode::	Edit Mode
Shortcut::	`Ctrl-Alt-LMB`, or `Shift-Ctrl-Alt-LMB` for extending the existing selection.

Holding `Ctrl-Alt` while clicking an edge selects that edge and then expands the selection as far as possible in the two directions perpendicular to it. (While this of course works for selecting edge “rings” that go all the way around a mesh, it also works if there’s no ring.)

You can additionally hold `Shift` to extend the current selection rather than replacing it.

See also

Mesh edit Select Edge Rings.

Snapping

Snapping lets you easily align UV elements to others. It can be toggled by clicking the magnet icon in the UV Editor's header, or more temporarily by holding `Ctrl`.

This page is about the Snap header button; for the Snap menu, see UV Editing.

Snap Target

Reference

Header:: *Snapping ▸ Snap To*
Shortcut:: `Shift-Ctrl-Tab`

Increment
: Snaps to grid points.

  This option snaps to an imaginary grid that starts at the selection's original location and has the same resolution as the grid displayed in the editor. In other words, it lets you move the selection in "increments" of the grid cell size.

Grid
: Snaps to grid points.

Vertex
: Snaps to the vertex that's closest to the mouse cursor.

Additional Options

Snap Base *Vertex*

See 3D Viewport Snapping for more information.

Affect

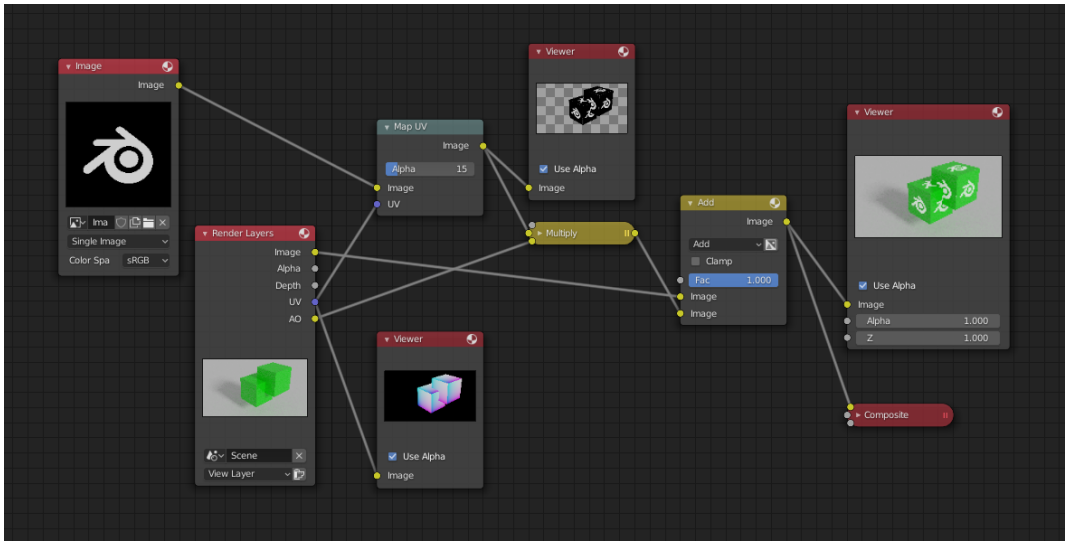
Specifies which transformations are affected by snapping. By default, snapping only happens while moving something, but you can also enable it for rotating and scaling.

Rotation Increment

Angle used in incremental snapping for the rotation operator. The second value is the **Rotation Precision Increment**, used for finer transformations and activated by default with the `Shift` key.

Compositor

The Compositor lets you manage nodes for compositing.



Nodes in the Compositor.

The use of the Compositor is explained in Compositing.

Texture Nodes

- Introduction
  - Using Texture Nodes
  - Header

Node Types

- Color Nodes
- Converter Nodes
- Distort Nodes
- Input Nodes
- Output Nodes
- Pattern Nodes
- Texture Nodes
- Group

Introduction

The Texture Node Editor allows creating custom textures by combining colors, procedural patterns, and images in various ways. This is a step up from the built-in textures, where you can select a type from a list and not much more.

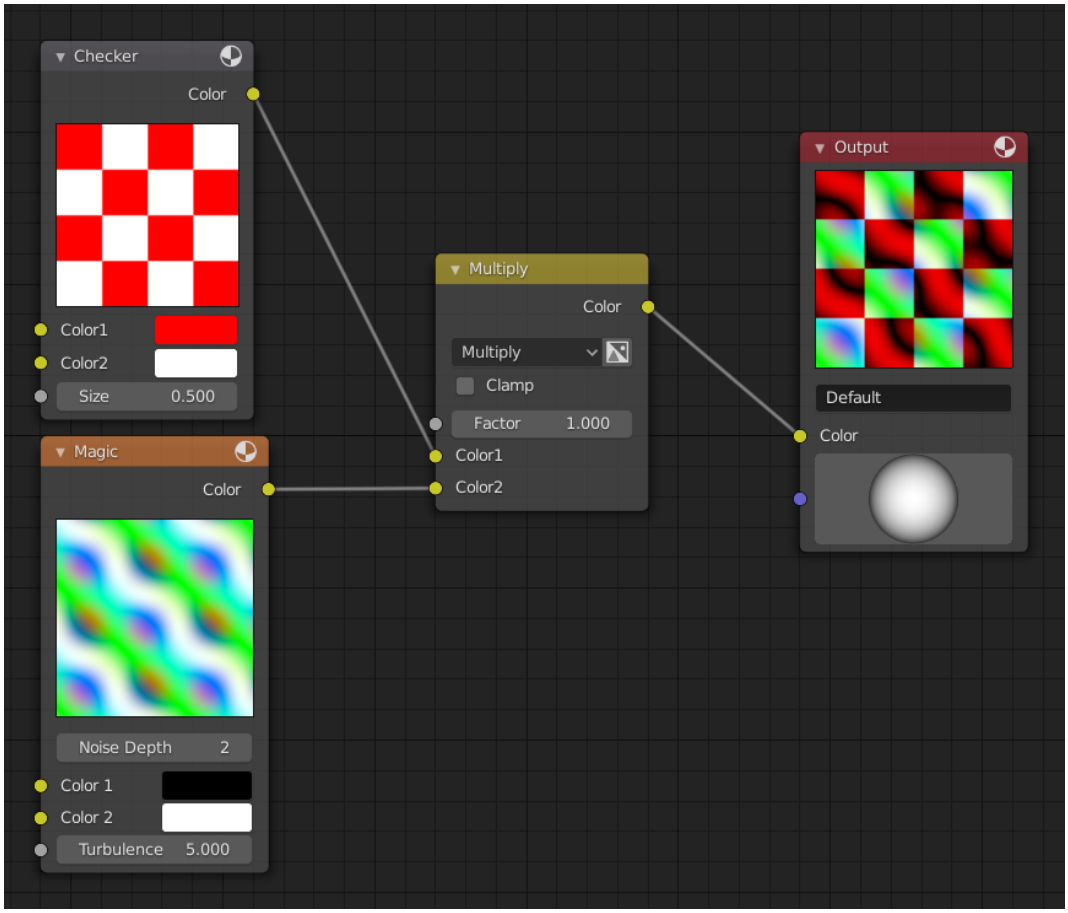
Note

Textures – both built-in ones and node-based ones – are a legacy feature. For their original main purpose, which was of course texturing objects, they have been replaced by Materials which are set up in the Shader Editor.

Today, the use of Textures is limited to:

- Brushes.
- The Displace Modifier.
- Influencing size, density etc. of particle systems.
- Influencing emission locations of fire/smoke simulations.
- Compositing.

In addition, the Displace modifier and fire/smoke simulations don't support node-based textures, instead only working with the built-in ones.



Combined textures based on nodes.

Using Texture Nodes

The default Blender layout has no workspace containing the Texture Node Editor. You need to manually open it in an area of choice.

Once the editor is open, you first need to set the empty *Texture Type* selector to *Brush*, after which you can use the Data-Block Menu to start creating textures. Note that you need to enable *Use Nodes* in the header before you can add nodes.

Header

See Nodes for the header items common to all node editors.

Texture Type

World

Deprecated – the scene's World Environment is now defined using a Material rather than a Texture.

Brush

Show brush Textures in the data-block menu. Because the other two types are deprecated, this effectively shows all Textures.

Line Style

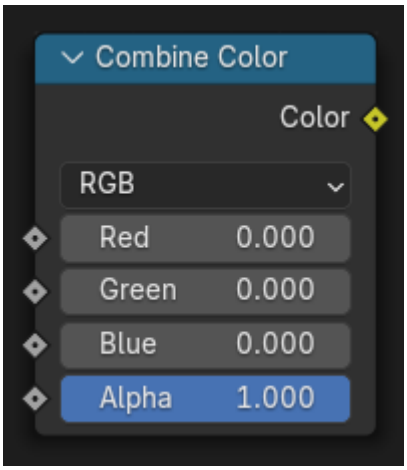
Deprecated – Line Styles for the Freestyle renderer are now defined using Materials rather than Textures.

Color Nodes

- Combine Color Node
- Invert Color Node
- Mix Node
- RGB Curves Node
- Hue/Saturation/Value Node
- Separate Color Node

Combine Color Node

Combines four grayscale channels into one color image, based on a particular Color Model.



Inputs

The inputs of this node depend on the Mode property (see below).

Alpha
: The opacity of the output color.

Properties

Mode
: The color model to use.

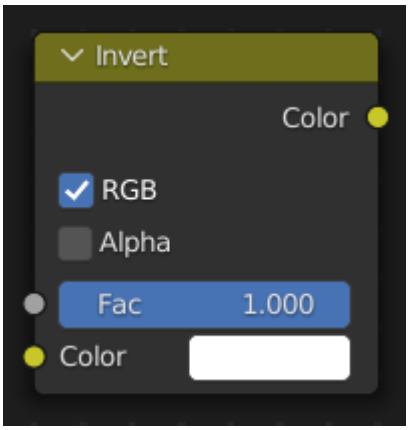
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Output

Color
: Standard color output.

Invert Color Node

Inverts the colors in the input image, producing a negative.



Inputs

Factor
: The amount of influence the node exerts on the image.

Color
: Standard color input.

Properties

In the compositing context, this node has the following properties:

RGB
: Invert the color channels.

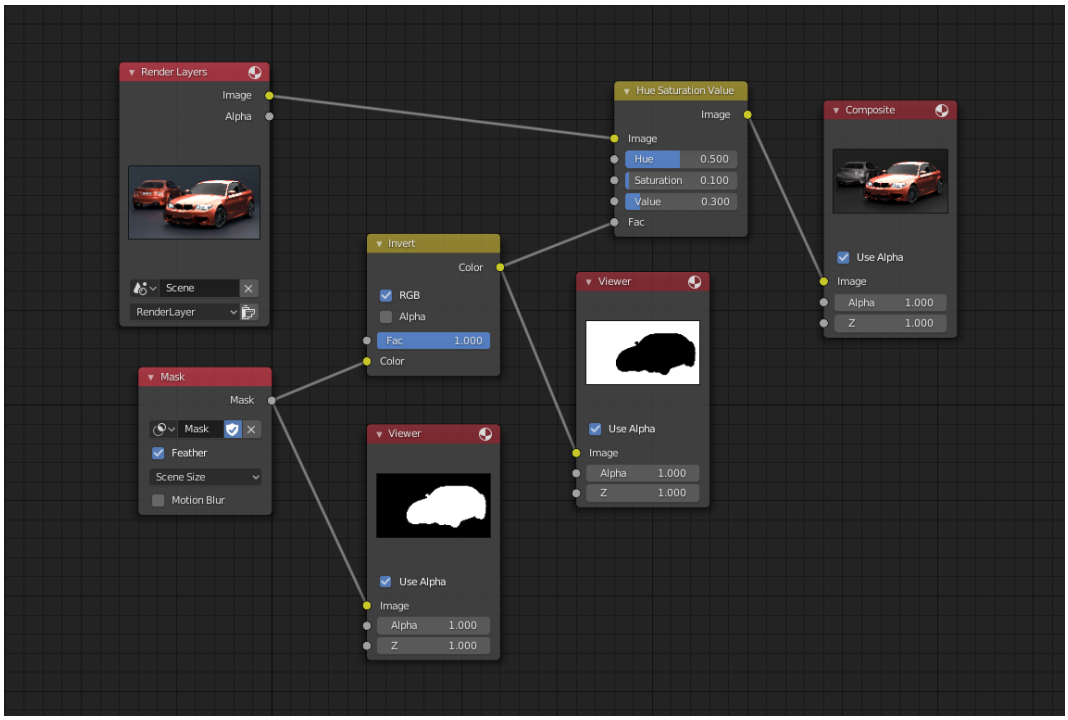
Alpha
: Invert the alpha channel.

Outputs

Color

Standard color output.

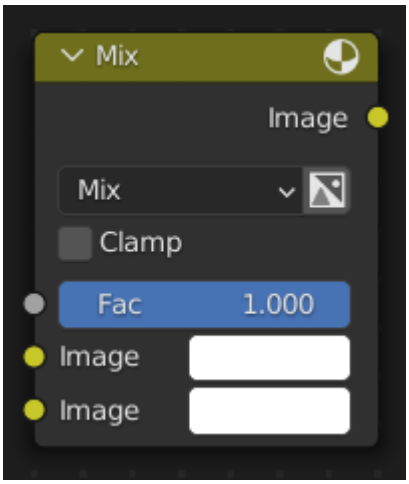
Example



The Invert node is used to invert the mask.

Mix Node

Blends two images together, much like how an image editing program blends two layers.



Inputs

Fac
: The opacity of the foreground image.

Image
: The background image. Determines the dimensions of the output.

Image
: The foreground image.

Keep in mind that, unlike image editing programs where the foreground layer is on top, the foreground slot in Blender is on the bottom.

Properties

Blending Mode
: The blending mode to use.

  Mix

Regular alpha blending. Typically called *Normal* in image editing programs.

Darken
: For each color component, takes the smallest of the two values being blended.

Multiply
: Multiplies the colors component by component. Blending with a white pixel (value 1.0) has no effect, while blending with a black one (0.0) always results in black.

Color Burn
: Inverts the background color, divides it by the foreground color, and inverts the result.

Lighten
: For each color component, takes the largest of the two values being blended.

Screen
: Inverts both colors, multiplies them, and inverts the result.

Color Dodge
: Divides the background color by the inverted foreground color.

Add
: Adds the two colors together.

Overlay
: Applies Multiply blending if the foreground color's lightness is below 0.5, or Screen blending if it's above.

Soft Light
: Like Overlay, but more subtle.

Linear Light

Applies Linear Burn blending (background + foregound - 1) if the foreground color's lightness is below 0.5, or Linear Dodge (background + foreground) if it's above.

Difference
: For each component, subtracts the lower value from the higher value.

Exclusion
: Adds the two colors, then subtracts their multiple twice.

Subtract
: Subtracts the foreground color from the background color.

Divide
: Divides the background color by the foreground color.

Hue
: Combines the saturation and value of the background color with the hue of the foreground color.

Saturation
: Combines the hue and value of the background color with the saturation of the foreground color.

Color
: Combines the value of the background color with the hue and saturation of the foreground color.

Value
: Combines the hue and saturation of the background color with the value of the foreground color.

Use Alpha
: Whether to use the alpha channel of the foreground image during mixing. The alpha channel of the background image is always used.

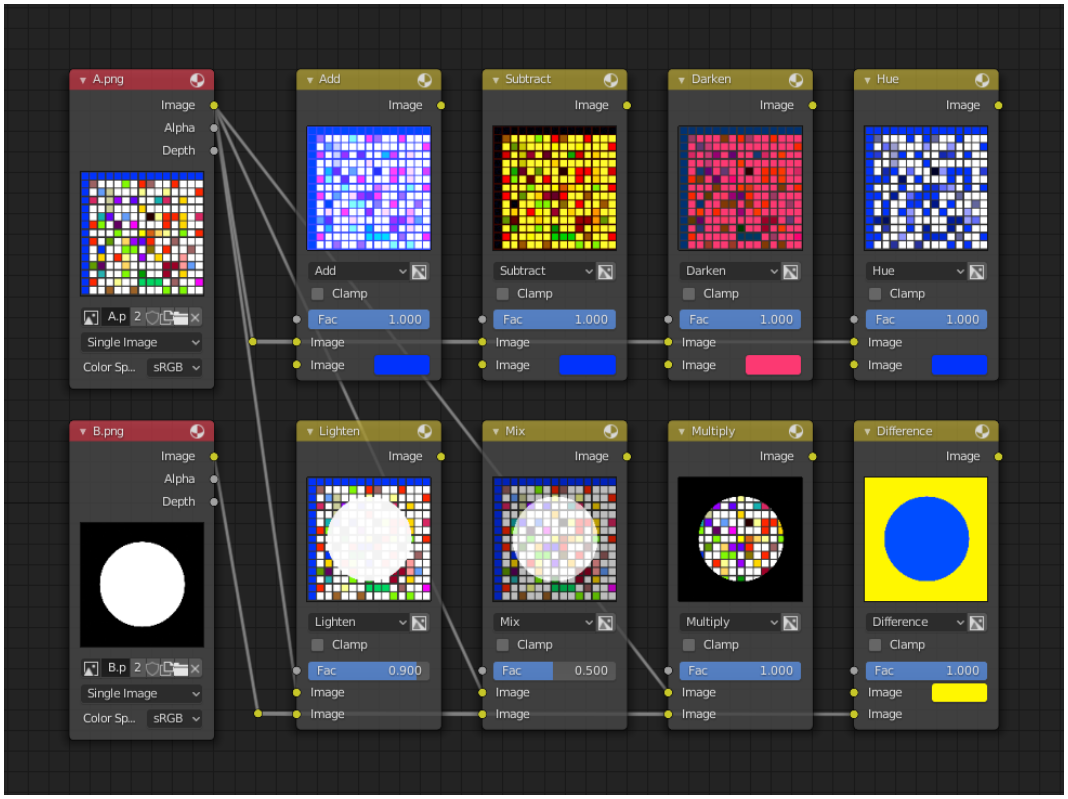
Clamp
: Clamp the output value to the [0.0, 1.0] range.

Outputs

Image
: The result of the mixing operation.

Examples

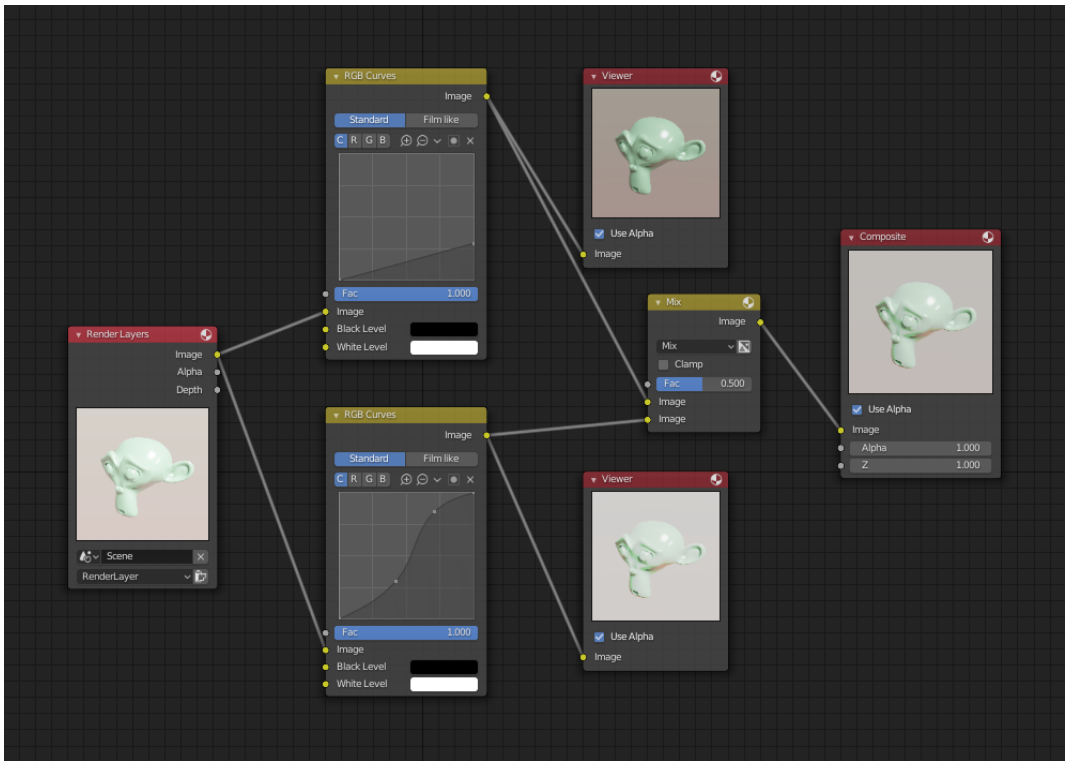
Below are examples of blending modes, as well as some practical use cases.



Blending a colored pattern with a flat color (top row) and a circular mask (bottom row).

Fixing overexposure

The Compositing setup below shows how to fix an overexposed render by darkening it and increasing contrast.



Example node setup showing two RGB Curves nodes and a Mix node for composition.

The top RGB Curves Node darkens the image by linearly scaling each color value to a smaller one.

The bottom curve node increases constract by making small values smaller and large values larger.

Finally, the Mix node blends the two together.

Watermark Images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising.

Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting

service, or simply to track their image's proliferation throughout the web.

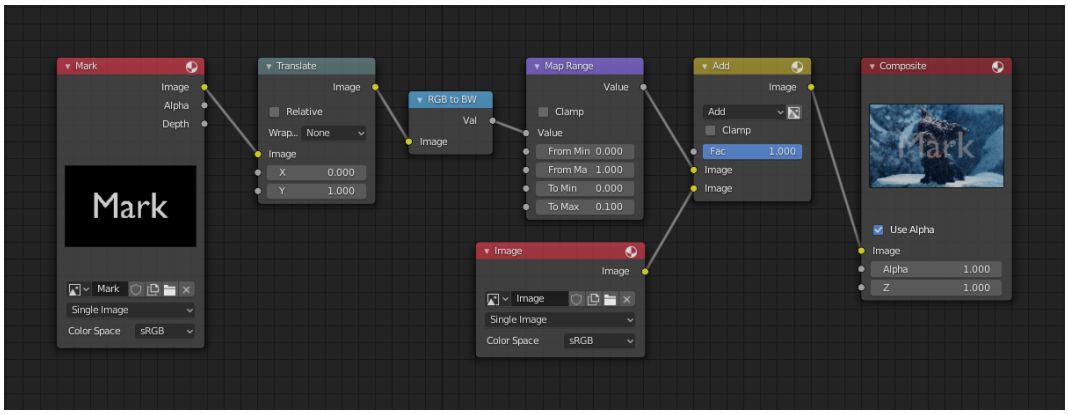
Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you.

In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the *RGB to BW node* to convert the color image to grayscale numbers, which we then feed into the *Map Range* node to reduce the mark to one-tenth of its original intensity.

The *Add* node (*Mix* node with blending mode *Add*) adds the corresponding pixels, making the ones containing the mark ever-so-slightly brighter.

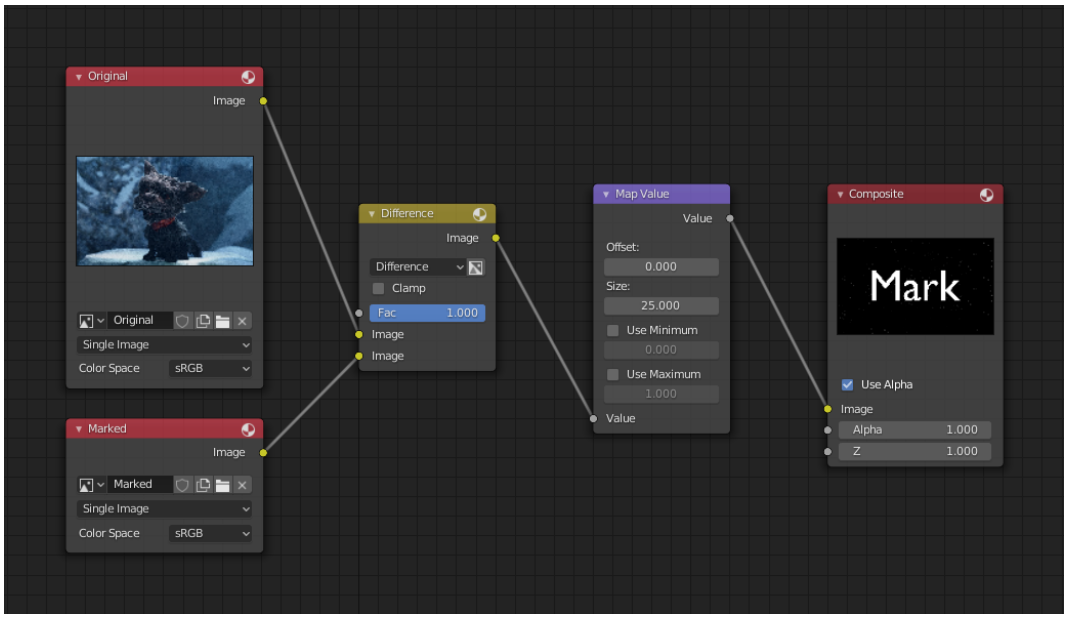


Embedding a watermark in an image.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Decoding an Image for your Watermark

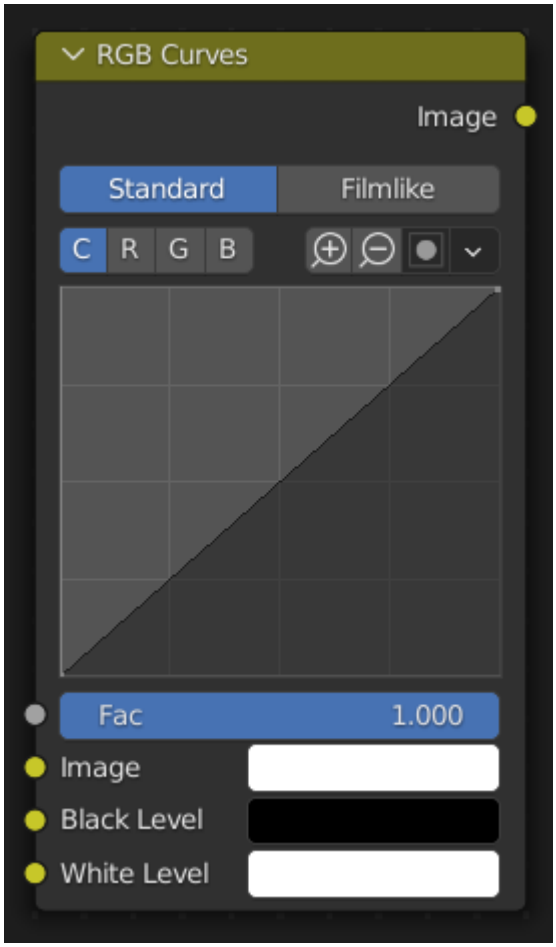
When you see an image that you think might be yours, use the node tree below to compare it to your stock image (pre-watermarked original). In this tree, the *Mix* node is set to Difference, and the *Map Value* node amplifies any difference. You can see how the original mark clearly stands out.



Checking an image for your watermark.

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.



Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*
: Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel
: The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve
: A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

Outputs

Image/Color
: Standard color output.

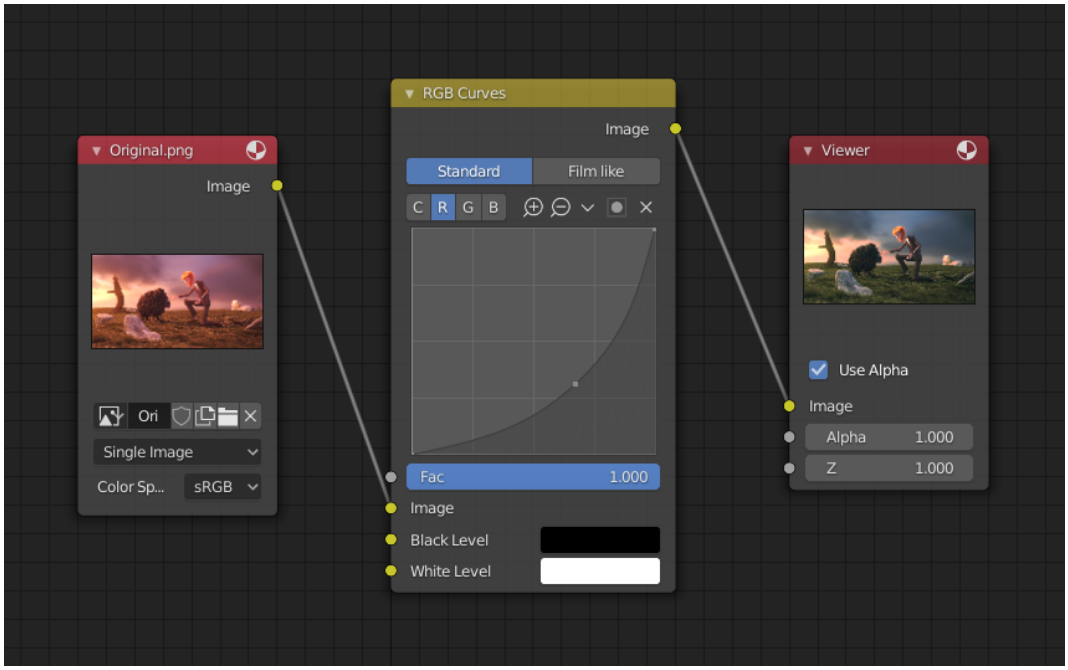
Examples

Below are some common curves you can use to achieve desired effects.



From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

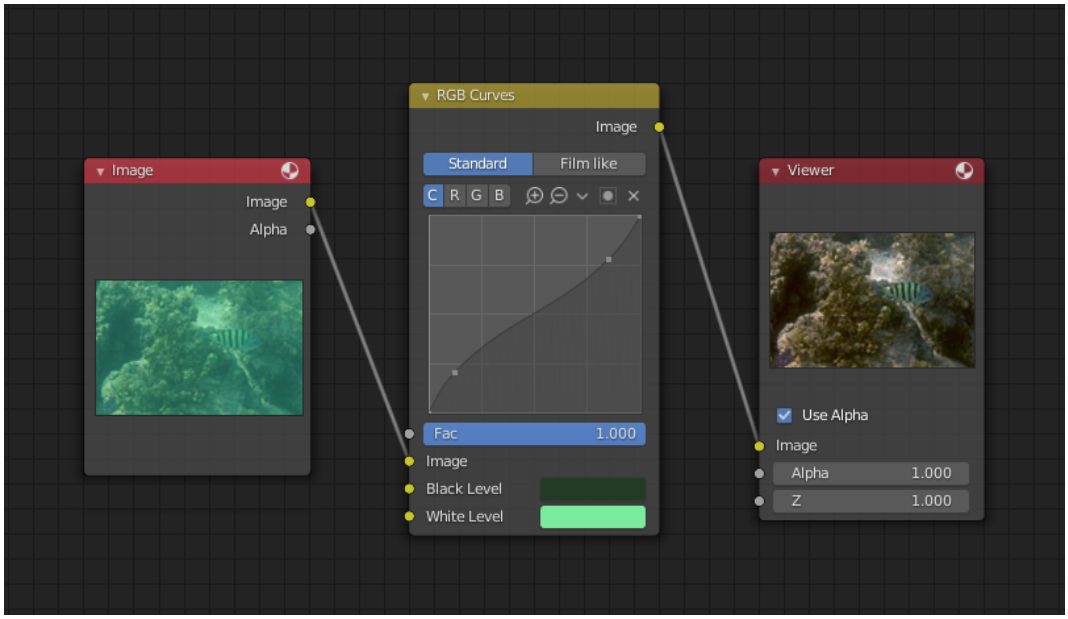


Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



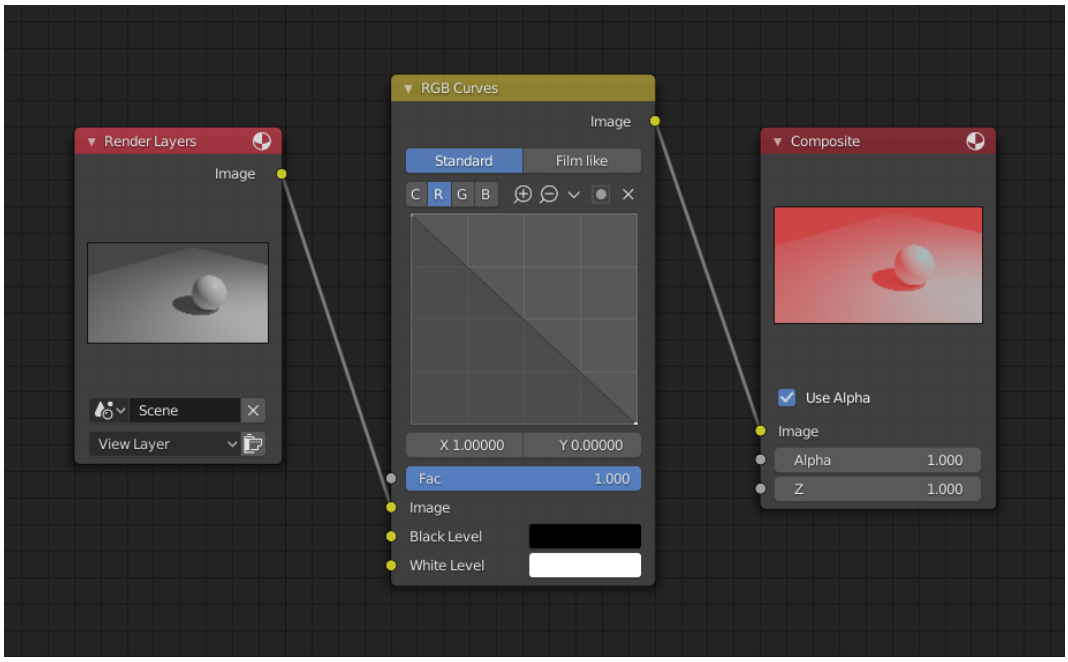
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

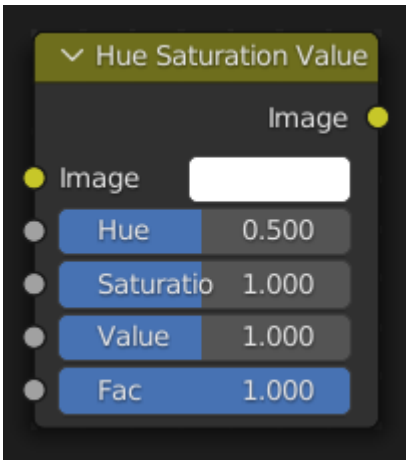
Effects



Changing colors by inverting the red channel.

Hue/Saturation/Value Node

The *Hue/Saturation/Value Node* applies a color transformation in the HSV Color Model.



Inputs

Factor
: The amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Hue
: The hue rotation offset, from 0 (-180°) to 1 (+180°). Note that 0 and 1 have the same result.

Saturation
: A value of 0 removes color from the image, making it black-and-white. A value greater than 1.0 increases saturation.

Value
: The value shift. 0 makes the color black, 1 keeps it the same, and higher values make it brighter.

Outputs

Image/Color
: Standard color output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are laid out on a circle
: If you apply a Hue offset of 1 (+180°) to a blue image, you get the diametrically opposite color, which is yellow. If you apply a Hue offset of 1 to that yellow image, you get blue again.

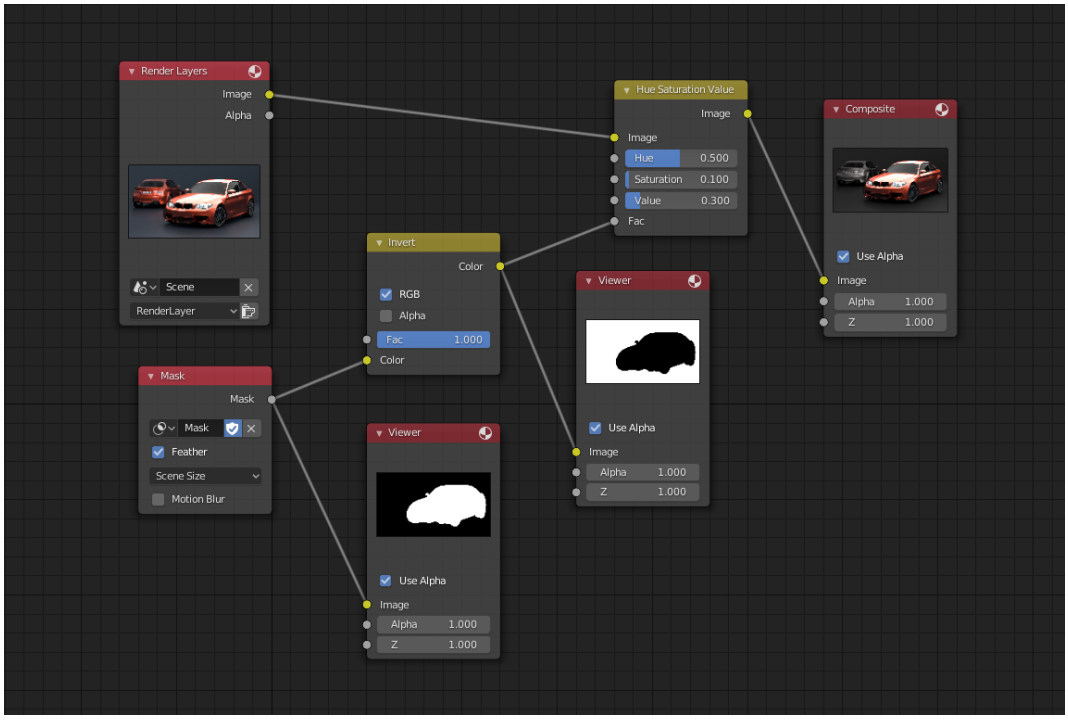
Grayscale images have no hue
: Trying to change the Hue or Saturation of a grayscale image has no effect. You can only brighten or darken it by adjusting the Value. To add color, use the Mix node instead.

Changing the effect over time
: The different values can be animated using a *Time Curve* node or by setting keyframes.

HSV Example



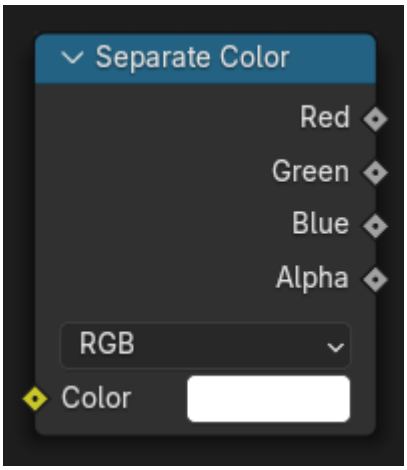
A basic example.



An example of using the Factor input for masking.

Separate Color Node

Splits an image into its channels, based on a particular Color Model.



Inputs

Color
: Standard color input.

Properties

Mode
: The color model to output.

RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Outputs

The outputs of this node depend on the Mode property (see above).

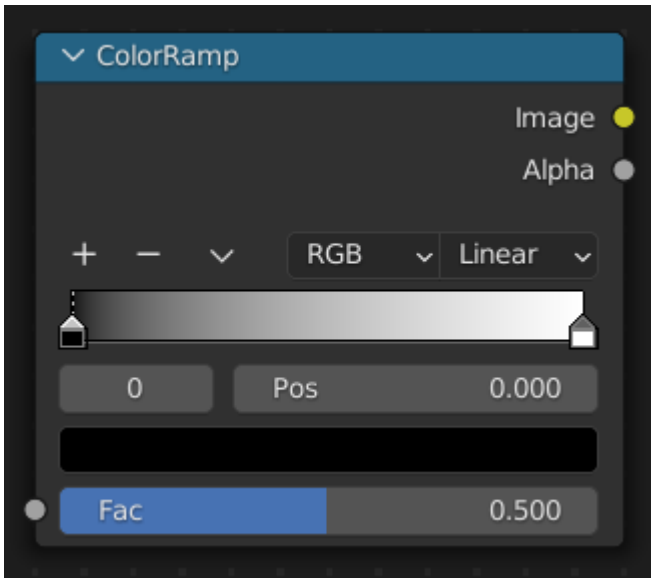
Alpha
: The opacity value.

Converter Nodes

- Color Ramp Node
- Distance Node
- Math Node
- RGB to BW Node
- Value to Normal Node

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Inputs

Factor
: The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp
: See Color Ramp Widget.

Outputs

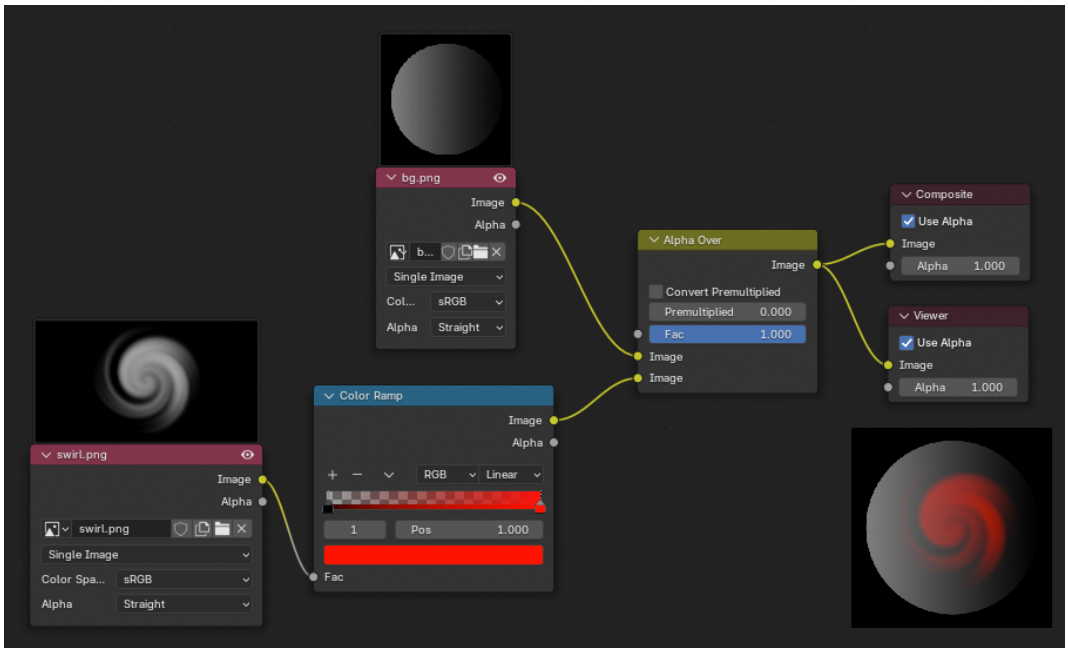
Image/Color
: Standard color output.

Alpha
: Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

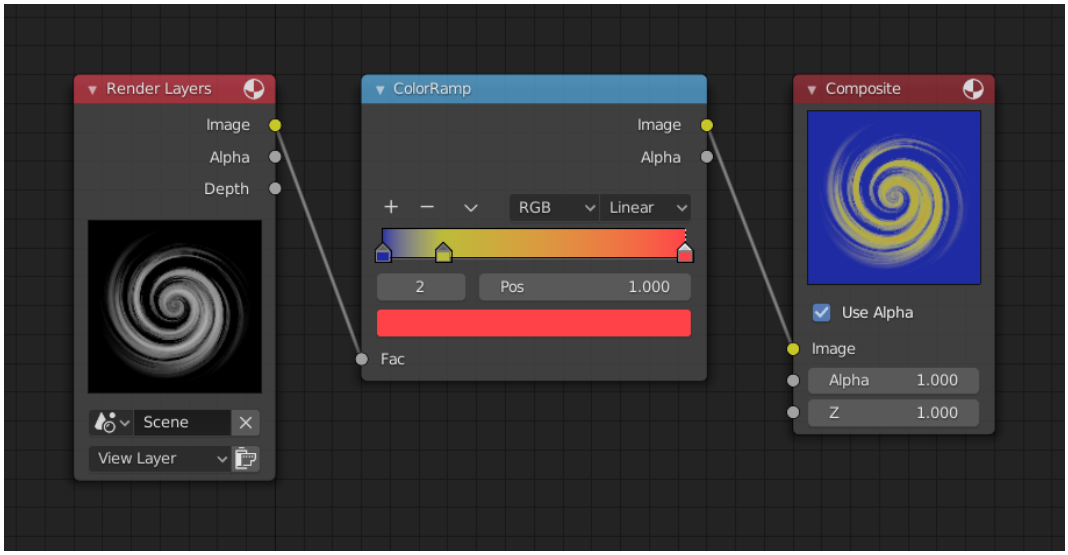


In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

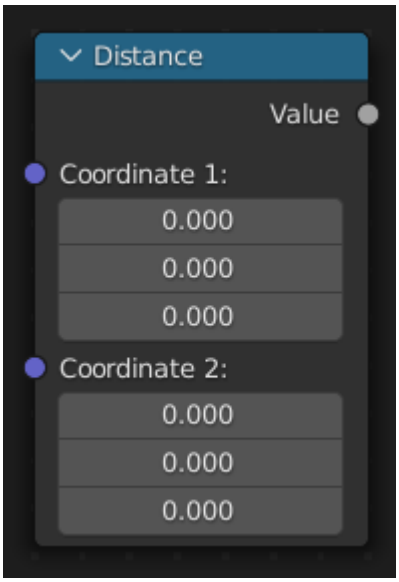
In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.



The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Distance Node

Computes the distance between two 3D coordinates.



Inputs

Coordinate 1
: First coordinate point.

Coordinate 2
: Second coordinate point.

Properties

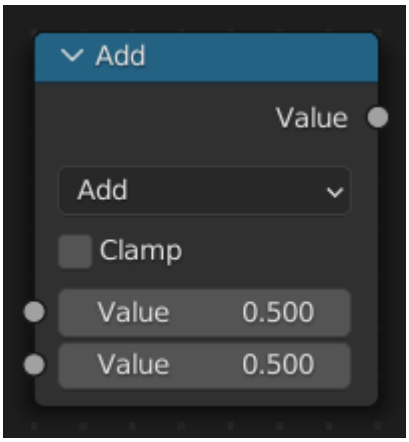
This node has no properties.

Outputs

Value
: Calculated distance.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value
: Input Value. Trigonometric functions read this value as radians.

Addend
: Input Addend.

Base
: Input Base.

Exponent
: Input Exponent.

Epsilon
: Input Epsilon.

Distance

Input Distance.

Min
Input Minimum.

Max
Input Maximum.

Increment
Input Increment.

Scale
Input Scale.

Degrees
Input Degrees.

Radians
Input Radians.

Properties

Operation
The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

One divided by the square root of the value.

Absolute:: The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises Euler's number to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: Smooth Minimum.

Smooth Maximum:: Smooth Maximum.

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo:: Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.

Snap:: Rounds the input value down to the nearest integer multiple of *Increment*.

Ping-pong:: Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine:: The Sine of the input value.

Cosine:: The Cosine of the input value.

Tangent:: The Tangent of the input value.

Arcsine:: The Arcsine of the input value.

Arccosine:: The Arccosine of the input value.

Arctangent:: The Arctangent of the input value.

Arctan2:: Outputs the Inverse Tangent of the first value divided by the second value measured in radians.

Hyperbolic Sine:: The Hyperbolic Sine of the input value.

Hyperbolic Cosine:: The Hyperbolic Cosine of the input value.

Hyperbolic Tangent:: The Hyperbolic Tangent of the input value.

Conversion

To Radians:: Converts the input from degrees to radians.

To Degrees:: Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

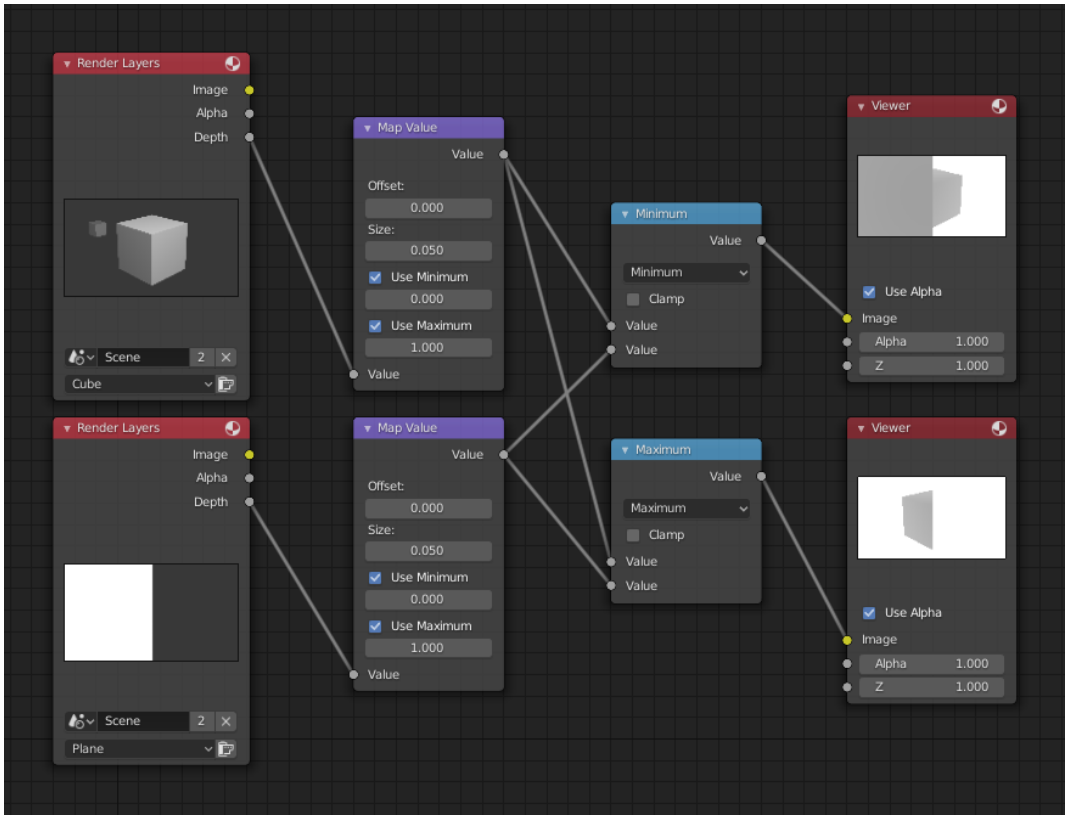
Outputs

Value

Numerical value output.

Examples

Manual Z-Mask



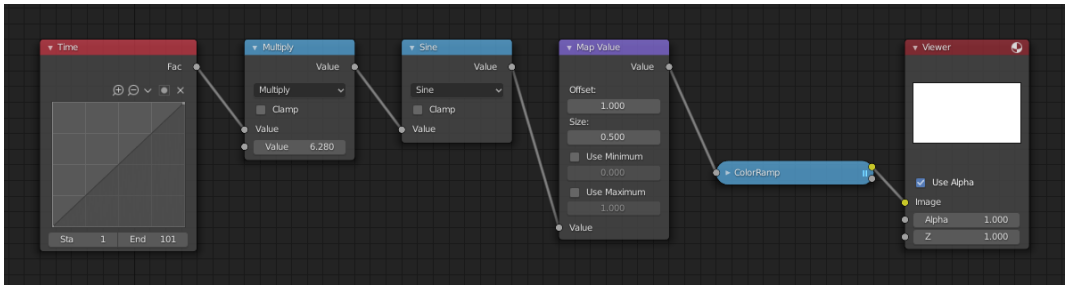
Minimum and maximum function example.

The top *Render Layers* node has a cube that is about 10 units from the camera. The bottom *Render Layers* node has a plane that covers the left half of the view and is 7 units from the camera. Both are fed through their respective *Map Value* nodes to multiply the depth value by 0.05 and clamp it to [0.0, 1.0], bringing it into a suitable range for displaying it as a color.

The Minimum node selects the smallest of the two depth values at each pixel. In the left half, it chooses the plane (because it's closer than the cube), and in the right half, it chooses the cube (because it's closer than the background, which is infinitely far away).

The Maximum node selects the largest of the two depth values at each pixel. In the left half, it chooses the cube (because it's farther away than the plane), and in the right half, it chooses the background (because it's farther away than the cube).

Using Sine Function to Pulsate



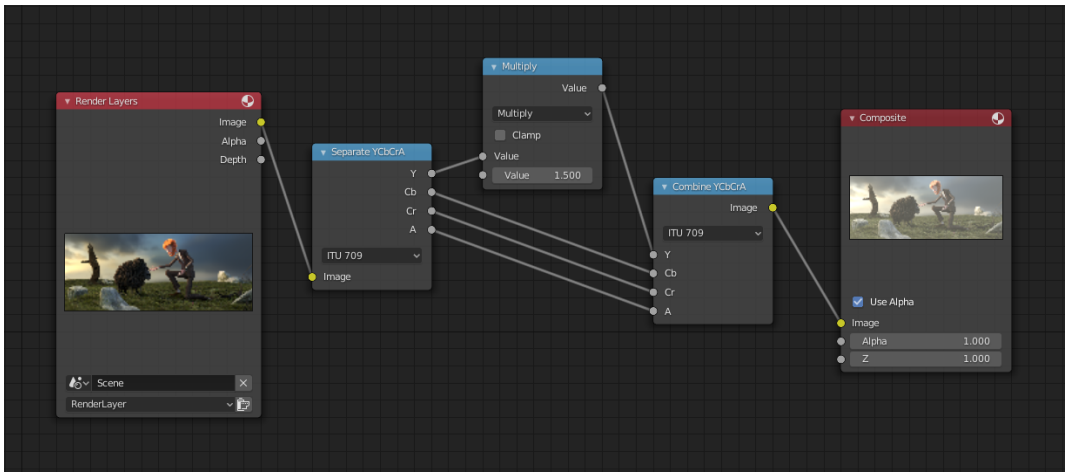
Using sine function example.

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. At frame 25, the output value is 0.25. That value is multiplied by 2 × pi (6.28) and converted to 1.0 by the Sine function, since \(sin(2 × pi/ 4) = sin(pi/ 2) = +1.0\).

Since the sine function can output values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by 0.5 (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a gray-scale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, \(sin(pi/ 2) = 1.0\). Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color “pulse”.

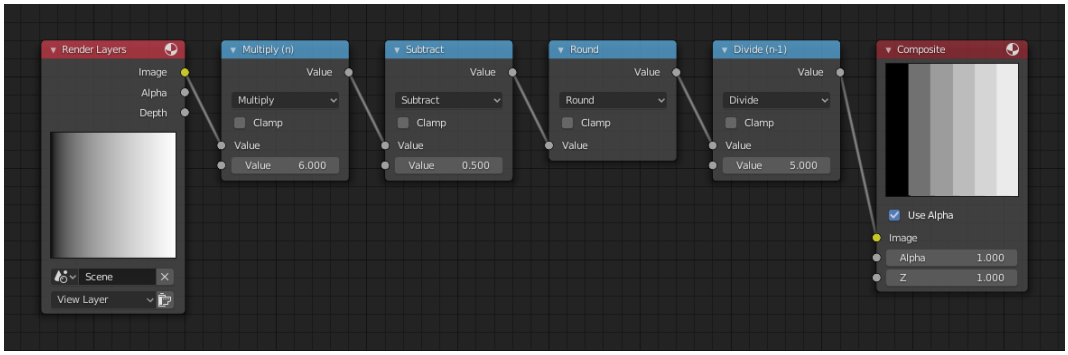
Brightening (Scaling) a Channel



Scaling a channel example.

This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high dynamic range image. For this particular example, there is also a *Brightness/Contrast node* that might give simpler control over brightness.

Restrict Color Selection (Posterization)



Posterization example.

In this example, we restrict the color values to be one of the six values: 0, 0.2, 0.4, 0.6, 0.8, 1.

To split up a continuous range of values between 0 and 1 to certain set of values, the following function is used: \(round(x × n - 0.5) / (n - 1)\), where

“n” is the number of possible output values, and “x” is the input pixel color. Read more about this function.

To implement this function in Blender, consider the node setup above. We string the Math nodes into a function that takes each color (values from 0 to 1), multiplies it up by six, the desired number of divisions (values become from 0 to 6), offsets it by 0.5 (-0.5 to 5.5), rounds the value to the nearest whole number (produces 0, 1, 2, 3, 4, 5), and then divides the image pixel color by five (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

In the case of a color image, you need to split it into separate RGB channels using *Separate/Combine Color* nodes and perform this operation on each channel independently.

RGB to BW Node



The *RGB to BW Node* makes a color image black-and-white by outputting its luminance.

Note

You can directly connect Color sockets to Value sockets in node graphs, which also converts the image to black-and-white. As such, this node is not always necessary.

Inputs

Image
: Color image input.

Properties

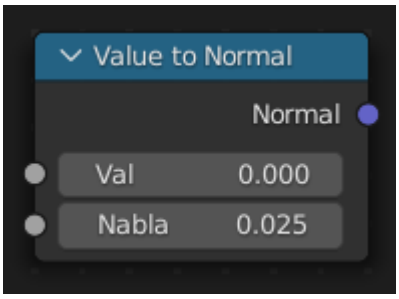
This node has no properties.

Outputs

Value
: Grayscale value output.

Value to Normal Node

Computes a normal map.



Inputs

Val
: The heightmap to compute the normal map from.

Nabla
: Size of derivative offset used for calculating normals.

Properties

This node has no properties.

Outputs

Normal
: Standard normal output.

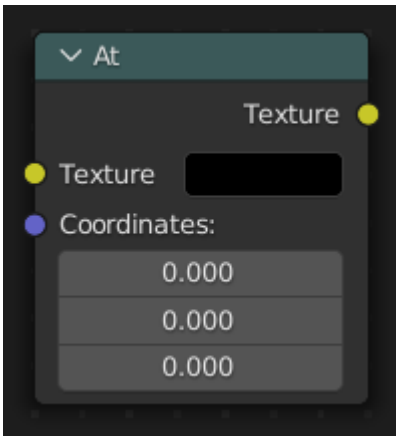
Distort Nodes

These nodes allow you to change the mapping of a texture.

- At Node
- Rotate Node
- Scale Node
- Translate Node

At Node

Returns the color of a texture at the specified coordinates.



Inputs

Texture
: Standard color input.

Coordinates
: The point at which to sample the color. For images, the space is between -1 and 1 for X and Y. If the coordinates are not spatially varying, the node will return a single color.

Properties

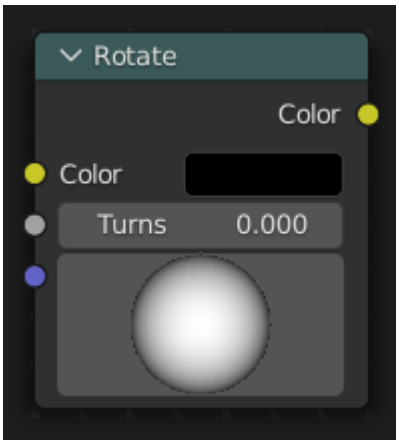
This node has no properties.

Outputs

Texture
: Standard color output.

Rotate Node

Rotate the texture coordinates of an image or texture.



Inputs

Color
: Standard color input.

Turns
: The number of times to rotate the coordinates 360 degrees about the specified axis.

Axis
: The axis to rotate the mapping about.

Properties

This node has no properties.

Outputs

Color
: Standard color output.

Scale Node

Scale the texture coordinates of an image or texture.



Inputs

Color
: Standard color input.

Scale
: The amount to scale the coordinates in each of the three axes.

Properties

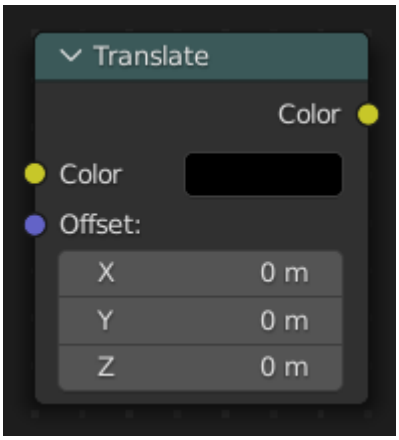
This node has no properties.

Outputs

Color
: Standard color output.

Translate Node

Translate the texture coordinates of an image or texture.



Inputs

Color
: Standard color input.

Offset
: The amount to offset the coordinates in each of the three axes.

Properties

This node has no properties.

Outputs

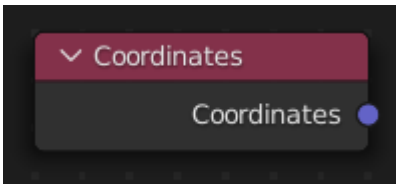
Color
: Standard color output.

Input Nodes

Input nodes provide input data for other nodes.

- Coordinates Node
- Image Node
- Texture Node
- Time Curve Node

Coordinates Node



Inputs

This node has no inputs.

Properties

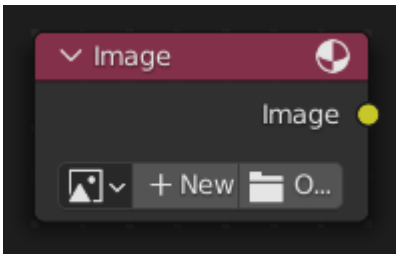
This node has no properties.

Outputs

Coordinates
: The Coordinates node outputs the local geometry coordinates, relative to the bounding box.

Image Node

The Image node can be used to load an external image.



Inputs

This node has no inputs.

Properties

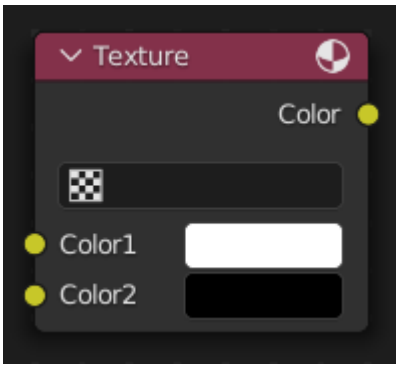
Image
: See Data-Block Menu.

Outputs

Color
: Standard color output.

Texture Node

The Texture node loads another node-based or non-node-based texture.



Inputs

These two colors can be used to remap a grayscale texture.

Color 1
: White Level.

Color 2
: Black Level.

Properties

Texture
: The texture to load – either from the current blend-file, or from a linked one.

Outputs

Color
: Standard color output.

Time Curve Node

The *Time Curve node* generates a factor value (from 0.0 to 1.0) between the scene start and end time, using a curve mapping.



Inputs

This node has no inputs.

Properties

Curve
: The Y value defined by the curve is the factor output. For the curve controls, see Curve widget.

Tip

Flipping the curve around reverses the time input, but doing so is easily overlooked in the node setup.

Start, End
: Start frame and End frame of the range of time specifying the values the output should last. This range becomes the X axis of the graph. The time input could be reversed by specifying a start frame greater than the end frame.

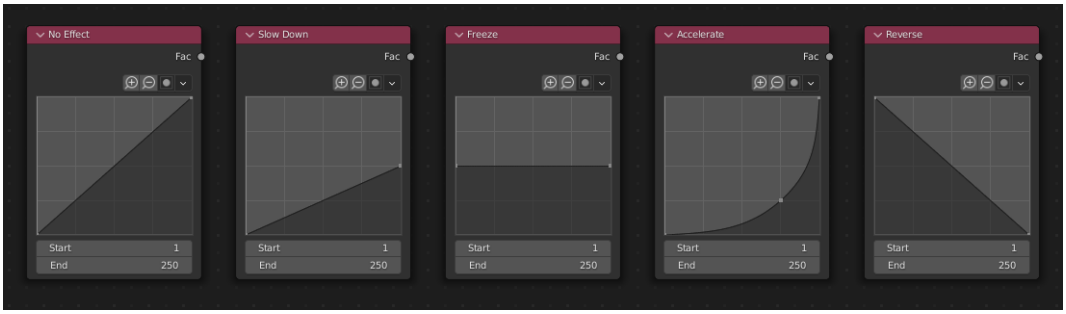
Outputs

Factor
: The Y value of the curve at the current frame.

Hint

The Map Value node can be used to map the output to a more appropriate value. With some curves, it is possible that the Time Curve node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

Example



Time controls from left to right: no effect, slow down, freeze, accelerate, reverse.

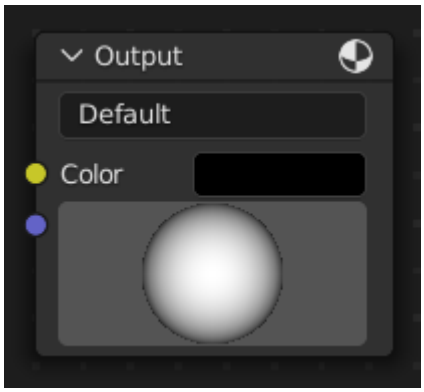
Output Nodes

These nodes serve as outputs for node-based textures.

- Output Node
- Viewer Node

Output Node

This node receives the result of the node-based texture.



Inputs

Color
: The color data that the texture renders.

Properties

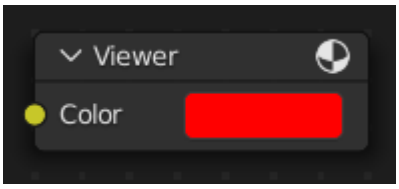
Output Name
: The name of the output. (Originally, it was possible for textures to have multiple outputs with different names.)

Outputs

This node has no outputs.

Viewer Node

The Viewer node can be used to preview the results of a node.



Inputs

Color
: Standard color input.

Properties

This node has no properties.

Outputs

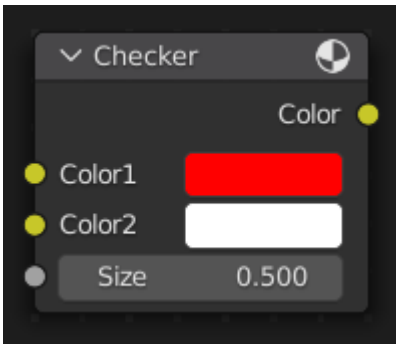
This node has no outputs.

Pattern Nodes

- Checker Node
- Bricks Node

Checker Node

The Checker node creates a checkerboard pattern.



Inputs

Color 1, Color 2
: The colors of the squares.

Size
: The scale of the checker pattern.

Properties

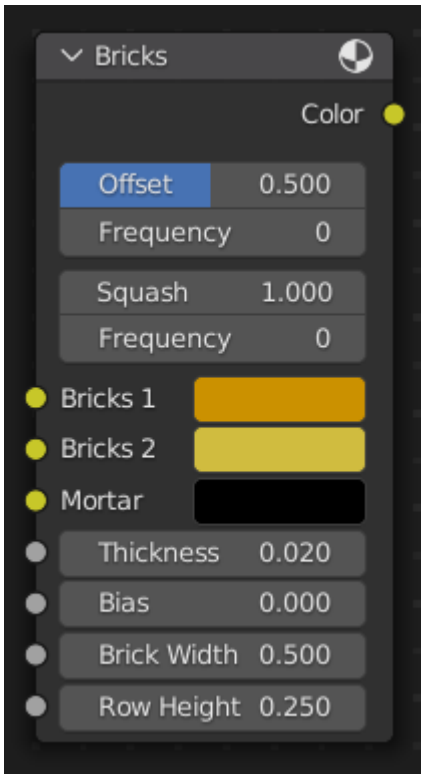
This node has no properties.

Outputs

Color
: Standard color output.

Bricks Node

The Bricks node creates a brick-like pattern.



Inputs

Bricks 1, Bricks 2
: Sets the color range of the bricks. Brick colors are chosen randomly between these two colors.

Mortar
: Sets the mortar color, in between the bricks.

Thickness
: Sets the thickness of the mortar.

Bias
: The bias of randomly chosen colors, between (-1 to 1). -1 Makes all bricks Color 1, and a value of 1 makes them all Color 2.

Brick Width
: Sets the horizontal size of all the bricks.

Row Height
: Sets the vertical size of all the bricks.

Properties

Offset
: The relative offset of the next row of bricks.

Frequency
: Offset every N rows. The brick pattern offset repeats every N rows.

Squash
: Scales the bricks in every N rows by this amount.

Frequency
: Squash every N rows.

Outputs

Color
: Standard color output.

Texture Nodes

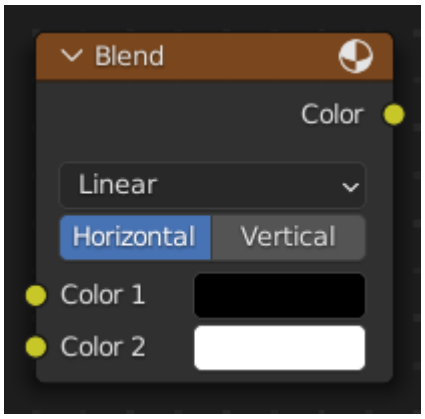
These nodes generate procedural textures, and function just like their non-node-based counterparts.

Common Options

Color 1/Color 2
: Remaps the procedural texture with these colors. These do not function in the Magic node.

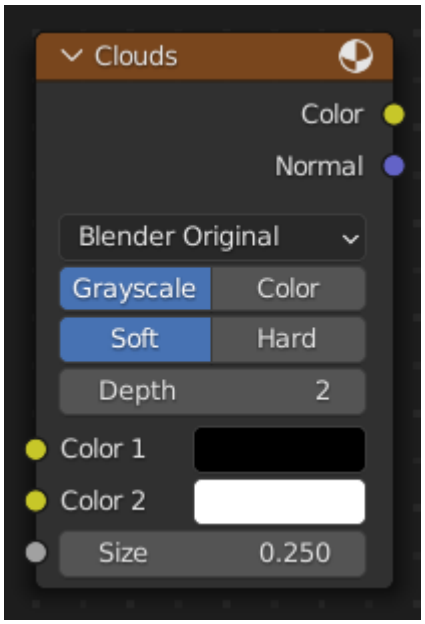
- Blend Node
- Clouds Node
- Distorted Noise Node
- Magic Node
- Marble Node
- Musgrave Node
- Noise Node
- Stucci Node
- Voronoi Node
- Wood Node

Blend Node



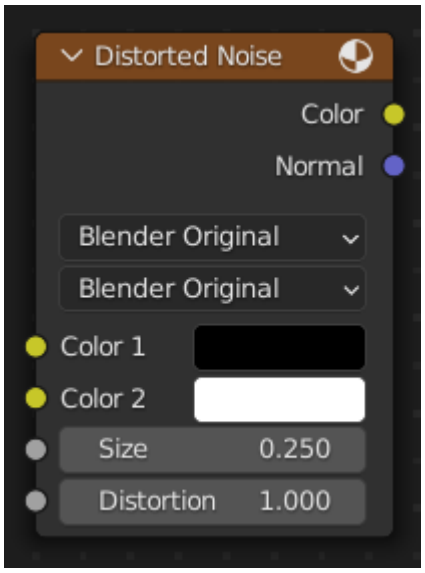
See Here.

Clouds Node



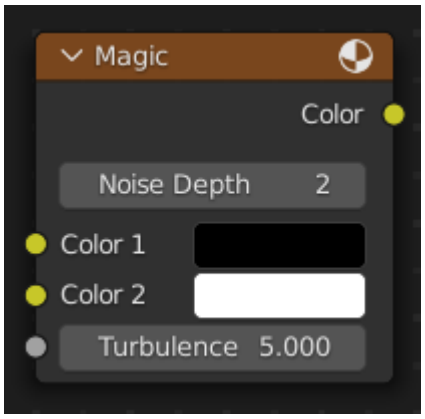
See Here.

Distorted Noise Node



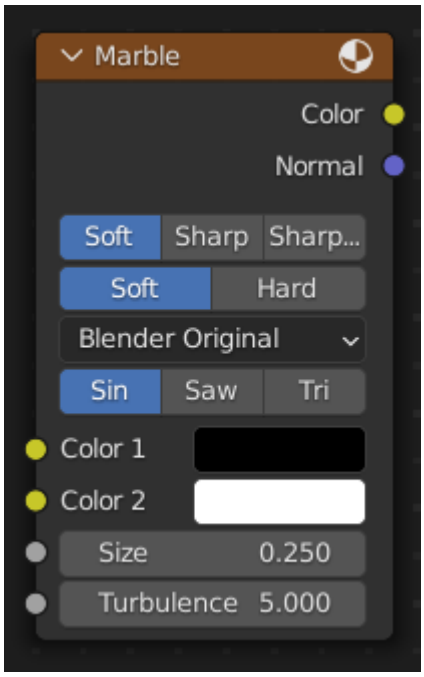
See Here.

Magic Node



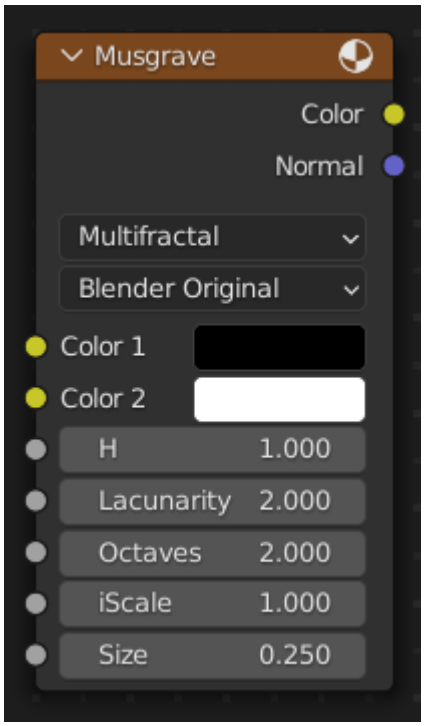
See Here.

Marble Node



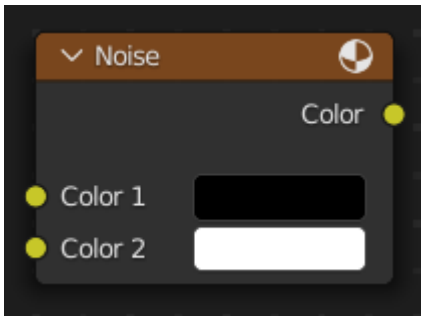
See Here.

Musgrave Node



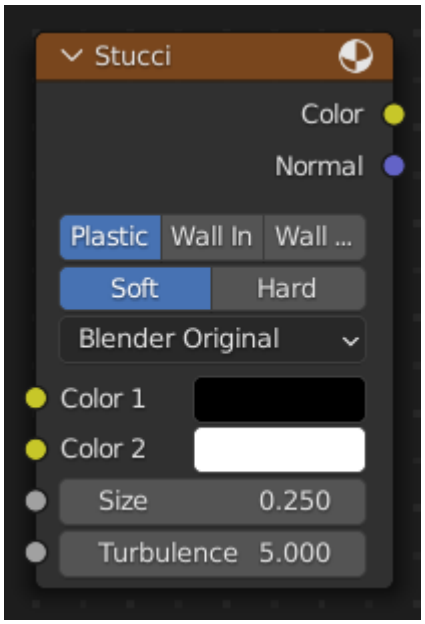
See Here.

Noise Node



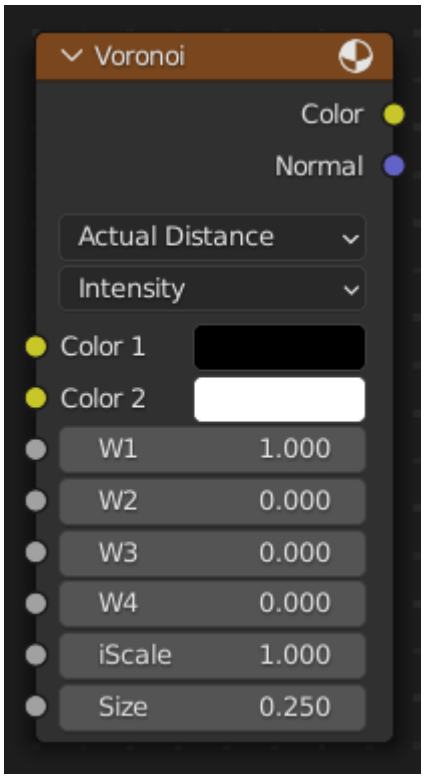
See Here.

Stucci Node



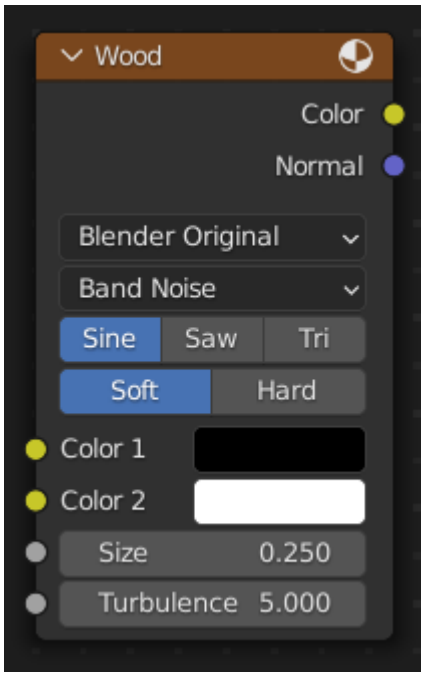
See Here.

Voronoi Node



See Here.

Wood Node



See Here.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Geometry Node Editor

The Geometry Node editor is used to edit Node Groups which are used by the Geometry Node Modifier. Such a node group can define many operations to modify an object's geometry.

A list of all Geometry Nodes is available in the modeling section.

Interface

Header

Geometry Nodes Type
: Geometry Nodes can have multiple contexts depending on the intended function of the node group. Changing the context adjusts the user interface to best fit the needs of the selected context.

Modifier::	Used to create node groups that will be used by the Geometry Nodes Modifier.
Tool::	Used to create node groups that will be used to create Node-Based Tools.

View
: Standard view menu.

Select
: Menu for Selecting Nodes.

Add
: Menu for adding new Geometry Nodes.

Node
: Menu for Editing Nodes.

Geometry Node Group
: Data-Block Menu for creating and selecting node groups.

Pin (pin icon)
: The pin button will keep the current node group selection fixed, instead of using the Active Modifier. When a node group is pinned, it will remain visible in the Geometry Node editor even when another object or modifier is selected elsewhere.

Parent Node Tree
: Jumps up a node group level. See Edit Group for details.

Snapping
: Snapping options. See Arranging Nodes for details.

Overlays
: See Overlays.

Toolbar

Select
: See Selecting Nodes.

Annotate
: See Annotations.

Links Cut
: See Cut Links.

Sidebar

Node

This tab gives access to the active node's properties.

Tool

This tab gives access to the active tool's settings.

View

This tab allows managing annotations.

Group

This tab allows you to edit the current node group's inputs and outputs.

> **Tip**
>
> In the Geometry Node Modifier, you can specify values for the root node group's inputs, as well as select destination Attributes for its outputs.

Tool Context

These popover menus are displayed in the header when the tool context is enabled. These properties determine where the tool is available in the user interface.

See Supported Modes & Data-Types for more information.

Types

The Object Types the tool supports.

Mesh
: The node tree supports Mesh Objects.

Curves
: The node tree supports Curve Objects.

Modes

The Object Modes the tool supports.

Object Mode
: The node group can be used in Object mode.

Edit Mode
: The node group can be used in edit mode.

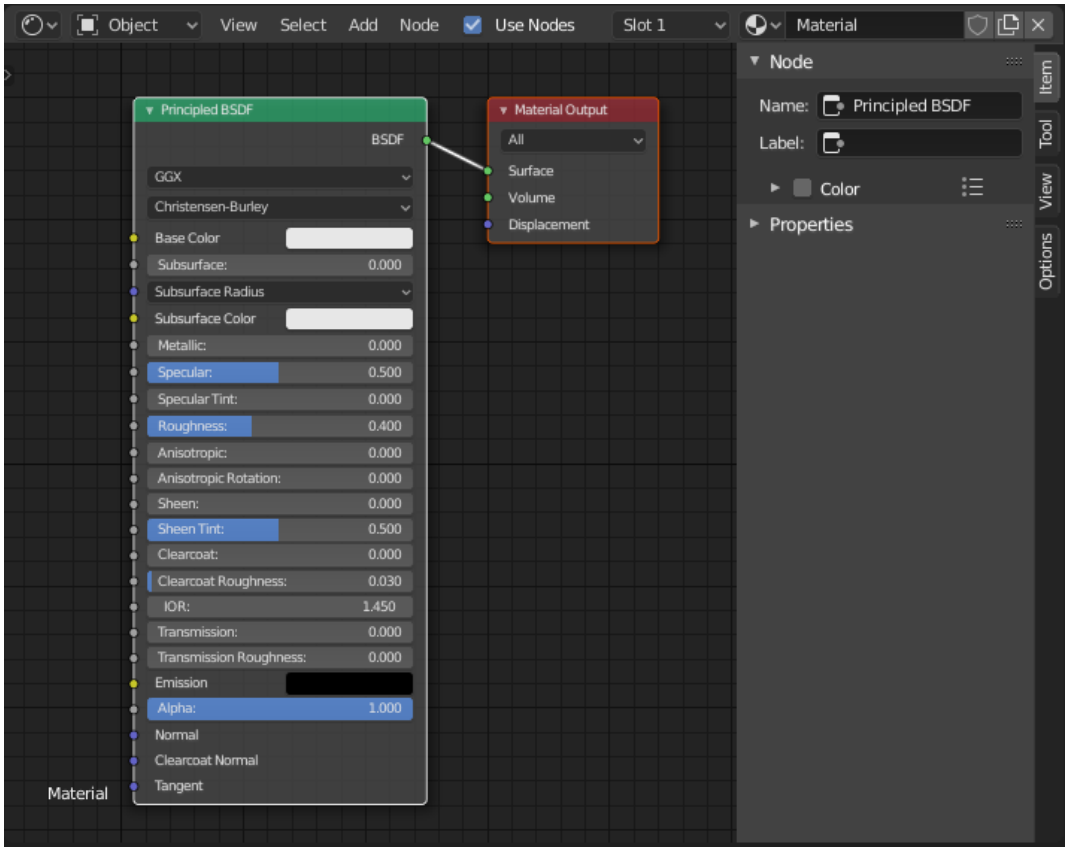
Sculpt Mode
: The node group can be used in Sculpt Mode.

Options

Wait for Click
: Wait for a mouse click input (`LMB`) before running the operator from a menu. This is useful for the Mouse Position Node.

Shader Editor

The Shader Editor is used to edit materials which are used for rendering. Materials used by Cycles and EEVEE are defined using a node tree. Therefore, the main window of the Shader editor is a node editor.



Shader Editor with the default material node tree.

A list of all shader nodes is available in the rendering section.

Header

Use Nodes

The Use Nodes setting is mostly a legacy setting and should always be checked for materials.

Slot
: The *Slot* menu can be used to select the active material slot on the active object. The material selector to the right of it can change the material that is in the selected slot.

Pin (pin icon)
: The pin button will keep the current material selection fixed. When a material is pinned, it will remain visible in the Shader editor even when another object or material is selected elsewhere.

Sidebar

Options

The Options panel in the Sidebar region contains the same settings that are also available in the Material tab in the Properties. They differ depending on the selected render engine. The settings are duplicated to make it possible to edit the entire material from the Shader editor.

Video Sequencer

- Introduction
  - Editor Layout
  - View Types
  - Performance

View Types

- Sequencer
  - Introduction
  - Channels
  - Navigating
  - Toolbar
  - Sidebar
  - Display
- Preview
  - Introduction
  - Header
  - Toolbar
  - Sidebar
  - Controls
  - Display
- Sequencer & Preview

Introduction

The Video Sequencer allows you to place images, videos, sounds, and scenes on a timeline and combine them into a new video. This section only describes its UI; to read more about its usage, see the Video Editing section.

Editor Layout

The Video Sequencer is composed of multiple regions. They are described in more detail in the next sections. Figure 1 shows the combined *Sequencer & Preview* view type:

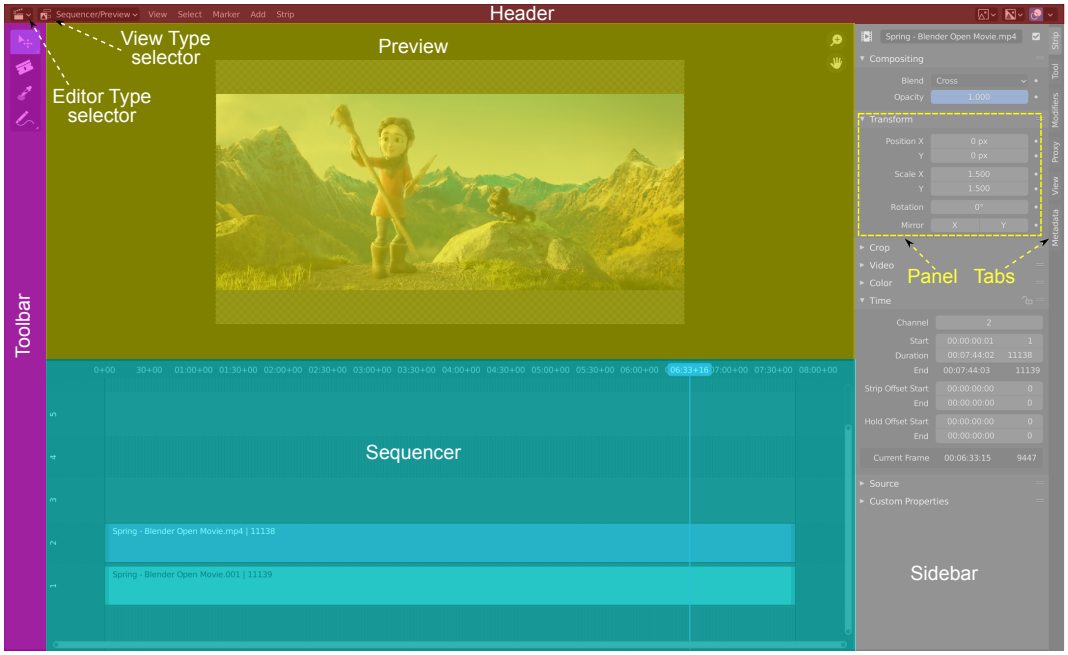


Figure 1: The Video Sequencer Editor shown in the Sequencer & Preview view type.

Header
: Contains menus and buttons for interacting with the editor. The header changes slightly depending on the selected view type (see below).

Preview
: Shows the output of the Sequencer at the time of the Playhead.

Sequencer
: Shows a timeline for managing the montage of strips.

Sidebar
: Shows the properties of the active strip. It's divided into panels and tabs. Toggle on or off with `N`.

Toolbar
: Shows a list of tools. Toggle on or off with `T`.

View Types

The Video Sequencer has three view types which can be changed using the View Type selector (see figure 1; top left).

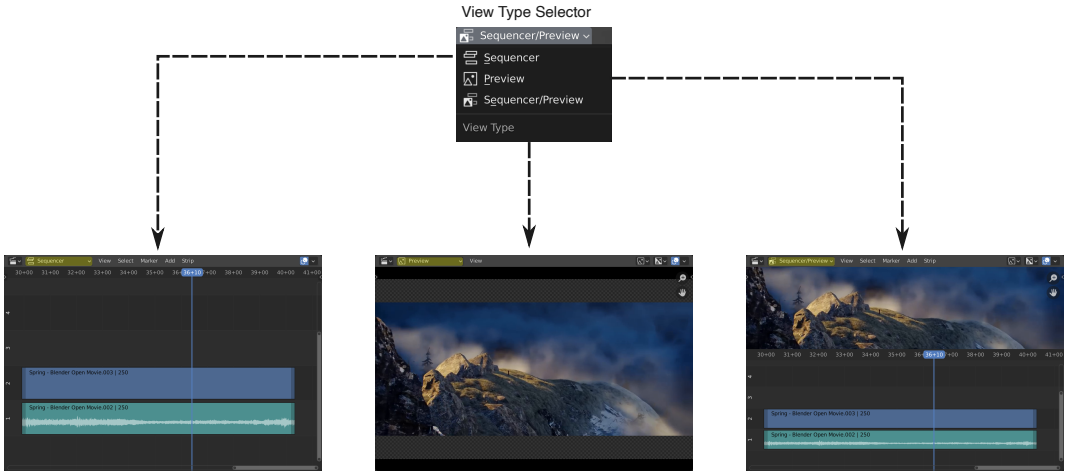


Figure 2: Three view types for the Video Sequence Editor

Sequencer
: View timeline and strip properties.

Preview
: View preview window and preview properties.

Sequencer & Preview
: Combined view of preview and timeline and their properties.

Tip

Rather than having one Video Sequencer in the *Sequencer & Preview* mode, it can be more useful to have one in the *Sequencer* mode and another in the *Preview* mode, the reason being that *Sequencer & Preview* lacks most of the *Preview* tools. Blender's default *Video Editing* workspace offers this layout.

Performance

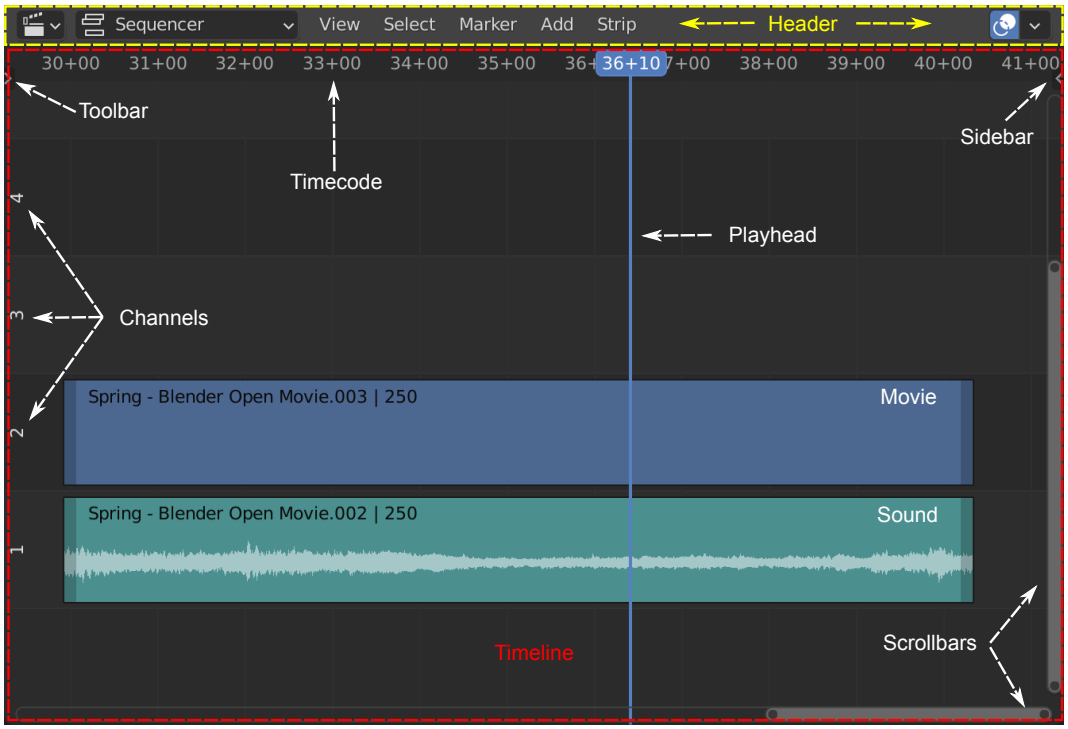
Playback performance can be improved in several ways.

The method with the most impact is to allow the Video Sequencer to cache generated frames. There are two levels of cache: a memory cache, which is enabled by default (and can be enlarged if RAM allows), and a disk cache, which is slower but has more capacity. Both of these can be configured in the Preferences.

Another way to improve performance is by using Strip Proxies. These are copies of source images and videos with a lower resolution and/or quality, making them faster to load than the originals.

Sequencer

The Sequencer view type shows a timeline and allows placing and editing strips.



The Sequencer view and its components.

- Introduction
- Channels
  - Channel Region
- Navigating
  - Header
  - Main View
- Toolbar
  - Introduction
  - Blade
- Sidebar
  - Strip

    - Modifiers
    - Cache
    - Proxy
- Display
    - Sequencer Overlays

Introduction

The Sequencer view is where most of the video editing happens. It shows a stack of channels, in which you can create strips.

Channels

A channel is a horizontal track that's similar to a layer in an image editing program: higher channels are displayed in front of lower ones.

Within each channel, you can create one or more strips, which contain either a segment of video content (a rendered scene, an external video file…) or an effect (color blending, blurring…). The X axis represents time, so the further a strip is placed to the right, the later it will play in the final video.

While a channel can contain multiple strips, they can't overlap each other. If you want two strips to play at the same time, you need to place them in different channels.

Channel Region

The Channel region sits on the left side of the editor and contains the channel properties listed below. Its visibility can be toggled with *View* ▸ *Channels*.

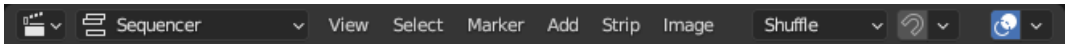
Name
: The name of the channel. Double-click to change.

Mute Channel
: Disable the entire channel so that none of its strips can be seen (or heard) in the final video. Note that you can also mute individual strips.

Lock Channel
: Lock the entire channel to protect all its strips against accidental changes. Note that you can also lock individual strips.

Navigating

Header



Video Sequencer Header.

View Menu

The View menu controls the editor's view settings.

Toolbar `T`
: Show or hide the Toolbar.

Sidebar `N`
: Show or hide the Sidebar.

Tool Settings
: Show or hide the settings for the currently selected tool.

Adjust Last Operation
: Displays a pop-up panel to alter properties of the last completed operation. See Adjust Last Operation.

Channels
: Show or hide the Channel Region.

Preview as Backdrop
: Display the current frame in the background.

Refresh All `Ctrl-E`

Reloads external files and refreshes the current frame preview. This is useful when you modified an external file or made a change in a scene that Blender didn't detect.

Frame Selected `NumpadPeriod`
: Zooms the display to show only the selected strips.

Frame All `Home`
: Zooms the display to show all strips.

Frame Scene/Preview Range
: Reset the horizontal view to the current scene frame range, taking the preview range into account if it is active.

Go to Current Frame `Numpad0`
: Centers the horizontal timeline on the current frame.

Zoom to Border `Shift-B`
: Click and drag to draw a rectangle and zoom to this rectangle.

Limit View to Contents
: Prevents you from panning higher than the highest used channel.

Show Markers
: Shows the marker region. When disabled, the *Marker* menu is also hidden and marker operators are not available in this editor.

Show Seconds `Ctrl-T`
: Shows seconds instead of frames on the time axis.

Sync Visible Range
: Synchronizes the horizontal panning and scale of the editor with other time-based editors that also have this option enabled. That way, they always show the same section of time.

Navigation

Play Animation `Spacebar`

Start or stop animation playback. This will start playback in all editors.

Go to Current Frame `Numpad0`

Scrolls the timeline so the current frame is in the center.

Jump to Previous Strip `PageDown`

Moves the playhead to the nearest strip border (start or end) that's before the current frame.

Jump to Next Strip `PageUp`

Moves the playhead to the nearest strip border (start or end) that's after the current frame.

Jump to Previous Strip (Center) `Alt-PageDown`

Moves the playhead to the nearest strip center that's before the current frame.

Jump to Next Strip (Center) `Alt-PageUp`

Moves the playhead to the nearest strip center that's after the current frame.

Range

Set Preview Range `P`

Interactively define the frame range used for preview playback/rendering.

As long as this range is active, playback will be limited to it, letting you repeatedly view a segment of the video without having to manually rewind each time. It also limits the range that gets rendered by *Sequence Render Animation* (see below).

Set Preview Range to Strips

Apply a preview range that encompasses the selected strips.

Clear Preview Range `Alt-P`

Clears the preview range.

Set Start Frame `Ctrl-Home`
: Set the Start frame of the scene to the current frame.

Set End Frame `Ctrl-End`
: Set the End frame of the scene to the current frame.

Set Frame Range to Strips
: Set the Start and End frames of the scene so they encompass the selected strips.

Sequence Render Image
: Show the current frame preview as a Render Result where you can save it as an image file.

Sequence Render Animation
: Save previews of the frames in the scene range (or the preview range, if active) to a video file or a series of image files. See the Output panel for details.

Note

Sequence Render Image and *Sequence Render Animation* don't render the final video by default – specifically, they don't render Scene Strips, instead using the preview's shading mode (which is initially Solid).

To output a video where the Scene Strips are rendered, use the *Render* menu in the top-bar, or change *Sidebar* ▸ *View* ▸ *Scene Strip Display* ▸ *Shading* to *Rendered*. The latter option is only available if the Video Sequencer is in the *Preview* or *Sequencer & Preview* mode.

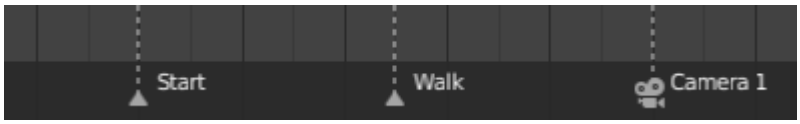
Export Subtitles
: Exports Text strips, which can act as subtitles, to a SubRip file (`.srt`). The exported file contains all Text strips in the video sequence.

Toggle Sequencer/Preview `Ctrl-Tab`
: Switch the editor mode between *Sequencer* and *Preview*.

Area
: Area controls. See the user interface documentation for more information.

Marker Menu

Markers are used to denote frames with key points or significant events within an animation. Like with most animation editors, markers are shown at the bottom of the editor.



Markers in animation editor.

See Editing Markers for details.

Main View

Adjusting the View

Use these shortcuts to adjust the view:

- Pan: `MMB`
- Horizontal scroll: use `Ctrl-Wheel`, or drag the horizontal scrollbar.
- Vertical scroll: use `Shift-Wheel`, or drag the vertical scrollbar.
- Zoom: `Wheel`
- Scale view: `Ctrl-MMB` and drag left/right (horizontal scale) or up/down (vertical scale). Alternatively, you can drag the circles on the scrollbars with `LMB`.

Playhead

The Playhead is the blue vertical line with the current time at the top. It can be moved in the following ways:

- Jump or scrub: click or drag `LMB` in the scrubbing area at the top of the timeline.
- Jump or scrub (alternative): click or drag `Shift-RMB` anywhere in the timeline. If you start dragging on a strip, that strip will be highlighted and displayed *solo* in the preview (all other strips are temporarily muted).
- Move in single-frame increments: `Left`, `Right`, or `Alt-Wheel`.
- Jump to the start or end frame of the scene (or preview range, if active): `Shift-Left` or `Shift-Right`.

While dragging with `LMB` or `Shift-RMB`, you can additionally hold `Ctrl` to snap to the start and end points of strips.

If scrubbing (or regular playback) performs poorly, you can speed it up by creating proxies.

Hint

The current frame is synchronized across all editors, so if you move the Playhead in the Timeline editor for example, it will move in the Video Sequence editor as well (and vice versa).

Toolbar

- Introduction
- Blade

Introduction

Select Box
: Selects or moves from the same tool. Click and drag to create a “box” selection.

Blade
: Create a cut along the strip.

Blade

Reference

Mode::	Sequencer Mode
Tool::	*Toolbar ▸ Blade*
Shortcut::	`K`, `Shift-K`

Cuts a strip in two. Specifically, it first shortens the strip so it only shows the content up to the cut point, then adds a second strip that shows the content after the cut point.

Splitting be done in two different ways:

- Select the tool in the Toolbar and click a strip at the time point where you want to split it.
- Alternatively, select one or more strips, place the Playhead at the time point where you want to split them, and press one of the keyboard shortcuts below.

You can choose between the following split types:

Soft `K`
: After splitting, it's still possible to restore the cut content in the new strips by dragging their handles.

Hard `Shift-K`
: After splitting, it's not possible to restore the cut content by dragging handles. However, you can still restore it by changing the Hold Offset in the Sidebar.

Sidebar

- Strip
  - Header
  - Compositing
  - Transform
  - Crop
  - Video
  - Color
  - Sound
  - Time
  - Source
- Modifiers
  - Common Options
  - Types
- Cache
  - Cache Settings
  - Strip Cache
- Proxy
  - Proxy Settings
  - Strip Proxy & Timecode

Strip

Header

Type
: Strip type, represented by an icon.

Name
: A text field to adjust the name of the strip, which is shown on the strip in the timeline.

  Color Tag
  : Strips are given a Default Color based on their type; using the color tag, you can assign a custom color to help organize your sequence.

Mute
: Uncheck to prevent the strip from producing output.

Compositing

Reference

Panel:: *Sidebar ▸ Strip ▸ Compositing*

Blend
: The method for blending the current strip with strips in lower channels. See Blend Modes for more information.

Opacity
: The opacity (alpha) of the strip.

  When this property is animated, the opacity is drawn as an overlay on the strip. The overlay will look like a dark section that follows the

animation curve. This can be hidden by disabling the F-Curves.

Transform

Reference

Panel:: *Sidebar ▸ Strip ▸ Transform*

Filter
: The technique used to estimate the values of pixels at non-integer coordinates within the image.

Auto::	Automatically choose filter based on scaling factor. • No scale, no rotation, integer positions: *Nearest* • Scaling up by more than 2x: *Cubic Mitchell* • Scaling down by more than 2x: *Box* • Otherwise: Bilinear
Nearest::	No interpolation; uses nearest neighboring pixel (fastest).
Bilinear::	Interpolate between 2×2 samples.
Cubic Mitchell::	Cubic Mitchell filter on 4×4 samples.
Cubic B-Spline::	Cubic B-Spline filter (blurry but no ringing) on 4×4 samples.
Box::	Averages source image samples that fall under destination pixel.

Position X, Y
: Used to move the frames along the X and Y axis.

Scale X, Y
: Scale the image on the X and Y axis.

Rotation
: Rotates the input two-dimensionally along the Z axis.

Mirror
: Mirrors the image along the X axis (left to right) or the Y axis (top to bottom).

Crop

Reference

Panel:: *Sidebar ▸ Strip ▸ Crop*

Used to crop the source image. Use *Top*, *Left*, *Bottom*, and *Right* to control the number of pixels that are cropped.

Video

Reference

Panel:: *Sidebar ▸ Strip ▸ Video*

Strobe
: Display every nth frame. For example, if you set this to 10, the strip will only display frames 1, 11, 21, 31, 41… of the source.

  It is important to realize that this property is a float value. This allows you to strobe effect synced exactly to a beat.

Reverse Frames
: Plays the strip backwards starting from the last frame in the sequence.

Color

Reference

Panel:: *Sidebar ▸ Strip ▸ Color*

Saturation
: Adjusts the vividness of colors in the image.

Multiply
: Multiplies the colors by this value. This will increase the brightness.

Multiply Alpha
: Multiply alpha along with color channels when using the *Multiply* option.

Convert to Float
: Converts input to float data.

Sound

Reference

Panel:: *Sidebar ▸ Strip ▸ Sound*

Working with sound is documented further at Sound Strip.

Volume
: Adjusts the perceived loudness or intensity of the sound.

  When this property is animated, the volume is drawn as an overlay on the strip. The overlay will look like a dark section that follows the animation curve. This can be hidden by disabling the F-Curves. The value is also reflected in the waveform.

Offset
: Offset of the sound from the beginning of the strip, expressed in seconds.

Mono
: Mixdown all audio channels into a single channel.

Pan

Used to pan the audio between speakers in multichannel audio. Only mono sources can be panned; if the source file is not mono, enable *Mono* to mix the channels together.



This value basically represents the angle at which it's played if you multiply the value by 90 degrees.

For stereo, output panning works from left (-1) to center (0) and finally right (1).

To address rear speakers, you can pan to those with higher values, where -2 is back left and 2 is back right.

Tip

For smooth animation you can assign values outside the soft bounds, since the angle wraps around over multiple rotations.

Note

The number of audio channels can be configured in the Audio Output settings.

Display Waveform

Display an approximate waveform of the sound file inside of the Sound strip. The waveform reflects strip volume and its animation using keyframes.

Clipping audio, i.e. values over 100% amplitude, will be shown in red.

This option is only visible if the Waveforms overlay is set to *Strip*.

Time

Reference

Panel:: *Sidebar ▸ Strip ▸ Time*

The Time panel is used to control source and timeline position of the strip.

Lock (padlock icon in panel header)
: Prevents the strip from being moved.

Show Retiming Keys
: Toggle visibility and selectability of Retiming keys.

Channel
: Changes the channel number, or row, of the strip.

Start
: Changes the starting frame of the strip, which is the same as selecting and moving the strip.

Duration
: Changes the length (in frames) of the strip. This works by changing the end frame, which is the same as selecting and moving the strip's right handle.

End
: Shows the ending time and frame of the strip.

Strip Offset Start/End
: Positive values will move the strip's handles inwards, making it start later than the start of the source material and stop before its end. This lets you trim down the source material to the part you need. You can enable the Offsets overlay to see the start and end of the full source file.

  Negative values will move the strip's handles outwards, making it start earlier than the start of the source material and stop after its end. This lets you show the first and/or last frame as a frozen image for some time.

Instead of adjusting these offsets in the Sidebar, you can also drag the strip's handles.

Hold Offset Start/End
: Used for trimming frames off the start/end of the source material. At first sight, this does the same as the *Strip Offset* properties, but you can in fact combine them to hold (freeze) a frame other than the first or last one. For example, if you set the *Hold Offset Start* to 10 and the *Strip Offset Start* to -20, the video will first show the 11th frame of the source for 21 frames, and then play the remaining frames.

Current Frame
: The Playhead's frame number relative to the start of the strip.

Source

Reference

Panel:: *Sidebar ▸ Strip ▸ Source*

The Source panel shows (and lets you change) the file which the strip points to, as well as how this file should be displayed.

File
: The full path of the source file.

Color Space
: The color space of the source file.

  The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Alpha Mode
: If the source file has an Alpha (transparency) channel, you can choose between Straight Alpha and Premultiplied Alpha.

Stream Index *Movie Strip*
: The video stream to use, in case there are multiple.

Deinterlace
: Applies deinterlacing to analog video.

Source Information
: Displays information about the strip's media.

  Resolution
  : Resolution of the active strip's image output.

  FPS *Movie Strip*
  : The frame rate encoded into the video file. If this value does not match the scene's Frame Rate, the perceived speed of the media will be wrong unless the speed is changed to account for the difference.

Options for Image Strips

Directory
: The directory that contains the source file(s).

Filename
: The name of the source file. For image sequences, this will be different for each frame.

Change Data/Files
: Opens a File Browser to let you select a new set of images (as an alternative to modifying the above textboxes). Same as *Strip* ▸ *Inputs* ▸ *Change Paths/Files*.

Options for Sound Strips

Sound
: Data-block menu to select a sound.

File Path

Path to the file used by the selected sound data-block.

Pack
: Pack the sound into the blend-file.

Caching
: Sound file is decoded and loaded into the RAM.

Source Information
: Displays information about the strip's media.

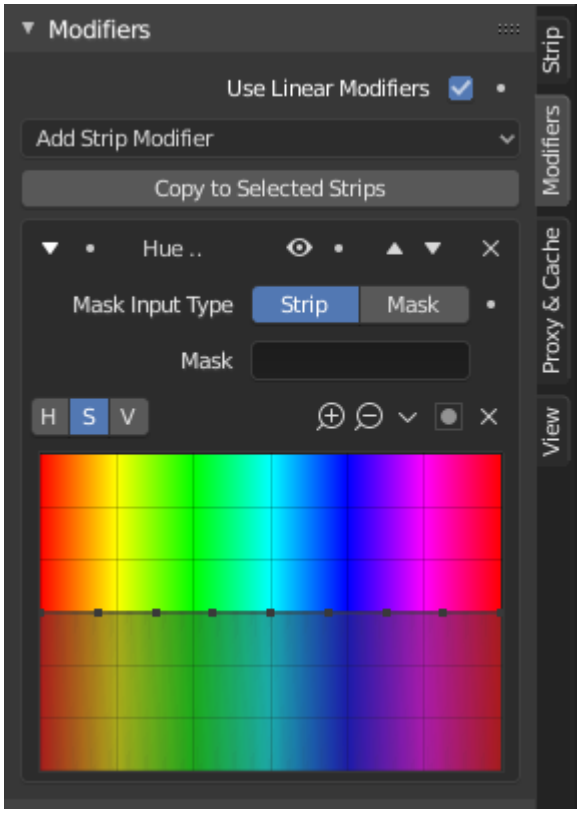
  Samplerate
  : The number of samples per second the audio is encoded at.

  Channels
  : The number of audio channels encoded into the audio stream.

Modifiers

Reference

Panel:: *Sidebar region ▸ Modifiers ▸ Modifiers*



Modifiers are used to make adjustments to the image, like contrast, brightness, saturation, color balance and applying masks.

You can add these modifiers directly to a media strip, or you can use them within an Adjustment Layer strip, making them apply to several media strips in one go.

Linear Modifiers
: Calculates modifiers in linear color space instead of the Sequencer color space.

  Calculating modifiers in linear space will match the image processing of the compositor. In most cases, this should be enabled; working in a non-linear workflow could have unpredictable results.

Copy to Selected Strips
: Copies the modifiers to the selected strips, either replacing their current modifiers or appending to them.

Common Options

Each modifier has several buttons at its top:

Mute (eye icon)
: Disables the modifier. Useful to compare the image with or without modifications.

Move (up/down arrow icon)
: These two buttons change the modifier's position in the stack which affects its computation order.

(Remove Strip Modifier)
: Deletes the modifier from the stack.

Masking

You can mask each modifier to limit the area of the image it affects. This can be done using either a Mask or another strip.

Mask Input Type
: Type of input data used for the mask.

Strip::	Use the grayscale representation of another strip's image.
Mask::	Use a Mask data-block.

Mask
: The Strip or Mask data-block to use.

Mask Time *Mask Input Only*
: How the start frame of the mask is calculated.

Relative::	Mask animation is offset to the start of the strip.
Absolute::	Mask animation is in sync with the scene frame.

Types

Currently, the following modifiers are supported:

Brightness/Contrast Modifier

Adjusts the brightness and contrast of the image.

Color Balance Modifier

Color balance adjustments, either by the Lift/Gamma/Gain or the Offset/Power/Slope method.

This modifier works similar to the Color Balance Node.



Depending on the selected method, the following operations can be applied to the color values in the sequencer color space:

Lift/Gamma/Gain

Lift

Increases the value of dark colors.

Gamma

Adjusts midtones.

Gain

Adjusts highlights.

Offset/Power/Slope (ASC-CDL)

The following formula is applied to each RGB color value separately: \(c_{out} = (c_{in}×s + o)^p\)

Slope

The multiplier \(s\) influences all color values except black. Its effect is stronger the brighter the source color is.

Offset

Shifts color values after applying Slope by adding the Offset \(o\) to them. Note that the selected value shown in the UI will be reduced by 1, so the default value of 1 means effectively no offset is applied.

Power
: Overall exponent \(p\), which mainly adjusts the midtones.

Curves Modifier

Color and RGB curves.

This modifier works the same as the RGB Curves Node.

Hue Correct Modifier

HSV multi points curves.

This modifier works the same as the Hue Correct Node.

Mask Modifier

The mask modifier is used to affect the Alpha Channel of the current strip.

Mask Input Type
: Type of input data used for the mask.

Strip:: Use the grayscale representation of another strip to affect the alpha of the current strip.

Mask:: Use a mask data-block to affect the alpha of the current strip.

Mask
: The Strip or Mask data-block to use.

Mask Time *Mask Input Only*
: How the start frame of the mask is calculated.

Relative:: Mask animation is offset to the start of the strip.

Absolute:: Mask animation is in sync with the scene frame.

Tone Map Modifier

Used to map one set of colors to another in order to approximate the appearance of high dynamic range images in a medium that has a more limited dynamic range.

This modifier works the same as the Tone Map Node.

White Balance Modifier

Used to adjust the white balance by choosing the color that should be white.

Sound Equalizer Modifier

This modifier can be used to emphasize or suppress sound frequencies. The range is limited to 35Hz - 20kHz and +/-35dB.

Cache

The Cache is used to save preview frames in memory, so they can later be displayed much faster than if they were rendered from scratch. Cache capacity can be set in the System tab of the Preferences.

See also

Which frames are cached can be visualized by enabling Show Cache.

Cache Settings

Reference

Panel:: *Sidebar ▸ Cache ▸ Cache Settings*

In this panel, you can select the preview rendering stages at which strip images should be cached. These settings apply to all strips.

Cache
: Raw
  : Cache raw images right after they're read from the drive, for faster tweaking of strip parameters at the cost of memory usage.

: Pre-processed
  : Cache strip images after applying cropping, scaling, saturation and so on, for faster tweaking of effects at the cost of memory usage.

: Composite
  : Cache strip images after blending with lower channels and applying effects, for faster tweaking of stacked strips at the cost of memory usage.

Final
: Cache the final rendered frame.

Display

Visualize cached images on the timeline.

Cache
: Raw
  : Visualize cached raw images on the timeline.

  Pre-processed
  : Visualize cached pre-processed images on the timeline.

  Composite
  : Visualize cached composite images on the timeline.

  Final
  : Visualize cached final images on the timeline.

Strip Cache

Reference

Panel:: *Sidebar ▸ Cache ▸ Cache Settings*

This panel sets the types of images that will be cached for the active strip. When enabled, these properties override the above Cache Settings.

Cache
: Raw
  : Cache raw images read from drive, for faster tweaking of strip parameters at the cost of memory usage.

  Preprocessed
  : Cache preprocessed images, for faster tweaking of effects at the cost of memory usage.

Composite

Cache intermediate composited images, for faster tweaking of stacked strips at the cost of memory usage.

Proxy

As projects involve increasingly high-resolution footage, the performance of the video preview can decrease drastically. To combat this, Blender can generate proxies – copies of the original footage stored at a lower quality and/or resolution – to maintain a smooth editing experience without compromising visual fidelity in the end result.

The quickest way to set up proxies for videos is to simply select a Proxy Render Size in the *View* tab (visible when the editor is in *Preview* or *Sequencer & Preview* mode). This will automatically enable the selected proxy resolution in all the strips and start generating the downscaled video files.

You can use the *Proxy* tab if you want to configure proxies in more detail (or create proxies for image sequences).

Proxy Settings

Reference

Panel:: *Sidebar region ▸ Proxy ▸ Proxy Settings*

Contains scene-wide proxy settings.

Storage
: How proxies are stored for the project.

  Per Strip:: Each strip can specify where to store its proxies (see below).

  Project:: All proxies are stored in one directory.
  Proxy Directory
  : The location to store the proxies for the project.

Set Selected Strip Proxies
: Shows a pop-over that lets you choose the resolution(s) to generate and whether to overwrite existing proxy files. Once you confirm with the *Set* button, your choices are applied to the selected strips. You can view and tweak the settings for individual strips in the *Strip Proxy & Timecode* panel (see below).

  In the *Preview* mode, where the *Proxy* tab is not available, this is instead done through the menu *View* ‣ *Proxy* ‣ *Setup*.

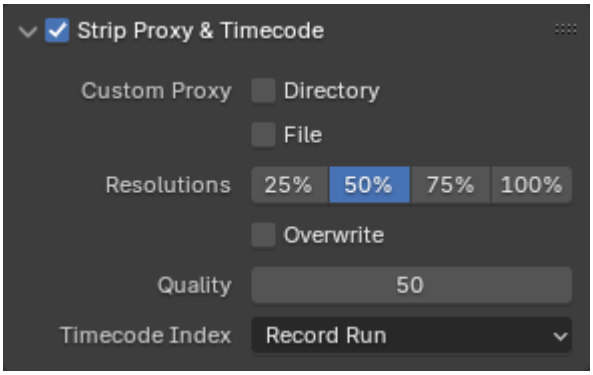
Rebuild Proxy and Timecode Indices
: Generates proxies and time indices for the selected strips.

  In the *Preview* mode, where the *Proxy* tab is not available, this is instead done through the menu *View* ‣ *Proxy* ‣ *Rebuild*.

Strip Proxy & Timecode

Reference

Panel:: *Sidebar region* ‣ *Proxy & Timecode* ‣ *Strip Proxy & Timecode*



Contains strip-specific proxy settings. The checkbox in the header can be used to enable/disable proxy generation.

Custom Proxy
: Directory
  : By default, all generated proxy videos are stored to the folder `<path of original footage>/BL_proxy/<clip name>`, but this can be changed to a custom directory using this option.

  File

Allows you to use preexisting proxies.

Resolutions
: The resolution(s) of the proxy videos to generate; multiple sizes can be selected.

Overwrite
: Whether to overwrite existing proxy files or keep them.

Quality
: The video/image quality for proxies.

Timecode Index
: When you are working with footage directly copied from a camera without preprocessing it, there might be numerous artifacts, mostly due to seeking to a given frame in the sequence. This happens because such footage usually does not have correct frame rate values in the file header. This issue can still arise when the source clip has the same frame rate as the scene settings. In order for Blender to correctly calculate the frames and frame rate there are two possible solutions:

1. Preprocess your video with e.g. MEncoder to repair the file header and insert the correct keyframes.
2. Use the Timecode Index option in Blender.

None:: Ignore generated timecodes, seek in movie stream based on calculated timestamp.

Record Run:: Seek based on timestamps read from movie stream, giving the best match between scene and movie times.

Record Run No Gaps:: Effectively convert movie to an image sequence, ignoring incomplete or dropped frames, and changes in frame rate.

Note

Record Run is the Timecode Index which usually is best to use, but if the source file is totally damaged, *Record Run No Gaps* will be the

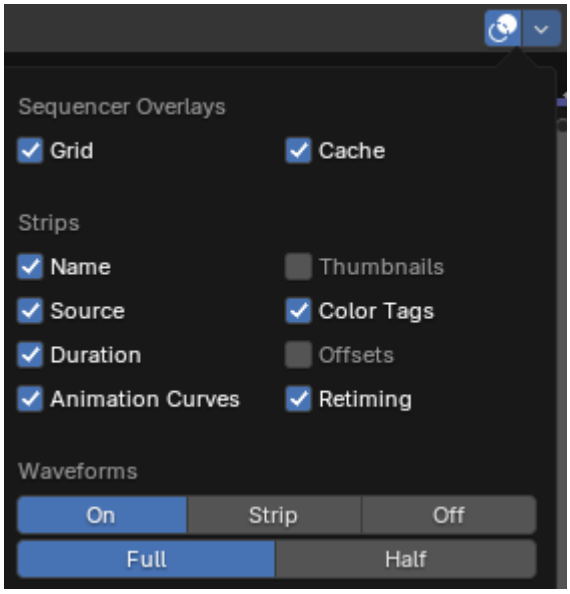
only chance of getting an acceptable result.

Display

Sequencer Overlays

Reference

Header::  *Overlays*



Overlays are information that is displayed on top of the sequencer region. The icon toggles all overlays in one go, while the drop-down button shows a pop-over where you can toggle individual ones:

Grid
: Shows vertical lines at regular time intervals.

Cache
: Visualize cached images on the timeline.

Strips

Name
: Shows the Name of each strip.

Source
: Shows the file path of each strip.

Duration
: Shows the length of each strip (in frames).

Animation Curves
: Shows animation curves for volume (Sound strips) and opacity (other strips).

Thumbnails
: Displays thumbnails across the full width of each Movie or Image strip. The thumbnail size depends on the vertical zoom level (which can be adjusted by dragging up and down with `Ctrl-MMB`). Zooming in results in taller strips with bigger, but fewer thumbnails. Zooming out results in narrower strips with smaller, but more thumbnails.

Color Tags
: Displays each strip in its designated custom color (if applied) rather than a color representing its type. To set a custom color, either click the *Color Tag* button next to the strip's name in *Sidebar ▸ Strip*, or use *Set Color Tag* in the strip's context menu.

Offsets
: Shows overflow bars of content that was trimmed from the strip (by moving the strip's handles). See Strip Offset Start/End.

Waveforms

Type
: Global options for waveform display on Sound strips.

On::	Enable waveforms for all strips.
Strip::	Use the Display Waveform option of each individual strip.

Off:: Disable waveforms for all strips.

Style

How Waveforms are displayed.

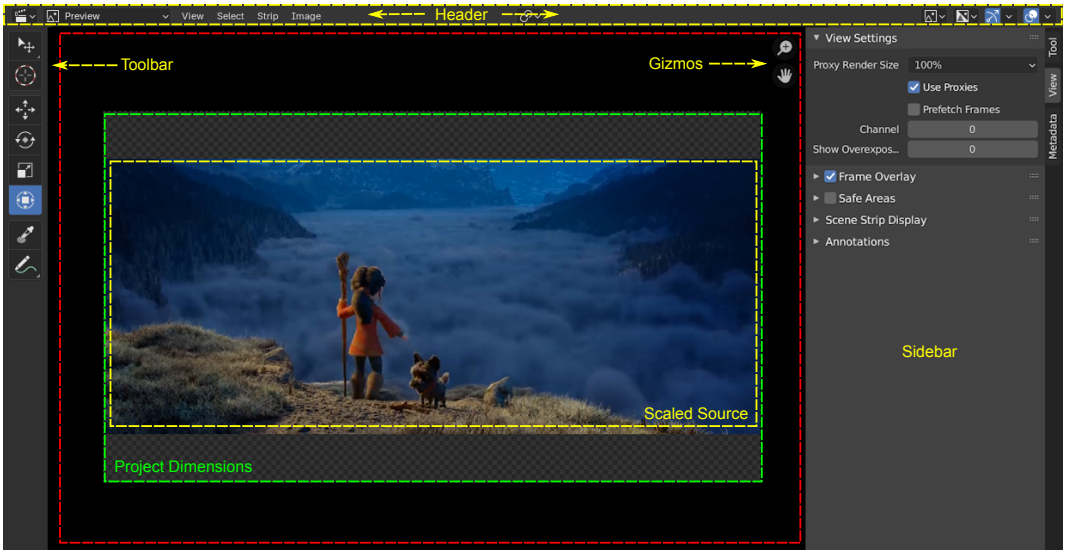
Full:: Displays the audio amplitude.
Half:: Displays the audio level.

Preview

- Introduction
- Header
  - View Menu
  - Select Menu
  - Strip Menu
  - Image Menu
  - Pivot Point
  - Display Mode
  - Display Channels
  - Gizmos
  - Overlays
- Toolbar
- Sidebar
  - Tool
  - View
  - Metadata
- Controls
  - Pivot Point
  - Preview Snapping
- Display
  - Display Mode
  - Gizmos
  - Overlays

Introduction

The Preview mode shows how the final edited video will look like. It also offers tools for moving, rotating, and scaling images, as well as scopes for analyzing color distribution.

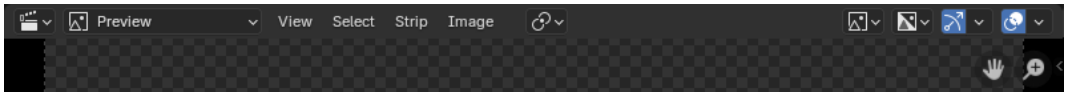


Preview mode of the Video Sequencer.

You can pan around the view with `MMB` and zoom with `Wheel` or `NumpadPlus`/ `NumpadMinus`. Alternatively, you can use the gizmos.

Pressing `Home` resets the view, maximizing the size of the preview within the editor's area.

Header



Header in Preview mode.

View Menu

Toolbar `T`
: Show or hide the Toolbar.

Sidebar `N`
: Show or hide the Sidebar.

Tool Settings
: Show or hide the settings for the currently selected tool.

Preview During Transform
: When enabled, previews the strip's new first/last frame while dragging its left/right handle.

Refresh All
: Reloads external files and refreshes the current frame preview. This is useful when you modified an external file or made a change in a scene that Blender didn't detect.

Frame Selected
: Pan and zoom the view to focus on the selected image.

Fit Preview in Window `Home`

Pan and zoom the view so that the entire video is visible. This enables *Zoom to Fit*.

Zoom
: Menu with convenient zoom levels and operations. The zoom levels are calculated based on the images resolution compared to the screen resolution.

  - 12.5% (1:8) `Numpad8` zoom out to a factor of 12.5%.
  - 25% (1:4) `Numpad4` zoom out to a factor of 25%.
  - 50% (1:2) `Numpad2` zoom out to a factor of 50%.
  - 100% (1:1) `Numpad1` resets the zoom to 100%.
  - 200% (2:1) `Ctrl-Numpad2` zoom in to a factor of 200%.
  - 400% (4:1) `Ctrl-Numpad4` zoom in to a factor of 400%.
  - 800% (8:1) `Ctrl-Numpad8` zoom in to a factor of 800%.

  Zoom In/Out `Wheel`
  : Zooms the view in or out.

  Zoom to Fit `Shift-Home`
  : Like *Frame All*, but uses as much space in the editor as possible.

  Zoom Region `Shift-B`
  : Zoom in the view to the nearest item contained in the border.

Auto Zoom
: As long as this option is enabled, the preview will automatically zoom to keep the video size synchronized with the editor size.

Proxy
: See Proxy.

Sequence Render Image
: Show the current frame preview as a Render Result where you can save it as an image file.

Sequence Render Animation
: Save previews of the frames in the scene range (or the preview range, if active) to a video file or a series of image files. See the Output panel for details.

Note

Sequence Render Image and *Sequence Render Animation* don't render the final video by default – specifically, they don't render Scene Strips, instead using the preview's shading mode (which is initially Solid).

To output a video where the Scene Strips are rendered, use the *Render* menu in the top-bar, or change *Sidebar* ▸ *View* ▸ *Scene Strip Display* ▸ *Shading* to *Rendered*.

Export Subtitles
: Exports Text strips, which can act as subtitles, to a SubRip file (`.srt`). The exported file contains all Text strips in the video sequence.

Toggle Sequencer/Preview `Ctrl-Tab`
: Switch the editor mode between *Sequencer* and *Preview*.

Area
: Area controls. See the user interface documentation for more information.

Select Menu

See Selecting.

Strip Menu

See Editing.

Image Menu

Clear
: Resets the position, rotation, or scale of the selected images.

Apply
: Scale to Fit
: : Resizes the selected images so that they're as large as possible while still fitting completely inside the video. They don't get cropped, and their aspect ratio stays the same.

: Scale to Fill
: : Resizes the selected images to that they fill the entire video space. They may get cropped, but their aspect ratio stays the same.

: Stretch to Fill
: : Resizes the selected images to match the video dimensions. They don't get cropped, but their aspect ratio may change.

Pivot Point

See Pivot Point.

Display Mode

See Display Mode.

Display Channels

Color & Alpha
: Display the preview image with transparency over a checkerboard pattern.

Color
: Ignore the transparency of the preview image (fully transparent areas will be black).

Gizmos

See Gizmos.

Overlays

See Overlays.

Toolbar

Tweak `W`

Lets you select images by clicking, and move them by dragging. Press `W` to cycle between this tool and *Select Box*.

Select Box

Lets you select one image by clicking, or multiple images by dragging a rectangle.

Cursor

Lets you move the 2D Cursor by clicking or dragging with `LMB`.

While dragging, you can press `X` or `Y` to constrain movement to an axis.

If you need extra precision, you can hold `Shift` to move the cursor more slowly than the mouse, or type a number to move it by an exact amount.

The header shows how far the cursor has traveled, including the distance along each axis.

Instead of this tool, you can also drag the mouse while holding `Shift-RMB` (works with all tools) or adjust the 2D Cursor Location in *Sidebar ▸ View*.

Note

By default, the 2D Cursor is only shown while dragging it. To make it permanently visible, enable the *2D Cursor* overlay.

Move `G`

Lets you move the selected images by dragging with `LMB`. Alternatively, you can press `G`, move the mouse, and finally click `LMB` to confirm (or `RMB` to cancel).

If the Active Tools gizmo is enabled, you can drag one of the colored arrows to only move along that one axis. You can also press `X` or `Y` while moving: press once to constrain to the corresponding global axis, a second time to constrain to the local axis, and a third time to remove the constraint again. Yet another way is to hold `MMB` and move the mouse horizontally or vertically.

If you need more precision, you can do one of the following while moving:

- Hold `Shift` to move more slowly.
- Type a number to move by an exact amount.
- Use the arrow keys.

The header shows how far the image has moved, including the offset along each axis.

Instead of using this tool, you can also adjust the Position in the Sidebar's *Strip* tab (only available in the *Sequencer* and *Sequencer & Preview* modes).

Rotate `R`

Lets you rotate the selected images by holding `LMB` and moving the mouse in a circle. Alternatively, you can press `R`, move the mouse, and finally click `LMB` to confirm (or `RMB` to cancel).

Images are rotated around the Pivot Point, so if it's off-center, the images will not just rotate but also move around it.

If you need more precision, you can do one of the following while rotating:

- Hold `Shift` to rotate more slowly.
- Hold `Ctrl` to rotate in increments of 5 degrees.
- Type a number to rotate by an exact amount.
- Use the arrow keys.

The header shows how much the image has rotated.

Instead of using this tool, you can also adjust the Rotation in the Sidebar's *Strip* tab (only available in the *Sequencer* and *Sequencer & Preview* modes).

Scale `S`

Lets you resize the selected images by dragging with `LMB`. Alternatively, you can press `S`, move the mouse, and finally click `LMB` to confirm (or `RMB` to cancel).

If the Active Tools gizmo is enabled, you can drag one of the colored lines to only scale along that one axis. You can also press `X` or `Y` while scaling: press once to constrain to the corresponding global axis, a second time to constrain to the local axis, and a third time to remove the constraint again. Yet another way is to hold `MMB` and move the mouse horizontally or vertically.

Images are scaled around the Pivot Point, so if it's off-center and you scale down, the images will not just become smaller but also move towards it.

If you need more precision, you can do one of the following while scaling:

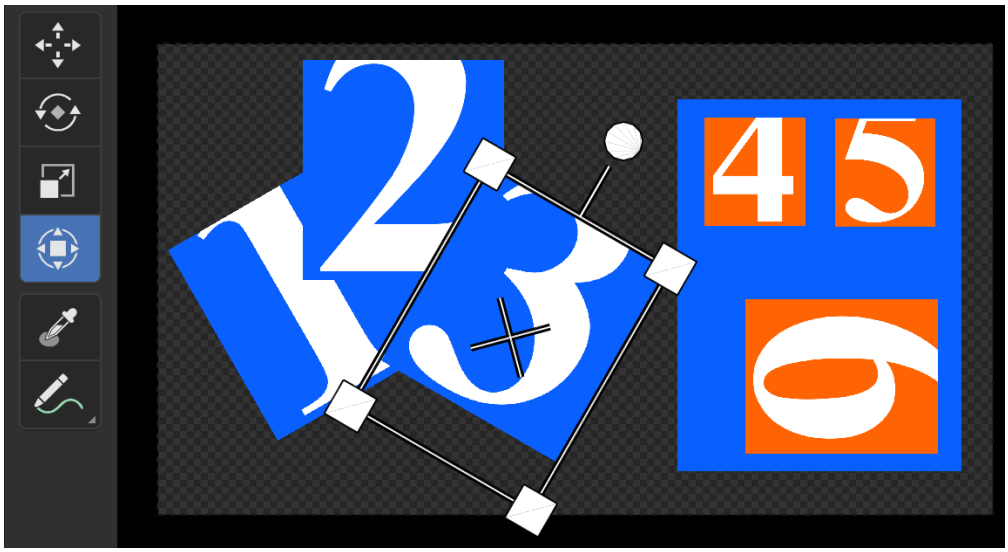
- Hold `Shift` to scale more slowly.
- Hold `Ctrl` to scale in increments of 10%.
- Type a number to scale by an exact factor (e.g. `.5` to make it half the size).
- Use the arrow keys.

The header shows the current scale factor.

Instead of using this tool, you can also adjust the Scale in the Sidebar's *Strip* tab (only available in the *Sequencer* and *Sequencer & Preview* modes).

Transform

Lets you move, rotate, and scale images all using one tool.



The Transform tool

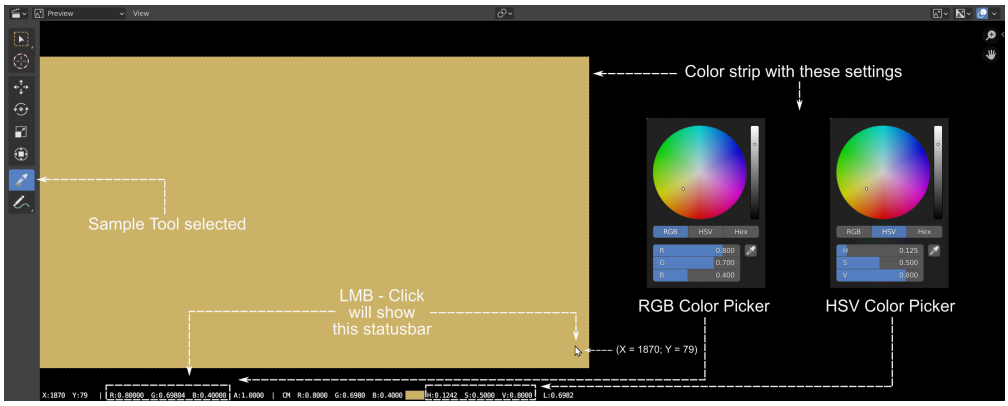
It works as follows:

- Drag the cross in the center to move the image.
- Drag the dot on the protruding line to rotate.
- Drag one of the corners to scale equally along both axes.
- Drag one of the sides to scale along just one axis.

Sample

Lets you sample a pixel's color by holding `LMB`. The editor will show the following information about it on the bottom:

- The X and Y coordinates, in pixels relative to the top left corner.
- The red, green, blue, and alpha components of the pixel, as decimal values between 0 and 1.
- The red, green, and blue components of the pixel with Color Management applied.
- The hue, saturation, value, and luminance components of the pixel with Color Management applied.



Sample tool example.

Annotate

Draw free-hand annotations.

Annotate Line

Draw a straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previously drawn annotations.

Sidebar

The Sidebar can be toggled with the menu item *View* ▸ *Sidebar* or with the shortcut `N`.

The image below shows two Video Sequencers, one in Preview mode and one in Sequencer mode, both with their Sidebar open.



Tool

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar* ▸ *Tool tab*

Settings for the active tool.

Drag
: What to do when dragging `LMB` on a place other than the tool's gizmo.

Active Tool
: Perform the same action as when dragging the gizmo.

Tweak
: Move the image under the mouse cursor.

Select Box
: Drag a selection rectangle and select all the images that are partially or completely inside it.

View

View Settings

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar ▸ View tab ▸ View Settings*

Proxy Render Size
: Controls the preview resolution. Lower values have worse detail but better performance.

No Display:: Disable the preview entirely.

Scene Size:: Preview at the full resolution without using proxies.

25%, 50%, 75%, 100%:: Preview at a downscaled resolution, optionally using proxies (see below). Even selecting 100% can give a performance benefit due to the reduced image quality and corresponding smaller file size.

Use Proxies
: Enable the use of proxies, which are copies of original footage stored at a lower resolution and/or quality for better preview performance.

Proxies can be configured in the *Proxy* tab of the Sidebar, which is however only visible in the *Sequencer* and *Sequencer & Preview* modes.

Prefetch Frames
: Automatically fill the cache with frames after the current frame in the background. Use this to achieve a more consistent playback speed. This feature currently doesn't support rendering Scene strips.

Channel
: Setting this to 0 shows all channels. Setting it to something higher will only show the channels up to and including that number.

Show Overexposed
: Hilight overexposed (bright white) areas using a zebra pattern. The threshold can be adjusted with the slider.

Show Missing Media
: Render missing images/movies with a solid magenta color. When disabled, missing content will render fully transparent.

Tip

Strips with missing content will be displayed as red in the timeline.

2D Cursor

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar ▸ View tab ▸ 2D Cursor*

The 2D Cursor is the white-red circle with a crosshair that is shown in the preview region (provided that the 2D Cursor overlay is enabled). It can be used as a Pivot Point for rotating and scaling images.

Location X, Y
: The location of the 2D Cursor relative to the center of the video. The edges are 0.5 away, so (0.5, 0.5) is the top right corner.

  The 2D Cursor's location can also be set with the Cursor tool or by dragging with `Shift-RMB`.

Frame Overlay

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar* ▸ *View tab* ▸ *Frame Overlay*

The Frame Overlay lets you display a reference frame for comparing to the current frame.

Set Overlay Region
: Lets you drag a rectangle to define the bounds of the overlay. Instead of clicking this button, you can also press `O` while hovering over the preview.

Frame Offset
: The time offset between the reference frame and the current frame, in frames.

Overlay Type
: How the reference frame should be displayed.

Rectangle::	Display part of the reference frame (defined by the *Overlay Region*) on top of the current frame.
Reference::	Display only the reference frame.
Current::	Display only the current frame.

  Tip

Each Video Sequencer editor can have its own *Overlay Type*. This means you can open two of them for showing the current frame and the reference frame next to each other.

Overlay Lock
: Keep displaying the same reference frame, even when moving to a different time point. This works by automatically adjusting the *Frame Offset*.

Safe Areas

Reference

Editor:: Video Sequencer
View Type:: Preview
Panel:: *Sidebar ▸ View tab ▸ Safe Areas*

Shows guides indicating the video area where content can be seen across all screens.

See also

Camera Safe Areas.

Scene Strip Display

Reference

Editor:: Video Sequencer
View Type:: Preview
Panel:: *Sidebar ▸ View tab ▸ Scene Strip Display*

Controls how Scene Strips are displayed in the preview.

Shading
: The shading mode to use.

Override Scene Settings
: Use the Workbench render settings from the current scene rather than the scenes referenced by the strips. Only available for the *Wireframe* and *Solid* shading modes.

Annotations

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar ▸ View tab ▸ Annotations*

For managing the Annotations in the Sequencer.

Metadata

Reference

Editor::	Video Sequencer
View Type::	Preview
Panel::	*Sidebar ▸ Metadata tab*

Lists information that has been encoded in the currently visible movie or image file (*not* the file referenced by the selected strip). This can include the filename, the creation date, the camera model etc. This also works for images produced by Blender; see Render Output for the metadata that can be included in this case.

Other graphics programs may also store metadata, but only the text in the header field “Comments” can be read.

Some of this metadata can also be made visible in the preview with the Metadata overlay.

Tip

The metadata can't be edited from Blender. Instead, you can use an external program such as exiftool. For example, the command to change the "Comments" field is:

```
exiftool --comments="My new comment" name-of-file.png
```

Note

Metadata is only displayed for images/movies that don't have an effect applied.

Controls

- Pivot Point
- Preview Snapping

Pivot Point

Reference

Header::  *Pivot Point*

Shortcut:: `Period`

The Pivot Point is the point around which images are rotated and scaled. It's indicated by the position of the selected tool's gizmo.

See also

The Pivot Point of the 3D Viewport

Bounding Box Center
: Use the center of the rectangle that's wrapped as tightly as possible around the selected images' origin points.

Median Point
: Use the averaged-out position of the selected images' origin points.

2D Cursor
: Use the location of the 2D Cursor, for when you want to specify the pivot point by hand.

Individual Origins
: Rotate/scale each image around its own origin, rather than rotating/scaling all of them around the same single point like the other options do.

Preview Snapping

The icon toggles snapping; you can also do this temporarily by holding `Ctrl` after starting to transform an image.

Images have multiple snap points; they can snap along their edges, corners, or center.

The drop-down arrow offers the following options:

Snap to

Borders
: Snap images to the edges of the render region.

Center
: Snap images to the horizontal and vertical center lines of the render region.

Other Strips
: Snap images to the snap points of other images.

Display

- Display Mode
  - Image Preview
  - Luma Waveform
  - RGB Parade
  - Chroma Vectorscope
  - Histogram
- Gizmos
- Overlays

Display Mode

Using this pop-up, you can choose between displaying the preview image or a scope that visualizes its color distribution.

Image Preview

Previews what the final video will look like, and lets you change the image layout using various tools.

Luma Waveform

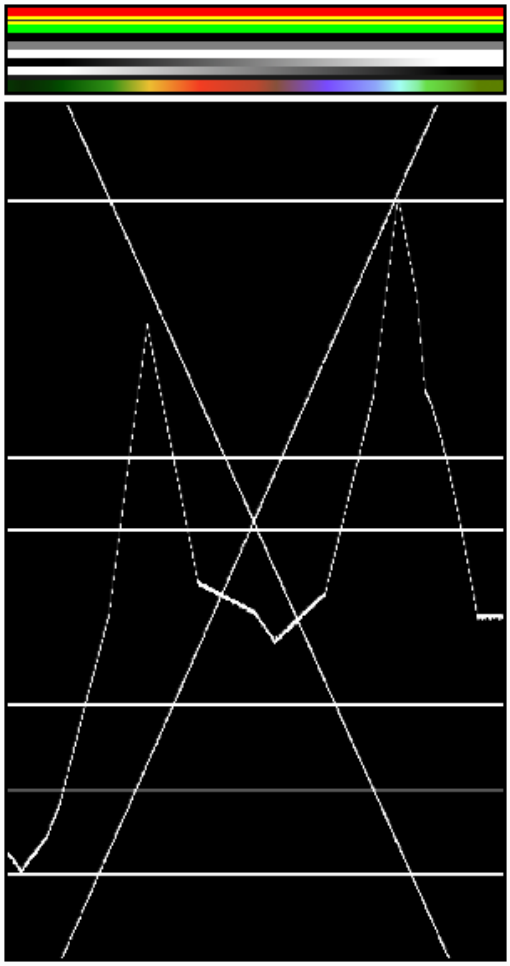
This scope visualizes the luminosity (brightness) distribution of the image, letting you see at a glance if there's enough contrast and if any areas are under- or overexposed.

The scope works by plotting a curve for each scanline in the current video frame. Another way of saying this is that each pixel column in the luma waveform is a brightness histogram of the corresponding pixel column in the frame. Specifically:

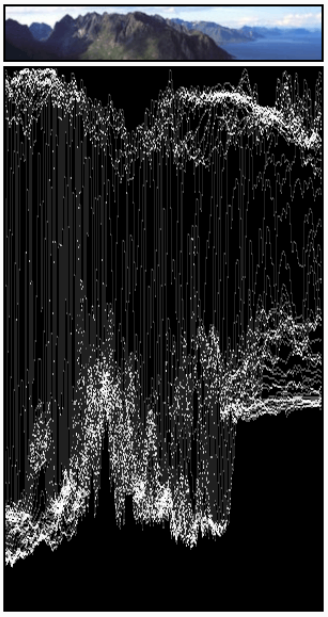
- The horizontal position of a pixel in the waveform refers to a pixel column in the frame.
- The vertical position of a pixel in the waveform refers to a brightness value, going from 0 at the bottom to 1 at the top.
- The brightness of a pixel in the waveform indicates how many pixels in the above frame column have the above brightness. If no pixels in the frame column have this brightness, the waveform pixel is black. If at least three pixels in the frame column have this brightness, the waveform pixel is white.

When this scope is selected, you have the following option in *Sidebar ▸ View ▸ View Settings*:

The examples below show two images and their corresponding luma waveforms.



The various horizontal lines in the luma waveform match the uniform-colored lines of the picture. Note that the ‘gray 20%’ one-pixel width line (inside the yellow strip) is represented in the Luma waveform by a gray line. The two lines drawing an “X” are from the two monochrome gradients. Finally, the broken



The curves are quite visible. We found a luma of 80-100% for the sky, a luma around 40% for the sea, and a luma of 10-20% for the mountains, growing around 40% for the sunny part.

line matches the colored gradient at the bottom.

RGB Parade

Shows three waveforms – for the red, green, and blue color channels – instead of just one for the overall image brightness.

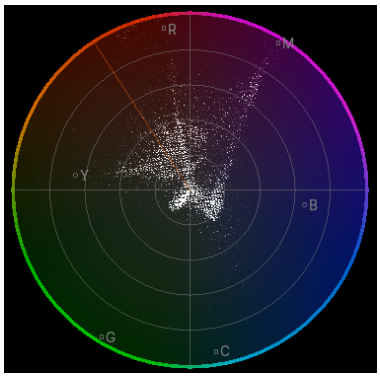
Chroma Vectorscope

This scope visualizes the color distribution of the image. Each point has:

- An angle indicating its hue.
- A distance-from-center indicating its saturation.
- A brightness indicating how many pixels in the video frame have the above hue and saturation.



Example image.



Corresponding Chroma Vectorscope.

Histogram

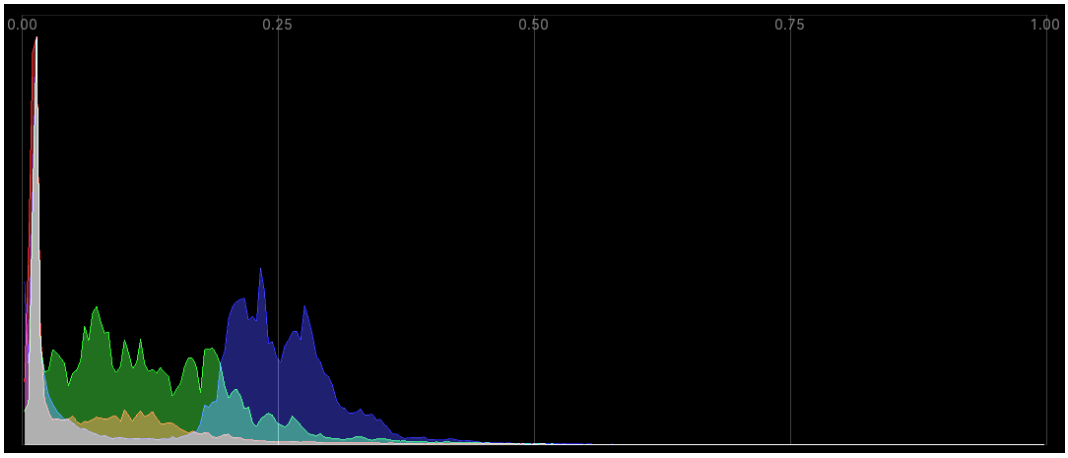
Shows three overlapping graphs, one for each color channel. Within each graph:

- The X axis corresponds to color intensity, going from 0 on the left (black) to 1 on the right (fully red/green/blue).
- The Y axis corresponds to number of pixels.

Use this mode to balance out the tonal range in an image. A well-balanced image should have nice and smooth distribution of color values.



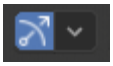
Example image.



Corresponding Histogram.

Gizmos

Reference

Header::  *Gizmos*

Clicking the icon toggles all gizmos in the Video Sequencer. The drop-down button displays a popover with more detailed settings, which are described below.

Navigate
: Enable/disable the navigation gizmo.

Active Tools
: Enable/disable the gizmo of the active tool.

Overlays

Reference

Header::  *Overlays*

Clicking the icon toggles all overlays in the Video Sequencer. The drop-down button displays a popover with more detailed settings, which are described below.

Image Outline
: Shows an outline around the selected images.

2D Cursor
: Shows the 2D Cursor.

Frame Overlay
: Shows the Frame Overlay for comparing the current frame to a reference frame.

Safe Areas
: Shows guides indicating the video area where content can be seen across all screens.

  See also

  Camera Safe Areas.

Metadata
: Shows file metadata.

Annotations
: Shows Annotations.

Sequencer & Preview

This view type shows both the preview and the sequencer inside one editor.

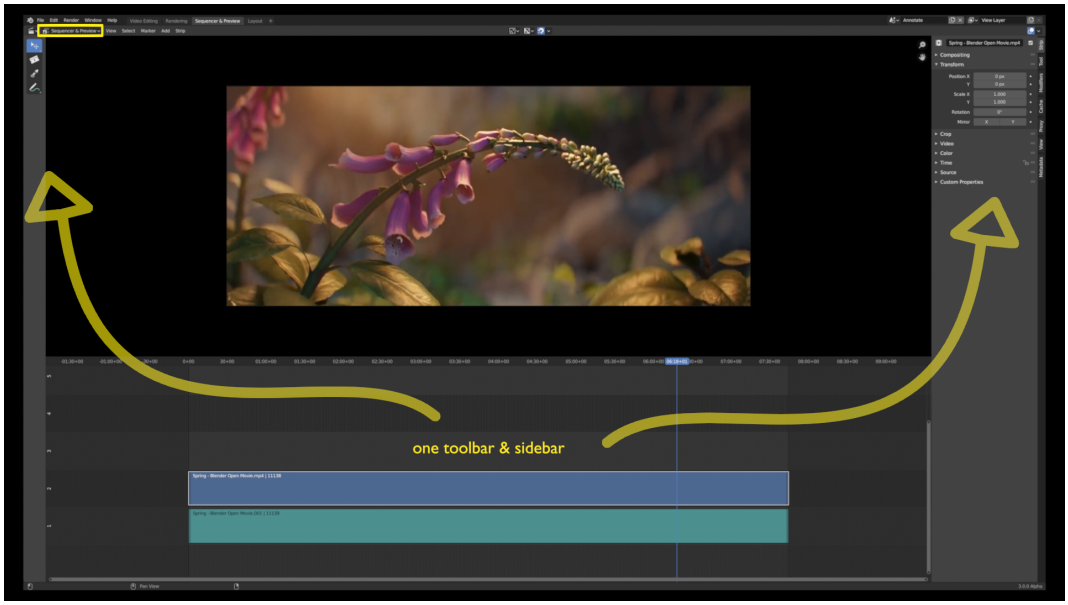


Figure 1: Combined Sequencer & Preview

In general, it's better to avoid this view type and instead have two editors, one serving as the *Preview* and the other as the *Sequencer*. Reasons for this include:

- Most of the *Preview* tools, such as Move and Rotate, are not available in *Sequencer & Preview*.
- You can't add a small editor (such as a File Browser) on the side that only takes up the height of the preview.
- You can't maximize the preview on another screen.

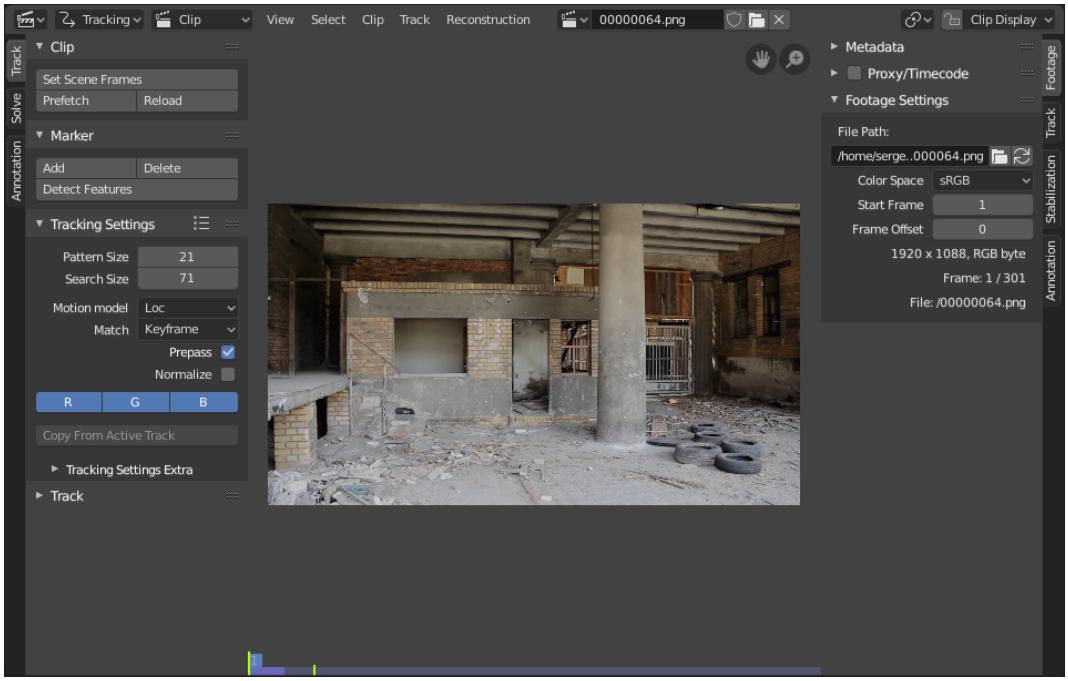
One way of getting two separate editors is to simply open the default *Video Editing* workspace. (You may need to click the "+" icon to the right of the tabs to find it.)

Movie Clip Editor

- Introduction
  - Header
- Display
  - Viewport Gizmos
  - Mask Display
  - Clip Display
- Sidebar Region
  - Footage
  - Track
  - Stabilization
  - View

Introduction

The Movie Clip Editor is used for tracking and masking movies.



Movie Clip Editor interface.

Header

Mode

Tracking
: For placing markers in a video and tracking their movement.

Mask
: For creating and animating masks.

View Type

Clip
: The default view, for placing and tracking markers.

Graph
: Plots the movement speed of the markers on a graph.

Dope Sheet
: Shows an overview of marker keyframes on a timeline.

View Menu

Toolbar `T`
: Show or hide the tab panel on the left for creating and manipulating markers and masks.

Sidebar `N`
: Show or hide the Sidebar.

Adjust Last Operation
: Display a pop-up panel to alter the properties of the last completed operation.

Frame Selected `NumpadPeriod`
: Zooms and pans the view to focus on the selected items.

Frame All `Home`
: Zooms and pans the view so that the whole video is visible.

Center View to Cursor *Mask Mode*
: Pans the view so that the 2D Cursor is in the center.

Zoom
: Menu with convenient zoom levels and operations. The zoom levels are calculated based on the images resolution compared to the screen resolution.

  - 12.5% (1:8) `Numpad8` zoom out to a factor of 12.5%.

- 25% (1:4) `Numpad4` zoom out to a factor of 25%.
- 50% (1:2) `Numpad2` zoom out to a factor of 50%.
- 100% (1:1) `Numpad1` resets the zoom to 100%.
- 200% (2:1) `Ctrl-Numpad2` zoom in to a factor of 200%.
- 400% (4:1) `Ctrl-Numpad4` zoom in to a factor of 400%.
- 800% (8:1) `Ctrl-Numpad8` zoom in to a factor of 800%.

Zoom In/Out `Wheel`
: Zooms the view in or out.

Zoom to Fit `F`
: Like *Frame All*, but uses as much space in the editor as possible.

Show Metadata
: Displays metadata encoded in the video, if available.

Area
: Area controls. See the user interface documentation for more information.

Select Menu

Menu for selecting markers and masks.

Clip Menu

Menu for loading movie clips and creating proxies.

Track Menu

Menu for performing tracking operations.

Reconstruction Menu

Menu for setting up the reconstruction of 3D information from the tracked points in the 2D video.

Add Menu

Circle
: Adds a circle-shaped mask.

Square
: Adds a square-shaped mask.

Mask Menu

Menu for editing masks.

Other

Clip
: A data-block menu used for loading and selecting movies. Both video files and image sequences can be used. When a movie clip is loaded into the Clip editor, extra panels are displayed in the interface.

Pivot Point
: See Pivot Points.

Proportional Editing *Mask Mode*
: See Proportional Editing.

Mask *Mask Mode*
: A data-block menu for creating and selecting masks.

Mask Display *Mask Mode*
: See Mask Display.

Toggle Lock Selection L
: Automatically pans the view to follow the selected markers, so that they remain in the same location on screen during tracking and playback.

  This option “locks the view onto the selection” and is not to be confused with the Lock option in the Sidebar, which instead prevents you from

changing the active marker.

Clip Display
: See Clip Display.

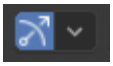
Gizmos
: See Viewport Gizmos.

Display

- Viewport Gizmos
  - Viewport Gizmos
- Mask Display
- Clip Display
  - Marker Display

Viewport Gizmos

Reference

Mode:: All Modes
Header::  *Gizmos*

Clicking the icon toggles all the gizmos in the Movie Clip Editor. The drop-down button displays a popover with more detailed settings, which are described below.

Viewport Gizmos

Navigate
: Toggle the visibility of the zooming and panning gizmos in the upper right corner.

Mask Display

This popover controls how masks are displayed in Mask mode.

Spline
: Toggles the display of the mask splines. Note that if they're hidden, you won't be able to edit them.

  Edge Display Type
  : Line style of the splines.

Overlay
: Visualizes masks by shading the whole clip.

  Overlay Mode

  Alpha Channel:: Displays just the masks as a grayscale image. Excluded areas are black, while included areas are white.

  Combined:: Displays the clip with excluded areas darkened.

Blending Factor
: How much excluded areas are darkened when using the "Combined" *Overlay Mode*.

Clip Display

This pop-over contains various display settings for both Tracking mode and Mask mode.

R, G, B
: Controls the color channels used for the frame preview. The tracking algorithm works with grayscale images, and with these options, you can check which combination of enabled and disabled channels will yield the best contrast and the least noise.

  Note that this only affects the preview. To select which channels to use for the actual tracking, use the Track tab in the Toolbar to set a default for newly created markers, or the Track tab in the Sidebar to configure existing markers.

Grayscale Preview (B/W)
: Shows the whole frame as a grayscale image.

Mute (eye icon) `M`
: Hides the movie clip and displays a black image instead. This helps to find markers that are tracked inaccurately or not at all.

Render Undistorted
: Applies the Lens settings to the video preview to undo lens distortion. Does not change the footage itself.

Show Stable
: Applies the 2D stabilization settings to the video preview. Does not change the footage itself.

Grid
: Displays a grid which is originally orthographic, but is distorted by the Lens settings. This can be used for manual calibration: the distorted grid lines should match lines in the footage that are meant to be straight.

Calibration
: Applies the Lens settings to annotation strokes. Like the Grid, this option also helps to perform manual calibration.

Display Aspect Ratio
: Changes the aspect ratio for displaying only. It does not affect the tracking or solving process.

Marker Display

Determines how markers are displayed in the editor.

Pattern
: Whether to show the pattern areas of tracks. Can be used to reduce clutter and check how good tracking is.

Search `Alt-S`
: Whether to show the search areas of *selected* tracks. Can be used to reduce clutter and check how good tracking is.

Path
: Shows past (red) and future (blue) positions of tracks relative to the current frame, visualizing how they move. This makes it easier to spot irregularities.

Length
: Length (in frames) of the *Path*.

Show Disabled `Alt-D`
: When unchecked, hides the tracks that are disabled on the current frame (except for the active track, i.e. the one that was selected last). This helps to make the view more clear and see if the tracking is accurate enough.

Info
: Displays the name and status of each selected track. The status can be “keyframed,” “tracked,” “disabled” and so on.

3D Markers

Shows the result of solving the markers' 3D locations based on their 2D movement. Each 3D location is projected back to the movie clip and displayed as a small point, which is colored green if it's close to the original 2D marker (meaning a good solve) or red if it's far away (meaning it needs to be tweaked).

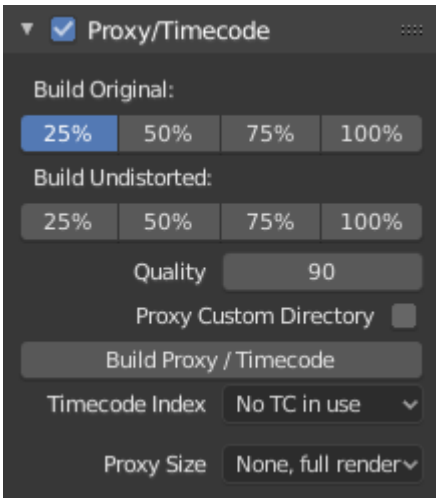
Display Thin

By default, marker areas are displayed as bright boxes with a black outline. This option displays them using thin dashed lines instead.

Sidebar Region

Footage

Proxy/Timecode



High-resolution video files can impact Blender's performance, slowing down scrubbing and other operations. To counter this, you can generate one or more proxies, which are copies of the original footage stored at a lower resolution and/or quality. These proxies can then be used as a less resource-heavy standin while working on the scene.

Build Original
: The proxy resolution(s) to generate based on the original, distorted footage.

Build Undistorted
: The proxy resolution(s) to generate based on the undistorted footage (that is, with the Lens settings applied to undo the distortion in the recording).

Quality
: The image quality for the proxies.

Proxy Custom Directory
: By default, proxies are stored to a `BL_proxy` subfolder next to the original file. Use this option to specify a different location.

Build Proxy/Timecode
: Generates proxies based on the settings above, as well as timecode files. Instead of using this button, you can also click *Clip* ▸ *Proxy* ▸ *Rebuild*

Proxy and Timecode Indices.

Timecode Index

When you are working with footage directly copied from a camera without preprocessing it, there might be numerous artifacts, mostly due to seeking to a given frame in the sequence. This happens because such footage usually does not have correct frame rate values in the file header. This issue can still arise when the source clip has the same frame rate as the scene settings. In order for Blender to correctly calculate the frames and frame rate there are two possible solutions:

1. Preprocess your video with e.g. MEncoder to repair the file header and insert the correct keyframes.
2. Use the Timecode Index option in Blender.

None:: Ignore generated timecodes, seek in movie stream based on calculated timestamp.

Record Run:: Seek based on timestamps read from movie stream, giving the best match between scene and movie times.

Record Run No Gaps:: Effectively convert movie to an image sequence, ignoring incomplete or dropped frames, and changes in frame rate.

Note

Record Run is the Timecode Index which usually is best to use, but if the source file is totally damaged, *Record Run No Gaps* will be the only chance of getting an acceptable result.

Proxy Render Size

Which proxy size to use for display. Depending on the Render Undistorted setting, Blender will use either the Original proxy or the Undistorted proxy.

Footage Settings

See Image Settings.

Track

See Track.

Stabilization

See 2D Stabilization.

View

2D Cursor

The 2D Cursor is the dashed crosshair in the main region. It can be used as a transformation pivot point by selecting the corresponding option in the editor's header.

Note that the 2D Cursor is only available in Mask mode, not in Tracking mode.

Location X, Y
: The relative location of the 2D Cursor, going from (0, 0) for the bottom left corner to (1, 1) for the top right corner.

  You can also position the 2D Cursor by clicking `Shift-RMB` in (or around) the video.

Annotations

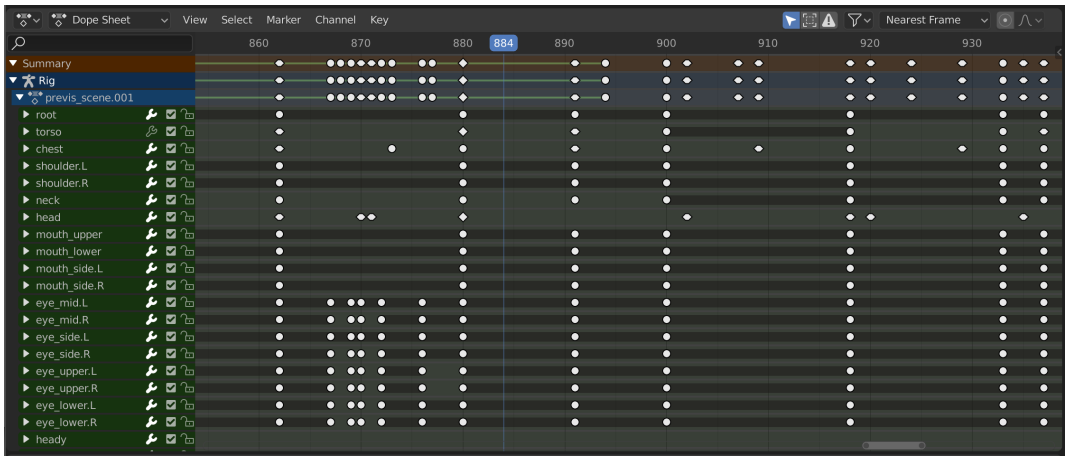
See Annotations.

Dope Sheet

- Introduction
  - Dope Sheet Modes
  - Main Region
- Navigating
  - View Menu
  - Filters
- Editing
  - Select Menu
  - Marker Menu
  - Channel Menu
  - Key Menu
  - Snap
  - Proportional Editing
- Modes
  - Action Editor
  - Shape Key Editor
  - Grease Pencil
  - Mask
- Sidebar
  - Action Panel
  - Custom Properties

Introduction

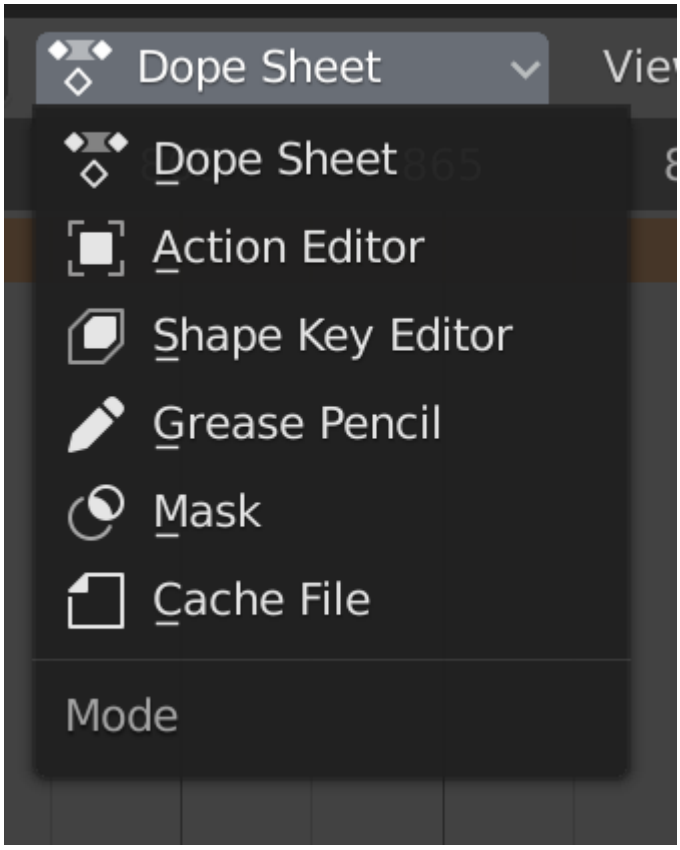
The Dope Sheet offers a bird's-eye view of the keyframes inside the scene. It's inspired by classical hand-drawn animation, where animators make use of a chart showing exactly when each drawing, sound, and camera move will occur, and for how long.



The Dope Sheet.

Dope Sheet Modes

The editor has several different modes that can be selected from a dropdown in the header. The default *Dope Sheet* mode gives an overview of most types of animatable data. For others, such as masks, you need to switch to a more specific mode.



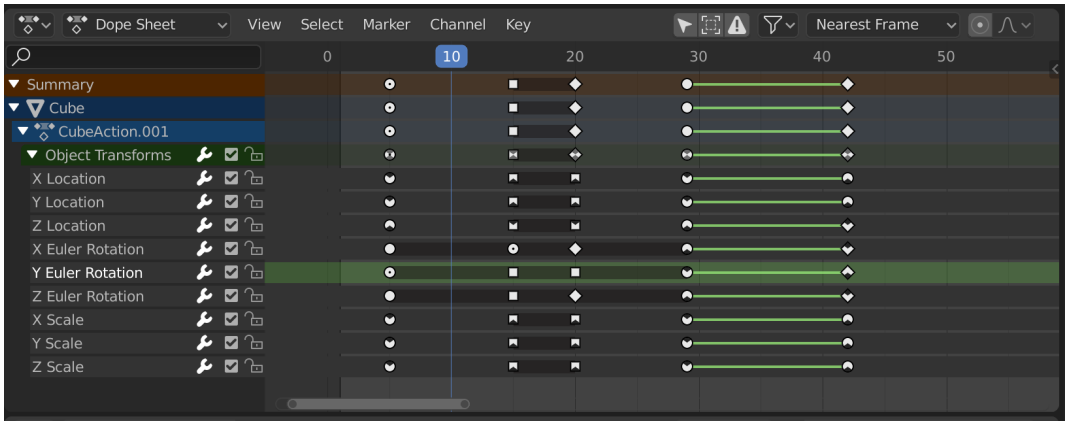
Dope Sheet modes.

The modes are as follows:

- Dope Sheet
- Action Editor
- Shape Key Editor
- Grease Pencil
- Mask
- Cache File: originally meant to show the baked animation data in Alembic files, but never implemented.

Main Region

The Dope Sheet Editor shows a stack of channels (animatable properties), and for each channel, a series of keyframes laid out along the time axis.



The Dope Sheet Editor with object channels.

Keyframes can take on various colors and shapes:

Gray	Unselected
Yellow	Selected
Other colors	Custom keyframe tag set by the user (*Key ▸ Keyframe Type*)
Diamond	Free Keyframe Handle (*Key ▸ Handle Type*)
Round	Auto-Clamped Keyframe Handle
Circle	Automatic Keyframe Handle
Square	Vector Keyframe Handle
Rhombus	Aligned Keyframe Handle

Gray bar between keys	Held key (the two keyframes are identical)
Green line between keys	The curve segment uses custom interpolation (*Key* ▸ *Interpolation Mode*)
Upwards arrow	Local maximum in curve (visible if *View* ▸ *Show Curve Extremes* is enabled)
Downwards arrow	Local minimum in curve

Keyframes can be selected by clicking and moved by dragging. See the Select and Key menus for more options.

Navigating

As with most editors, you can:

- Pan the view vertically (channels) and horizontally (time) by dragging `MMB`.
- Zoom in and out by rolling `Wheel` or dragging `Ctrl-MMB`.

You can also use the scrollbars for this.

View Menu

Sidebar `N`
: Shows or hides the Sidebar Region.

Adjust Last Operation
: Displays a pop-up panel to alter the properties of the last completed operation. See Adjust Last Operation.

Channels
: Shows or hides the Channels region (the list of animated property names on the left).

Frame Selected `NumpadPeriod`
: Pans and zooms the view to focus on the selected keyframes.

Frame All `Home`
: Pans and zooms the view to show all keyframes.

Frame Scene/Preview Range
: Reset the horizontal view to the current scene frame range, taking the preview range into account if it is active.

Go to Current Frame `Numpad0`

Pans the view so the Playhead is in the center.

Multi-Word Match Search

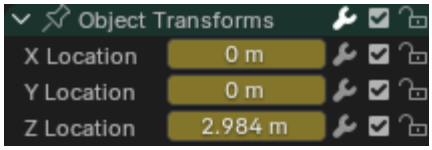
Lets you filter by multiple search terms instead of just one (in the search textbox above the channel list and in the Filters popover). The terms are space-separated, so you can for example type “loc rot” to find all channels that have “loc” or “rot” in their name. If this option were disabled, the list would only show channels containing the text “loc rot”, of which there are likely none.

Realtime Updates

Whether to update other views (such as the 3D Viewport) while you’re moving keyframes around. If disabled, the other views only get updated once you finish the move.

Show Sliders

Shows a value slider next to each channel. Adjusting such a slider automatically creates a keyframe.



Sliders.

Show Handles and Interpolation

Displays keyframes using shapes that represent their Bézier handle type. In addition, if a keyframe uses a non-default interpolation type for the curve segment that comes after it, this is indicated by a green line.



Handle types.

See Handles & Interpolation Display.

Show Curve Extremes

Detects keys where the curve changes direction, and marks them by displaying an arrow inside their shape. Local maxima (hills) are shown as up



arrows, while local minima (valleys) are shown as down arrows. Extreme markers.

A keyframe may show both arrows, namely when it's part of a summary row containing a channel with a maximum and one with a minimum.

Auto-Merge Keyframes
: Automatically merge keyframes that end up on the same frame after transformation.

Show Markers
: Shows the marker region (provided any markers have been defined). When disabled, the Marker menu is also hidden and marker operators are not available in this editor.

Show Seconds `Ctrl-T`
: Shows timing in seconds instead of frames.

Sync Visible Range
: Synchronizes the horizontal panning and scale of the editor with other time-based editors that also have this option enabled. That way, they always show the same section of time.

Set Preview Range `P`
: Lets you drag a box to define a time range for previewing. As long as this range is active, playback will be limited to it, letting you repeatedly view a segment of the animation without having to manually rewind each time.

  You can change the start or end frame using the corresponding button in the Timeline editor's Playback popover. Alternatively, you can simply run *Set Preview Range* again.

Clear Preview Range `Alt-P`
: Clears the preview range.

Set Preview Range to Selected `Ctrl-Alt-P`
: Applies a preview range that encompasses the selected keyframes.

Toggle Graph Editor `Ctrl-Tab`
: Changes the area's editor to the Graph Editor.

Area
: Area controls. See the user interface documentation for more information.

Filters

These filters are available in the funnel dropdown button in the header.

Summary
: Toggles the "Summary" row at the top of the Channels region. This row shows the union of all keyframes across all channels.

Only Show Selected
: Only show keyframes belonging to objects/bones/… that are selected.

Note

If this option is enabled, the Dope Sheet may not show all material keyframes of the selected objects. Instead, it only shows the keyframes belonging to the selected nodes in the Shader Editor.

Show Hidden
: Show keyframes from objects/bones/… that are hidden.

Only Show Errors
: Only show channels that have errors (for example, because they try to animate a property that doesn't exist on the object).

Search
: Filters the channel list by a search term (or multiple search terms if *Multi-Word Match Search* is enabled).

Filtering Collection
: Select a collection to only show keyframes from objects in that collection.

Filter by Type
: Filter curves by property type.

Sort Data-Blocks
: Sorts data-blocks alphabetically to make them easier to find.

  If your playback speed suffers because of this (should only really be an issue when working with lots of objects), you can turn it off.

Editing

Select Menu

See also

Selecting.

All `A`
: Selects all keyframes.

None `Alt-A`
: Deselects all keyframes.

Invert `Ctrl-I`
: Inverts the selection.

Box Select `B`
: Lets you drag a box and selects the keyframes inside it.

Box Select (Axis Range) `Alt-B`
: Lets you drag a box and selects the keyframes inside the corresponding time range, even if they're above or below the box.

Circle Select `C`
: Displays a circle around the cursor, which you can drag over keyframes to select them.

Lasso Select `Ctrl-RMB`
: Lets you draw a freehand shape and selects the keyframes inside it.

More `Ctrl-NumpadPlus`

Expand the selection to include the neighbors (in time) of the currently selected keys.

Less `Ctrl-NumpadMinus`
: Deselect keyframes with fewer than two selected neighbors.

Select Linked `L`
: Select keys that are on the same channel as a key that's already selected.

Columns on Selected Keys `K`
: Selects keys that are on the same frame as a key that's already selected.

Column on Current Frame `Ctrl-K`
: Selects all the keys that are on the current frame.

Columns on Selected Markers `Shift-K`
: Selects keys that are on the same frame as a selected marker.

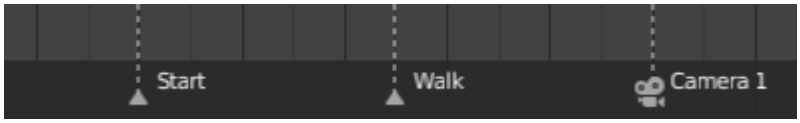
Between Selected Markers `Alt-K`
: Selects keys that lie between the leftmost and rightmost selected markers.

Before Current Frame `[`
: Select the keys that lie before (or on) the current frame. You can also click `Shift-Ctrl-LMB` anywhere to the left of the Playhead.

After Current Frame `]`
: Select the keys that lie after (or on) the current frame. You can also click `Shift-Ctrl-LMB` anywhere to the right of the Playhead.

Marker Menu

Markers are used to denote frames with key points or significant events within an animation. Like with most animation editors, they're shown at the bottom.



Markers in animation editor.

There are some options that are exclusive to the Dope Sheet editor:

Sync Markers
: Whether to also move the selected markers when moving the selected keyframes.

Show Pose Markers *Action Editor*
: Instead of showing the global scene markers, show the local pose markers (which only exist inside the action). While this option is active, the *Add Marker* menu item will also create pose markers instead of scene markers.

Make Markers Local *Action Editor*
: Converts the selected scene markers into pose markers, making them only visible inside the currently selected action.

For information about the other marker tools, see Editing Markers.

Channel Menu

See Graph Editor Channels.

Key Menu

Most items in this menu are documented on the Graph Editor's Editing page. One important difference is that scaling keyframes in the Dope Sheet Editor only moves them along the time axis (with the Playhead serving as the pivot point); it doesn't change their values.

The Dope Sheet editor has the following additional menu items:

Slide `Shift-T`
: Lets you stretch one set of keyframes across time while compressing an adjacent set to compensate, leaving the combined duration the same.

  To use this operator, first select a range of three or more keyframes, then place the mouse cursor somewhere in the middle and press `Shift-T`. The range will be temporarily split in two at the location of the cursor, indicated by a dashed vertical line. If you now move the mouse, the two halves of the range will change in length, and the keyframes within them will move accordingly. Click `LMB` to confirm or `RMB` to cancel.

Keyframe Type `R`
: Sets the type of the selected keyframes.

Snap

The toggle button enables/disables automatic keyframe snapping. The dropdown button shows a popover with the following options:

Snap To
: Type of element to snap to.

  Frame:: Snap to full frames.
  Second:: Snap to seconds.
  Nearest Marker:: Snap to the nearest Marker.

Absolute Time Snap
: When disabled, keyframes will move in increments of *Snap To*. For example, if you selected *Second* and have a keyframe that's currently on 0:06+5, dragging it to the right will snap it to 0:07+5. Its time increases by a second, and its subsecond offset of 5 frames remains the same.

  When enabled, keyframes will snap to multiples of *Snap To*. Taking the above example, the keyframe would snap to 0:07+0, removing the subsecond offset.

Proportional Editing

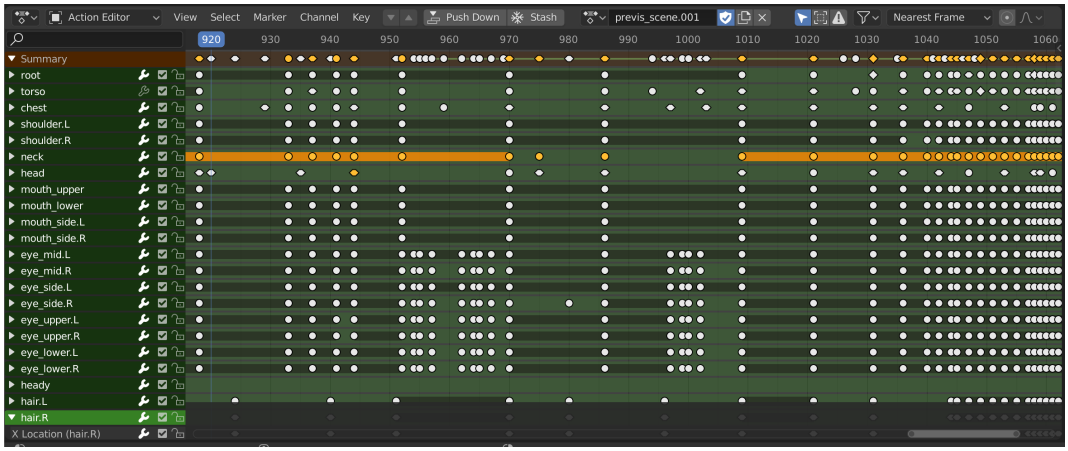
See Proportional Editing.

Modes

- Action Editor
  - Header
- Shape Key Editor
- Grease Pencil
  - Channels Region
  - Header
  - Main Region
  - Sidebar
- Mask

Action Editor

While the *Dope Sheet* mode lets you work with keyframes of many objects at the same time, the *Action Editor* mode focuses on the keyframes inside a single action.

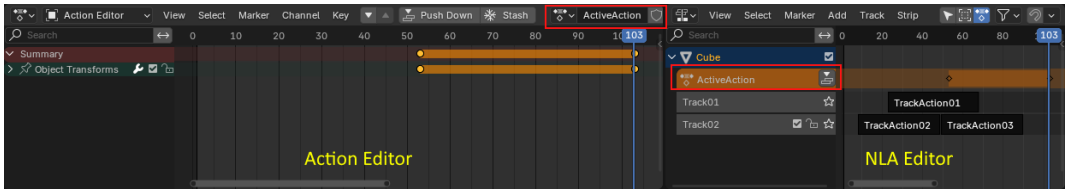


The Action Editor.

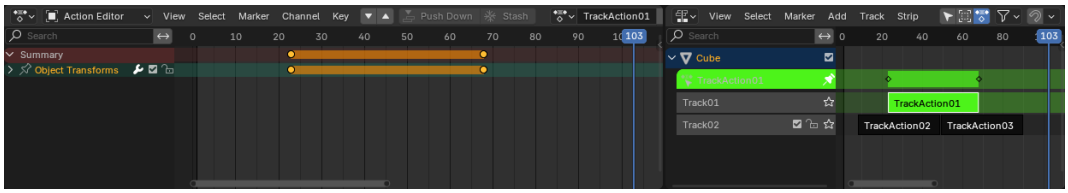
An action is a reusable animation segment – a collection of F-curves, where each curve describes how a certain property changes over time. Objects can reference one action as their *active action* and additional ones through Nonlinear Animation tracks.

Header

Previous/Next Layer (down/up arrows)
: Switches to editing the action in the track below/above the current one, automatically entering or leaving Tweak Mode as necessary. Also transfers the *Disable NLA stack* setting to the *Solo* setting and vice versa (see NLA tracks).



By default, the Action Editor shows the selected object's active action, which is stored in the Action Track at the top.



After clicking *Previous Layer*, we enter Tweak Mode for the action in the second track. The NLA Editor highlights it in green, and the Action Editor lets us edit its keyframes.

Push Down (strips with down arrow icon)

Creates a new NLA track below the Action Track and moves the active action into it. This is the same as clicking Push Down Action in the NLA editor.

Stash (snowflake icon)

Creates a new *muted* NLA track at the bottom and moves the active action into it. In effect, this sets the action aside for later use, disabling it so it no longer affects the animation. Later, you can choose to either unmute it again or delete it.

If you click *New Action* in the data-block menu for an object that already has an active action, that previous action will be stashed automatically.

Note

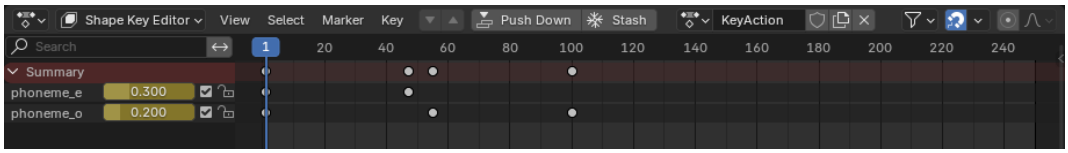
Both *Push Down* and *Stash* leave the object without an active action (meaning the Action Editor becomes empty and the action can no longer be edited). If you still want to make changes to the action, you can select

it in the NLA editor and press `Tab` to enter Tweak Mode, or use the Previous/Next Layer buttons as described above.

Action
: A data-block menu that lets you change – or clear – the object's active action.

Shape Key Editor

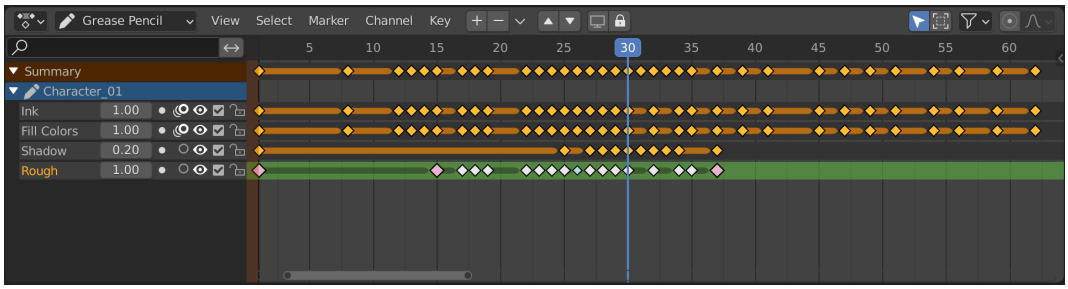
This mode shows the shape keys of the active object and lets you create/adjust keyframes for their values.



The Shape Key Editor.

Grease Pencil

This mode lets you adjust the timing of a Grease Pencil object's animation frames. It is especially useful for blocking out shots.



Channels Region

The Channels region shows the Grease Pencil object in light blue and its layers in gray. Layers have the following settings:

Opacity
: The layer's opacity.

Use Mask
: Toggle the layer's masks on or off.

Onion Skinning
: Toggle onion skinning.

Visibility (eye icon)
: Toggle layer visibility in the viewport and in render.

Show all keyframes (checkbox)
: When unchecked, the layer gets frozen in its current state, and moving to a different keyframe will no longer change its appearance.

Lock (padlock)
: Locked layers can't be edited.

Header

Add New Layer
: Adds a layer.

Remove Layer
: Removes the active layer.

Move Layer
: Moves the active layer down/up.

Isolate Layers (screen icon)
: Toggle whether the active layer is the only one that can be edited and is visible.

Isolate Layers (padlock icon)
: Toggle whether the active layer is the only one that can be edited.

Insert Keyframe

You can press `I` while hovering over the Dope Sheet Editor to insert a keyframe. It'll create a copy of the active frame if Additive Drawing is enabled, and a blank frame otherwise.

Copying Frames

It is possible to copy frames from one layer to another, or from object to object, using the *Copy* and *Paste* tools in the *Key* menu. Note that keyframes will be pasted into selected layers, so make sure you have a destination layer selected.

Main Region

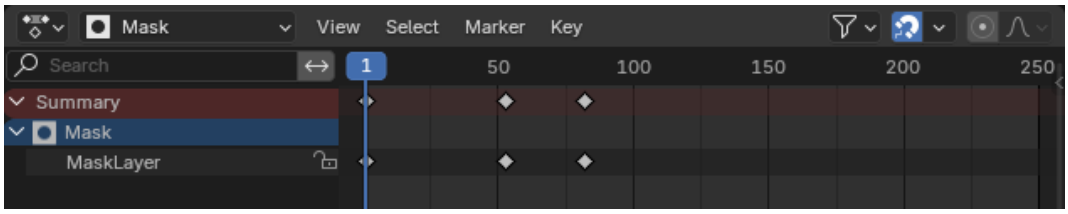
The keyframes can be manipulated like any other data in the *Dope Sheet*. Interpolated keyframes (alias breakdowns) are visualized as smaller light blue points.

Sidebar

The Sidebar contains a copy of the Grease Pencil Layer Properties.

Mask

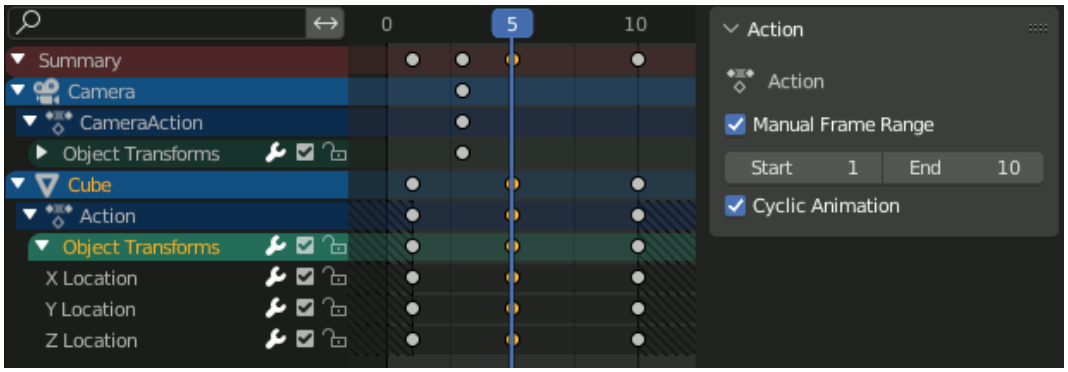
This mode shows all the masks in the blend-file (that have at least one layer) and lets you adjust their keyframes.



The Mask mode of the Dope Sheet Editor.

Sidebar

Action Panel



Actions with and without a Manual Frame Range in Dope Sheet.

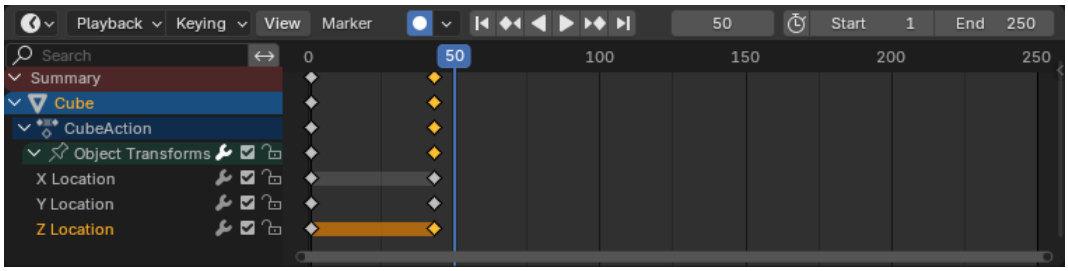
When the editor is in Action Editor mode, or in Dope Sheet mode with a channel selected that belongs to an action, this panel allows changing some settings of that action. See Action Properties for details.

Custom Properties

Create and manage your own properties to store data in the action's data block. See the Custom Properties page for more information.

Timeline

The *Timeline* editor is used to jump to different frames, manipulate keyframes, and control animation playback.



The Timeline.

Main View

The X axis represents time, with the numbers 0/50/100/… being frame numbers. The blue line is the *Playhead* indicating the current frame, and the diamond shapes are *Keyframes*, points where you specified a certain value for a certain property at a certain time.

Adjusting the View

Panning is done by dragging `MMB`.

Zooming is done by dragging `Ctrl-MMB`, rolling the mouse `Wheel`, or pressing `NumpadMinus`/`NumpadPlus`.

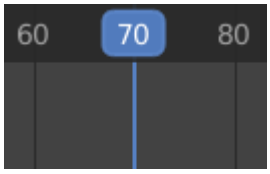
You can also use the scrollbars located at the bottom and the right of the editor.

Playhead

The *Playhead* is the blue vertical line showing the current frame number.

It can be moved to a new position by clicking or dragging `LMB` in the scrubbing area at the top.

You can also move it in single-frame increments by pressing `Left` or `Right`, or jump to the beginning or end frame by pressing `Shift-Left` or `Shift-Right`.



Playhead.

Frame Range

The *Frame Range* determines the length of the scene's animation. By default, it's set to start at frame 1 and end at frame 250. You can change this using the Start/End inputs in the Timeline header, or in the Output Properties.

Keyframes

By default, the timeline only shows keyframes for selected items. You can make it show all keyframes by unchecking *View ▸ Only Show Selected*.

You can click a keyframe to select it (and deselect all others), or click it while holding `Shift` to add it to the selection (or remove it if it was already selected). You can also drag a box to select multiple keyframes in one go.

To move the selected keyframes, simply drag one of them. Alternatively, you can press `G`, move the mouse, and click `LMB` to confirm (or `RMB` to cancel). You can also press `S` to scale the keyframes in relation to the Playhead.

Markers

See the Markers page for more information.

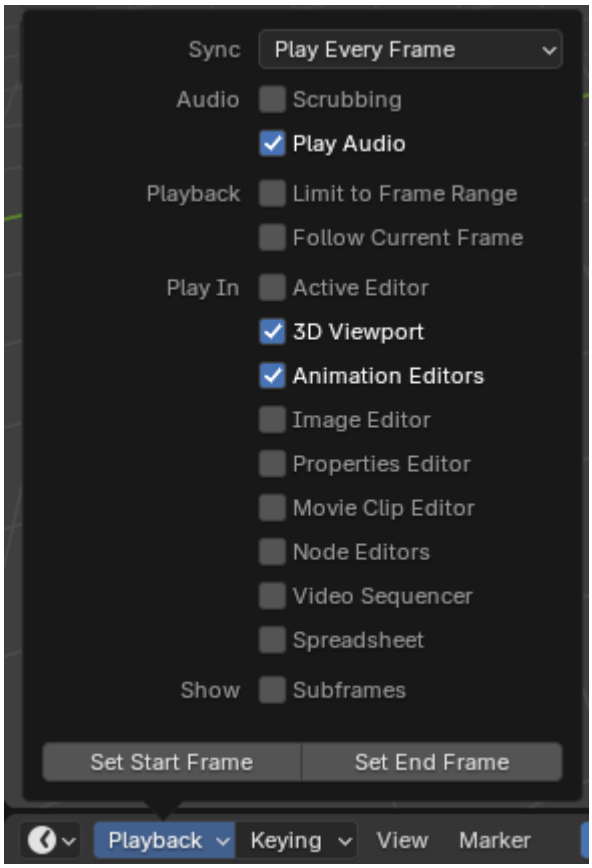
Header



Popovers for Playback and Keying; transport controls; and frame controls

Popovers

Playback Popover



Sync

If animation playback can't keep up with the desired Frame Rate, the actual frame rate (shown in the top left corner of the 3D Viewport) will turn red, and the *Sync* option determines how the situation should be handled.



3D Viewport red FPS.

Play Every Frame

Play every frame, even if this results in the animation playing slower than intended.

Frame Dropping
: Drop frames if playback becomes slower than the scene's frame rate.

Sync to Audio
: Drop frames if playback becomes too slow to remain synced with audio.

Audio

Scrubbing
: Play bits of the sound in the animation (if there is any) while you drag the Playhead around.

Play Audio
: Uncheck to mute all sound.

Playback

Limit to Frame Range
: Don't allow moving the Playhead outside of the Frame Range using the mouse.

Follow Current Frame
: Automatically pan the view to catch up when the Playhead goes off screen.

Play In
: Which editors to update on each animation frame. If an editor is unchecked, it'll only be updated once playback stops (with some exceptions where it'll update on each frame anyway). When starting playback in either the Graph Editor, Dope Sheet or the NLA Editor, all editors will play back regardless of the settings. This is a feature requested by animators to easily play back all views.

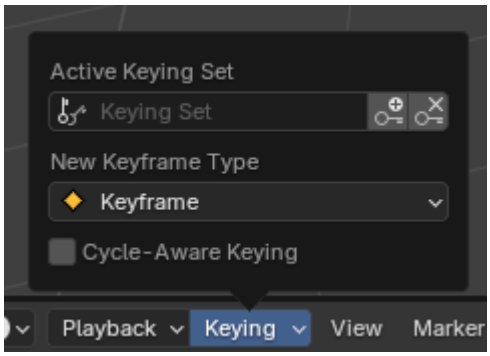
Show

Subframes
: Display and allow changing the current scene subframe.

Set Start/End Frame

Set the scene's start/end frame to the current frame. If the Preview Range is active (see Frame Controls), that one is changed instead.

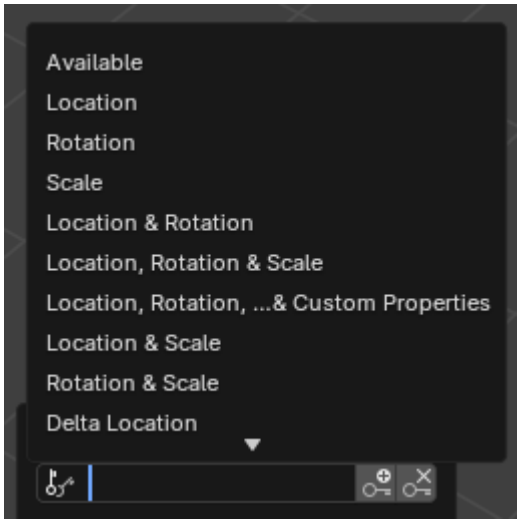
Keying Popover



The *Keying* popover contains options that affect keyframe insertion.

Active Keying Set

A *Keying Set* is a named collection of animatable properties. If you select one and then press `I` while not hovering over any input field, Blender will create keyframes for the properties in that keying set.



Timeline Keying Sets.

If you don't have a keying set selected, you'll get keyframes on a default set of properties instead (e.g. Location/Rotation/Scale for objects).

There are a number of predefined keying sets, but you can also create your own in the Keying Sets panel.

Insert Keyframes `I`

Insert keyframes on the current frame.

Delete Keyframes `Alt-I`

Delete keyframes on the current frame.

New Keyframe Type
: The keyframe type for newly created keyframes.

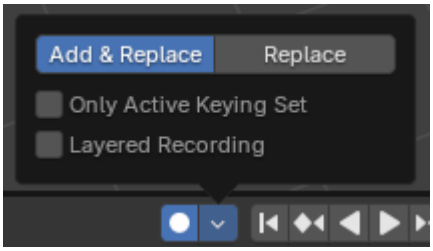
Cycle-Aware Keying
: When inserting keyframes into trivially cyclic curves, special handling is applied to preserve the cycle integrity (most useful while tweaking an established cycle):

  - If a key insertion is attempted outside of the main time range of the cycle, it is remapped back inside the range.
  - When overwriting one of the end keys, the other one is updated accordingly.

  In addition, when adding a new curve into an action with a Manual Frame Range and *Cyclic Animation* enabled, the curve is automatically made cyclic with the period matching the frame range. For convenience, this check and conversion is also done before adding the second keyframe to such a curve.

Auto Keying

When the record button () is enabled, Blender will automatically create keyframes on the current frame whenever you transform an object or bone in the 3D Viewport (or change one of its transform properties in the Properties Editor).



Auto Keying button.

One special use case is to record a camera path as you fly through the scene. See Fly/Walk Navigation.

Note

Auto Keying only works for transform properties (Location, Rotation, Scale). It won't create a keyframe if you change, say, the color of a material – you still have to do that manually.

Mode
: Add & Replace
: Add or replace keyframes as needed.

: Replace
: Only replace existing keyframes.

Only Active Keying Set
: By default, Auto Keying will create keyframes even for properties that are not in the active keying set. Use this checkbox to change that.

Layered Recording
: Adds a new NLA Track for every pass made over the animation to allow non-destructive tweaking.

Menus

View Menu

Adjust Last Operation
: Displays a pop-up panel to alter properties of the last completed operation. See Adjust Last Operation.

Channels
: Show or hide the Channels region (the tree of objects and animatable properties on the left).

Frame All `Home`
: Pans and zooms the view so that all keyframes are visible.

Frame Scene/Preview Range

Reset the horizontal view to the current scene frame range, taking the preview range into account if it is active.

Go to Current Frame `Numpad0`
: Centers the Timeline to the Playhead.

Show Markers
: Shows the Markers region (if any markers are defined). When disabled, the Marker Menu is also hidden and marker operators are not available in this editor.

Show Seconds `Ctrl-T`
: Shows the time on the X axis and the *Playhead* as timestamps instead of frame numbers. A timestamp such as `01:03+02` means "1 minute, 3 seconds, 2 frames."

Sync Visible Range
: Synchronizes the horizontal panning and scale of the editor with other time-based editors that also have this option enabled. That way, they always show the same section of time.

Only Show Selected
: Only show keyframes related to the selected items. This could be objects, bones, nodes, and so on.

Note

If this option is enabled, the Timeline may not show all material keyframes of the selected objects. Instead, it only shows the keyframes belonging to the selected nodes in the Shader Editor.

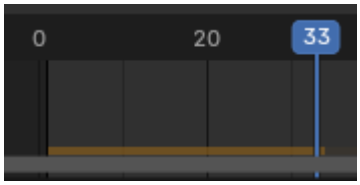
Only Show Errors
: Only show curves and drivers that are disabled or have errors. Useful for debugging.

Cache

Show Cache

Which simulation caches to show on the timeline.

Baked simulations will be shown as fully opaque, cached simulations will be slightly transparent, and invalid caches will be slightly transparent with dark diagonal stripes.



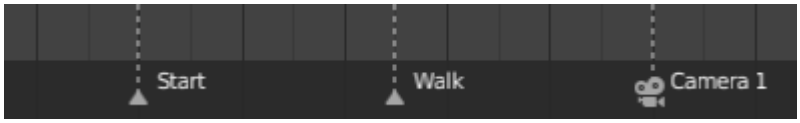
Timeline Cache.

Area

Area controls. See the user interface documentation for more information.

Marker Menu

Markers are used to denote frames with key points or significant events within an animation. Like in most animation editors, they're shown at the bottom of the Timeline.



Markers in an animation editor.

For descriptions of the different marker tools, see Editing Markers.

Transport Controls

These buttons are used to set the current frame and control playback.



Transport controls.

Jump to Start `Shift-Left`
: Sets the Playhead to the start of the frame range.

Jump to Previous Keyframe `Down`
: Moves the Playhead to the previous keyframe.

Rewind `Shift-Ctrl-Spacebar`
: Starts playing the animation in reverse.

Play `Spacebar`
: Starts playing the animation.

Jump to Next Keyframe `Up`
: Moves the Playhead to the next keyframe.

Jump to End `Shift-Right`
: Sets the Playhead to the end of the frame range.

Pause `Spacebar`
: Stops playing the animation.

Frame Controls

Current Frame `Alt-Wheel`
: The number of the frame that's currently being displayed in the 3D Viewport. This is also the location of the Playhead.

Use Preview Range
: The Preview Range is an alternative Frame Range that you can use for focusing on a particular part of the animation. It lets you repeatedly play a short segment without having to manually rewind or change the frame range of the entire scene.

  This range only affects the preview in the 3D Viewport; it doesn't affect rendering.

  The boundaries of the Preview Range are shown in dark orange. You can quickly configure and enable it by pressing `P` and dragging a box.

To disable it, you can press `Alt-P`.

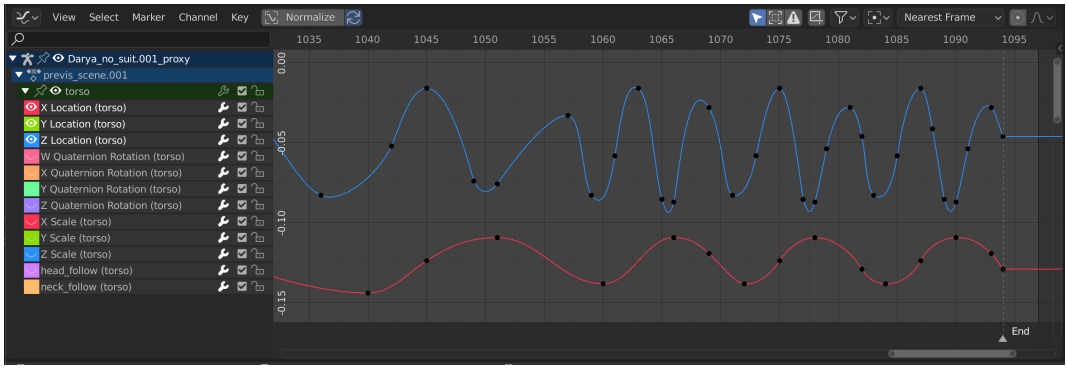
Start/End Frame
: The start/end frame of the scene (or the preview range, if active).

Graph Editor

- Introduction
  - Main Region
  - Header
  - Sidebar Region
- Channels
  - Introduction
  - Editing
- F-Curves
  - Introduction
  - Editing
  - Properties
  - F-Curve Modifiers

Introduction

The Graph Editor lets you edit animation curves, which determine how properties change over time.



The Graph Editor.

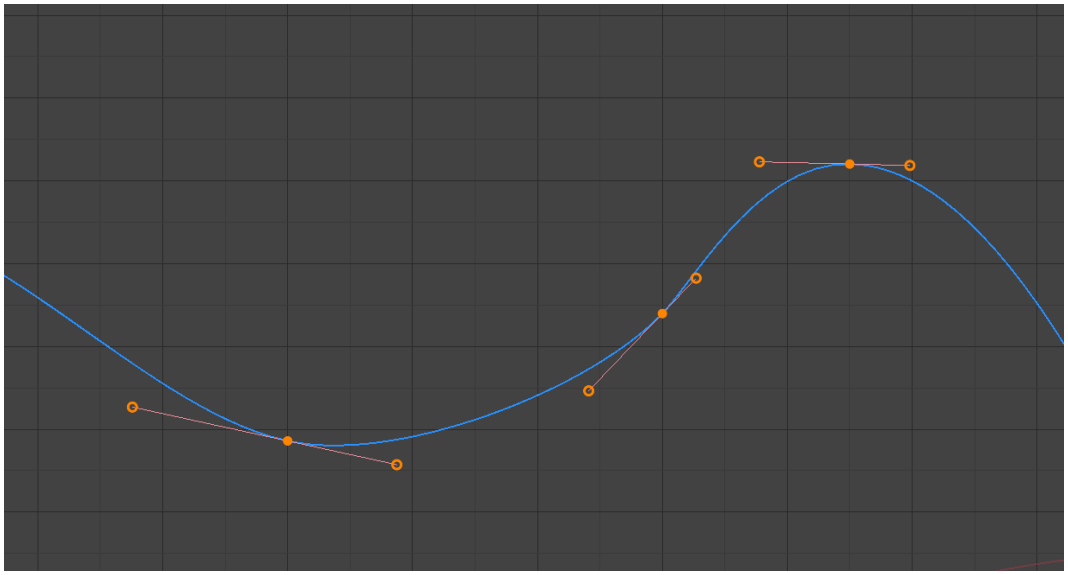
Main Region

The curve view allows you to view and edit F-Curves. An F-Curve has several key parts:

Curve
: The curve describes how the value of a property (Y axis) evolves over time (X axis).

Keyframes
: Keyframes are user-defined values on certain frames and are represented by little black discs that become orange when selected. The values on the other frames are calculated automatically by interpolating between these keyframes.

Handles
: Each keyframe has two handles – points that can be dragged around to influence the shape of the curve around it.



A simple curve. The discs are keyframes, and the circles are their handles.

See also

See F-Curves for more info.

Navigation

As with most editors, you can:

Pan
: Pan the view by dragging with `MMB`.

Zoom
: Zoom in and out with the mouse `Wheel`.

Scale View
: Scale the view horizontally or vertically by dragging with `Ctrl-MMB`.

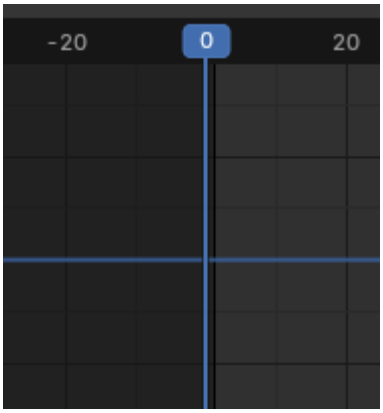
You can also use the scrollbars.

Tip

You can focus the view on the curve of an animated property by right clicking it and choosing *View in Graph Editor*. If you want to set up a hotkey for this, you need to open the Keymap preferences, open the *User Interface* category, click *Add New*, fill in the operator name `anim.view_curve_in_graph_editor`, and finally choose a shortcut. Normally this can be done more easily by right clicking the context menu item and choosing *Assign Shortcut*, but in this case, the shortcut would be added to the wrong category and not work.

Playhead & 2D Cursor

The current frame is represented by a vertical blue line called the *Playhead*. As in the Timeline, you can move it by clicking or dragging with `LMB` in the scrubbing area at the top.



Graph Editor 2D Cursor.

Combined with the horizontal blue line, the Playhead forms the *2D Cursor* which can be used as a pivot point for rotating and scaling. You can disable the horizontal line using *View ▸ Show Cursor* or *Sidebar ▸ View ▸ Show Cursor*.

The 2D Cursor can be moved by clicking or dragging with `Shift-RMB` or by adjusting its coordinates in the View tab of the Sidebar.

Header

View Menu

Sidebar `N`
: Shows or hides the Sidebar Region.

Adjust Last Operation

Displays a pop-up panel to alter properties of the last completed operation. See Adjust Last Operation.

Channels
Shows or hides the Channels Region.

Frame Selected `NumpadPeriod`
Pans and zooms the view to focus on the selected keyframes.

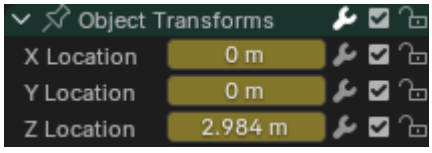
Frame All `Home`
Pans and zooms the view to show all keyframes.

Frame Scene/Preview Range
Reset the horizontal view to the current scene frame range, taking the preview range into account if it is active.

Go to Current Frame `Numpad0`
Centers the area to the Playhead.

Realtime Updates
Whether to update other views (such as the 3D Viewport) while you're moving keyframes around. If disabled, the other views only get updated once you finish the move.

Show Sliders
Shows a value slider next to each channel. Adjusting such a slider automatically creates a keyframe.



Sliders.

Auto-Merge Keyframes
Automatically merge keyframes that end up on the same frame after transformation.

Auto-Lock Key Axis
Automatically locks the movement of keyframes to the axis that best matches the direction of the mouse cursor.

Show Markers
: Shows the marker region. When disabled, the Marker Menu is also hidden and marker operators are not available in this editor.

Show Cursor
: Toggles the visibility of the horizontal blue line (see Playhead & 2D Cursor).

Show Seconds `Ctrl-T`
: Show timing in seconds instead of frames. As an example, the timestamp `01:03+02` means "1 minute, 3 seconds, 2 frames."

Sync Visible Range
: Synchronizes the horizontal panning and scale of the editor with other time-based editors that also have this option enabled. That way, they always show the same section of time.

Show Extrapolation
: Toggles the visibility of the extrapolated portion of curves.

Show Handles `Ctrl-H`
: Toggles the display of keyframe handles.

Only Selected Keyframes Handles
: Only shows the handles for the selected keyframes.

Set Preview Range `P`
: Lets you drag a box to define a time range for previewing. As long as this range is active, playback will be limited to it, letting you repeatedly view a segment of the animation without having to manually rewind each time.

  You can change the start or end frame using the corresponding button in the Timeline editor's Playback popover. Alternatively, you can simply run *Set Preview Range* again.

Clear Preview Range `Alt-P`
: Clears the preview range.

Set Preview Range to Selected `Ctrl-Alt-P`
: Applies a preview range that encompasses the selected keyframes.

Toggle Dope Sheet
: Changes the area's editor to the Dope Sheet Editor.

Area
: Area controls. See the user interface documentation for more information.

Select Menu

All `A`
: Selects all keyframes and handles.

None `Alt-A`
: Clears the selection.

Invert `Ctrl-I`
: Inverts the selection.

Box Select `B`
: Lets you drag a box and selects the keyframes and handles inside it.

Box Select (Axis Range) `Alt-B`
: Lets you drag a box and selects the keyframes and handles inside the corresponding time range, even if they're above or below the box.

Box Select (Include Handles)
: Selects keyframes and their handles inside the defined box.

Circle Select `C`

Displays a circle around the cursor, which you can drag over keyframes and handles to select them.

Lasso Select `Ctrl-RMB`
: Lets you draw a freehand shape and selects the keyframes and handles inside it.

Columns on Selected Keys `K`
: Selects keys that are on the same frame as a key that's already selected.

Column on Current Frame `Ctrl-K`
: Selects all the keys that are on the current frame.

Columns on Selected Markers `Shift-K`
: Selects keys that are on the same frame as a selected marker.

Between Selected Markers `Alt-K`
: Selects keys that lie between the leftmost and rightmost selected markers.

Before Current Frame `[`
: Select the keys that lie before (or on) the current frame. You can also click `Shift-Ctrl-LMB` anywhere to the left of the Playhead.

After Current Frame `]`
: Select the keys that lie after (or on) the current frame. You can also click `Shift-Ctrl-LMB` anywhere to the right of the Playhead.

Select Handles
: Selects the handles of the currently selected keyframes.

Select Keys
: Selects the keyframes of the currently selected handles.

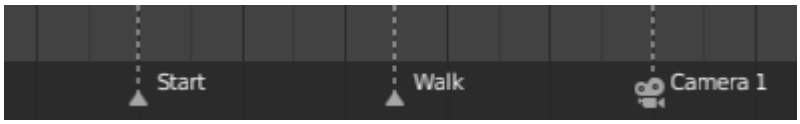
Select More `Ctrl-NumpadPlus`
: Expands the selection to include the neighbors (in time) of the currently selected keys.

Select Less `Ctrl-NumpadMinus`
: Deselects keyframes with fewer than two selected neighbors.

Select Linked
: Selects keys that are on the same curve as a key that's already selected.

Marker Menu

Markers are used to denote frames with key points or significant events within an animation. Like with most animation editors, they're shown at the bottom.



Markers in animation editor.

For descriptions of the different marker tools, see Editing Markers.

Channel Menu

See Editing Channels.

Key Menu

See Editing F-Curves.

Normalize

Scales the display of each curve so that they all (appear to) occupy the same value range, going from -1 to 1. This can make editing easier when you're

working with curves whose value ranges are far apart.

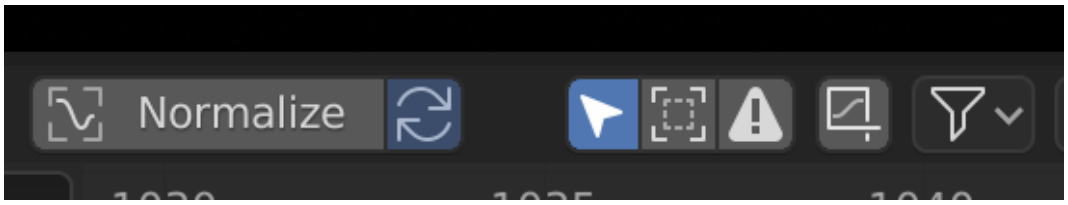
When you enable this option, the view is zoomed accordingly and the area outside the normalized value range is darkened.

If a preview range is defined, keyframes within the range are normalized, while the others are scaled proportionally.

Auto Normalization

Automatically recalculate curve normalization on every curve edit.

View Controls



View controls.

Show Only Selected
: Only show curves belonging to objects/bones/… that are selected.

Show Hidden
: Show keyframes from objects/bones/… that are hidden.

Show Only Errors
: Only show channels that have errors (for example, because they try to animate a property that doesn't exist on the object).

Create Ghost Curves (framed F-Curve icon)
: Creates a snapshot of the current curves and shows it in the background so that you can use it as a reference. Click the button again to clear the snapshot.

Filter (funnel icon)

Search
: Filters the channel list by a search term.

Filtering Collection
: Select a collection to only show keyframes from objects in that collection.

Filter by Type
: Filter curves by property type.

Sort Data-Blocks
: Sorts data-blocks alphabetically to make them easier to find.

  If your playback speed suffers because of this (should only really be an issue when working with lots of objects), you can turn it off.

Transform Controls



Transform controls.

Pivot Point
: Pivot point for rotating and scaling.

  Bounding Box Center
  : Center of the smallest possible box around the selected keyframes.

  2D Cursor
  : The intersection between the Playhead and the horizontal Cursor line.

  Individual Centers
  : Rotate/scale each handle around its keyframe.

Snap

The icon toggles snapping on or off. The dropdown offers the following options:

Snap To
: Type of element to snap to.

  Frame
  : Snap to full frames.

  Second
  : Snap to seconds.

  Nearest Marker
  : Snap to the nearest Marker.

Absolute Time Snap
: When disabled, keyframes will move in increments of *Snap To*. For example, if you selected *Second* and have a keyframe that's currently on 0:06+5, dragging it to the right will snap it to 0:07+5. Its time increases by a second, and its subsecond offset of 5 frames remains the same.

  When enabled, keyframes will snap to multiples of *Snap To*. Taking the above example, the keyframe would snap to 0:07+0, removing the subsecond offset.

Proportional Editing `0`
: See Proportional Editing.

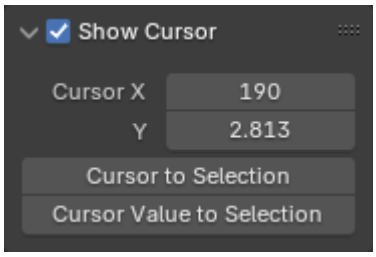
Sidebar Region

View Tab

Show Cursor
: Toggles the visibility of the 2D Cursor's horizontal line.

Cursor X, Y

Shows, and lets you change, the X coordinate (current frame) and Y coordinate (value) of the 2D Cursor.



View Tab.

Cursor to Selection
: Places the 2D Cursor at the average time and value of the selected keyframes.

Cursor Value to Selection
: Places the 2D Cursor at the average value of the selected keyframes, leaving its time unchanged.

F-Curve Tab

See F-Curve Properties.

Modifiers Tab

See F-Curve Modifiers.

Channels

- Introduction
  - Channels Region
- Editing
  - Delete Channels
  - Un/Group Channels
  - Toggle/Enable/Disable Channel Settings
  - Toggle Channel Editability
  - Extrapolation Mode
  - Add F-Curve Modifier
  - Show/Hide
  - Expand/Collapse Channels
  - Move
  - Revive Disabled F-Curves
  - Keys to Samples
  - Samples to Keys
  - Sound to Samples
  - Bake Channels
  - Discontinuity (Euler) Filter
  - Frame Selected Channels

Introduction

Channels Region



The Channels region.

This region is found on the left side of time-based editors like the Timeline, the Dope Sheet Editor, and the Graph Editor. It shows a tree of items (objects, bones…) and their animated properties, with the latter also being

called “channels.” Each channel has an associated F-curve describing how its value changes over time.

The rows are color-coded as follows:

- Dark blue: scenes, objects
- Light blue: actions, shape keys etc.
- Green: channel groups
- Gray: channels

Search `Ctrl-F`
: Lets you filter the channels by typing a part of their name. Click the Invert button to instead show channels that *don’t* include the search text.

Controls

The headers contain the following toggle buttons:

Pin (pin icon)
: Keep the row and its children visible even when selecting a different object.

Hide (eye icon)
: Hides the keyframes and curve associated with the channel.

Modifiers (wrench icon)
: Deactivates the modifiers of the curve.

Mute (checkbox)
: Deactivates the curve, making the animation behave as though it doesn’t exist.

Lock `Tab` (padlock icon)
: Prevent the curve from being edited.

Note

This also works in the Nonlinear Animation Editor, but note that it only locks the strips there, not the underlying F-curves.

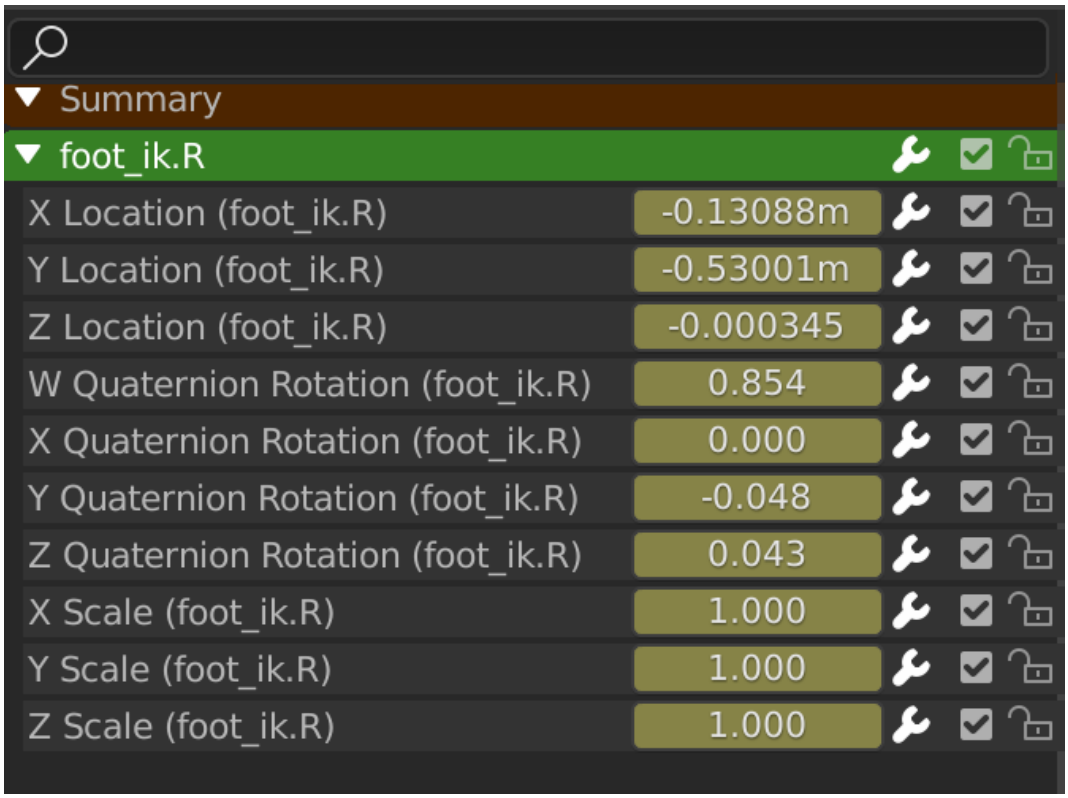
Selection

- Select single header: click `LMB`
- Add/Remove single header to/from selection: click `Ctrl-LMB`
- Select range: click `Shift-LMB`
- Select All: `A`
- Deselect All: press `Alt-A` or double-tap `A`
- Box Select: drag `LMB`
- Box Add: drag `Shift-LMB`
- Box Remove: drag `Ctrl-LMB`
- Select all keyframes in the channel: double-click `LMB` on its header.

Editing

- Rename (anything but a channel): double-click `LMB`
- Delete selected: `X` or `Delete`
- Lock selected: `Tab`

Sliders



The Action editor showing sliders.

If you enable *View ▸ Show Sliders*, the region will show a value slider next to each channel. Changing such a slider will change the value of the curve at the current frame, creating a keyframe if one doesn't already exist.

Editing

Delete Channels

Reference

Menu::	*Channel ▸ Delete Channels*
Shortcut::	`Delete`, `X`

Removes the selected channels from the current action.

Warning

Make sure the mouse cursor is hovering over the channel region before using the keyboard shortcuts. If it's hovering over the main region, you'll only delete the selected keyframes, not the full channels.

Un/Group Channels

Reference

Menu::	*Channel ▸ Un/Group Channels*
Shortcut::	`Ctrl-Alt-G`, `Ctrl-G`

Un/Groups the selected channels into a collection that can be renamed by double-clicking its name. Grouping channels helps keep the view more organized.

Toggle/Enable/Disable Channel Settings

Reference

Menu:: *Channel ▸ Toggle/Enable/Disable Channel Settings*
Shortcut:: `Shift-W`, `Shift-Ctrl-W`, `Alt-W`

Toggles, enables, or disables a certain setting for the selected channels:

Protect
: When a channel is protected (closed padlock icon), it can't be edited. Instead of pressing `Shift-W` and selecting *Toggle*, you can also simply press `Tab`.

Mute
: When a channel is muted (empty checkbox), it doesn't affect the animation.

Toggle Channel Editability

Reference

Menu:: *Channel ▸ Toggle Channel Editability*
Shortcut:: `Tab`

Locks or unlocks a channel for editing.

Extrapolation Mode

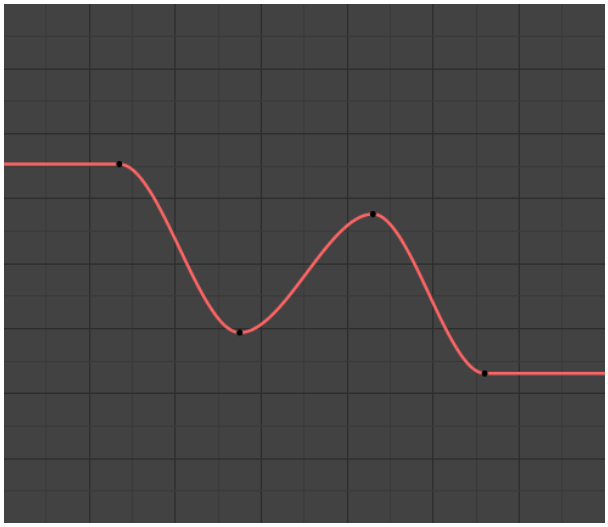
Reference

Menu:: *Channel ▸ Extrapolation Mode*
Shortcut:: `Shift-E`

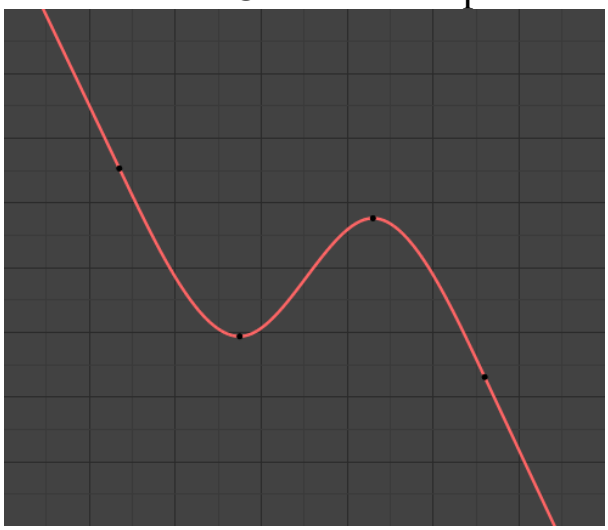
Changes how the curve behaves before its first keyframe and after its last keyframe.

Constant:: Continue in a straight line, keeping the same value as the first/last keyframe. This is the default.

Linear::



Constant extrapolation.



Linear extrapolation.

Continue in a straight line, keeping the same slope as on the first/last keyframe.

Make Cyclic:: Repeat the whole curve. This works by adding a Cycles modifier.

Clear Cyclic:: Remove the above modifier, making the curve non-repeating again.

Add F-Curve Modifier

Reference

Menu::	*Channel ▸ Add F-Curve Modifier*
Shortcut::	`Shift-Ctrl-M`

Shows a submenu from where you can add a modifier to the active curve. Settings for these modifiers can be found in *Sidebar ▸ Modifiers*.

Show/Hide

Hide Selected Curves `H`
: Hides the selected curves.

Hide Unselected `Shift-H`
: Hides all curves except the selected ones.

Reveal Curves `Alt-H`
: Shows all previous hidden curves.

Expand/Collapse Channels

Reference

Menu::	*Channel ▸ Expand/Collapse Channels*
Shortcut::	`NumpadPlus`, `NumpadMinus`

Expands or collapses the selected headers.

Move

Reference

Menu::	*Channel ▸ Move…*

Lets you reorder the selected channels in the list:

- To the top `Shift-PageUp`

- Up one line `PageUp`
- Down one line `PageDown`
- To the bottom `Shift-PageDown`

Revive Disabled F-Curves

Reference

Menu:: *Channel ▸ Revive Disabled F-Curves*

Clears the "disabled" tag from all F-Curves to get broken F-Curves working again. (A curve is broken if it references a property that doesn't exist.)

Keys to Samples

Reference

Menu:: *Channel ▸ Keys to Samples*
Shortcut:: `Alt-C`

Switches the selected curves from interpolating between a set of keyframes to using a sampled value at each full frame. **This is a destructive process that removes the ability to edit the curve.** It's mainly used to reduce the file size with large datasets, as samples take up less space than keyframes.

Between samples (on subframes), the curve interpolates linearly.

Samples to Keys

Reference

Menu:: *Channel ▸ Samples to Keys*

Switches the selected curves from using samples to using keyframes, making them editable. Note that this creates a keyframe on every frame.

Sound to Samples

Reference

Menu:: *Channel ▸ Sound to Samples*

Creates a sampled curve based on a sound file. Use *Samples to Keys* if you need to edit it.

Lowest Frequency
: Cutoff frequency of a high-pass filter that is applied to the audio data.

Highest Frequency
: Cutoff frequency of a low-pass filter that is applied to the audio data.

Attack Time
: Value for the hull curve calculation that tells how fast the hull curve can rise. The lower the value, the steeper it can rise.

Release Time
: Value for the hull curve calculation that tells how fast the hull curve can fall. The lower the value, the steeper it can fall.

Threshold
: Minimum amplitude value needed to influence the hull curve.

Accumulate
: Only the positive differences of the hull curve amplitudes are summarized to produce the output.

Additive
: The amplitudes of the hull curve are summarized. If *Accumulate* is enabled, both positive and negative differences are accumulated.

Square
: Gives the output as a square curve. Negative values always result in -1, and positive ones in 1.

  Square Threshold
  : All values lower than this threshold result in 0.

Bake Channels

Reference

Menu:: *Channel ▸ Bake Channels*

Generates new keyframes for the selected curves.

Frame Range
: The range that will be baked. Defaults to the scene range or preview range.

Frame Step
: Distance between keyframes. Can be used to create a keyframe every 10 frames or even every half frame.

Remove Outside Range
: Removes existing keys outside the specified baking range.

Interpolation Type
: The interpolation type for the new keys.

Bake Modifiers
: If enabled, the new keyframes are based on the modified curve, and the modifiers get deleted.

  If disabled, the new keyframes are based on the original curve, and the modifiers stay applied.

Discontinuity (Euler) Filter

Reference

Menu:: *Channel ▸ Discontinuity (Euler) Filter*

Cleans up Euler rotation channels that suffer from Gimbal Lock. The channels of all three euler rotation axes need to be selected for this to work.

Frame Selected Channels

Reference

Menu:: *Channel ▸ Frame Selected Channels*
Shortcut:: `NumpadPeriod`

Pans and zooms the view to show all keyframes of the selected curves. You can also click a channel with `Alt-MMB`.

F-Curves

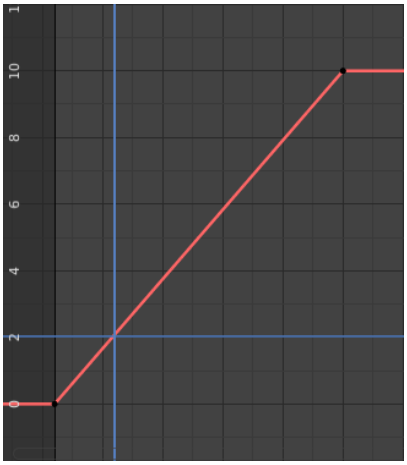
- Introduction
  - Direction of Time
- Editing
  - Transform
  - Snap
  - Mirror
  - Jump to Selected
  - Insert
  - Copy/Paste
  - Duplicate
  - Delete
  - Handle Type
  - Interpolation Mode
  - Easing Type
  - Density
  - Blend
  - Smooth
- Properties
  - Active F-Curve
  - Active Keyframe
- F-Curve Modifiers
  - Interface
  - Adding a Modifier
  - Types of Modifiers

Introduction

Blender lets you animate almost any property, going from the X coordinate of an object to the transparency of a material. The evolution of a property's value over time is described by a *function curve*, or F-Curve for short.

An important aspect of F-Curves is that they can interpolate. This saves you the effort of manually configuring a value on every single frame, which would be highly impractical. Instead, you define just a few values on key frames, and let the curve calculate the values on all the other frames.

The example curve on the right has two such keyframes (indicated by black dots): one on frame 0 with value 0, and another on frame 25 with value 10. The curve automatically calculates the values for the other frames, such as for frame 5 where the value is 2.



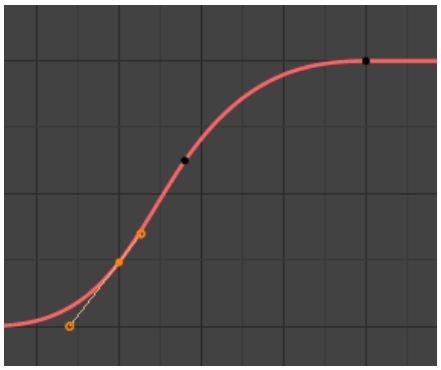
Example of interpolation.

Direction of Time

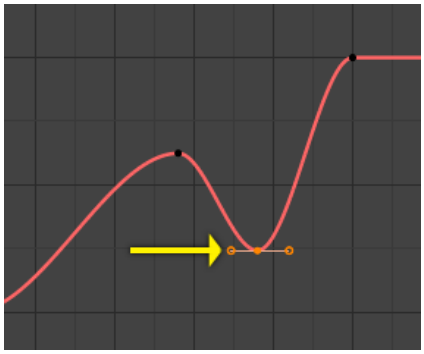
F-Curves are similar to Curve objects in that they interpolate between a set of user-defined control points. However, because their purpose is to define a *single* value on every frame, there's an important difference: F-curves can't be closed or otherwise made to turn back on themselves. They always continue going further to the right.

If you try to make a curve go left by dragging one control point past another, it switches the order of the points to prevent this.

Two control points switching: the curve cannot go back in time!



Before moving the second keyframe.



After moving the second keyframe.

Editing

Transform

Reference

Mode::	Edit Mode
Menu::	*Key* ▸ *Transform*

An F-Curve can be edited by transforming the locations of its keyframes.

Move, Rotate, Scale
: Like other elements in Blender, keyframes can be moved, rotated, and scaled as described in Basic Transformations.

Extend `E`
: Lets you quickly move the selected keyframes that are on a certain side of the Playhead. This is handy if you need to, say, move all the keyframes after a certain time point to the right to make space for new ones.

  To use this operator, first select some or all keyframes and place your mouse cursor to the left or right of the Playhead. Then, press `E`, move the mouse to move (only) the keyframes on that side of the Playhead, and press `LMB` to confirm (or `RMB` to cancel).

Tip

You can also change the *Key Frame* and *Value* properties in *Sidebar* ▸ *F-Curve* ▸ *Active Keyframe* if you want to specify exact numbers.

While transforming keyframes, you can hold `Shift` to move them more slowly for better precision, or `Ctrl` to move them in coarse increments.

Snap

Reference

Menu::	*Key* ▸ *Snap*
Shortcut::	`Shift-S`

Apart from using the snapping operators in this menu, you can also turn on snapping in the header.

Selection to Current Frame
: Set the selected keyframes' time to the current frame.

Selection to Cursor Value
: Set the selected keyframes' value to that of the 2D Cursor.

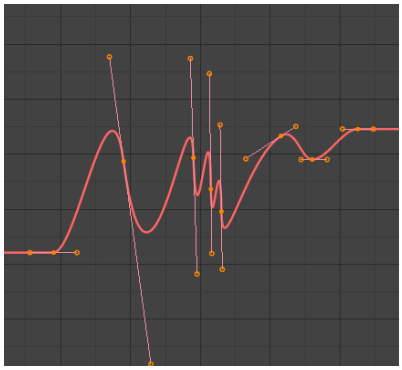
Selection to Nearest Frame
: Round the time of each keyframe to the nearest frame.

Selection to Nearest Second
: Round the time of each keyframe to the nearest second. You can use *View* ▸ *Show Seconds* to show seconds instead of frames at the top of the editor.

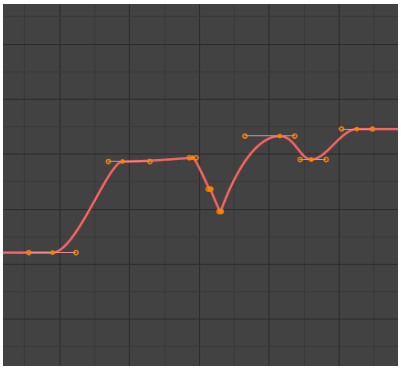
Selection to Nearest Marker
: Set the time of each keyframe to that of the nearest marker.

Flatten Handles
: Flatten the *Bézier* handles for the selected keyframes.

Flatten Handles snapping example.



Before Flatten Handles.



After Flatten Handles.

Equalize Handles
: Ensure selected keyframes' handles have equal length.

  Side
  : Which handles to affect (left, right, or both).

  Handle Length
  : Length to make selected keyframes' Bézier handles.

  Flatten
  : Make the values of the handles the same as their respective keyframes.

Cursor to Selected `Ctrl-G`
: Changes the time and value of the 2D Cursor to the average time and value of the selected keyframes.

Cursor Value to Selection
: Changes the value of the 2D Cursor to the average value of the selected keyframes.

Mirror

Reference

Menu:: *Key ▸ Mirror*
Shortcut:: `Ctrl-M`

Mirrors the selected keyframes across a reference point.

By Times over Current Frame
: Mirror horizontally across the current frame.

By Values over Cursor Value
: Mirror vertically across the 2D Cursor's value.

By Times over Zero Time
: Mirror horizontally across frame 0.

By Values over Zero Value
: Mirror vertically across value 0.

By Times over First Selected Marker
: Mirror horizontally across the first selected marker.

Jump to Selected

Reference

Menu:: *Key ▸ Jump to Selected*
Shortcut:: `Ctrl-G`

Places the 2D Cursor at the average time and value of the selected keyframes.

Insert

Reference

Menu:: *Key ▸ Insert*
Shortcut:: `I`

Adds new keyframes and selects them. Previously selected keyframes stay selected too.

All Channels
: Insert a keyframe on all visible and editable F-Curves using each curve's current value.

Only Selected Channels
: Insert a keyframe on the selected F-Curves using each curve's current value.

Only Active F-Curve
: Insert a keyframe on the active F-Curve using the curve's current value.

Active Channels at Cursor
: Insert a keyframe on the active F-Curve at the 2D Cursor's value.

Selected Channels at Cursor
: Insert a keyframe on the selected F-Curves at the 2D Cursor's value.

Copy/Paste

Reference

Menu::	*Key* ▸ *Copy*, *Key* ▸ *Paste*
Shortcut::	`Ctrl-C`, `Ctrl-V`

Use `Ctrl-C` to copy the selected keyframes and `Ctrl-V` to paste them. After pasting, the Adjust Last Operation panel provides some extra options:

Frame Offset
: Offsets the pasted keyframes horizontally so that…

  Frame Start
  : …the first one lands on the current frame.

  Frame End

…the last one lands on the current frame.

Frame Relative
…they land at the same distance from the current frame as when they were copied.

No Offset
…they stay at their original frames.

Value Offset
Offsets the pasted keyframes vertically so that…

Left Key
…the first one has the value of the existing keyframe to the left of the Playhead.

Right Key
…the last one has the value of the existing keyframe to the right of the Playhead.

Current Frame Value
…the first one has the value of the curve at the current frame.

Cursor Value
…the first one has the value of the 2D Cursor.

No Offset
…they keep their original values.

Type
Mix
Integrates the pasted keyframes with existing ones, only overwriting those that share a frame.

Overwrite All
Removes all previous keyframes in the target F-Curves.

Overwrite Range

Within each F-Curve, remove the existing keyframes that are in the range of the keyframes pasted into it.

Overwrite Entire Range
: Within each F-Curve, remove the existing keyframes that are in the range of all pasted keyframes combined.

Flipped
: If you copied keyframes from one or more pairs of symmetrically opposite bones, enabling this option will paste the keyframes of the left bones into the curves of the right ones and vice versa. In addition, the values are inverted, effectively mirroring the animation.

Duplicate

Reference

Menu::	*Key ▸ Duplicate*
Shortcut::	`Shift-D`

Duplicates the selected keyframes. You can reposition them by moving the mouse.

Delete

Reference

Menu::	*Key ▸ Delete*
Shortcut::	`X`, `Delete`

Pressing `X` or `Delete` opens a pop-up menu from where you can delete the selected keyframes.

Handle Type

Reference

Menu:: *Key ▸ Handle Type*
Shortcut:: `V`

Sets the handle type of the selected keyframes.

Interpolation Mode

Reference

Menu:: *Key ▸ Interpolation Mode*
Shortcut:: `T`

Sets the interpolation mode of the selected keyframes. This determines the curve interpolation between each keyframe and the next.

Easing Type

Reference

Menu:: *Key ▸ Easing Type*
Shortcut:: `Ctrl-E`

Sets the easing mode of the selected keyframes. This determines whether easing is applied to the left side, right side, or both sides of the curve segments between each keyframe and the next.

Density

Decimate

Reference

Menu:: *Key ▸ Density ▸ Decimate (Ratio)*
Menu:: *Key ▸ Density ▸ Decimate (Allowed Change)*

Simplifies an F-Curve by removing the keyframes that influence its shape the least.

Mode
: How to pick the number of keyframes to delete.

  Ratio
  : Deletes a certain percentage of keyframes.

    Remove
    : The percentage of keyframes to remove.

  Error Margin
  : Deletes as many keyframes as possible while ensuring the F-Curve's shape changes no more than a certain amount.

    Max Error Margin
    : How much the decimated curve may deviate from the original.

Bake Keyframes

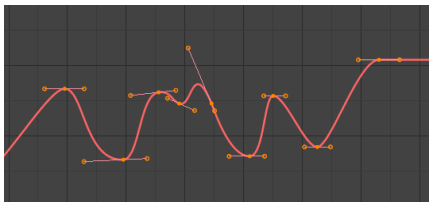
Reference

Menu:: *Key ▸ Density ▸ Bake Keyframes*
Shortcut:: `Shift-Alt-O`

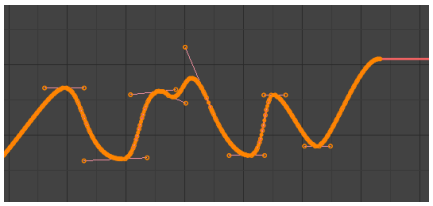
Creates a keyframe at every frame.

See also

Bake Channels, which offers options on what range to bake and how.



F-Curve before baking.



F-Curve after baking.

Clean Keyframes

Reference

Menu:: *Key* ▸ *Density* ▸ *Clean Keyframes*
Shortcut:: X

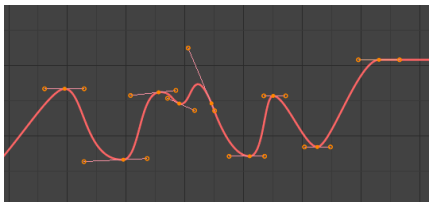
Finds redundant keyframes among the selected ones and deletes them. A keyframe is seen as redundant if it has the same value as its neighbors – even if the curve segments around it aren't flat.

Tip

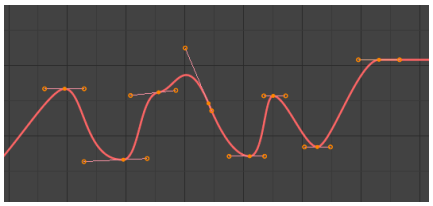
This operator is likely to change the shape of the affected curves, so it's best run after e.g. bulk keyframe insertion on all the bones of an armature (which creates useless keyframes on bones that haven't moved) and before tweaking the curves by hand.

Threshold
: Value threshold. By increasing this, you can also delete keyframes that *almost* have the same value as their neighbors.

Channels
: Cleans all the keyframes (even unselected ones) in the selected F-Curves. If a curve is left with only one keyframe, it's deleted entirely.



F-Curve before cleaning.



F-Curve after cleaning.

Blend

Reference

Menu:: *Key ▸ Blend*
Shortcut:: `Alt-D`

Adjusts the values of the selected keyframes by a certain percentage. Select a blending operator, move the mouse left or right to adjust the factor, and click `LMB` to confirm (or `RMB` to cancel).

Several blending operators work based on “neighboring keyframes.” This means that they divide the selected keyframes into contiguous groups, then reference the unselected keyframes immediately before and after each group.

Breakdown

Reference

Menu:: *Key ▸ Blend ▸ Breakdown*

Sets the value of the selected keyframes to an interpolation of their neighbors.

Factor
: At -1, the keyframes are set to the value of the left neighbor.

At 1, they're set to the value of the right neighbor.

For other factors, they're set to an interpolation between the two neighbor values, with 0 being right in the middle.

Blend to Neighbor

Reference

Menu:: *Key ▸ Blend ▸ Blend to Neighbor*

Moves each selected keyframe towards the value of the left or right neighbor by a certain percentage.

Blend
: When negative, each keyframe moves *Blend* percent to the value of the left neighbor.

  When positive, they move to the right neighbor.

  When zero, they keep their original values.

Blend to Default Value

Reference

Menu:: *Key ▸ Blend ▸ Blend to Default Value*

Moves the selected keyframes towards the property's default value by a certain percentage.

Factor
: How much to change the keyframes' values, going from 0 (no change) to 1 (reset to the default value).

See also

The Reset to Default operator resets any property to its default value without the need of keyframing.

Ease

Reference

Menu:: *Key ▸ Blend ▸ Ease*

Makes the selected keyframes follow an S-curve. While the slider is visible (so after activating the operator but before confirming with `LMB`), you can press `Tab` to toggle which of the following settings to edit:

Curve Bend
: A negative value gives more weight to the left side, while a positive value gives more weight to the right. A value of 0 results in a balanced curve.

Sharpness
: A low value results in an almost straight diagonal line, while a high value results in a steep rise/drop in the curve.

Blend Offset

Reference

Menu:: *Key ▸ Blend ▸ Blend Offset*

Moves the selected keyframes up or down – all by the same amount – until the first/last one matches the left/right neighbor.

Offset Factor
: At -1, the first selected key gets aligned to its left neighbor.

  At 1, the last selected key gets aligned to its right neighbor.

At 0, nothing changes.

Blend to Ease

Reference

Menu:: *Key ▸ Blend ▸ Blend to Ease*

Blends the selected keys to either an “ease in” or an “ease out” curve.

Blend

At -1, the keys will follow an “ease in” curve, with small value changes in the beginning and large changes towards the end.

At 1, the keys will follow an “ease out” curve, with large value changes in the beginning and small changes towards the end.

At 0, nothing changes.

Match Slope

Reference

Menu:: *Key ▸ Blend ▸ Match Slope*

Blends the selected keys towards a straight line going through two keys just outside the current selection.

Factor

Negative values use the two keys to the left of the selection.

Positive values use the keys to the right.

At zero, nothing changes.

Push Pull

Reference

Menu:: *Key ▸ Blend ▸ Push Pull*

Moves the selected keys towards, or away from, the straight line going through the first and last selected key.

Factor
: At 0, the keys will lie on the straight line.

  At 1, they keep their original values.

  At 2, each key's value will be twice as far from the straight line as before.

Shear Keys

Reference

Menu:: *Key ▸ Blend ▸ Shear Keys*

Shears the selected keyframes – that is, changes their value by an amount that increases as they get further away in time from a reference keyframe. By default, this reference keyframe is the leftmost selected one, but you can instead use the rightmost one by pressing `D`.

Shear Factor
: How much to shear. Negative values move keyframes downwards, while positive ones move them up.

Direction
: Whether to use the leftmost or the rightmost selected keyframe as a reference.

Scale Average

Reference

Menu:: *Key ▸ Blend ▸ Scale Average*

Scales the selected keyframes vertically, using their average value as the pivot.

Factor
: At 0, the keyframes will all have the average value.

  At 1, they keep their original values.

  At 2, each keyframe's value will be twice as far from the average as before.

Scale from Neighbor

Reference

Menu:: *Key ▸ Blend ▸ Scale from Neighbor*

Scales the selected keyframes vertically, using a keyframe just outside the selection as the pivot. By default, this is the neighbor to the left of the selection, but you can instead use the right one by pressing `D`.

Factor
: The scale factor to apply.

Reference Key
: Whether to use the left or right neighbor as the pivot.

Time Offset

Reference

Menu:: *Key ▸ Blend ▸ Time Offset*

Shifts the values of the selected keyframes so that the resulting F-Curve appears to move in time. Works best with dense keyframes.

As the curve leaves the selected keyframes' time range on one end, it wraps back in on the other, offset vertically so that the ends connect and there is no jump.

Frame Offset
: By how many frames to shift the F-Curve. The slider is limited to the range -10 … 10, but you can type larger numbers too.

Smooth

Reference

Menu:: *Key ▸ Smooth*
Shortcut:: `Alt-S`

Smooth (Gaussian)

Reference

Menu:: *Key ▸ Smooth ▸ Smooth (Gaussian)*

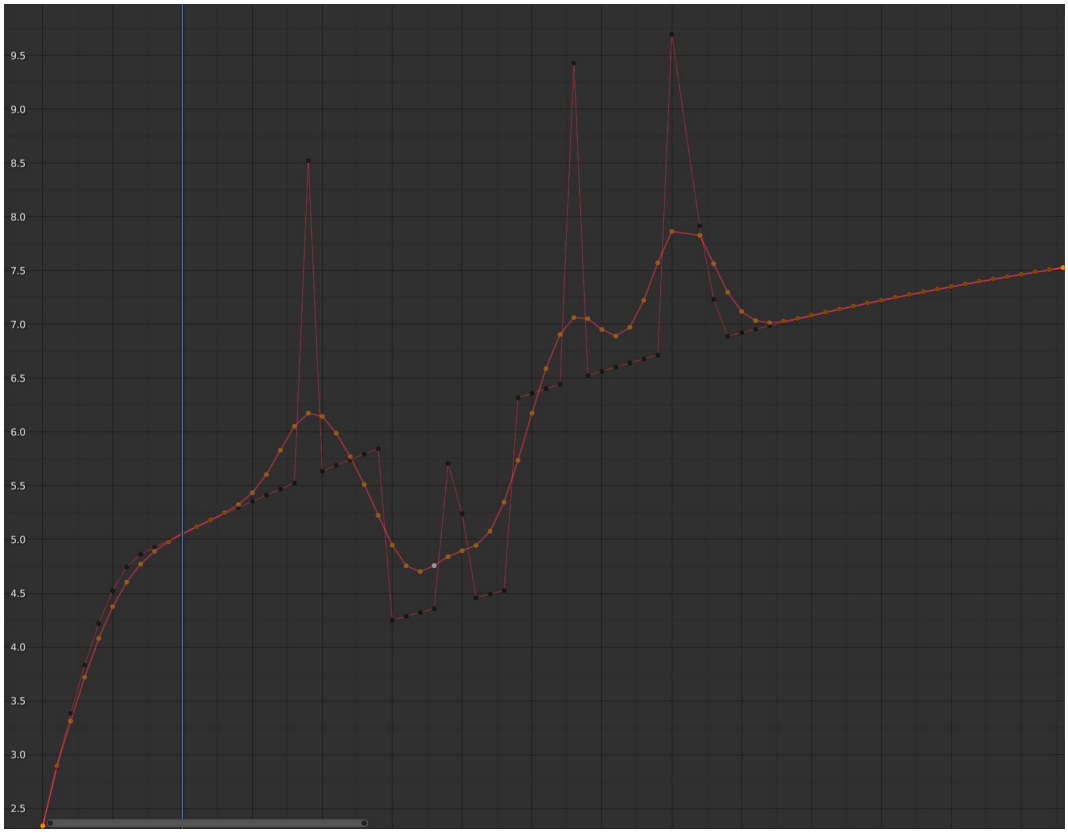
Smooths the selected keyframes using a Gaussian kernel. Click the menu item, move the mouse left or right to adjust the strength, and click `LMB` to confirm (or `RMB` to cancel).

Factor
: How strongly the smoothing should be applied.

Sigma
: The shape of the gaussian distribution. Lower values mean a sharper curve, giving keys that are close to each other more weight. A high value behaves like a simple average filter.

Filter Width

A wider filter looks at more keyframes, producing a smoother result. At a width of 1, the filter only looks at the keyframes to the immediate left and right for a weighted average.



F-Curve after applying the Gaussian Smooth with the original curve overlayed.

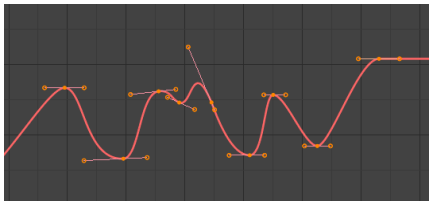
Smooth (Legacy)

Reference

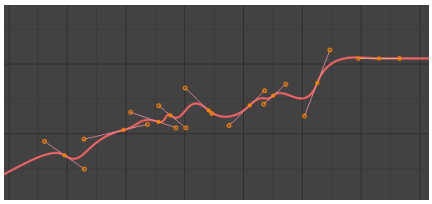
Menu:: *Key ▸ Smooth ▸ Smooth (Legacy)*
Shortcut:: `Alt-O`

There is also an option to smooth the selected curves, but beware: its algorithm seems to be to halve the distance between each keyframe and the

average linear value of the curve, which gives quite a strong smoothing! Note that the first and last keys seem to be never modified by this tool.



F-Curve before smoothing.



F-Curve after smoothing.

Butterworth Smooth

Reference

Menu:: *Key ▸ Smooth ▸ Butterworth Smooth*

Smooth the selected keyframes using a Butterworth filter. Click the menu item, move the mouse left or right to adjust the frequency, and click `LMB` to confirm (or `RMB` to cancel).

This filter is ideal for smoothing large amounts of data because it preserves the peaks of the animation. The downside is that it can introduce a ripple effect when the key values change rapidly.

Frequency Cutoff
: The lower the value, the smoother the curve. There is an implicit maximum at which the value no longer changes the curve, which is at half the sample rate. The sample rate in this case is the scene frame rate multiplied by the **Samples per Frame** of this operator.

Filter order
: Higher values mean the frequency cutoff is steeper.

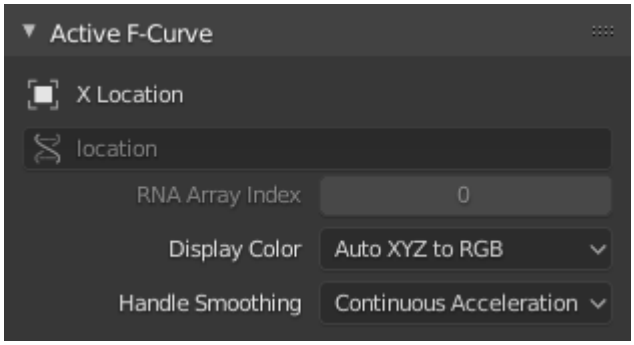
Samples per Frame

Properties

Active F-Curve

Reference

Panel:: *Sidebar region ▸ F-Curve ▸ Active F-Curve*



Active F-Curve panel.

This panel displays properties for the active F-Curve.

Channel Name
: The name of the property that's animated by the curve.

Data Path
: The programmatic path to the property.

RNA Array Index
: The index into the property, for multi-value properties. As an example, the Location property has three values (X, Y and Z), which means it can have three different curves with indices 0, 1 and 2.

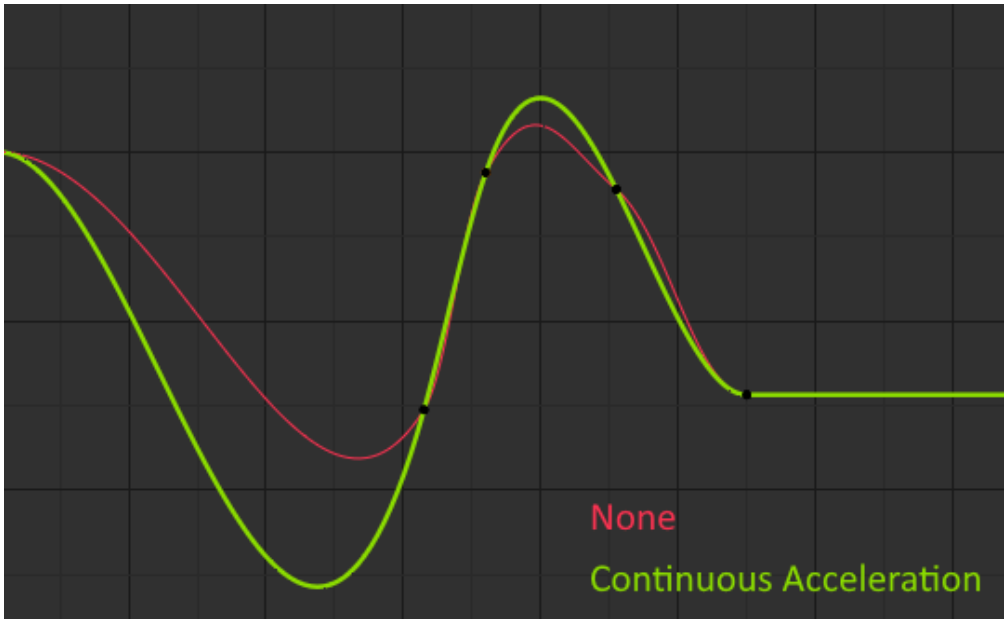
Display Color
: How to determine the color of the F-Curve in the Graph editor.

Auto Rainbow
: Assigns a unique color to each curve that uses this setting.

Auto XYZ to RGB
: Detects curves that animate an X/Y/Z coordinate and colors them red/green/blue accordingly.

User Defined
: Lets you choose the color yourself.

Handle Smoothing
: How to compute the Bézier handles when using the *Automatic* or *Auto Clamped* handle type.



Handle smoothing mode comparison.

None
: Only directly adjacent key values are considered when computing the handles.

  This older method is very simple and predictable, but it can only produce truly smooth curves in the most trivial cases. Notice that the red curve in the image above has a few kinks in the middle.

Continuous Acceleration

A system of equations is solved in order to avoid or minimize jumps in acceleration at every keyframe.

It produces much smoother curves out of the box, but necessarily means that any changes in the key values may affect interpolation over a significant stretch of the curve; although the amount of change decays exponentially with distance. This change propagation is stopped by any key with *Free*, *Aligned*, or *Vector* handles, as well as by extremes with *Auto Clamped* handles.

The mode also tends to overshoot and oscillate more with fully *Automatic* handles in some cases (see the image above). So it is recommended to use *Auto Clamped* by default, and only switch to *Automatic* handles in places where this is desired behavior. That effect can also be reduced by adding in-between keys.

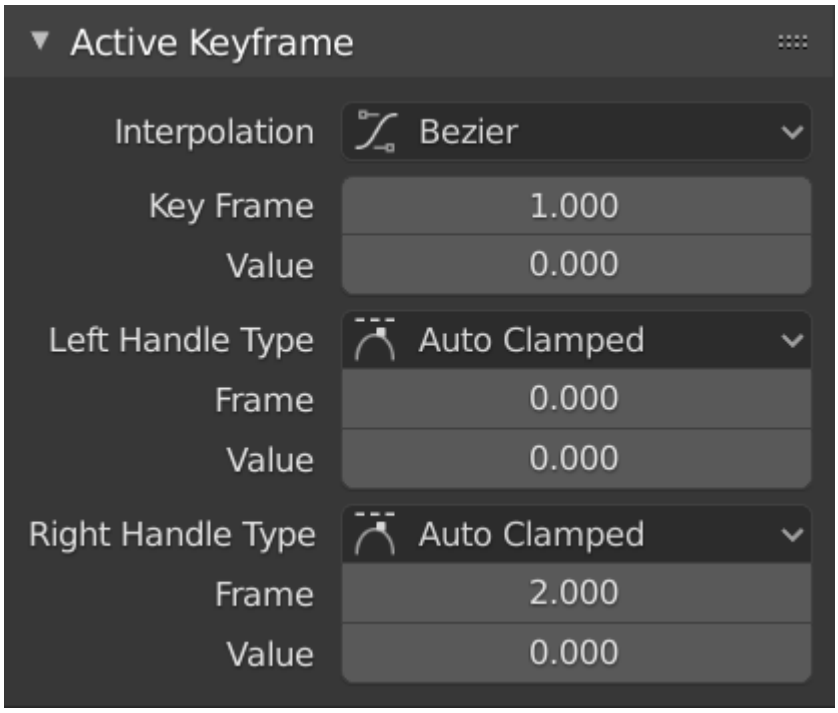
Tip

Considering the upsides and downsides of each mode, *Continuous Acceleration* should be better suited for limited animation, which uses a small number of interpolated keys with minimal manual polish. In case of highly polished high key rate animation, the benefits of smoothing may not outweigh the workflow disruption from more extensive change propagation.

Active Keyframe

Reference

Panel:: *Sidebar region ▸ F-Curve ▸ Active Keyframe*



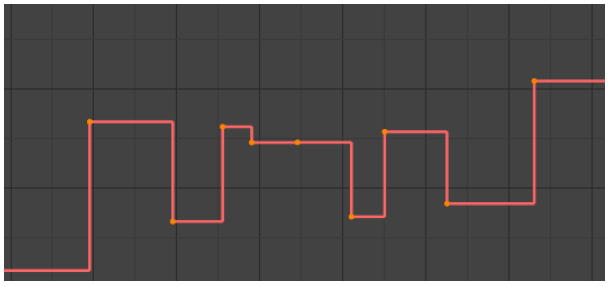
Active Keyframe panel.

Interpolation

The interpolation to use between the active keyframe and the next.

Interpolation

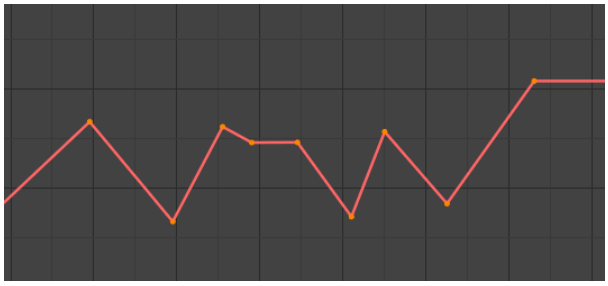
Constant::



Constant.

The curve holds the value until the next keyframe, producing a stair step effect with very abrupt changes. Normally only used during the initial “blocking” stage in pose-to-pose animation workflows.

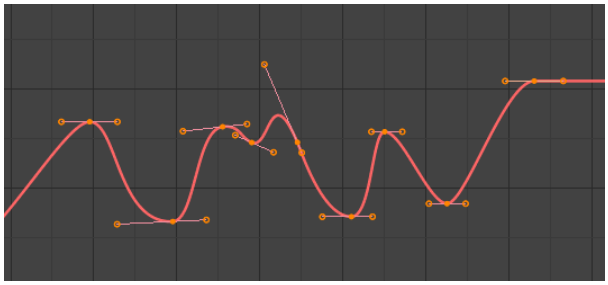
Linear::



Linear.

The curve goes from one keyframe to the next in a straight line, which prevents abrupt changes in value but not in speed. By creating two keyframes with Linear interpolation and extrapolation, you can easily make a straight line curve that goes on forever.

Bézier::



Bézier.

The default interpolation, which is smooth in both values and speed.

Note

F-Curves for properties that only accept discrete values will always have a stair step pattern, no matter which option you chose.

Easing (by strength)

A set of interpolations that specialize in accelerating or decelerating an value, “easing” it from a stationary into a moving state or vice versa.

See also

For more info and a few live demos, see easings.net and Robert Penner's Easing Functions.

Dynamic Effects

These additional easing types imitate (fake) physics-based effects like bouncing.

Back

With *Ease In*, the value first moves away from the target and then shoots towards it. With *Ease Out*, it goes towards the target, overshoots it, and then returns.

Back

The size and direction (i.e. above/below the curve) of the overshoot.

Bounce

Makes the value bounce a few times with exponential decay, like a tennis ball that was dropped on the floor.

Elastic

Makes the value overshoot the target, then rebound and undershoot it, then overshoot it again… with ever-decreasing intensity until it eventually settles.

Amplitude

How far the value initially overshoots its target.

Period

How much time there is between each oscillation.

Easing

This option only applies to the *Easing* and *Dynamic Effects* categories above.

Automatic Easing
: Automatically pick the most commonly used type: *Ease In* when using one of the *Easing* interpolations, and *Ease Out* when using one of the *Dynamic Effects*.

Ease In
: The value accelerates, moving slowly at the beginning of the curve segment and speeding up towards the end.

Ease Out
: The value decelerates, moving quickly at the beginning of the curve segment and slowing down towards the end.

Ease In Out
: The value moves slowly in the beginning, speeds up towards the middle, and slows down again towards the end.

Key Frame
: The frame of the active keyframe.

Value
: The value of the active keyframe.

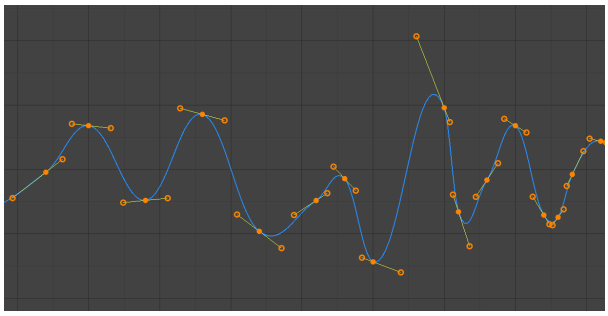
Left/Right Handle Type
: When the keyframe's interpolation is set to *Bézier*, the shape of the curve around it is influenced by its handles. Each handle has its own type which determines if (and how) Blender positions it automatically, and if not, how much freedom you have in positioning it manually.

  You can change the handles' types in multiple ways: using these dropdowns in the Sidebar; through the menu *Key* ▸ *Handle Type*; or by pressing V and selecting from the popup menu.

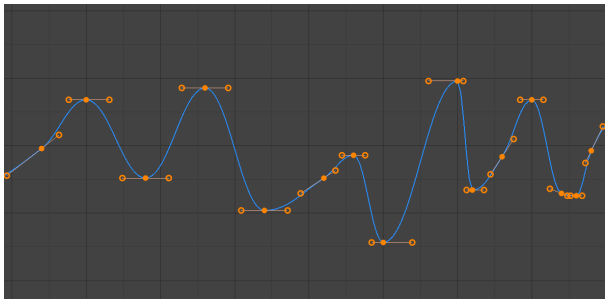
  If you want to change an automatic handle's position, just drag it; you don't need to manually change its type first.

  Automatic:: Automatic handles that produce smooth curves.

Auto Clamped::



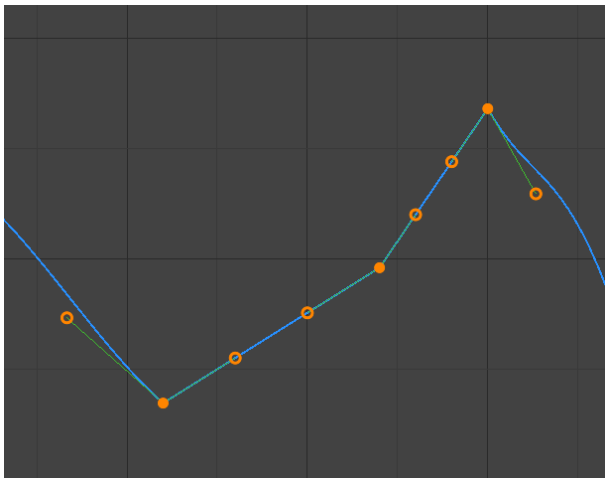
Auto handles.



Auto clamped handles.

Automatic handles clamped to prevent overshoots and changes in the curve direction between keyframes (S-shapes).

Vector::



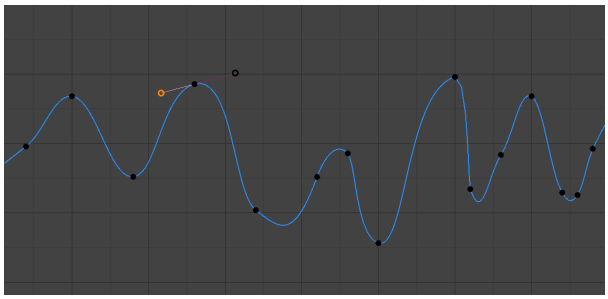
Vector handles.

Automatic handles that produce straight curve segments (like linear interpolation).

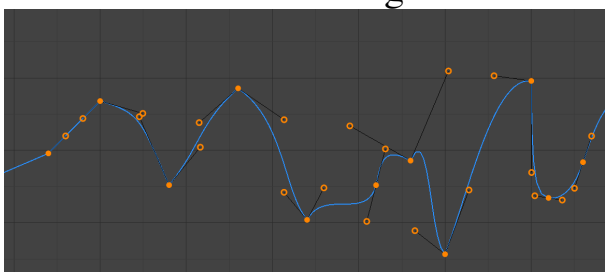
Aligned::

Manual handles, which are however locked together to always point in opposite directions. This results in a curve that is always smooth at the control point.

Free::



Aligned handles.



Free handles.

Manual handles that can be moved independently, and thus can result in a sharp change of direction.

Frame, Value

The frame and value for the left/right handle of the active keyframe.

F-Curve Modifiers

Reference

Panel:: *Sidebar region ▸ Modifiers*

F-Curve modifiers are similar to object modifiers, in that they add non-destructive effects that can be adjusted at any time and layered to create more complex effects.

Modifiers are evaluated from top to bottom. You can change their order by dragging the dots in their top right corner.

Interface

Name
: By default, modifiers are named after their function, but this can be changed.

Mute
: Click the checkbox in a modifier's header to disable it.

Delete
: Click the cross in a modifier's header to delete it.

Influence
: Lets you blend between the original curve and the modified one.

Restrict Frame Range
: Start/End
  : The frame on which the modifier's effect starts/ends.

  Blend In/Out
  : The number of frames, relative the start/end values above, it takes the modifier to fade in/out.

Adding a Modifier



Modifiers panel.

Modifiers can be managed on the *Modifiers* tab of the Sidebar. Select an F-Curve (in the channel region or by selecting one of its keyframes), then click the *Add Modifier* dropdown and choose the modifier to add.

Types of Modifiers

Generator Modifier

Creates a polynomial function. These are basic mathematical formulas that represent lines, parabolas, and other more complex curves, depending on the values used.

See also

The Wikipedia Page for more information on polynomials.

Mode
: Method used to represent the equation.

  Expanded Polynomial
  : Equation in the form $y = A + Bx^1 + Cx^2 + ... + Dx^n$.

  Factorized Polynomial
  : Equation in the form $y = (Ax + B)(Cx + D)$.

Additive
: Add the polynomial to the curve rather than replacing it.

Order
: The highest power of x for this polynomial.

Coefficient
: The constants A, B, C… in the equation.

Built-in Function Modifier

These are additional formulas, each with the same options to control their shape. Consult mathematics reference for more detailed information on each function:

Type
: The built-in function to use:

- Sine
- Cosine
- Tangent
- Square Root
- Natural Logarithm
- Normalized Sine: \(sin(x)/x\)

Additive
: Add the function to the curve rather than replacing it.

Amplitude
: Adjusts the Y scaling.

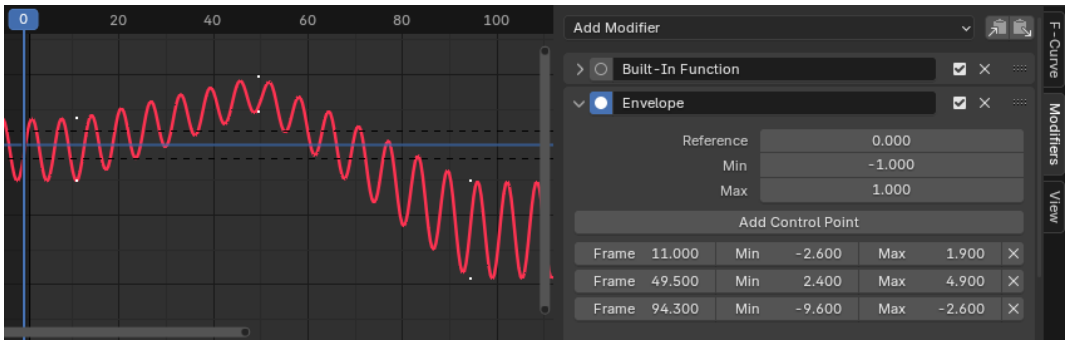
Phase Multiplier
: Adjusts the X scaling.

Phase Offset
: Adjusts the X offset.

Value Offset
: Adjusts the Y offset.

Envelope Modifier

Lets you reshape the curve. First, you define an envelope, which consists of two horizontal lines that more or less match the curve's lower and upper bounds. Then, you add control points, where each point can push, squeeze, and stretch the envelope (and the curve along with it) at a certain frame.



The Envelope modifier.

Reference
: The value which the envelope is centered around.

Min/Max
: The offset from the reference value to the envelope's initial lower/upper bound.

Add Control Point
: Adds a control point at the current frame.

Point
: Frame
: : The frame of the control point.

: Min/Max
: : The offset from the reference value to the envelope's adjusted lower/upper bound at this frame.

Cycles Modifier

Makes the curve repeat itself.

Note

The Cycles Modifier can only be the first modifier.

Before/After Mode
: No Cycles
  : Do not repeat the curve before/after the original.

: Repeat Motion
  : Repeats the curve, keeping the values of each copy the same.

: Repeat with Offset
  : Repeats the curve, offsetting each copy vertically so that its first keyframe matches the previous last keyframe.

: Repeat Mirrored
  : Repeats the curve, flipping every other copy horizontally.

Count
: The number of copies to create. A value of 0 means infinite.

Trivially Cyclic Curves

When the *Cycle Mode* for both ends is set to either *Repeat Motion* or *Repeat with Offset*, and no other options of the modifier are changed from their defaults, it defines a simple infinite cycle.

This special case receives some additional support from other areas of Blender:

- Automatic Bézier handle placement is aware of the cycle and adjusts to achieve a smooth transition.
- The Cycle-Aware Keying option can be enabled to take the cycle into account when inserting new keyframes.

Noise Modifier

Modifies the curve with a noise formula. This is useful for adding subtle or extreme randomness to animated movements, like camera shake.

Blend Type

Replace::	Adds noise in the range [-0.5, 0.5].
Add::	Adds noise in the range [0, 1].
Subtract::	Subtracts noise in the range [0, 1].
Multiply::	Multiplies by noise in the range [0, 1].

Scale
: Changes the horizontal scale of the noise. Higher values make for less dense oscillation.

Strength
: Changes the vertical scale of the noise.

Offset
: Offsets the noise in time.

Phase
: Adjusts the random seed of the noise.

Depth
: Adjusts how detailed the noise function is.

Limits Modifier

Limits the curve to specific time and value ranges.

Minimum, Maximum X
: Removes the original curve data to the left of the minimum frame and to the right of the maximum, replacing it by Constant extrapolation.

Minimum, Maximum Y
: Clamps the curve values, never letting them go below the minimum or above the maximum.

Stepped Interpolation Modifier

Gives the curve a stepped appearance by sampling it every N frames and making it hold its value after each sample. In a sense, this lowers the curve's frame rate by letting it change its value less frequently, producing choppy movement as a result.

Step Size
: The number of frames to hold each step.

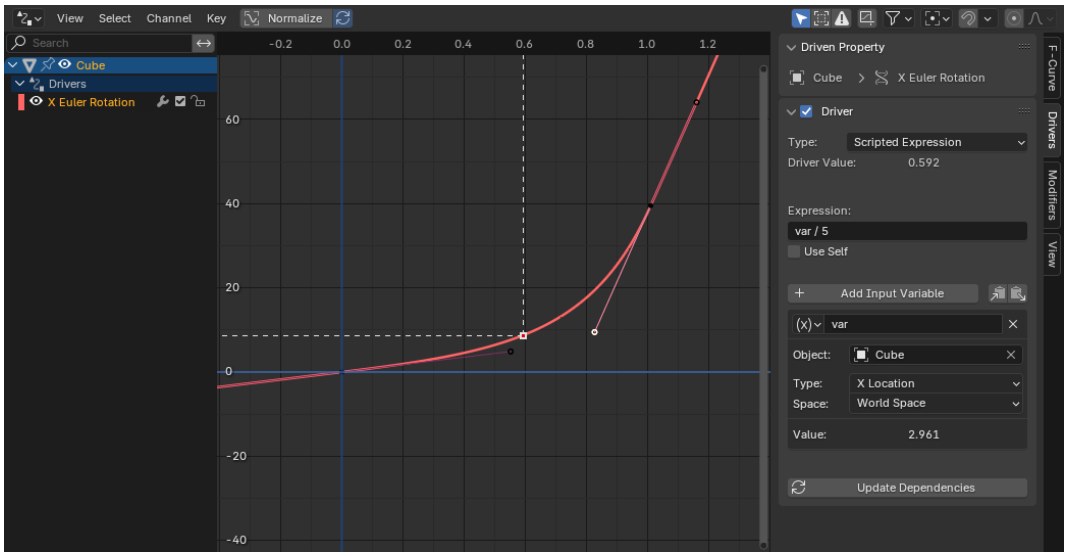
Offset
: The number of frames to offset the sample points.

Start Frame
: The frame where to start applying the effect.

End Frame
: The frame where to stop applying the effect.

Drivers Editor

This editor lets you set up Drivers, which calculate the value for a property based on other properties. In other words, they make a set of source properties "drive" the target property, and can thus serve as an alternative to animating the property by hand.



The Drivers Editor, showing how you might drive a cube's rotation based on its position.

The user interface is largely the same as that of the Graph Editor, with two important differences:

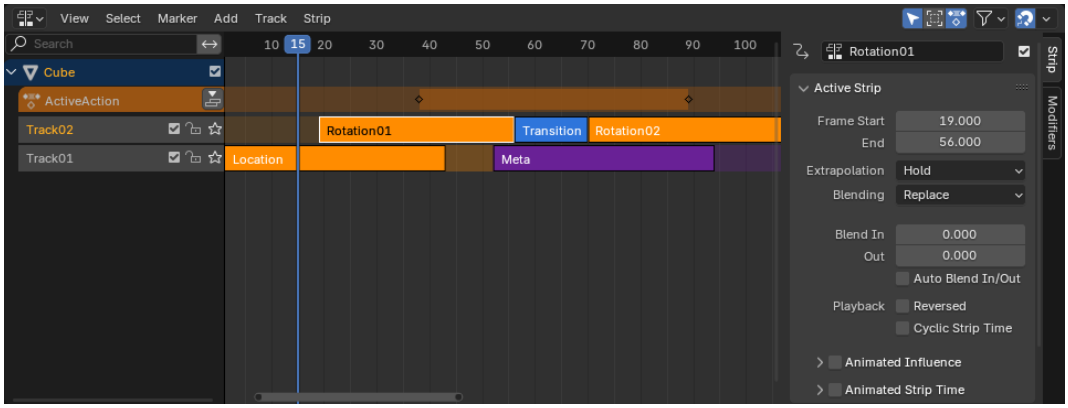
- The Sidebar has an additional Drivers tab. This is where the source properties are brought together to calculate an intermediate value for the target property.
- The curve doesn't represent the property's value over time, but a mapping from the above intermediate value (X axis) to the final value (Y axis).

Nonlinear Animation

- Introduction
  - Main Region
  - Header
- Tracks
  - Action Track
- Strips
  - Action Strips
  - Transition Strips
  - Sound Strips
  - Meta Strips
- Editing
  - Track
  - Strip
- Sidebar
  - Edited Action
  - Strip
  - Modifiers

Introduction

The NonLinear Animation editor, or NLA editor for short, lets you animate on a higher level. Instead of working with individual keyframes, it works with actions, which are named, reusable animation segments.



The NLA editor.

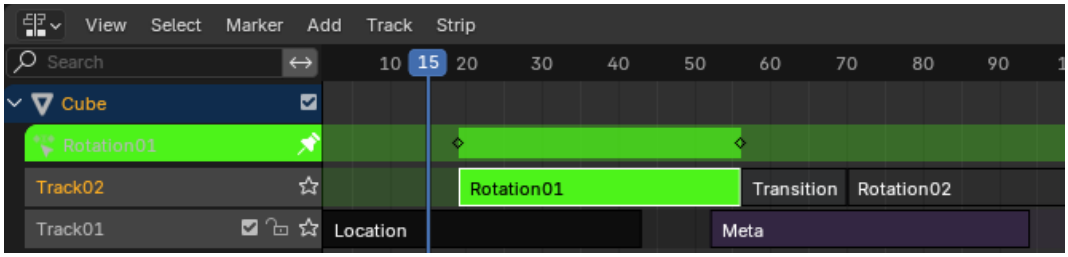
Main Region

The editor displays a stack of tracks which work like layers in an image editing program. Higher tracks take precedence over lower ones, although you can also choose to blend them.

Each track can contain any number of strips – typically Action Strips, which are instances of actions.

The top track hilighted in orange is special: this is the Action Track. Unlike the other tracks, it doesn't contain strips – instead, it contains the object's *active action*, which is where new keyframes are added to by default.

Editors like the Timeline and the Dope Sheet Editor normally only show the keyframes of this active action. If you want to edit another action, you can select it in the NLA editor and press `Tab` to enter Tweak Mode.



Tweaking an action. Notice that it's shown in both its original track and the Action Track. The active action is temporarily hidden.

Header

View Menu

Sidebar `N`
: Shows or hides the Sidebar Region.

Adjust Last Operation
: Displays a pop-up panel to alter properties of the last completed operation. See Adjust Last Operation.

Channels
: Shows or hides the Track Region.

Frame Selected `NumpadPeriod`
: Pans and zooms the view to focus on the selected strips.

Frame All `Home`
: Pans and zooms the view to show all strips.

Frame Scene/Preview Range
: Reset the horizontal view to the current scene frame range, taking the preview range into account if it is active.

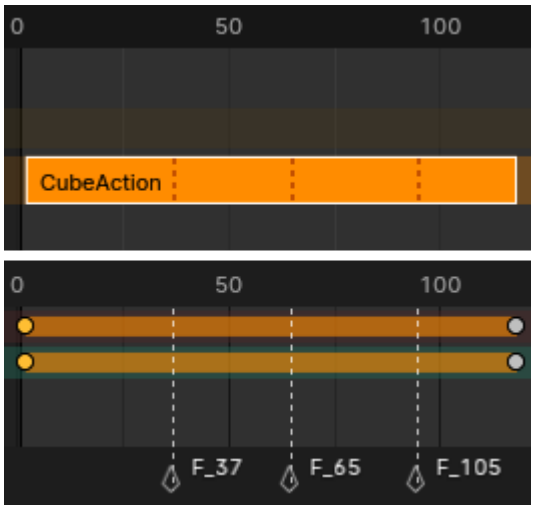
Go to Current Frame `Numpad0`
: Centers the view on the Playhead.

Realtime Updates
: Whether to update other views (such as the 3D Viewport) while you're moving strips around. If disabled, the other views only get updated once you finish the move.

Show Control F-Curves
: Shows a graph on top of each strip that uses Animated Influence.

Show Markers
: Shows the marker region (provided any markers have been defined). When disabled, the Marker Menu is also hidden and marker operators are not available in this editor.

Show Local Markers
: Shows action-local markers (which you can create in the Action Editor). This can be useful to align strips to each other.



Local markers shown in the NLA Editor (top) and the Action Editor (bottom).

Show Seconds `Ctrl-T`
: Shows timing in seconds instead of frames.

Sync Visible Range

Synchronizes the horizontal panning and scale of the editor with other time-based editors that also have this option enabled. That way, they always show the same section of time.

Set Preview Range `P`
: Lets you drag a box to define a time range for previewing. As long as this range is active, playback will be limited to it, letting you repeatedly view a segment of the animation without having to manually rewind each time.

  You can change the start or end frame using the corresponding button in the Timeline editor's Playback popover. Alternatively, you can simply run *Set Preview Range* again.

Clear Preview Range `Alt-P`
: Clears the preview range.

Set Preview Range to Selected `Ctrl-Alt-P`
: Applies a preview range that encompasses the selected strips.

Area
: Area controls. See the user interface documentation for more information.

Select Menu

All `A`
: Selects all strips.

None `Alt-A`
: Deselects all strips.

Invert `Ctrl-I`
: Inverts the current selection.

Box Select `B`
: Lets you drag a box and selects the strips that are partially or completely inside it.

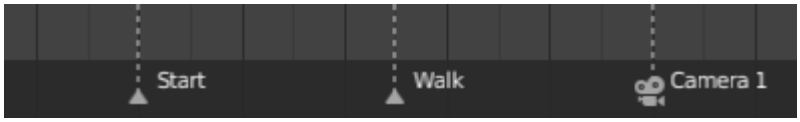
Box Select (Axis Range) `Alt-B`
: Lets you drag a box and selects the strips that overlap the corresponding time range, even if they're above or below the box.

Before Current Frame `[`
: Selects all the strips that start before (or on) the current frame.

After Current Frame `]`
: Selects all the strips that end after (or on) the current frame.

Marker Menu

Markers are used to denote frames with key points or significant events within an animation. Like with most animation editors, they're shown at the bottom.



Markers in animation editor.

For descriptions of the different marker tools, see Editing Markers.

Add Menu

Action `Shift-A`
: Adds a strip referencing an action to the active track.

Transition `Shift-T`
: Adds a transition strip between the two selected action strips.

Sound `Shift-K`

Adds a strip that controls when the Speaker object plays its sound clip.

Selected Objects
: Makes the selected objects appear in the NLA Editor without adding an action or track to them.

See Strips for details on the various strip types.

Track Menu

Contains tools for working with NLA tracks. See Editing Tracks for details.

Strip Menu

Contains tools for working with NLA strips. See Editing Strips for details.

Filters

Only Show Selected
: Only shows tracks belonging to objects that are selected.

Show Hidden
: Shows tracks from objects that are hidden.

Include Missing NLA
: Shows the Action Track even if there is no action in it.

Search
: Filters the track list by a search term.

Filtering Collection
: Select a collection to only show tracks from objects in that collection.

Filter by Type
: Filter tracks by target type.

Sort Data-Blocks

Sorts data-blocks alphabetically to make them easier to find.

If your playback speed suffers because of this (should only really be an issue when working with lots of objects), you can turn it off.

Snap

The toggle button enables/disables automatic strip snapping. The dropdown button shows a popover with the following options:

Snap To

Type of element to snap to.

Frame::	Snap to full frames.
Second::	Snap to seconds.
Nearest Marker::	Snap to the nearest Marker.

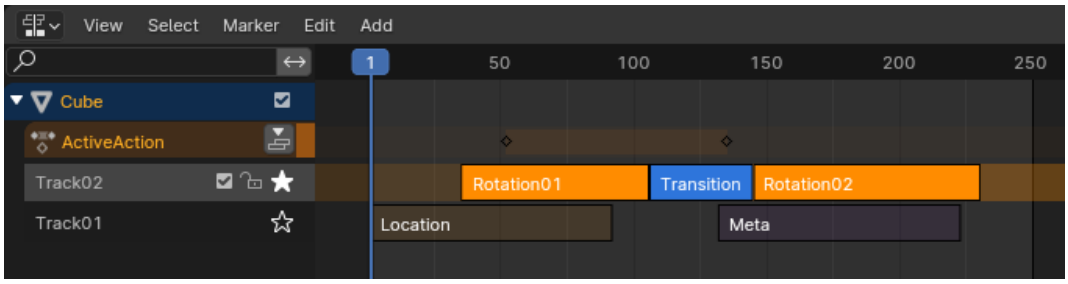
Absolute Time Snap

When disabled, strips will move in increments of *Snap To*. For example, if you selected *Second* and have a strip that currently starts on 0:06+5, dragging it to the right will snap it to 0:07+5. Its time increases by a second, and its subsecond offset of 5 frames remains the same.

When enabled, strips will snap to multiples of *Snap To*. Taking the above example, the strip would snap to 0:07+0, removing the subsecond offset.

Tracks

A track plays one or more actions in sequence. You can create multiple tracks to play several actions at the same time.



NLA Tracks and Strips.

The track region has the following properties:

Disable NLA stack (checkbox in blue object header)
: When unchecked, mutes all the tracks except the Action Track.

Track name
: Double-click to change. (Not possible for the Action Track, as this one simply displays the name of the action.)

Mute (checkbox in gray track header)
: When unchecked, the track stops contributing to the animation. Its strips receive a dotted outline to indicate this. Note that you can also mute individual strips.

Lock (padlock icon)
: Prevents changes from being made to this track. This is useful, for example, if you want to move the strips in all the tracks except for a few.

Solo (star icon)

Mutes all other tracks, including the Action Track, so that only this track contributes to the animation. This is useful for inspecting the track without any distractions from others.

Action Track

The topmost track with the orange header holds the action that's being edited. Normally this is the object's active action, but if you select a strip and press `Tab` to enter Tweak Mode, you can temporarily make that one editable instead – in the Action Editor or the Graph Editor, for example.

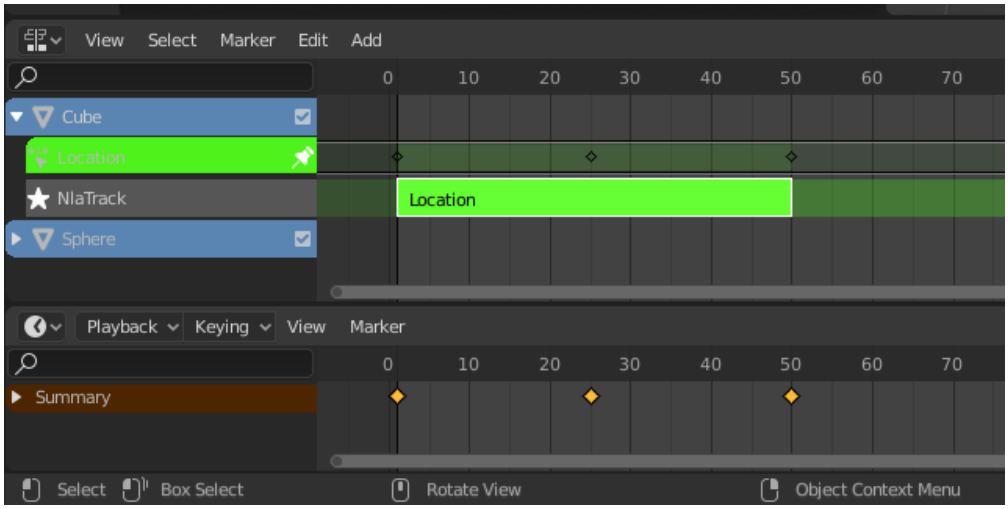
The Action Track has one of the following buttons:

Push Down Action
: Not available in Tweak Mode. Creates a new track below the Action Track and moves the active action into it as a strip, leaving the Action Track empty. (If you create a keyframe after this, Blender will automatically create a new active action to hold it.)

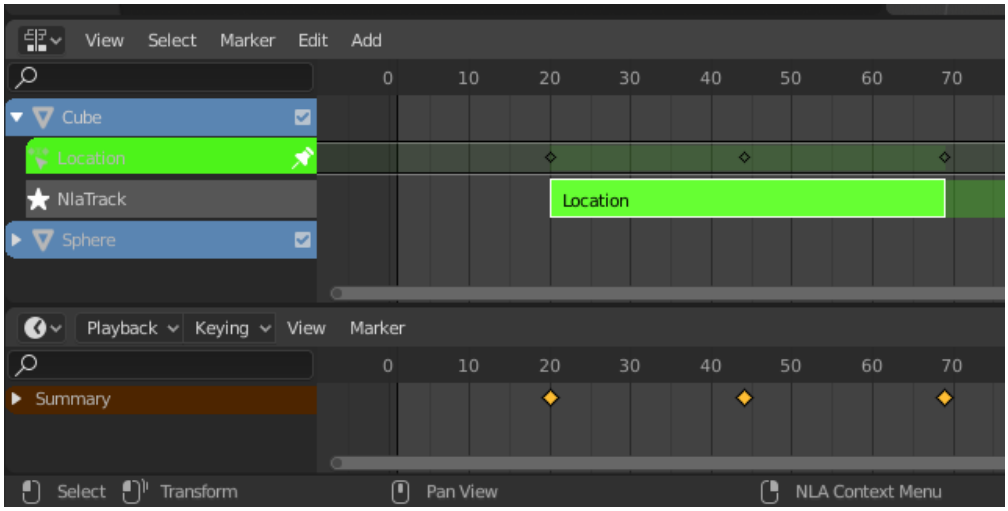


Push Down Action button.

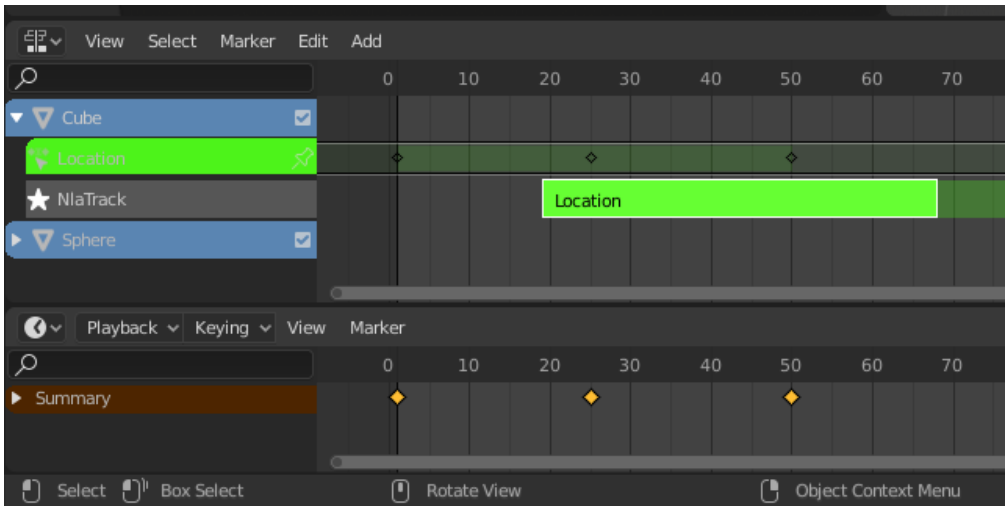
Pin
: Only available in Tweak Mode. When unchecked, the action's keyframes are shown at their original time points, rather than their new time points resulting from the strip being moved and scaled.



Strip at its original time point.



Strip moved. Notice that the keyframes are now shown to start at frame 20, which is also how the animation will behave. Within the action, however, they still start at frame 1.



After unchecking the Pin icon, the keyframes are shown at their original time points.

Strips

A strip tells the animation when something happens and for how long. There are a few different types which are described below.

Action Strips

An action strip plays the keyframes inside an action. You can create one using *Add ▸ Action*. Another way is to click Push Down Action in the NLA's Action Track – this will create a strip based on the object's active action.

Multiple strips can reference the same action, so that you can potentially change multiple parts of the animation by editing a single set of keyframes.

A strip can be shorter than its underlying action, be it through cropping, speeding up, or both. It can also be longer than its underlying action, be it through extending, slowing down, or both. See the Sidebar for details.

Transition Strips

A transition strip interpolates between two neighboring action strips. Select them and click *Add ▸ Transition*.



Transition Strip.

Sound Strips

These strips control when a Speaker plays its sound clip.

Meta Strips

A meta strip groups other strips together, letting you move, scale, and copy them as one combined unit.

Editing

- Track
  - Add
  - Add Above Selected
  - Delete Tracks
  - Move
  - Remove Empty Animation Data
- Strip
  - Transform
  - Snap
  - Split
  - Duplicate
  - Linked Duplicate
  - Delete
  - Make Meta
  - Remove Meta
  - Toggle Muting
  - Bake Action
  - Apply Scale
  - Clear Scale
  - Sync Action Length
  - Make Single User
  - Start Editing Stashed Action
  - Start Tweaking Strips Actions (Full Stack)
  - Start Tweaking Strips Actions (Lower Stack)

Track

Add

Reference

Editor::	Nonlinear Animation
Menu::	*Track ▸ Delete*

Adds a new track below the Action Track.

Add Above Selected

Reference

Editor::	Nonlinear Animation
Menu::	*Track ▸ Add Above Selected*

Adds a new track above each selected one.

Delete Tracks

Reference

Editor::	Nonlinear Animation
Menu::	*Track ▸ Delete*
Shortcut::	`Delete`, `X`

Deletes the selected tracks and the strips they contain.

When using the keyboard shortcuts, make sure the mouse cursor is hovering over the track region, as otherwise, Blender will only delete the selected

strips.

Move

Reference

Editor::	Nonlinear Animation
Menu::	*Track ▸ Move*

To Top `Shift-PageUp`
: Moves the selected tracks to the top.

Up `PageUp`
: Moves the selected tracks up by one.

Down `PageDown`
: Moves the selected tracks down by one.

To Bottom `Shift-PageDown`
: Moves the selected tracks to the bottom.

When using the keyboard shortcuts, make sure the mouse cursor is hovering over the track region, as otherwise, Blender will move the selected strips to different tracks (rather than moving the tracks).

Remove Empty Animation Data

Reference

Editor::	Nonlinear Animation
Menu::	*Track ▸ Remove Empty Animation Data*

Removes objects that don't have drivers, NLA tracks, or an active action from the NLA editor to reduce clutter. This essentially does the opposite of *Add ▸ Selected Objects*.

Strip

Transform

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Transform*

Move `G`
: Move the selected strips in time or to a different track.

Extend `E`
: Lets you quickly move the selected strips that are on a certain side of the Playhead. This is handy if you need to, say, move all the strips after a certain time point to the right to make space for new ones.

  To use this operator, first select some or all strips and place your mouse cursor to the left or right of the Playhead. Then, press `E`, move the mouse to move (only) the strips on that side of the Playhead, and press `LMB` to confirm (or `RMB` to cancel).

  If a strip straddles the Playhead, only its starting/ending point will be moved (again depending on the position of the mouse cursor).

Scale `S`
: Scales the selected strips, using the Playhead as the pivot point.

Swap

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Transform ▸ Swap*

Shortcut:: `Alt-F`

Swap the order of the selected strips in their track.

Move Up

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Transform ▸ Move Up*
Shortcut:: `PageUp`

Move selected strips up a track if there is room.

Move Down

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Transform ▸ Move Down*
Shortcut:: `PageDown`

Move selected strips down a track if there is room.

Snap

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Snap*

Selection to Current Frame
: Move the start of the selected strips to the current frame.

Selection to Nearest Frame

Move the start of the selected strips to the nearest full frame.

Selection to Nearest Second
: Move the start of the selected strips to the nearest second.

Selection to Nearest Marker
: Move the start of the selected strips to the nearest marker.

Split

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Split*
Shortcut::	`Y`

Split the selected strips in two at the current frame.

Duplicate

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Duplicate*
Shortcut::	`Alt-D`

Creates copies of the selected strips, duplicating any actions they reference. Editing the keyframes in a copied strip therefore doesn't affect the original.

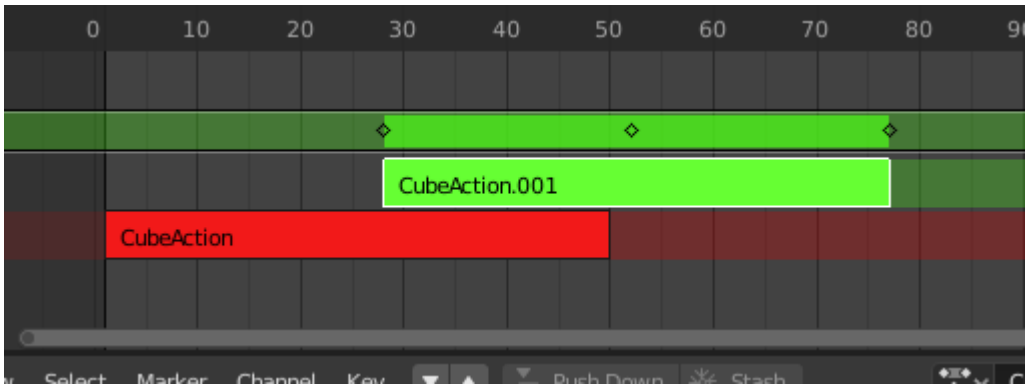
Linked Duplicate

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Linked Duplicate*

Shortcut:: `Shift-D`

Creates copies of the selected strips, reusing any actions they reference. Editing the keyframes in a copied strip therefore also affects the original (and vice versa). Blender warns you about this by hilighting the other strip in red.



Linked duplicated strip being edited.

Delete

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Delete*
Shortcut:: `Delete`, `X`

Deletes the selected NLA-Strips.

Make Meta

Reference

Editor:: Nonlinear Animation
Menu:: *Add ▸ Make Meta*
Shortcut:: `Ctrl-G`

Groups the selected NLA-strips into a meta strip.



Select two or more strips.



Combine them into a meta strip.

Remove Meta

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Remove Meta*
Shortcut::	`Ctrl-Alt-G`

Ungroups the selected meta strips, replacing them by their contents.

Toggle Muting

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Toggle Muting*
Shortcut::	`H`

Mutes or unmutes the selected strips. Muted strips have a dotted border and don’t influence the animation.

Bake Action

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Bake Action*

Reference

Editor:: 3D Viewport
Mode:: Object and Pose Modes
Menu:: *Header ▸ Object ▸ Animation ▸ Bake Action…*

The final motion of objects and bones depends not only on the keyframed animation, but also on F-Curve modifiers, drivers, and constraints. The *Bake Action* operator computes this final motion and creates a corresponding keyframe on every scene frame.

This can be useful for adding deviation to a cyclic action like a Walk Cycle, or to create a keyframe animation from drivers or constraints.

Start Frame
: Start frame for baking.

End Frame
: End frame for baking.

Frame Step
: Number of frames to skip forward while baking each frame.

Only Selected Bones
: Only key selected bones (Pose baking only).

Visual Keying
: Keyframe from the final transformations (with constraints applied).

Clear Constraints
: Remove all constraints from keyed object/bones, and do ‘visual’ keying.

Clear Parents
: Bake animation onto the object then clear parents (objects only).

Overwrite Current Action
: Bake animation into the current action instead of creating a new one (useful for baking only part of bones in an armature).

Clean Curves
: After baking curves, remove redundant keys.

Bake Data
: Which data's transformations to bake.

Pose:: Bake bone transformations.
Object:: Bake object transformations.

Channels
: Which channels to bake.

Location:: Bake location channels.
Rotation:: Bake rotation channels.
Scale:: Bake scale channels.
B-Bone:: Bake B-Bone channels.
Custom Properties:: Bake custom properties.

Apply Scale

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Apply Scale*
Shortcut:: `Ctrl-A`

Applies the scale of the selected strips to their referenced actions.

Clear Scale

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Clear Scale*
Shortcut:: `Alt-S`

Resets the scale of the selected strips.

Sync Action Length

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Sync Action Length*

Resets the strip's length to that of its underlying action, ensuring that it (only) plays from the action's first keyframe to its last.

See also

The Sync Length Now button in the Sidebar, which does the same thing.

Make Single User

Reference

Editor:: Nonlinear Animation
Menu:: *Strip ▸ Make Single User*
Shortcut:: `U`

Duplicates actions where necessary so that each selected strip has its own action that's not used by any others. This way, you can edit the keyframes in the selected strips knowing that you won't affect any other part of the animation.

Note

This does not recursively go inside meta strips.

Start Editing Stashed Action

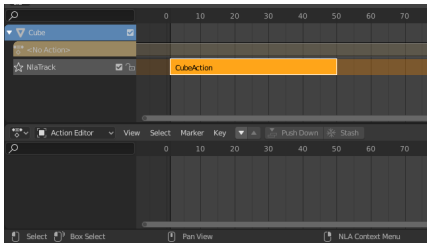
Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Start Editing Stashed Action*
Shortcut::	`Shift-Tab`

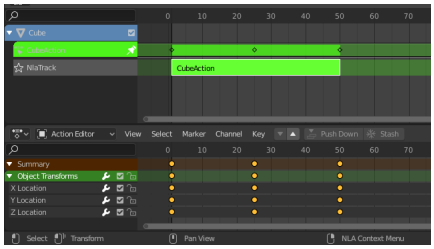
Enters Tweak Mode for the selected strip's action, making its keyframes available for editing in e.g. the Graph Editor. In addition, marks the strip's track as *Solo*, muting all the other tracks – this way, they no longer influence the animation and you can focus exclusively on the action you're editing.

While the menu item refers to stashed (muted) actions, this only reflects the typical use case. It works on unmuted actions as well.

When you're done editing, click *Strip ▸ Stop Editing Stashed Action* or press `Shift-Tab` again.



Strip in NLA mode.



Strip in Tweak mode.

Start Tweaking Strips Actions (Full Stack)

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Start Tweaking Strips Actions (Full Stack)*
Shortcut::	`Tab`

Enters Tweak Mode for the selected strip's action, making its keyframes available for editing. Leaves all the other tracks enabled so that you can still see their effects while making changes.

When you're done, click *Strip ▸ Stop Tweaking Strips Actions* or press `Tab` again.

Note

For transitions above the tweaked strip, keyframe remapping will fail for channel values that are affected by the transition. A workaround is to tweak the active strip without evaluating the upper NLA stack.

Start Tweaking Strips Actions (Lower Stack)

Reference

Editor::	Nonlinear Animation
Menu::	*Strip ▸ Start Tweaking Strips Actions (Lower Stack)*

Enters Tweak Mode for the selected strip's action, making its keyframes available for editing. Mutes any tracks above the current one so that they don't influence the animation while making changes.

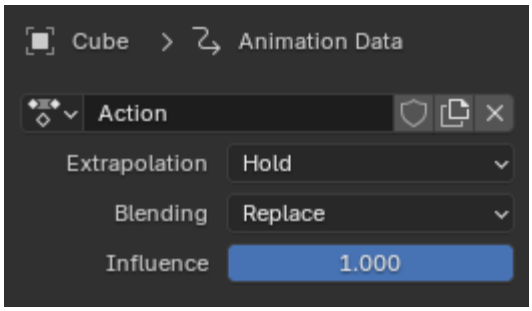
When you're done, click *Strip ▸ Stop Tweaking Strips Actions* or press `Tab`.

Sidebar

Edited Action

Reference

Panel:: *Sidebar ▸ Edited Action*



Edited Action panel.

Contains settings for the object's active action. Only visible if the Action Track is selected.

Action
: A data-block menu where you can see, change, and clear the object's active action. See also the Action Editor's Action.

Extrapolation
: Determines whether the action will influence the frames before/after its boundaries once it has been pushed down into a strip. (As long as it's still the active action, *Hold* is used regardless of the choice.)

  Hold
  : The property values at the action's first keyframe also apply to the earlier frames (if the strip is the first in the track). The values at its last keyframe also apply to the later frames (up to the next strip).

Hold Forward

The property values at the action's last keyframe also apply to the later frames (up to the next strip).

Nothing

The animated properties return to their default values outside of the strip boundaries.

Blending

How to combine the action's property values with those of the tracks below.

Replace

Overwrites the values produced by the lower tracks. If *Influence* is less than 1, a linear interpolation between the previous and new values is used instead.

Multiply, Subtract, Add

Blends the action's values with those of the lower tracks using a simple calculation. If *Influence* is less than 1, a linear interpolation between the previous values and these calculated values is used.

$result = mix(previous, previous (+-×) value, influence)$

Combine

Depending on the type of each property, one of the following methods is automatically chosen:

Axis/Angle Rotation

$result = previous + value × influence$

This results in averaging the axis and adding the amount of rotation.

Quaternion Rotation

Quaternion math is applied to all four channels of the property at once:

$result = {previous} \times {value} ^ {influence}$

Proportional (Scale)
$result = previous \times (value / default) ^ {influence}$

Others
$result = previous + (value - default) \times {influence}$

Note

Since this blending mode uses quaternion multiplication to calculate the Quaternion Rotation properties, it always drives all four channels during playback, and *Insert Single Keyframe* is forced to insert all four keys. Other types of channels can still be keyed individually.

Influence
How much the action contributes to the result of the NLA stack.

Strip

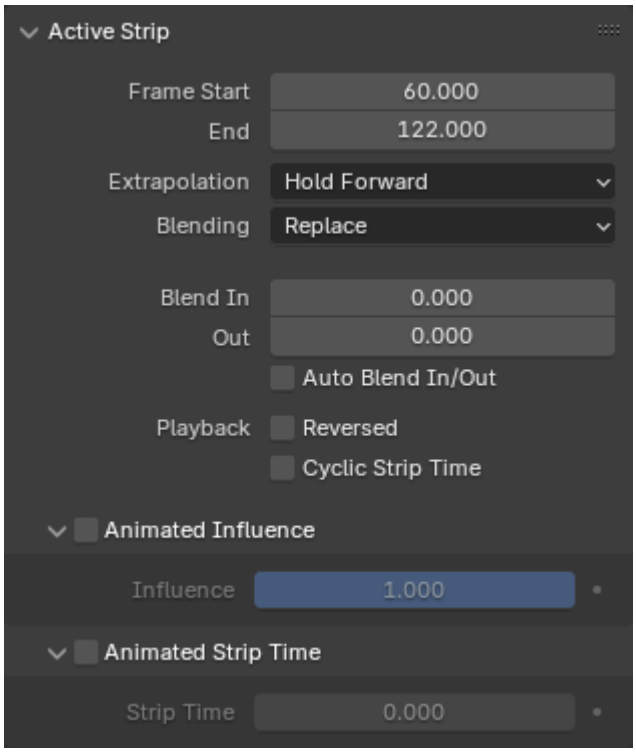
Name
Name of the strip.

Mute (checkbox)
When unchecked, the strip will no longer contribute to the animation. It's shown with a dotted outline to indicate this.

Active Strip

Reference

Panel:: *Sidebar ▸ Strip ▸ Active Strip*



Active Strip panel.

Contains common strip properties.

Frame Start
: The frame where the strip begins. Changing this will move the strip while keeping its duration constant.

Frame End
: The frame where the strip ends. Changing this will also change the *Action Clip Frame End*, thereby cropping or extending the action. If you instead want to speed it up or slow it down, scale it by using *Strip* ‣ *Transform* ‣ *Scale* or adjusting the *Playback Scale* setting.

Extrapolation
: See Extrapolation.

Blending
: See Blending.

Blend In, Out

How many frames it takes for the strip's influence to ramp up at the start and wind down at the end.

Auto Blend In/Out
Calculates *Blend In/Out* automatically by looking at the strips in the track above or below that overlap the current strip in time.



Two strips with Auto Blend enabled.

Playback
Reversed
Makes the strip play backwards.

Cyclic Strip Time
Whether to wrap the *Animated Strip Time* back to the start if it exceeds the Action Clip Frame End.

Animated Influence

Lets you manually specify, and animate, how strongly the strip affects the animation. This is an alternative to using the *(Auto) Blend In/Out* settings above.

To create an influence keyframe, first type a value, then either click *Insert Keyframe* in its context menu or press `I` while hovering over it. You can see the keyframes in e.g. the Graph Editor.

Animated Strip Time

Lets you manually specify, and animate, the frame at which the underlying action is sampled.

Note

Although the setting is called *Strip Time*, its value is a frame number inside the action, not inside the strip. If you have an action going from

frame 1 to frame 50 that's referenced by a strip going from frame 101 to 150, you'd set the *Strip Time* to 1 to see the first keyframe, not 101.

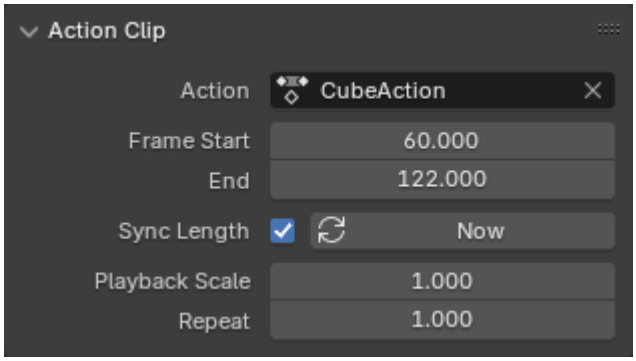
In combination with *Cyclic Strip Time*, this lets you play the action's keyframes multiple times in a single strip. As an example, say that the action's keyframes are between frames 1 and 50. If you animate the strip time to instead go from 1 to 100, the keyframes will play twice (at twice the speed).

In practice, however, it's easier to use the *Repeat* setting described below.

Action Clip

Reference

Panel:: *Sidebar region ▸ Strip ▸ Action Clip*



Action Clip panel.

Contains properties specific to Action strips.

Action
: The action referenced by the strip.

Frame Start, End
: How much of the action to use. By adjusting these, you can crop or extend the action (and the strip, as its *Frame End* will change

accordingly). If you extend the action, F-Curve Extrapolation will kick in.

One case where these settings can be useful is in cyclic animation where the first and last keyframes of the action have the same value (meaning this value applies for two frames when the animation restarts). By reducing the *Frame End*, you can exclude the last keyframe and have the value apply for only one frame instead.

Sync Length
: Automatically sets *Frame Start*/*End* to the action's first/last keyframe when exiting Tweak Mode.

Now
: Sets *Frame Start*/*End* to the action's first/last keyframe.

Playback Scale
: Makes the animation play more quickly (scale < 1) or slowly (scale > 1) than the original action.

Repeat
: Makes the action play multiple times.

Action

Reference

Panel:: *Sidebar region* ▸ *Strip* ▸ *Action*

See Action Properties.

Modifiers

Reference

Panel:: *Sidebar region* ▸ *Modifiers*

Strip modifiers let you make non-destructive changes to all the curves inside the strip's action.

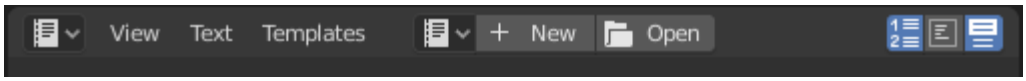
See F-Curve Modifiers.

Text Editor

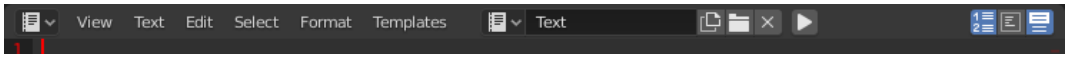
This editor can be used to write Python scripts, Open Shading Language scripts, or just plaintext notes. To open it, you can switch to the *Scripting* workspace or press `Shift-F11` to replace the current editor.

Header

The newly opened Text editor is empty, with a very simple header. More options become available when a text file is created or opened.



Text header.



Text header with a text loaded.

Editor Type
: The standard editor selection button.

Menus
: Editor's menus.

Resolve Conflict
: Resolves modified file conflicts when an external text file is updated from another program.

  Reload from Disk
  : Opens the file from drive again, overriding any local changes.

  Make Text Internal
  : Converts the external text data-block into an internal one.

Ignore
: Hides the warning message until the external text file is modified externally again.

Text
: A data-block menu to select a text or to create a new one. After that the header will change.

Run Script (play icon)
: Executes the text as a Python script `Alt-P`. See Template Menu.

Show
: Toggles for line numbers, word wrapping, and syntax highlighting.

Script Node Update (refresh icon)
: When an OSL-file is opened, this updates the Shader Script node with new options and sockets from the script.

View Menu

Sidebar `Ctrl-T`
: Show or hide the Sidebar.

Line Numbers
: Displays the text file's line numbers on the left of the Main View.

Word Wrap
: Wraps words that don't fit into the horizontal space by pushing them to a new "pseudo line".

Syntax Highlight
: Colors special words, in the Main View, that are used in the Python programming language.

Highlight Line
: Emphasizes the active line by altering the color of the background.

Zoom In/Out
: Increase/decrease the font size of text in the main view.

Navigation

Top `Ctrl-Home`
: Moves the view and cursor to the start of the text file.

Bottom `Ctrl-End`
: Moves the view and cursor to the end of the text file.

Line Begin `Home`
: Moves the cursor to the start of the current line.

Line End `End`
: Moves the cursor to the end of the current line.

Previous Line `Up`
: Moves the cursor to the same position in the line above the current line.

Next Line `Down`
: Moves the cursor to the same position in the line below the current line.

Previous Word `Ctrl-Left`
: Moves the cursor to the beginning of the previous word. If the cursor is in the middle of a word, the cursor is moved to the beginning of the current word.

Next Word `Ctrl-Right`
: Moves the cursor to the end of the next word. If the cursor is in the middle of a word, the cursor is moved to the end of the current word.

Text Menu

New `Alt-N`

Creates a new text Data Block.

Open `Alt-O`.
: Loads an external text file that is selected via the File Browser.

Reload `Alt-R`
: Reopens (reloads) the current buffer (all non-saved modifications are lost).

Edit Externally
: Edit text file in external text editor. The external editor can be configured in the User Preferences.

Save `Alt-S`
: Saves an already open file.

Save As `Shift-Ctrl-Alt-S`.
: Saves text as a new text file. A File Browser is opened to select the directory to save the file along with giving the file a name and extension.

Register
: Runs the text data-block as a Python script on loading the blend-file. Read more about the registration of Python modules in API documentation.

Live Edit
: Runs the Python script each time you make a change.

Run Script `Alt-P`
: Executes the text as a Python script. See Running Scripts for more information.

Edit Menu

Undo/Redo
: See Undo & Redo.

Cut `Ctrl-X`
: Cuts out the marked text into the clipboard.

Copy `Ctrl-C`
: Copies the marked text into the clipboard.

Paste `Ctrl-V`
: Pastes the text from the clipboard at the cursor location in the Text editor.

Duplicate Line `Ctrl-D`
: Duplicates the current line.

Move Line(s) Up `Shift-Ctrl-Up`
: Swaps the current/selected line(s) with the above.

Move Line(s) Down `Shift-Ctrl-Down`
: Swaps the current/selected line(s) with the below.

Find & Replace `Ctrl-F`
: Shows the *Find & Replace* panel in the Sidebar.

Find & Set Selection `Ctrl-G`
: Finds the next instance of the selected text.

Jump To `Ctrl-J`
: Shows a pop-up, which lets you select a line number where to move the cursor to.

Text Auto Complete `Tab`
: Shows a selectable list of words already used in the text.

Text to 3D Object
: Converts the text file to a Text Object either as *One Object* or *One Object Per Line*.

Select Menu

All `Ctrl-A`
: Selects the entire text file.

Line `Shift-Ctrl-A`
: Selects the entire current line.

Word double-click `LMB`
: Selects the entire current word.

Top `Shift-Ctrl-Home`
: Selects everything above the cursor.

Bottom `Shift-Ctrl-End`
: Selects everything below the cursor.

Line Begin `Shift-Home`
: Selects everything between the beginning of the current line and the cursor.

Line End `Shift-End`
: Selects everything between the cursor and the end of the current line.

Previous Line `Shift-Up`
: Selects everything between the cursor and the position of the cursor one line above.

Next Line `Shift-Down`
: Selects everything between the cursor and the position of the cursor one line below.

Previous Word `Shift-Ctrl-Left`
: Selects everything between the cursor and the beginning of the previous word. If the cursor is in the middle of a word, select everything to the beginning of the current word.

Next Word `Shift-Ctrl-Right`
: Selects everything between the cursor and the end of the next word. If the cursor is in the middle of a word, select everything to the end of the

current word.

Format Menu

Indent `Tab`
: Inserts a tab character at the cursor.

Unindent `Shift-Tab`.
: Unindents the selection.

Toggle Comments `Ctrl-Slash`.
: Toggles whether the selected line(s) are a Python comment. If no lines are selected, the current line is toggled.

Convert Whitespace
: Converts indentation characters *To Spaces* or *To Tabs*.

Template Menu

Contains a number of templates for both Python and Open Shading Language scripts.

Main View

Typing on the keyboard produces text in the text buffer.

As usual, pressing, dragging and releasing `LMB` selects text. Pressing `RMB` opens the context menu.

Tip

The Text editor is also handy when you want to share your blend-file with others: you can leave a note that explains how the file is structured. Be sure to keep the editor visible when saving so they'll see it!

Sidebar

Find & Replace

Find Text `Ctrl-F`
: Searches for instances of a text that occur after the cursor. Using the eyedropper icon will search for the currently selected text and sets the selection to the match. *Find Next* searches for the next instance of the text.

Replace Text `Ctrl-H`
: Searches for the text specified in *Find Text* and replaces it with the new text. Using the eyedropper icon will set the currently selected text as the replace text. *Replace* searches for the next match and replaces it. *Replace All* searches for the match and replaces all occurrences of the match with the new text.

Case
: Search is sensitive to uppercase and lowercase letters.

Wrap
: Search again from the start of the file when reaching the end.

All
: Search in all text data-blocks instead of only the active one.

Properties

Margin
: Shows a vertical margin line to help keep text lines at a reasonable length. The position of this margin line is specified by *Margin Column*.

Font Size `Ctrl-WheelUp`
: The size of the font used to display text.

Tab Width
: The number of character spaces to display tab characters with.

Indentation
: Whether to use *Tabs* or *Spaces* for indentation.

Footer

The Text editor footer displays whether the text is saved internally or externally and if there are unsaved changes to an external file. For external files, this region also displays the file path to the text file.

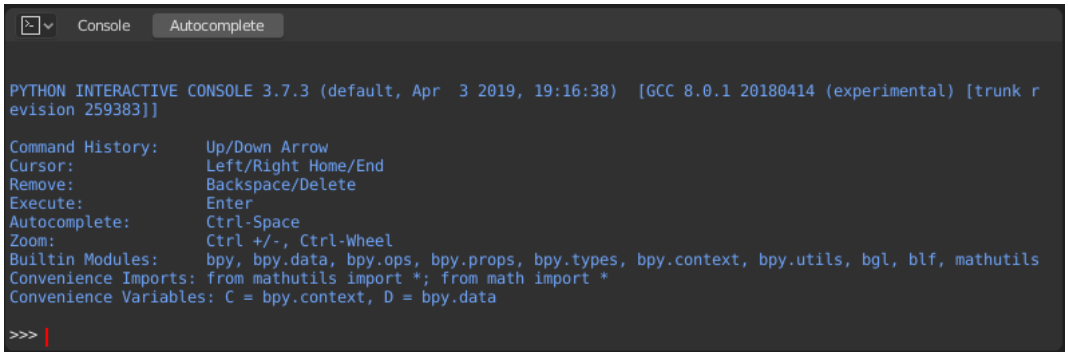
Usage

Running Scripts

The most notable keystroke is `Alt-P` which executes the current text as a Python script. You can access not just the standard Python modules, but also a whole bunch of Blender-specific ones; see Scripting & Extending Blender.

Python Console

The Python Console offers a quick way to test code snippets and explore Blender's API. It executes whatever you type on its >>> prompt and has command history and auto-complete.



Python Console.

Interface

Header Menus

View Menu

Zoom In / Zoom Out
: Increases/decreases the font size.

Move to Previous Word `Ctrl-Left`
: Moves the cursor to the beginning of the previous word. If the cursor is in the middle of a word, the cursor is moved to the beginning of the current word.

Move to Next Word `Ctrl-Right`
: Moves the cursor to the end of the next word. If the cursor is in the middle of a word, the cursor is moved to the end of the current word.

Move to Line Begin `Home`
: Moves the cursor to the start of the current line.

  `Shift-Home`: Selects all text between the cursor and the start of the current line.

Move to Line End `End`
: Moves the cursor to the end of the current line.

  `Shift-End`: Selects all text between the cursor and the end of the current line.

Console Menu

Clear All
: Refreshes the console, giving the view a fresh start. Note that command history is not cleared.

Clear Line `Shift-Return`.
: Removes everything from the prompt line.

Delete Previous Word `Ctrl-Backspace`
: Deletes everything between the cursor and the beginning of the previous word (separated by periods). If the cursor is in the middle of a word, deletes everything to the beginning of the current word.

Delete Next Word `Ctrl-Delete`
: Deletes everything between the cursor and the end of the next word. If the cursor is in the middle of a word, deletes everything to the end of the current word.

Copy as Script `Shift-Ctrl-C`
: Copies the full history buffer to the clipboard. This can be pasted into a text file to be used as a Python script.

Cut `Ctrl-X`
: Copies the selected text into the clipboard and deletes it.

Copy `Ctrl-C`
: Copies the selected text into the clipboard.

Paste `Ctrl-V`
: Pastes into the command line.

Indent `Tab`
: Inserts a tab character at the cursor.

Unindent `Shift-Tab`
: Unindents the selection.

Backward in History `Up`
: Changes the current command to the previous one from the command history.

Forward in History `Down`
: Changes the current command to the next one from the command history.

Autocomplete `Tab`
: See Auto Completion.

Main View

Key Bindings

- `LMB` – Moves the cursor along the input line.
- `Left` / `Right` – Moves the cursor by one character.
- `Ctrl-Left` / `Ctrl-Right` – Moves the cursor by one word.
- `Shift-Left` / `Shift-Right` – Selects characters to the left/right.
- `Shift-Ctrl-Left` / `Shift-Ctrl-Right` – Selects words to the left/right.
- `Ctrl-A` Selects all text and text history.
- `Backspace` / `Delete` – Erase characters.
- `Ctrl-Backspace` / `Ctrl-Delete` – Erase words.
- `Return` – Execute command.
- `Shift-Return` – Add to command history without executing.

Usage

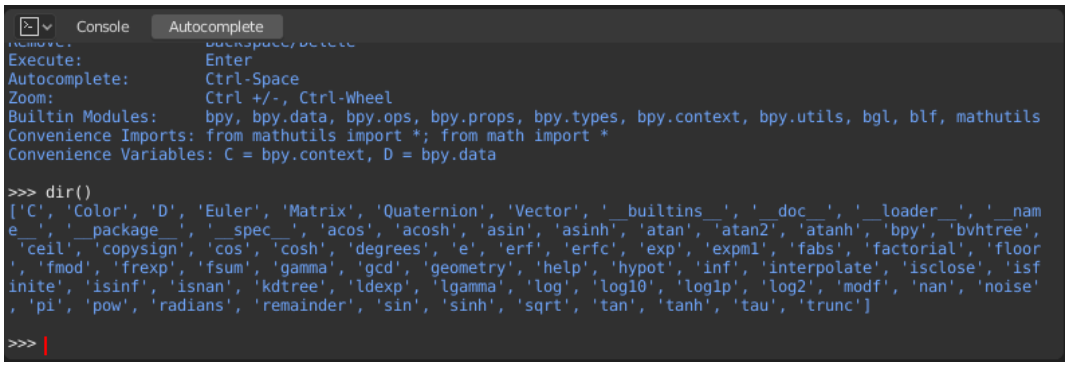
Aliases

Some variables and modules are available for convenience:

- `C`: Quick access to `bpy.context`.
- `D`: Quick access to `bpy.data`.
- `bpy`: Top level Blender Python API module.

First Look at the Console Environment

To see the list of global functions and variables, type `dir()` and press `Return` to execute it.



Auto Completion

The Console can preview the available members of a module or variable. As an example, type `bpy.` and press `Tab`:



The submodules are listed in green. Attributes and methods will be listed in the same way, with methods being indicated by a trailing `(`.

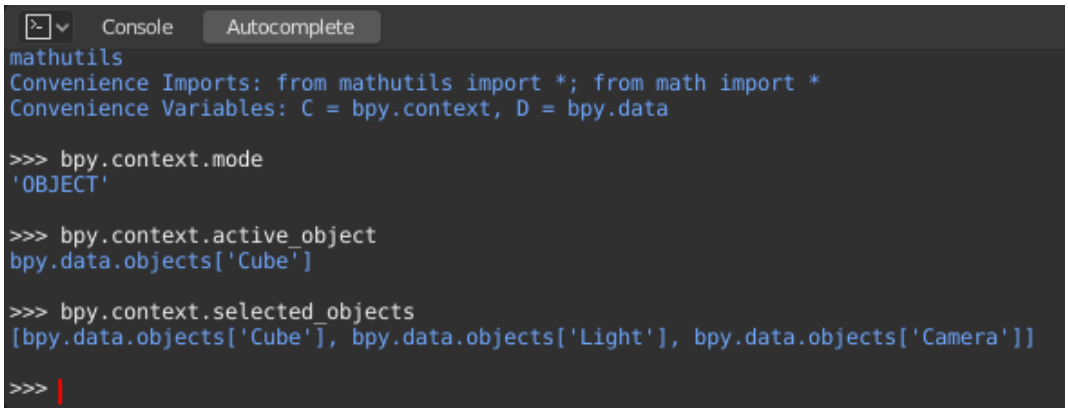
Examples

bpy.context

This module gives you access to the current scene, the currently selected objects, the current object mode, and so on.

Note

For the commands below to show the proper output, make sure you have selected object(s) in the 3D Viewport.



Get the current 3D Viewport mode (Object, Edit, Sculpt, etc.):

```
bpy.context.mode
```

Get the active object:

```
bpy.context.object
bpy.context.active_object
```

Change the active object's X coordinate to 1:

```
bpy.context.object.location.x = 1
```

Move the active object by 0.5 along the X axis:

```
bpy.context.object.location.x += 0.5
```

Change all three location coordinates in one go:

```
bpy.context.object.location = (1, 2, 3)
```

Change only the X and Y coordinates:

```
bpy.context.object.location.xy = (1, 2)
```

Get the selected objects:

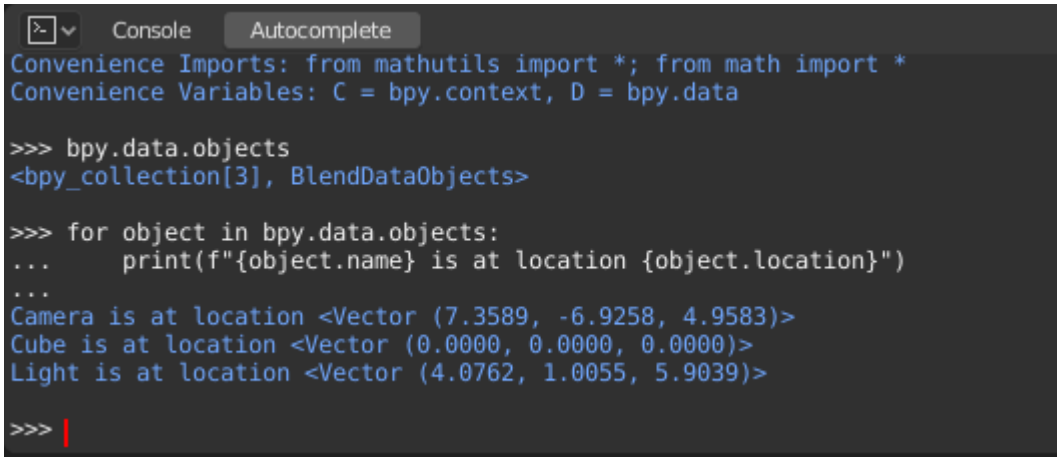
```
bpy.context.selected_objects
```

Get the selected objects excluding the active one:

```
[obj for obj in bpy.context.selected_objects if obj !=
bpy.context.object]
```

bpy.data

Gives you access to all the data in the blend-file, regardless of whether it's currently active or selected.

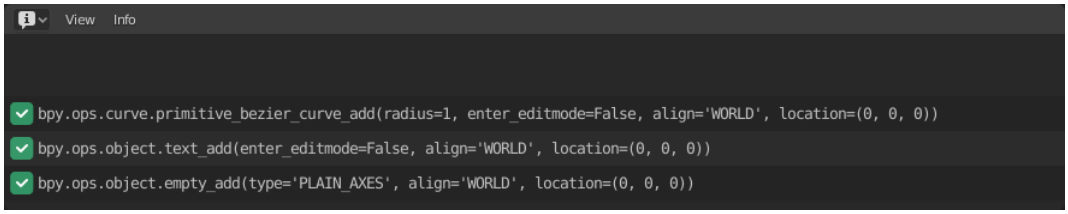


bpy.ops

“Operators” are actions that are normally triggered from a button or menu item but can also be called programmatically. See the bpy.ops API documentation for a list of all operators.

Info Editor

The Info editor logs the executed operators as well as errors, warnings, and informational messages. You can select an entry by clicking it, optionally holding `Shift` to add it to the existing selection.



Info Editor.

Interface

View Menu

Area
: Area controls. See the user interface documentation for more information.

Info Menu

Select All `A`
: Selects all entries.

Deselect All `Alt-A`
: Deselects all entries.

Invert Selection `Ctrl-I`
: Selects non-selected entries and deselects selected ones.

Toggle Selection

Selects all entries if there are currently no selected ones, and deselects them otherwise.

Box Select `B`
: Lets you drag a box and adds the entries that overlap it to the selection.

Delete `X`, `Delete`
: Removes the selected entries from the log.

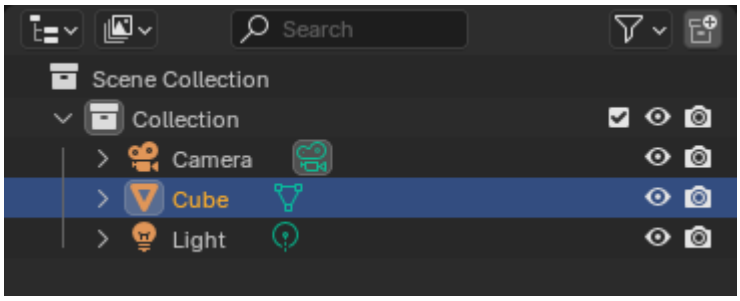
Copy `Ctrl-C`
: Copies the selected entries to the clipboard.

Outliner

- Introduction
- Interface
- Selecting
- Editing
- Usage

Introduction

The *Outliner* shows the content of the blend-file in a tree. You can use it to:

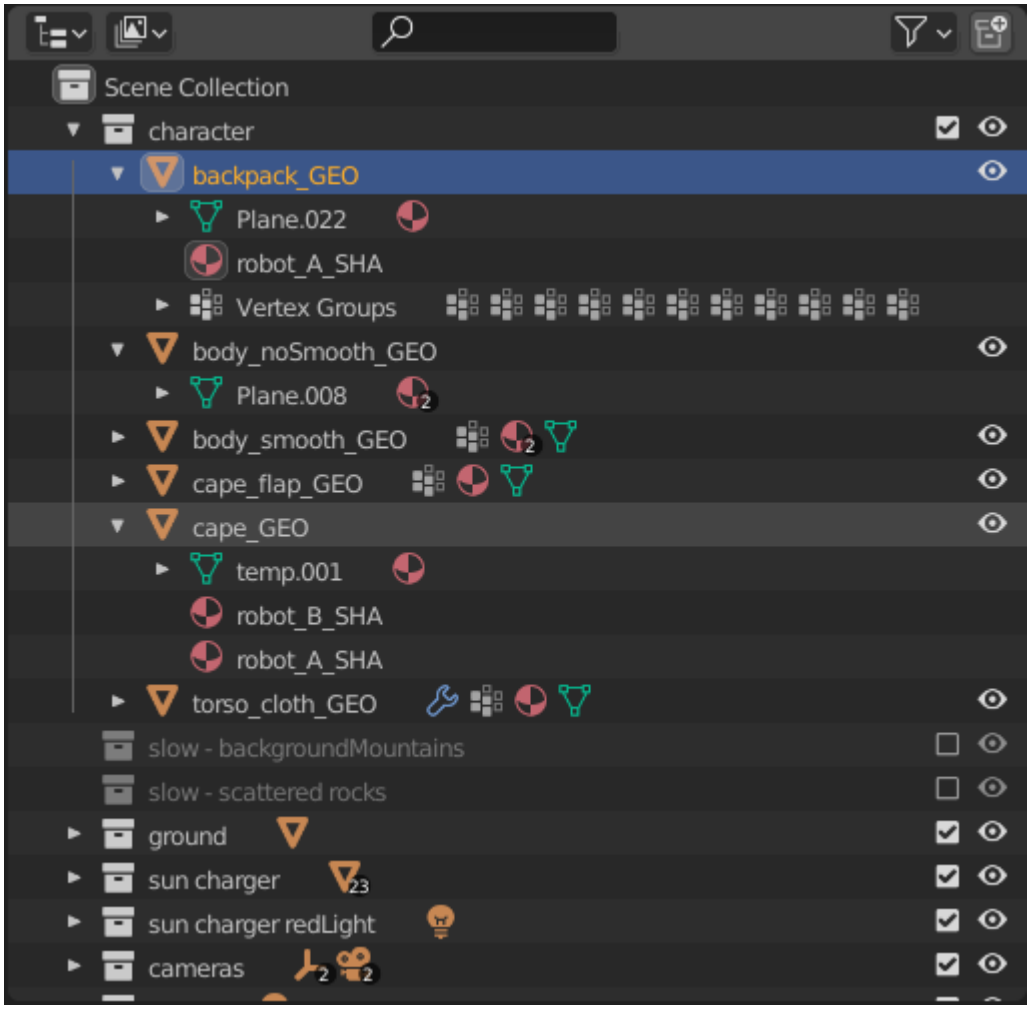


The Outliner editor.

- Get an overview of the data in the scene.
- Select and deselect objects.
- Make objects unselectable or invisible in the 3D Viewport.
- Exclude objects from rendering.
- Duplicate objects.
- Delete objects.
- Manage parent/child relationships and collections.

Items with an arrow on the left can be expanded. Click it with `LMB` to expand a single item, drag `LMB` to expand multiple items, or click `Shift-LMB` to expand an item recursively.

Example



The Outliner with different kinds of data.

Interface

Header

Display Mode

This header dropdown lets you choose what the Outliner should show.

Scenes:: Shows the view layers, collections, and objects across all scenes.

View Layer:: Shows the collections and objects in the current view layer of the current scene.

Video Sequencer:: Shows the images and videos that are used in the Video Sequencer.

Blender File:: Lists all data in the current blend-file. On the right side of the list, a shield icon shows the number of users – clicking it adds or removes a fake user.

Data API:: Lists every data-block in the file along with any properties that it might have.

Library Overrides::

Shows the library overrides. Separated further into two view modes:

Properties:: Shows the data-blocks that have overridden properties in a list grouped by type. You can expand each data-block to see and change these properties.

Hierarchies::

Shows the overridden data-blocks in a tree that visualizes their hierarchy. This includes parent data-blocks that were overridden implicitly. For example, if you created an override for a material, this tree would show the hierarchy object > mesh > material.
This view also shows a column of icons on the right that let you toggle whether each override is editable.

Unused Data:: Lists the data-blocks that are unused or only have a fake user. You can add/remove a fake user by clicking the shield icon on the right.
Unused data-blocks are automatically deleted when saving and reloading the file. You can also delete them manually by clicking *Purge* in the header.

Search

The textbox lets you filter the tree by typing a substring. You can focus it using `Ctrl-F` or clear it using `Alt-F`.

Filter

The funnel icon in the header offers further control over what is displayed in the editor. Depending on the *Display Mode*, some options are not available.

Restriction Toggles
: Set which Restriction Toggles should be visible.

Sort Alphabetically
: Sort the entries alphabetically.

Sync Selection
: Whether to synchronize the Outliner selection to and from the 3D Viewport and Video Sequencer editors.

Show Mode Column
: Show the column for toggling the object interaction mode.

Search

Exact Match
: Only show the items whose name fully matches the search text rather than only containing it as a substring.

Case Sensitive
: Take lower/upper case into account when comparing the search text to the item names.

Filter

All View Layers
: Show all the view layers in the scene instead of only the active one. Combined with disabling the *Objects* filter, this gives a compact overview of all the collections in relation to the view layers.

Collections
: Show the collections in the scene hierarchy. Only the collections themselves are hidden when this option is disabled; the objects within them remain visible.

Objects
: Show the objects in the scene hierarchy. Disabling this gives you an overview of just the collections.

Object State
: List the objects based on their state or restrictions. The results can be inverted using the *Invert* toggle button.

  All:: Show all objects.

Visible:: Only show the objects that are visible in the 3D Viewport. This takes both the *Hide in Viewports* and *Disable in Viewports* settings into account; see Restriction Toggles.

Selected:: Only show the object(s) that are currently selected in the 3D Viewport.

Active:: Only show the active object (typically the one that was selected last).

Selectable:: Only show the objects that can be selected in the 3D Viewport; see Restriction Toggles.

Object Contents
: List relevant materials, modifiers, mesh data and so on as children of each object.

Object Children
: Show child objects as child nodes in the Outliner tree. When disabled, child objects are shown as sibling nodes instead (unless they're in a different collection than their parent, in which case they're not shown in the parent's collection at all).

Meshes/Lights/…
: Lets you filter out objects by type.

System Overrides
: Shows the data-block properties that are defined/controlled automatically (e.g. to make data-blocks point to overridden data instead of the original). Only available in the *Library Overrides* Display Mode.

Miscellaneous

Some options in the header will only show if compatible with the active Display Mode.

New Collection *View Layer*
: Add a new collection inside the selected one.

Filter by Type *Blender File Unused Data*

Restrict the type of the data-blocks shown in the Outliner.

Keying Sets (Data API) *Data API*
: Add/Remove the selected property to/from the active Keying Set.

Drivers
: Add/Remove Drivers to the selected item.

Purge (Orphan Data) *Unused Data*
: Opens a dialog to remove unused data-blocks from both the current blend-file or any Linked Data (cannot be undone).

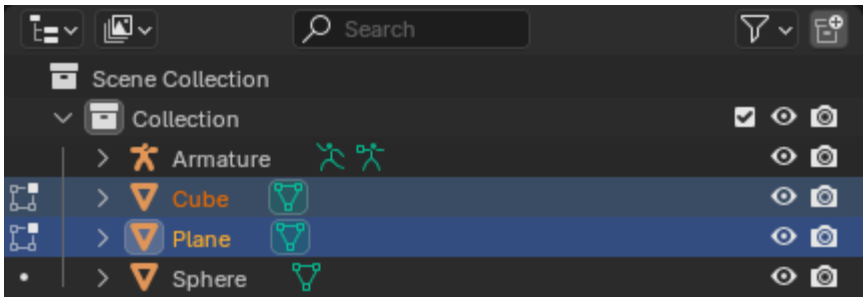
  Local Data-Blocks
  : Removes unused data-blocks from the current blend-file.

  Linked Data-Blocks
  : Removes unused data-blocks from Linked Data.

  Recursive Delete
  : Removes data-blocks only used by unused data-blocks, ensuring that no orphaned data-blocks remain after execution.

Main Region

Object Interaction Mode



Mode icons. Two objects are currently in Edit Mode; a third could be added.

If a selected object is in an interaction mode other than the default Object Mode, the Outliner shows an icon representing this mode on the left.

If the active object has such an icon, the Outliner also shows a dot next to objects of the same type. You can click such a dot to switch over to a different object while staying in the same mode.

If the mode supports Multi-Object Editing, you can also click a dot with `Ctrl-LMB` to *add* an object to the mode.

You can click the mode icon of the active object to switch it (and any other objects in case of Multi-Object Editing) back to Object Mode. You can also `Ctrl-LMB` the mode icon of a selected – but not active – object to switch only that object back to Object Mode.

Restriction Toggles

The right side of the Outliner shows a series of toggle icons for every collection, object, bone, modifier, and constraint. These can be used to make the item invisible, unselectable, and so on.



Restriction toggles.

Note

Only a few icons are shown by default. You can use the Filter pop-over to show additional ones.

Clicking an icon with `Shift-LMB` toggles it for the item and all its children.

Clicking a collection's icon with `Ctrl-LMB` enables it for the collection (and its parent/child collections) and disables it for all others. Clicking again enables it for the others again.

Exclude from View Layer (checkbox) *Collections*
: Uncheck to disable the collection for the current View Layer. Its contents will be hidden in the 3D Viewport, the render, and even the Outliner.

Disable Selection (mouse cursor icon)

Toggles whether the object can be selected in the 3D Viewport. This can be useful for, say, references images that you only want to display and never select/move.

Hide in Viewports (eye icon)

Toggles the visibility of the object or collection in (only) the 3D Viewport, for the current view layer. The render is not affected.

As an alternative to clicking this icon, you can press `H` while hovering over the 3D Viewport to hide the selected objects, or `Alt-H` to unhide all objects.

This setting only applies within the current blend-file: when you Link or Append it to another blend-file, all collections and objects will be visible there.

Objects hidden this way are still part of the view layer, so they still get evaluated and affect playback performance.

See also

Collections can be hidden for individual 3D Viewports; see Local Collections in the Sidebar.

Disable in Viewports (screen icon)

Toggles the visibility of the object or collection in (only) the 3D Viewport, for all view layers. The render is not affected.

This setting is separate from *Hide in Viewports*. An object needs to have both settings enabled to be visible. You can use this one for "long-term invisibility," keeping an object invisible even after pressing `Alt-H`.

This setting carries over to other blend-files when linking or appending.

Objects hidden this way are no longer part of the view layer, so they no longer get evaluated and don't affect playback performance.

Disable in Renders (camera icon)

Toggles the visibility of the object or collection in (only) the render, for all view layers. The 3D Viewport is not affected.

This is typically used for supporting objects that help modeling and animation yet don't belong in the final image.

Holdout *Collections*

Makes the objects in the collection cut a fully transparent hole into the render output of the view layer.

See also

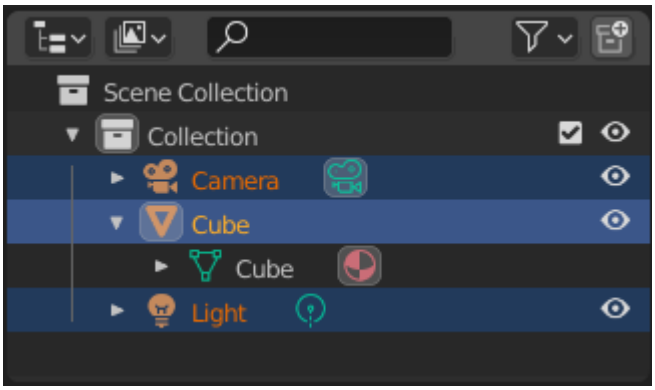
Holdout Shader Node

Indirect Only *Collections Cycles*

Objects in the collection only contribute to indirect light.

Selecting

Selection is done with LMB (and/or the context menu) on the row of a data-block. Single selections will also activate the data-block. The rows of selected data-blocks are highlighted blue, with the active data-block highlighted in a lighter blue.



Selected data-blocks with the Cube active.

Clicking in empty space below the list of data-blocks will deselect all.

Note

By default, selecting data-blocks in the Outliner will select the respective objects, bones, and sequences in the 3D Viewport and Video Sequencer. Selections in the 3D Viewport and Video Sequencer will be synced to each Outliner. To disable selection syncing, turn off the toggle in the filter popover.

Children of a data-block can also be selected by clicking the icon that is displayed to the right of the parent data-block's name.

Selecting Multiple Data-Blocks

Extend the selection one data-block at a time using Ctrl–LMB. Each data-block added to the selection this way will be made the active data-block.

Select a range of elements from the active element using `Shift-LMB`. To select a range without deselecting the previous selection, use `Shift-Ctrl-LMB`.

Select all children of an item by double clicking the item's icon.

A click and drag from any location in the Outliner other than a name or icon will begin a box selection. Use `Shift` to add and `Ctrl` to subtract from existing selections with box select. Box select can also be started with `B`.

To select all items use `A`; `Alt-A` will deselect all items.

The keyboard arrow keys can be used to navigate and select in the Outliner. Keyboard selection and navigation starts from the active data-block.

`Up`	Select the previous element in the list.
`Down`	Select the next element in the list.
`Shift-Up`	Select the previous element without deselecting.
`Shift-Down`	Select the next element without deselecting.
`Left`	Close the data-block or select the parent.
`Right`	Open the data-block to view children or select the first child.
`Shift-Left`	Close this and all child data-blocks.

`Shift-Right`	Open this and all child data-blocks.

Properties Editor Sync

When clicking the icon (not the name) of an item in the Outliner, the Properties editor will automatically switch to the relevant tab. This feature can be disabled using the Properties editor's Sync with Outliner option.

Editing

Context Menu

Show the context menu for a data-block with `RMB` on the icon or name. Depending on the type of the selected data-block(s), you will have all or part of the following options:

Copy/Paste
: Copies/pastes the selected data-blocks.

Delete `X`, `Delete`
: Removes all usages of the selected data-blocks. Objects are removed from all scenes, materials are removed from all meshes, and so on.

Note

Pressing these shortcuts while hovering over the 3D Viewport will instead *Unlink* the selected objects, removing them only from the current scene.

Delete Hierarchy
: As above, but also affects child collections/objects. Note that if you run this on a collection, child objects that (also) belong to another collection will not be deleted.

Select
: Adds the items that are selected in the Outliner to the selection in the 3D Viewport. This is only useful when Sync Selection is disabled, as when it's enabled (which is the default), the Outliner selection is synchronized to the 3D Viewport automatically.

Select Hierarchy
: Adds the children of the selected items to the selection in the Outliner. If *Sync Selection* is enabled, this also adds them to the selection in the

3D Viewport.

Deselect
: Removes the items that are selected in the Outliner from the selection in the 3D Viewport.

Unlink
: Removes the current usage of the data-block while keeping any others. Objects are only removed from the current scene, materials are only removed from the current mesh, and so on.

Collections

Collections let you organize the content of a scene. They can contain objects as well as other collections.

New
: Creates a new collection.

Duplicate Collections
: Recursively duplicates the collection including all child collections, objects, and object data.

Duplicate Linked
: Recursively duplicates the collection including child collections and objects, but reuses object data.

Instance to Scene
: Creates a new collection instance.

Visibility
: Controls the collection's visibility in the 3D Viewport and the final render.

  Isolate
  : Shows the selected collection (as well as its child and parent collections) and hides all the others.

  Show/Hide

Changes the Hide in Viewports setting for the selected collections.

Show/Hide Inside
: Changes the Hide in Viewports setting for the selected collections and all their children.

Enable/Disable in Viewports
: Changes the Disable in Viewports setting for the selected collections.

Enable/Disable in Renders
: Changes the Disable in Renders setting for the selected collections.

View Layer
: Controls the collection's interactions with the View Layer.

Disable/Enable in View Layer
: Changes the Exclude from View Layer setting for the selected collections.

Set/Clear Holdout
: Changes the Holdout setting for the selected collections.

Set Color Tag
: Assigns or clears a collection's color tag for the selected collection.

ID Data

Unlink
: Removes the current usage of the data-block while keeping any others (e.g. removing a material from only the current mesh).

Make Local
: Turns an externally linked data-block into a local one.

Make Single User
: This menu item is not currently functional. You can use the *User Count* button in the Data-Block Menu instead.

Delete
: Deletes the selected data-block.

Remap Users
: Replaces all usages of the selected data-block by a different one. For example, you could use this to globally replace a material by another.

Copy/Paste
: Copies/pastes selected data-blocks.

Add/Clear Fake User
: Adds/removes a fake user, which prevents unused data-blocks from getting automatically deleted when saving and reloading the blend-file.

Rename `F2`
: Renames the selected data-block.

Select Linked
: Selects the data-blocks that use the currently selected one (e.g. selecting all the objects that use the selected material). See Select Linked.

Mark as Asset

See Creating an Asset.

Clear Asset

See Removing Assets.

Clear Asset (Set Fake User)

See Removing Assets.

Library Override

See Library Overrides.

View

Show Active `Period`
: Centers the tree view to the active item.

Expand/Collapse All `Shift-A`
: Expands/collapses every single item in the tree.

Show Object Hierarchy `Home`
: Expands all objects that have child objects, and collapses all objects that don't.

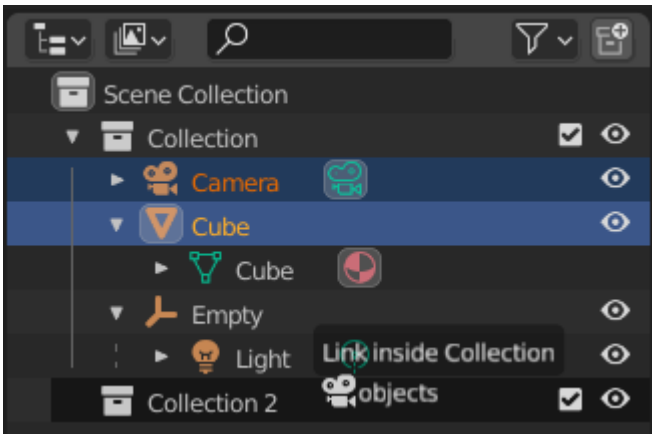
Show/Hide One Level `NumpadPlus`/ `NumpadMinus`
: Expands/collapses a level down/up the tree across all items.

Usage

Relations Management

You can move an object (or collection) to a different parent collection by dragging and dropping.

You can link an object (or collection) to a parent collection by dragging and then holding `Ctrl` while dropping. This way, you can make the object (or child collection) part of multiple parent collections at the same time.



Linking objects to a collection.

You can parent an object to another by dragging and then holding `Shift` while dropping.

Note

Drag and drop will attempt to operate on the entire selection. Selected data-blocks that are incompatible with the operation will remain unmodified.

Modifiers, Constraints, and Visual Effects

You can manage Modifiers, Constraints, and Visual Effects from the Outliner in a couple of ways:

- You can drag and drop individual items to change their order within the stack or to copy them to another object.
- You can drag and drop the group item (e.g. *Modifiers*) to copy the whole stack to another object. The target object's existing stack will be replaced.
- You can apply and delete items using the context menu.

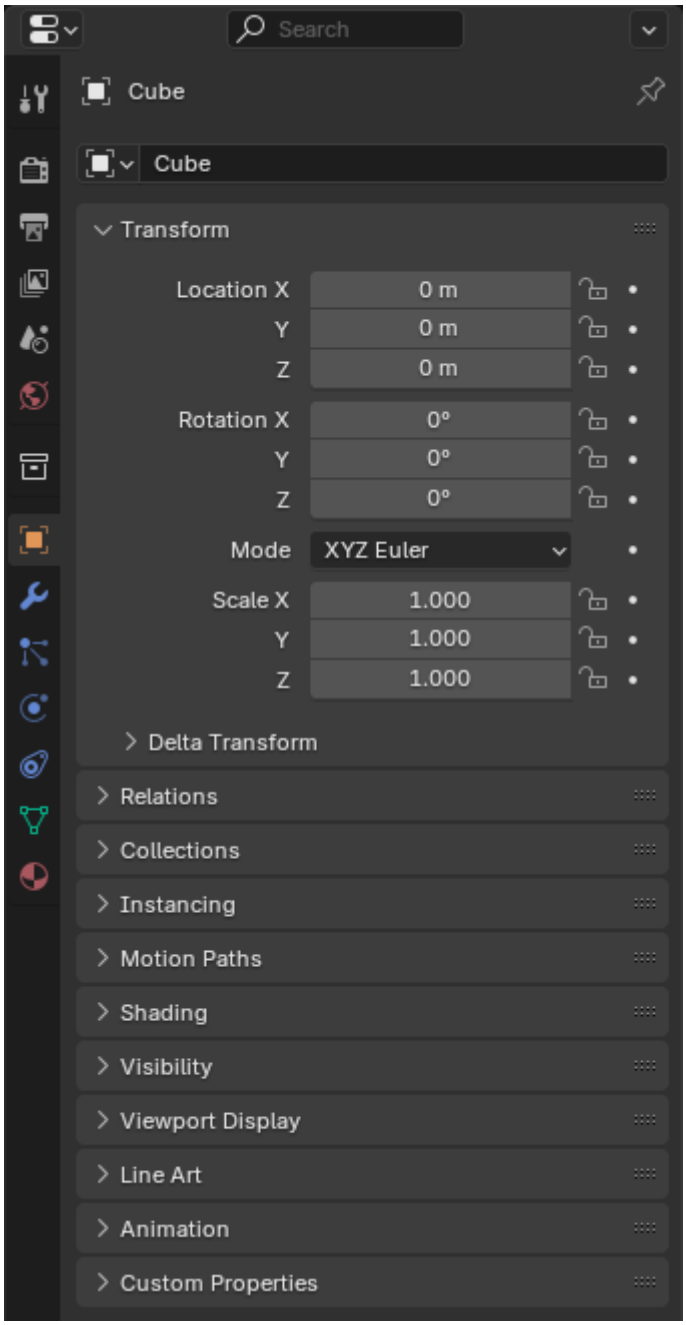
Drag & Dropping to 3D Viewport

Dragging an object from the Outliner to the 3D Viewport creates a duplicate – a new object with its own copy of the underlying object data.

Dragging object data from the Outliner to the 3D Viewport creates a linked duplicate – a new object that references the same underlying object data.

Properties

The Properties editor shows and allows editing of many active data, including the active scene and object.



The Properties, with *Object* properties shown.

Tabs

The Properties editor has several categories, which can be chosen via tabs (the icons column to its left). Each tab regroups properties and settings of a data type, and is documented in its own manual sections, linked below.

Active Tool and Workspace Settings

This first tab contains settings for the active tool (in the 3D Viewport) and the current workspace.

Scene

These tabs contain settings for the active scene.

- Render: EEVEE, Cycles or Workbench settings
- Output
- View Layer
- Scene
- World

Collection

This tabs contain settings for the active Collection.

Object

These tabs are used to add features, and to change properties for the active object. Depending on the type of the active object, some of those will be hidden.

- Object
- Modifiers (or Grease Pencil Modifiers)
- Object Visual Effects
- Particles
- Physics

- Object Constraints

Object Data

The main tab of that category (often the only one) always has the same name, *Object Data*, but its icon will change based on the actual type of the active object.

Geometry Objects:

- Mesh
- Curve
- Surface
- Text
- Metaball
- Grease Pencil

Rigging and Deformation Objects:

- Armature
  - Bone
  - Bone Constraints
- Lattice

Other Types of Objects:

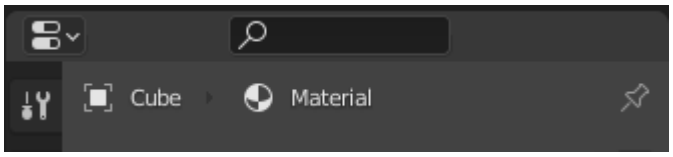
- Empty
- Speaker
- Camera
- Light
- Light Probe

Object Shading

Depending on the type of the active object, some of those will be hidden.

- Material
- Texture

Header



The header of the Properties.

Display Filter `Ctrl-F`
: Allows you to search for a property by its name. The results will be highlighted with there corresponding panel also highlighted and expanded. The search also works across multiple Tabs; graying out tabs with no search results. You can start a search using `Ctrl-F` or clear a search with `Alt-F`. This filter is useful if you do not remember where a property is within the Properties.

Data Context
: Just below the header is a list of icons and text items which show the owner of the properties being edited, together with some dependency context if needed. In the example above, the material “Material” is used by the active object “Cube”.

Toggle Pin ID
: By toggling on the pin icon to the right, Blender will display only the currently shown data-block’s properties in that editor, disregarding further selection changes. Toggle off that pin to switch back to default behavior, showing active data-block’s properties.

Options

Sync with Outliner
: Sync tabs with the Outliner data-block selection. See Properties Editor Sync for more information.

Always::	Always, when possible, change Property editor tabs when clicking an icon in an Outliner.
Never::	Never change Property editor tabs when clicking an icon in an Outliner.
Auto::	Change tabs only when the Property editor shares a border with an Outliner.

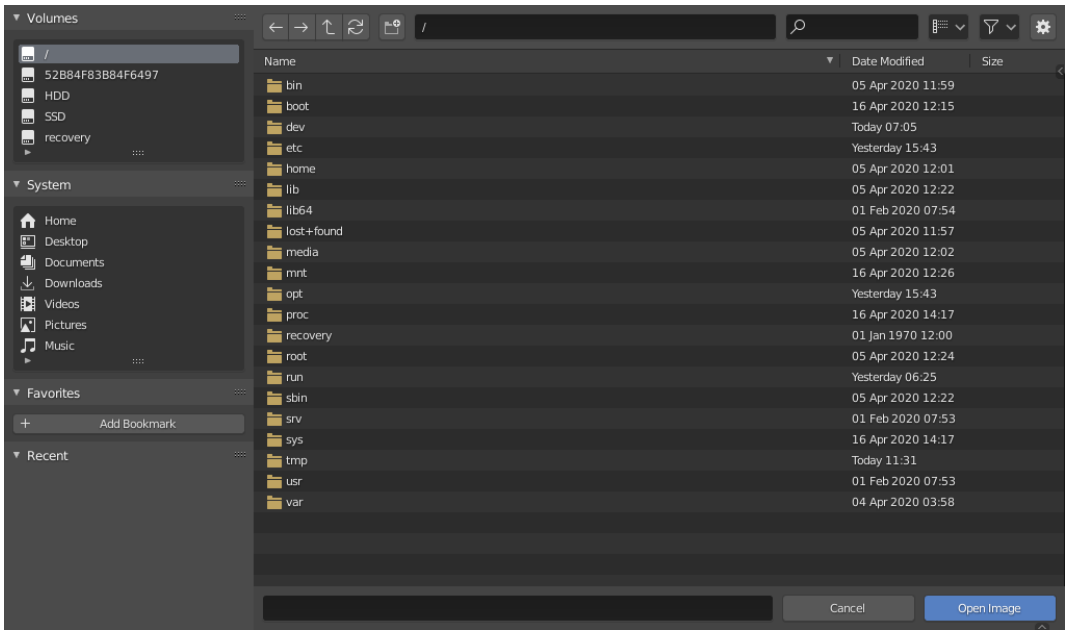
File Browser

The File Browser is used in all file-related operations. These include:

- Opening and saving blend-files.
- Browsing the content of other blend-files when appending or linking data-blocks (see Linked Libraries).
- Importing from/exporting to other file formats.
- Updating the locations of previously imported media (images, videos, fonts…).

The most common way to use this editor is through modal operators (like opening or saving a blend-file). The File Browser will appear in a new window, wait for you to select a file, and then close again.

You can also use the File Browser like a regular, permanently visible editor. In fact, the predefined Video Editing workspace uses it this way. This lets you drag-and-drop media from the browser straight into e.g. the 3D Viewport or the Video Sequencer, saving you some overhead.



The File Browser.

Interface

Main Region

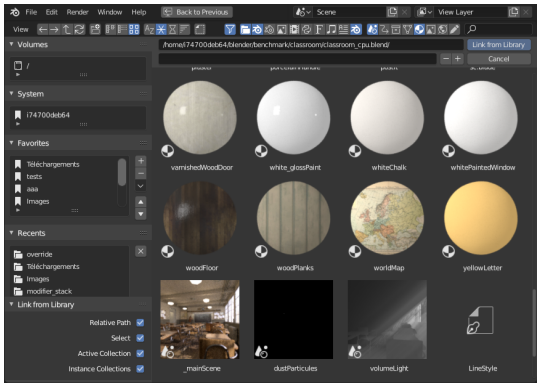
The main region lists files, folders, or blend-file contents. Hovering over an item will show a tooltip with extra information.

Previews

In its *Thumbnail* display mode, the File Browser supports many types of previews. These include:

- Image and video formats
- Fonts
- Blend-files
- Internal Data-blocks

In order to get previews for data-blocks, these must first be generated. See Blend-Files Previews.



The File Browser in *Thumbnail* mode.

Directory Region

Above the file list, there's a textbox showing the current folder path, along with buttons for navigating.

Previous Folder `Backspace`, `Alt-Left`, `Mouse4`
: Move to previous folder in navigation history.

Next Folder `Shift-Backspace`, `Alt-Right` `Mouse5`
: Move to next folder in navigation history.

Parent Directory `P`, `Alt-Up`
: Move up to parent directory.

Refresh File List `R`, `NumpadPeriod`
: Refresh current folder.

Create New Directory `I`
: Create a new directory inside the current one.

Directory `Ctrl-L`
: The current folder path. `Tab` will auto-complete an existing path. If you type a nonexistent path, you will be prompted to create it.

Search `Ctrl-F`
: Filter items by name. The wildcard `*` will match anything, e.g. `bl*er` will match both `blender` and `blogger`. There is always an implicit wildcard at the start and end of the search text, so `blender` will also match `test_blender_file.blend`. This field can also be used to filter some specific file extension (e.g. `.png` will list all PNG files).

Display Mode
: Control how files are displayed.

  Vertical List:: Displays files and folders in a vertical list.
  Horizontal List:: Displays files and folders in a horizontal list.
  Thumbnails:: Shows previews.

Display Settings

Size
: The size of the thumbnails.

Recursions

The number of directory levels to show at once in a flat way.

None::	List only the current directory's content.
Blend File::	List the whole content of a blend-file (only available when linking or appending data-blocks).
One Level::	List all subdirectories' content, one level of recursion.
Two Levels::	List all subdirectories' content, two levels of recursion.
Three Levels::	List all subdirectories' content, three levels of recursion.

Hint

Showing several levels of directories at once can be handy to e.g. see your whole collection of textures, even if you have arranged them in a nice set of directories to avoid having hundreds of files in a single place.

In the *Append/Link* case, showing the content of the whole blend-file lets you link different types of data-blocks in a single operation.

Warning

The more levels you show at once, the more time it will take to list them all.

Sort By

Sorts items by one of the four methods:

Name::	Sort the file list alphabetically.
Extension::	Sort the file list by extension/type.
Modified Date::	Sort files by modification time.
Size::	Sort files by size.

Filter Settings

The toggle with the funnel icon controls whether filtering is enabled or not. The dropdown button next to it shows the filtering options.

File Types
: Filters files by categories, like folders, blend-files, images, etc.

Blender IDs
: When appending or linking, you can also filter by data-block categories, like scenes, animations, materials, etc.

Show Hidden `H`
: Shows hidden files (starting with a `.`).

Execution Region

These controls are at the bottom of the editor.

File Name
: Text field to edit the file name and extension. Turns red to warn you about overwriting an existing file. `Tab` will auto-complete to existing names in the current directory.

  Increase/Decrease Number `+`, `-`
  : Adds/increases or removes/decreases a trailing number in your file name (used e.g. to store different versions of a file).

Cancel `Esc`
: Closes the File Browser and cancels the operation.

Confirm `Return`
: Confirm the current directory and file name. You can also double-click a file or data-block in the main region.

Quick Access Region

The region on the left contains a few panels that let you quickly jump to certain directories with a single click.

Bookmarks

A custom list of folders that you use often. You can use the buttons to the right of the list to add/remove/move items.

System

Common directories such as the home directory in Linux or the "Documents" folder in Windows.

Volumes

Drives and network mounts.

Recent

Recently accessed folders.

Clicking the down arrow button to the right reveals *Clear Recent Items* to fully clear this list.

You can control how many folders appear in this list with the *Recent Files* number field of the Save & Load tab in the Preferences.

Operator Options Region

The right region shows the options of the calling operator. Besides the common actions listed below, many import/export add-ons will also expose their options there.

Open, Save, Save As Blender File
: See Opening & Saving.

Open, Replace, Save As Image
: See Supported Graphics Formats.

Link/Append from Library
: See Linked libraries.

For the common option:

Relative Path
: See Relative Paths.

Header Region

The header only contains two menus, one with the standard editor *View* controls and the other to list a few Selecting operators for the sake of discoverability. These menus are not visible when the browser is in a modal window.

Navigating

Entering a Directory `Return`
: Double-click a directory to enter it.

Parent Directory `P`
: Takes you up one level of directory.

File Drop

You can also drag and drop a file or directory from your file manager into the Blender File Browser. This will navigate to the item and select it.

Selecting

Select
: Click `LMB` to select a single item. Additionally hold `Ctrl` to add/remove that item to/from the selection, or `Shift` to select a range of items.

Dragging
: Dragging with `LMB` starts a box selection.

Note

You can always select several entries in the File Browser – the last selected one is considered the active one. If the calling operation expects a single path (like e.g. the main blend-file *Open* one), it will get that active item's path, and the other selected items will be ignored.

Arrow Keys

It is also possible to select/deselect files by "walking" through them using the arrow keys:

- Press an arrow key to select the next/previous file in the list and deselect all the others.
- Hold `Shift` to keep the current selection (and add to it).
- Hold `Shift-Ctrl` to invert the selection as you pass over it.

If no file is selected, the arrow key navigation selects the first or last file in the directory, depending on the arrow direction.

Editing

The following operations are available in the file list's context menu.

External
: Use the operating system to perform an action on the file or directory. The options listed below might not be available on all operating systems.

Open::	Open the file.
Open Folder::	Open the folder.
Edit::	Edit the file.
New::	Create a new file of this type.

Find File:: Search for files of this type.
Show:: Show this file.
Play:: Play this file.
Browse:: Browse this file.
Preview:: Preview this file.
Print:: Print this file.
Install:: Install this file.
Run As User:: Run as specific user.
Properties:: Show OS Properties for this item.
Find in Folder:: Search for items in this folder.
Command Prompt Here::
Open a command prompt here.

Delete `Delete`, `X`

Delete the currently selected files or directories by moving them to the operating system's "trash".

Note, on Linux deleting directories requires KDE or GNOME.

Rename `F2`

Change the name of the currently selected file or directory.

Asset Browser

The *Asset Browser* is the main interface for organizing and using assets.

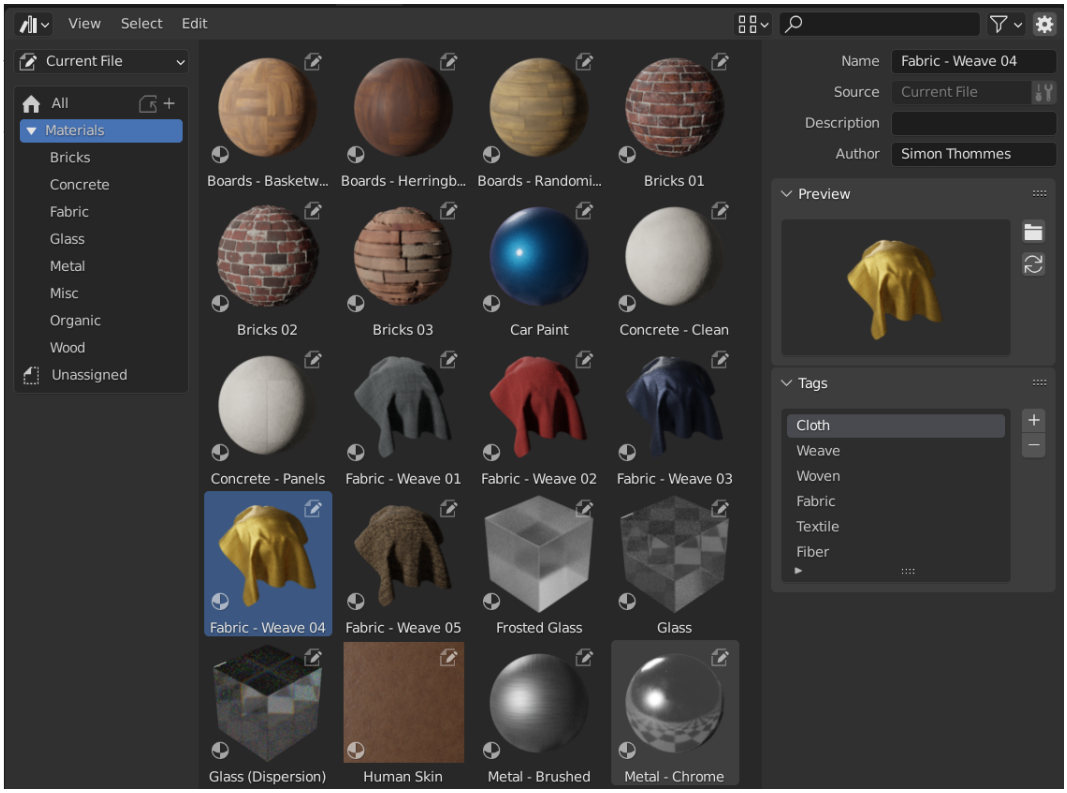
See also

Asset Libraries
: For general information on Blender's asset library system, including how to create and edit assets, and design choices.

Asset Catalogs
: For organizing assets.

Pose Library
: Built on top of the Asset Browser.

Interface



Asset Browser, showing materials in an asset library.

Main Region

The center region of the Asset Browser lists the assets contained in the selected catalog.

Click `LMB` to select a single asset. Additionally hold `Ctrl` to add/remove that asset to/from the selection, or `Shift` to select a range of assets. You can also drag `LMB` to perform a box select.

The region has a context menu with the following operations:

Refresh Asset Library `R`
: Refreshes the list.

Clear Asset
: See Removing Assets.

Clear Asset (Set Fake User)
: See Removing Assets.

Open Blend File
: Opens the blend-file containing the asset.

Display Size
: Changes the size of the preview thumbnails.

Asset Library Region

The region on the left lets you select an asset library and shows its catalogs. You can show/hide this region by pressing `T`.

Asset Library
: The asset library whose catalogs to show.

  All Libraries:: Show catalogs from all available libraries.
  Current File:: Show the catalogs in the current blend-file (even if that file is not yet part of an asset library). See The Current File Asset Library for more information.
  Essentials:: Show the catalogs that come bundled with Blender.

  Any libraries that you added in the File Path Preferences are listed here too.

Copy Bundle to Asset Library
: Shown when *Asset Library* is set to *Current File* and the current blend-file is an asset bundle that's not yet part of an asset library.

  Lets you select a target asset library, then opens a File Browser in that library's root folder so you can save the current blend-file there. Once saved, the assets in the blend-file become available as part of the library.

Catalogs

Tree view that shows the catalogs of the selected asset library. A catalog is a group of assets; when you select one, only the assets in that catalog and its child catalogs will be listed.

You can rename a catalog by double-clicking it, or assign it to a different parent catalog by dragging and dropping.

Add-ons and features like the Pose Library can show custom panels here.

Asset Details Region

The region on the right shows the metadata of the active asset. You can show/hide this region by pressing `N` or clicking the gear icon in the header.

Only metadata of assets contained in the current blend-file can be edited.

Name
: The asset's name. Unique for the asset data type within the same blend-file.

Source
: The full path of the blend-file that contains the asset.

  Open Blend File
  : Opens the blend-file that contains the asset in a new Blender instance. When this instance is closed, the Asset Browser will be automatically refreshed.

License
: Optional name of the license under which this asset is distributed. Not used by Blender itself.

Copyright
: Optional copyright notice. Not used by Blender itself.

Description
: Optional asset description. Not used by Blender itself.

Author
: Optional field for the asset author. Not used by Blender itself.

Preview

Shows the preview image of the asset. See Asset Previews.

Load Custom Preview
: Opens a File Browser where you can select a new image for the asset preview.

Generate Preview
: Autogenerate a new preview for the asset.

Preview
: Menu of additional preview operators.

  Render Active Object
  : Generates a preview based on the 3D Viewport's Active object. This is useful for node groups, which cannot automatically generate their own preview.

Tags

Panel for viewing and editing asset tags. These do not have any meaning to Blender and can be chosen freely. When using the search field to filter the assets, the assets whose tags (partially) match the search term will also be shown.

Note

Depending on the current mode of the object and the selected asset types, more panels may be shown. For example, see Pose Library.

Using Assets

As a general rule, **an asset can be used by dragging it from the Asset Browser to the desired location**. Objects and worlds can be dragged from the Asset Browser into the scene. Materials can be dragged onto the object that should use them. Geometry nodes can also be dragged onto objects to add a Geometry Nodes Modifier. The use of pose assets is different, and is described in Pose Library.

There are several things that can happen when an asset is used, depending on the following configuration of the Asset Browser:

Import Method

Determines how data is managed when an asset is imported. This option can be found in the center of the Asset Browser header (when an asset library other than *Current File* or *Essentials* is selected):

Follow Preferences

Use the import method set in the File Path Preferences.

Link

Same as File ▸ Link…

The asset will be linked to the current blend-file, and thus be read-only. Later changes to the asset file will be reflected in all files that link it.

Append

Same as File ▸ Append…

The asset and all its dependencies will be copied into the current file. Dragging a material into the scene three times will result in three independent copies. Dragging an object into the scene three times will also result in three independent copies.

“Dependencies” in this case means everything the asset refers to. For an object, this can be its mesh and materials, but also other objects used by modifiers, constraints, or drivers.

Since the file now has its own copy of the asset, later changes to the asset file will not be reflected in the file it's appended to.

Append (Reuse Data)

Specific to the Asset Browser.

The first time an asset is used, it will be appended, including its dependencies, just as described previously. However, Blender will keep track of where it originated, and the next time the asset is used, as much data as possible will be reused. Dragging a material into the scene three times will only load it once, and just assign the same material three times. Dragging an object into the scene three times will create three copies of the object, but all copies will share their mesh data, materials, etc.

Since the file now has its own copy of the asset, later changes to the asset file will not be reflected in the file it's appended to.

Tip

Instancing Collections

Mimics the Instance Collections option when appending from the file browser

Some asset types such as collections can be created as an instanced collection. This is done by enabling the *Instance* option after dragging collection assets into the 3D Viewport. By enabling this option, an empty object is added that uses an instance of the collection. If this option is disabled, the full collection hierarchy will be added to the scene.

Collection Assets from the current file will always be instanced.

Note that all regular Blender operations are available after the asset has been added to the current file. For example, you could choose to link an object to the scene; this will also link its mesh and its materials. Then you can make the object itself local (*Object ▸ Relations ▸ Make Local… ▸*

Selected Objects), while keeping the mesh and materials linked to the asset files. This will result in a local, and thus editable, object, and keep the mesh and materials automatically up to date with any changes in the asset library.

Asset Previews

Preview images are typically generated automatically when you mark a data-block as an asset. Objects are captured from their local -Y axis, while collections are captured from the global -Y axis (as these don't have a local axis).

If the auto-generated preview image isn't sufficient, you can replace it by a custom one.

For previews of pose assets, see Controlling the Look of Preview Images.



Preview panel in the Asset Browser.

Asset Bundles

Asset bundles are blend-files that do not reference any other file, and whose name ends in `_bundle.blend`. Any textures and other external files need to be packed into the current blend-file.

Asset bundles can be copied to an asset library via the Asset Browser:

- Open the asset bundle blend-file.
- Switch its Asset Browser to *Current File* (if it's not set to that already).
- Click on *Copy Bundle to Asset Library*.
- Choose the asset library to copy it to.
- A File Browser will open at the root folder of the selected asset library. Choose the desired location of the blend-file, and click the *Copy to Asset Library* button.

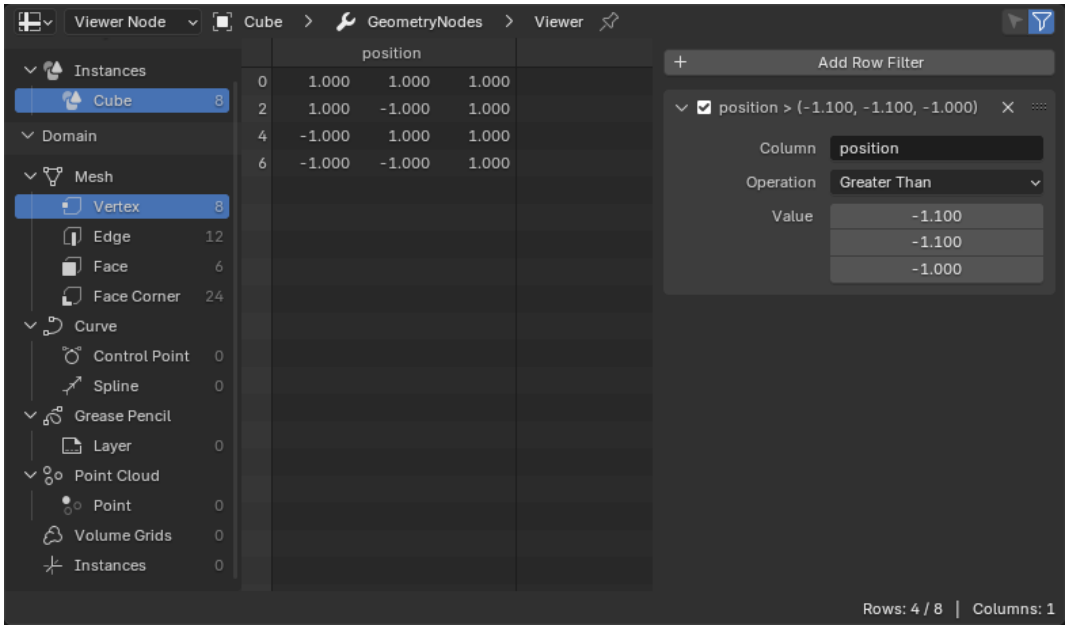
- The blend-file will be saved at the chosen location, and any catalogs of the asset bundle will be merged into the target asset library.

Note

Both the word “asset” and the word “bundle” are commonly used, and not necessarily with the same meaning as described here. Not everything that’s presented as an “asset bundle” will have the *Copy to Asset Library* functionality available; for that, the bundle file needs to adhere to the definition above.

Spreadsheet

The Spreadsheet editor is used to inspect the geometry attributes of the active object, typically in order to debug geometry nodes.



The Spreadsheet editor.

Header

Object Evaluation State

The state for which to display data.

Evaluated:: Display data from the object with all modifiers applied.

Original:: Display data from the original object without any modifiers applied.

Viewer Node:: Display the data that's fed into the active Viewer Node.

You can also toggle between *Evaluated* and *Viewer Node* by clicking the eye icon in the Viewer node's header.

Breadcrumbs
: Shows the name of the active object, and (if *Object Evaluation State* is set to *Viewer Node*) the name of the Geometry Nodes modifier and the active Viewer node.

  You can click one of the arrows between the names to hide the modifier.

Toggle Pin
: Click to "lock" the editor to the currently active object, and have it keep displaying that object's data even if another object becomes active. Click again to unlock.

Selected Only
: This option is only available if the object is in Edit Mode. When checked, only data for the selected geometry elements is shown.

Use Filter
: Whether to use the filters that are defined in the Sidebar (see below).

Main Region

The main view shows the actual spreadsheet. Column names and row indices remain visible when scrolling down or to the side.

Note

Tooltips give more detail about the value, depending on the type. For example, *Byte Color* attributes are displayed as scene linear floats, but the actual integer values are displayed when hovering over the float values, and *Matrix* attribute values are only displayed in tooltips.

Data Set Region

With region on the left, you can choose what to display data for. The top tree lets you pick from the hierarchy of geometries, such as a mesh inside an instance. The bottom tree lets you pick a domain, such as mesh vertices or curve splines.

Each tree item shows the number of elements inside.

Sidebar

In the Sidebar, you can define filters so that only the rows matching these filters are displayed. Click *Add Row Filter* and set up the properties described below.

Enabled
: Uncheck to temporarily disable the filter.

Column
: The name of the column to filter on. If there is no column with the specified name, the filter will be grayed out and ignored.

  If you want to filter on an attribute from another domain, you can use the Store Named Attribute Node to create a copy that's converted to the current domain, then filter on that.

Operation
: For numerical columns, you can select one of the following comparison operators. Other columns only support *Equal To*.

  Equal To:: Only display rows whose value for the column is equal to the filter value (within the specified threshold).

  Greater Than:: Only display rows whose value for the column is greater than the filter value.

  Less Than:: Only display rows whose value for the column is less than the filter value.

Value
: The filter value to compare the row value to.

Threshold
: How much the row's value is allowed to deviate from the filter value before it is excluded.

Status Bar

The status bar shows how many rows and columns there are, and how many rows remain after filtering.

Preferences

- Introduction
  - Managing Preferences

Sections

- Interface
- Viewport
- Lights
- Editing
- Animation

- Get Extensions

- Add-ons
- Themes

- Input
- Navigation
- Keymap

- System
- Save & Load
- File Paths

- Experimental

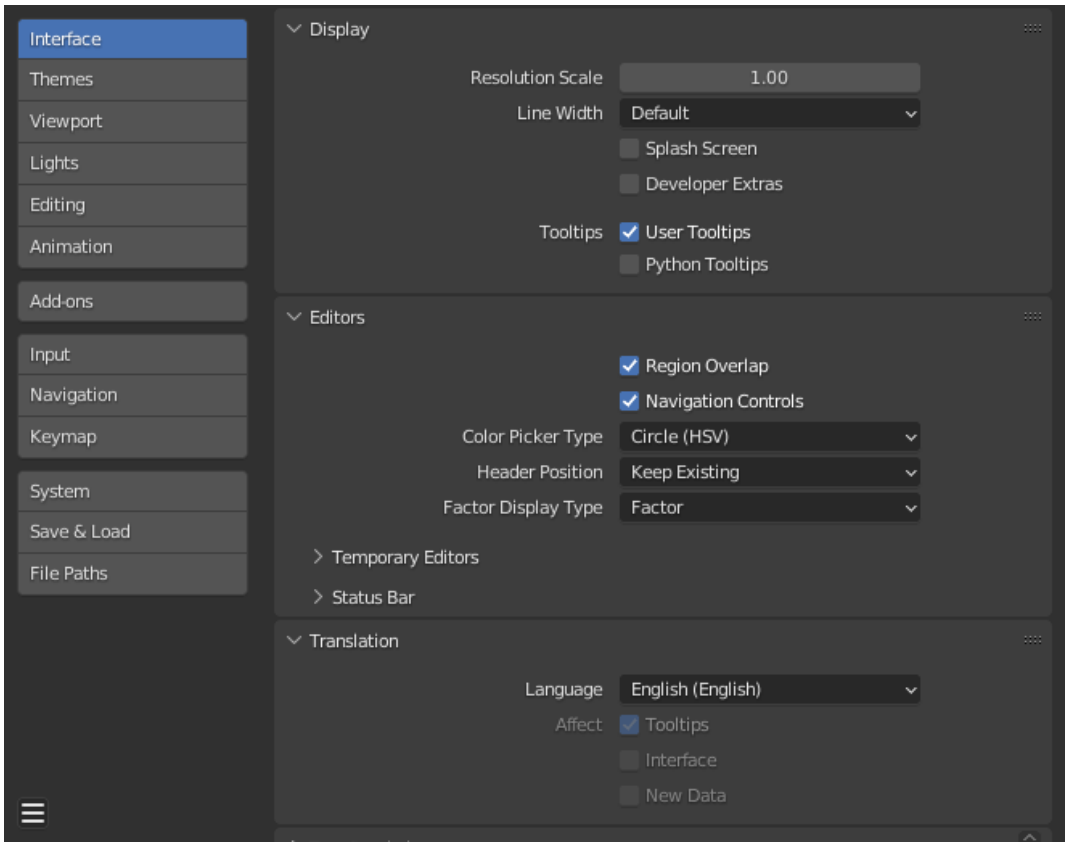
Introduction

Reference

Menu:: *Edit ▸ Preferences…*
Shortcut:: `Ctrl-Comma`

This chapter explains how to change Blender's default configuration with the *Preferences* editor.

The Blender *Preferences* contains settings to control how Blender behaves. At the left of the editor, the available options are grouped into sections.



Blender Preferences window.

Managing Preferences

Default preferences are managed from the ≡ menu in the preferences window.

The following items are available in this menu:

Auto-Save Preferences
: By default changes to preferences are saved on exit, this allows changes to the keymap and Quick Favorites menu to be stored and used between Blender Sessions.

  When disabled, a *Save Preferences* button is shown to manually perform the operation.

Revert to Saved Preferences
: Undoes any unsaved modifications, loading the previously saved state.

Load Factory Preferences
: Completely undo all the modifications made to the preferences, resetting to the state used before making customizations.

  Note

  After running *Load Factory Preferences*, auto-save will be disabled for the current session.

  This allows you to switch back to the factory settings for testing or following tutorials for example, without the risk of accidentally auto-saving over the preferences you have manually configured.

  If you wish to save these as your preferences, run *Save Preferences* manually.

  Note

  This only resets the preferences and will not affect settings stored in the startup file. This includes app templates, area locations, and any

Blender properties not part of the preferences.

These must be reverted though *File ▸ Defaults*.

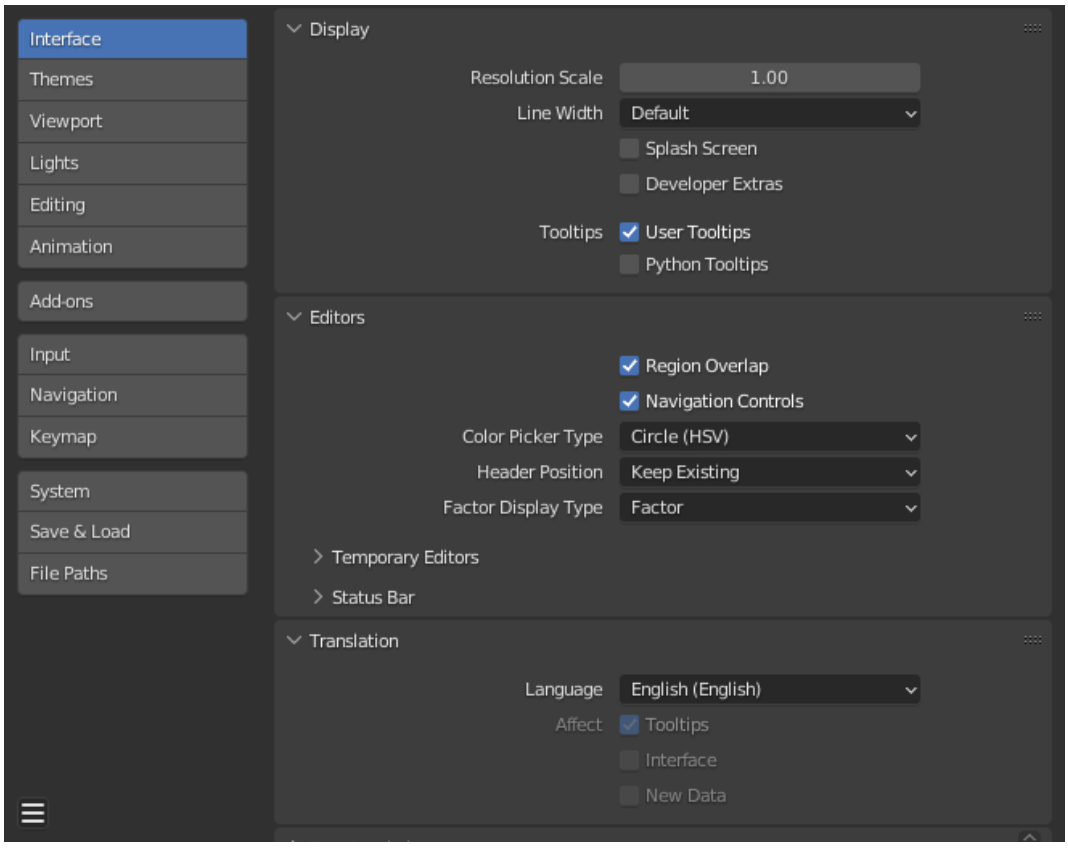
Tip

It can be valuable to make a backup of your preferences in the event that you lose your configuration.

See the directory layout section to see where your preferences are stored.

Interface

Interface configuration lets you change how UI elements are displayed and how they react.



Display

Resolution Scale
: Adjusts the size of fonts and buttons relative to the automatically detected DPI. During typical usage, you may prefer to use zoom which is available in many parts of Blender interface.

Line Width
: Scale of lines and points in the interface e.g. button outlines, edges and vertex points in the 3D Viewport.

Thin, Default, Thick

Splash Screen
Display the Splash Screen when starting Blender.

Developer Extras
Show settings and menu items which are intended to help developers, this includes:

- Operator Search
- Sequencer Cache Settings

Button Context Menu
Online Python Reference
To open the Python reference manual.

Copy Python Command
To copy the expression used when pressing the button.

Edit Source
To edit Python source code that defines the button.

Edit Translation
The option to edit UI translations (only available when the *Manage UI translations* add-on is also enabled).

3D Viewport
Show Indices
The option to show mesh vertex/edge/face indices in the overlay popover.

Preferences
Experimental Tab
Work in progress features can be enabled here which are currently being tested.

Tooltips
User Tooltips

When enabled, a tooltip will appear when your mouse pointer is over a control. This tip explains the function of what is under the pointer, shows the associated hotkey (if any).

Python Tooltips
: Displays a property's Python information below the tooltip.

Search – Sort by Most Recent
: Show most recently selected items at the top of search results, otherwise search results are sorted alphabetically.

Editors

Region Overlap
: This makes regions overlap the viewport. It means that the *Toolbar* and *Sidebar* regions, will be displayed overlapping the main area.

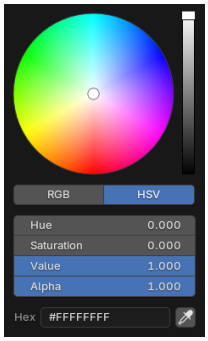
Navigation Controls
: Show navigation controls at top right of the area. This impacts the 3D Viewport as well as image spaces.

Note

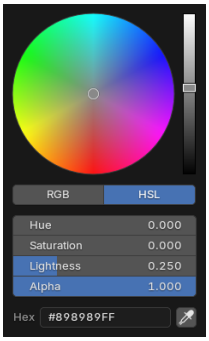
If you are familiar with navigation key shortcuts, this can be disabled.

Color Picker Type
: Choose which type of Color Space you prefer for the Color picker. It will show when clicking `LMB` on any color field.

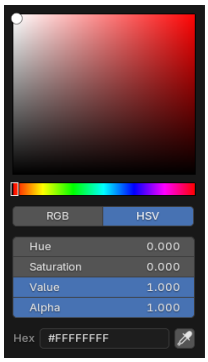
Color Picker types.



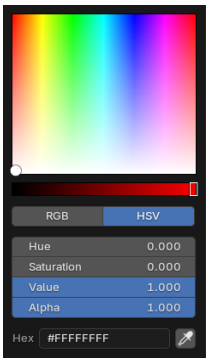
Circle HSV.



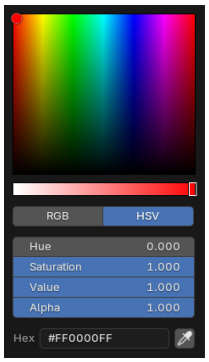
Circle HSL.



Square (SV + H).



Square (HS + V).



Square (HV + S).

Header Position

The default header position when opening a new editor.

Keep Existing:: Uses top for most editor types and the positions saved in the start-up file.

Top/Bottom:: Always positions the header at the top or the bottom of the editor.

Factor Display Type

How factor value types are displayed in the user interface.

Factor:: Values are displayed as float numbers between 0.0 and 1.0.

Percentage:: Values are expressed as a percentage between 0 and 100.

Temporary Editors

When performing certain operations, Blender will open a new window. The behavior of these operations can be configured here.

Render In
: When rendering, the user interface can do any of:

  Keep User Interface::
  : The user interface does not change and the render is computed in the background.

  Maximize Area::
  : A new Image editor is opened as a temporary window in full screen mode.

  Image Editor::
  : The area that is the largest on screen is replaced placed by a temporary Image editor.

  New Window::
  : A new Image editor is opened as a regularly sized temporary window.

File Browser
: When opening files from the computer, the user interface can do any of:

  Maximize Area::
  : A new File Browser editor is opened as a temporary window in full screen mode.

  New Window::
  : A new File Browser editor is opened as a regularly sized temporary window.

Status Bar

Preferences that affect the Status Bar.

Show

Scene Statistics
: Shows information about the data in the active scene.

- **Collection**: The name of the active Collection.
- **Active Object**: The name of the active selected object.

- **Geometry**: Information about the current scene depending on the mode and object type. This can be the number of vertices, faces, triangles, or bones.
- **Objects**: The number of selected objects and the total count of objects.

Scene Duration
: Shows the total amount of time of the playback along with the current frame number and total frame count. The format of the duration text is determined by the Timecode Style.

System Memory
: Shows an estimate of Blender's RAM consumption. On a single-instance single-machine scenario, this estimate provides a measurement against the hardware limit of the machine.

Extensions Updates
: Shows the number of extensions with available updates.

Blender Version
: Shows the version number of Blender that is currently running.

Language

Language
: The language used for translating the user interface (UI). The list is broken up into categories determining how complete the translations are.

Translate
: Tooltips
  : Translates the descriptions when hovering over UI elements.

  Interface
  : Translates all labels in menus, buttons, and panels.

  New Data
  : Translates the names of new data-blocks.

Text Rendering

Anti-Aliasing
: Enable interface text Anti-Aliasing. When disabled, texts are rendered using straight text rendering (filling only absolute pixels).

Subpixel Anti-Aliasing
: Render text for optimal horizontal placement.

Hinting
: Adjust font hinting, controls the spacing and crispness of text display.

Interface Font
: Replacement for the default user interface font.

Mono-space Font
: Replacement for the default mono-space interface font *(used in the Text editor and Python Console)*.

Menus

Open on Mouse Over

Select this to have the menu open by placing the mouse pointer over the entry instead of clicking on it.

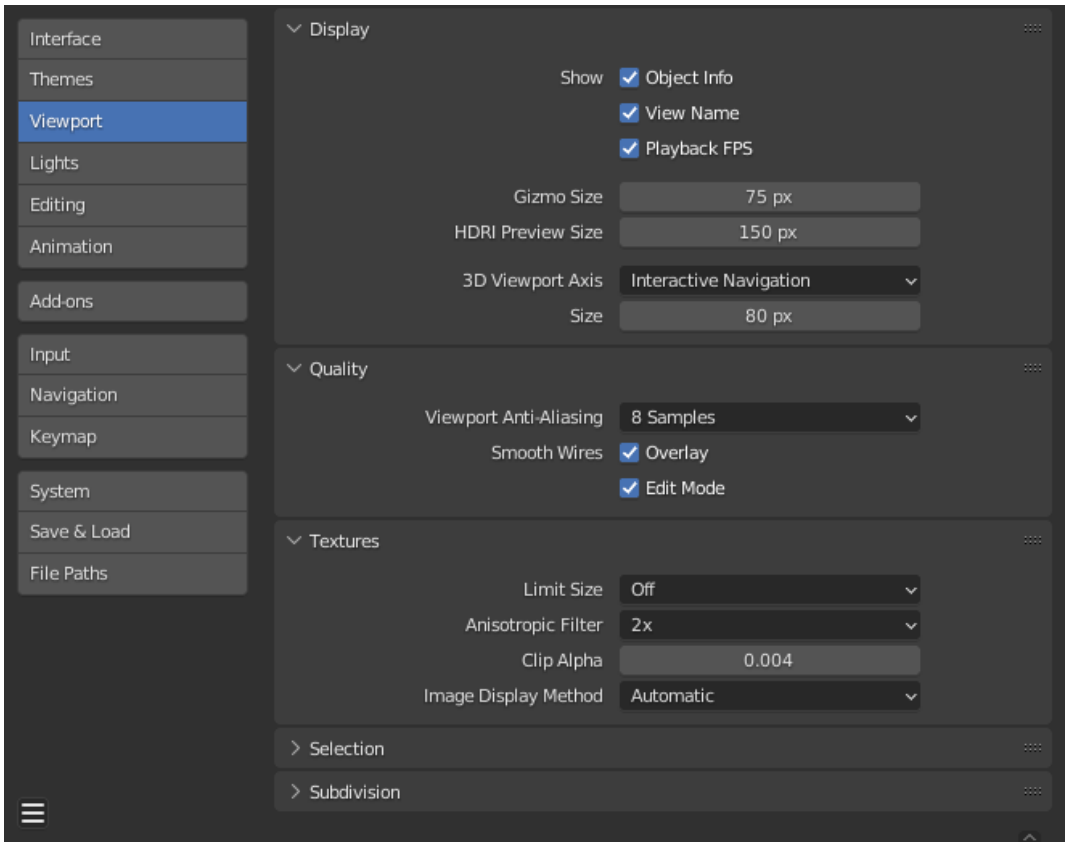
Top Level
: Time delay in 1/10 second before a menu opens (*Open on Mouse Over* needs to be enabled).

Sub Level
: Same as above for sub menus (for example: *File ▸ Open Recent*).

Pie Menus

Animation Timeout

Viewport



Blender Preferences Viewport section.

Display

Text Info Overlay

Object Info

Display the active Object name and frame number at the top left of the 3D Viewport.

View Name

Display the name and type of the current view in the top left corner of the 3D Viewport. For example: “User Perspective” or “Top Orthographic”.

Playback Frame Rate (FPS)

Show the frames per second screen refresh rate while an animation is played back. It appears in the top left of the 3D Viewport, displaying red if the frame rate set cannot be reached.

Frame Rate Samples

Calculate the FPS displayed in the viewport based on an average of the current and previously displayed frames. A value of zero uses the number of frames in 1.0 second.

- More samples represent the average FPS over a longer period of time, however sudden changes to performance result in a more gradual increase/decrease over time.
- Fewer samples shows an FPS which more closely matches the actual performance, however the value may jitter - making the FPS difficult to comprehend.

Gizmo Size

Diameter of the gizmo.

HDRI Preview Size

Diameter of the HDRI sphere overlay.

3D Viewport Axes

Interactive Navigation::

Display the axis as an interactive gizmo. Click sets the viewport to display along this axis and dragging orbits the view.

Simple Axes:: Display simple, less intrusive axis in the viewport.

Brightness

How vivid the colors of the simple axis are.

Off:: Disables the viewport axis.

Size

Diameter of the *3D Viewport Axis* widget.

Fresnel – Edit Mode
: Enable a fresnel effect on edit mesh overlays. It improves shape readability of very dense meshes, but increases eye fatigue when modeling lower poly.

Quality

Viewport Anti-Aliasing
: Control the Anti-Aliasing for higher quality rendering.

Smooth Wires
: Overlay
  : Display overlays with smooth wire, without this wires will be rendered aliased. To increase the visibility you can disable this for Edit Mode specificity (see below), since edges do not blend into other shaded regions.

  Edit Mode
  : Display smooth wire in Edit Mode, without this wires will be rendered aliased.

Textures

Limit Size
: Limit the maximum resolution for pictures used in textured display to save memory. The limit options are specified in a square of pixels (e.g: the option 256 means a texture of 256×256 pixels). This is useful for game engineers, whereas the texture limit matches paging blocks of the textures in the target graphic card memory.

Anisotropic Filtering
: Sets the level of anisotropic filtering. This improves the quality of textures that are rendered at the cost of performance.

Clip Alpha

Clip alpha below this threshold in the 3D Viewport. Note that, the default is set to a low value to prevent issues on some GPUs.

Image Display Method
: Method to render images; the following options are supported:

Automatic::	Automatically use *GLSL* which runs on the GPU for performance but falls back to the CPU for large images which might be slow when loaded with the GPU.
2D Texture::	Uses CPU for display transform and render images as a 2D texture.
GLSL::	Fastest method using GLSL for display transform and render images as a 2D texture.

Selection

GPU Depth Picking
: This option uses an alternative method of picking which uses depth information to select the front-most elements. It is only used for selecting with the cursor (not box select, lasso, circle select, etc.).

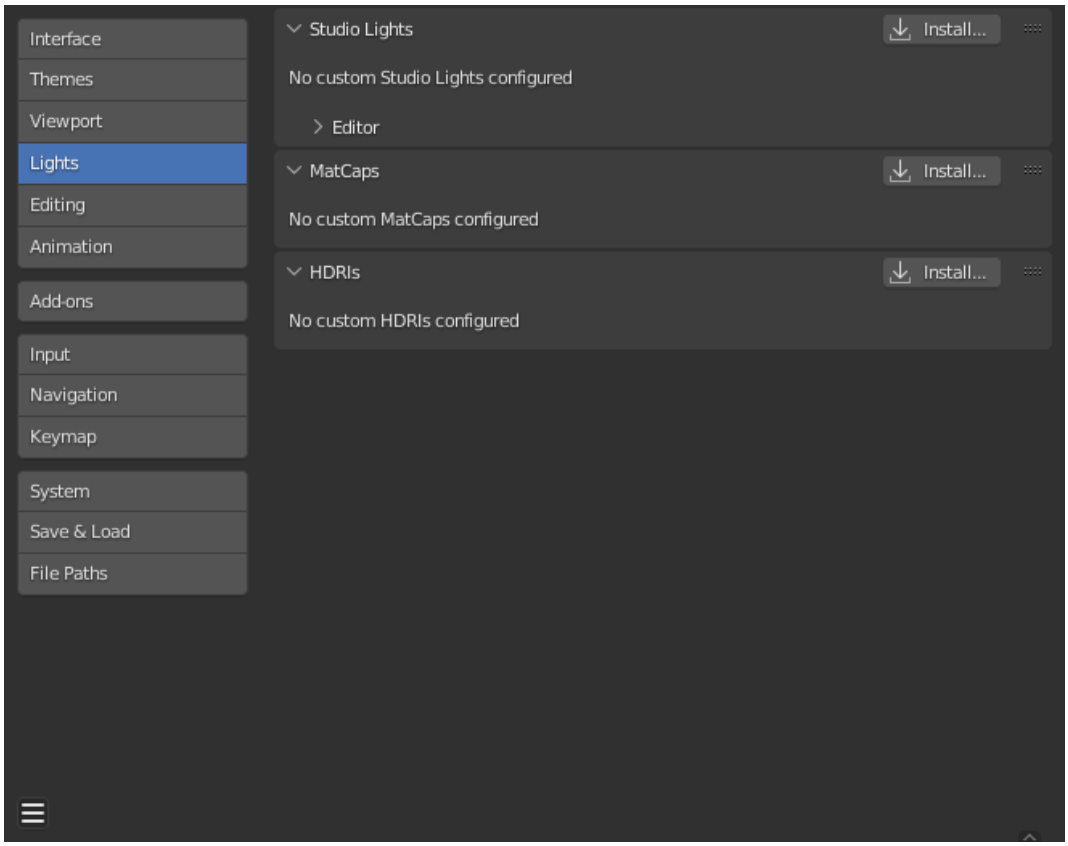
  Performance varies depending on your OpenGL hardware and drivers.

Subdivision

GPU Subdivision
: Under certain circumstances, the GPU will be used to subdivide a mesh with a Subdivision Surface modifier. This typically results in increased subdivision performance.

  This feature is not supported on Qualcomm GPUs on Windows

Lights



Blender Preferences Lights section.

Studio Lights

Studio Lights are used to illuminate the 3D Viewport during *Solid View* and will not be rendered. Unlike lights in the scene, the lighting direction follows the viewport orientation.

Editor

There are up to four virtual light sources.

The Light toggles allow you to enable or disable individual lights. At least one of the four lights must remain enabled for the 3D Viewport. The lights are equal, except for their direction and color. You can control the direction of the lights, as well as their diffuse and specular colors.

Use Light
: Toggles the specific light.

Diffuse
: This is the constant color of the light.

Specular
: This is the highlight color of the light.

Smooth
: Smooth the shading from this light.

  This has the effect of lighting to be less direct.

Direction
: The direction of the light, (see Direction Buttons).

  The direction of the light will be the same as shown at the sphere surface.

Ambient Color
: The color of unlit areas.

MatCaps

This panel manages MatCap image files which can used to light the view when MatCap shading is enabled.

Two kinds of images are supported for MatCaps. Regular image files and multilayered OpenEXR files. When using multilayered OpenEXR files, the layer named “diffuse” will be used as a diffuse pass, the layer named

“specular” will be used as a specular pass. Regular images will be handled as “diffuse” and will not support specular highlighting.

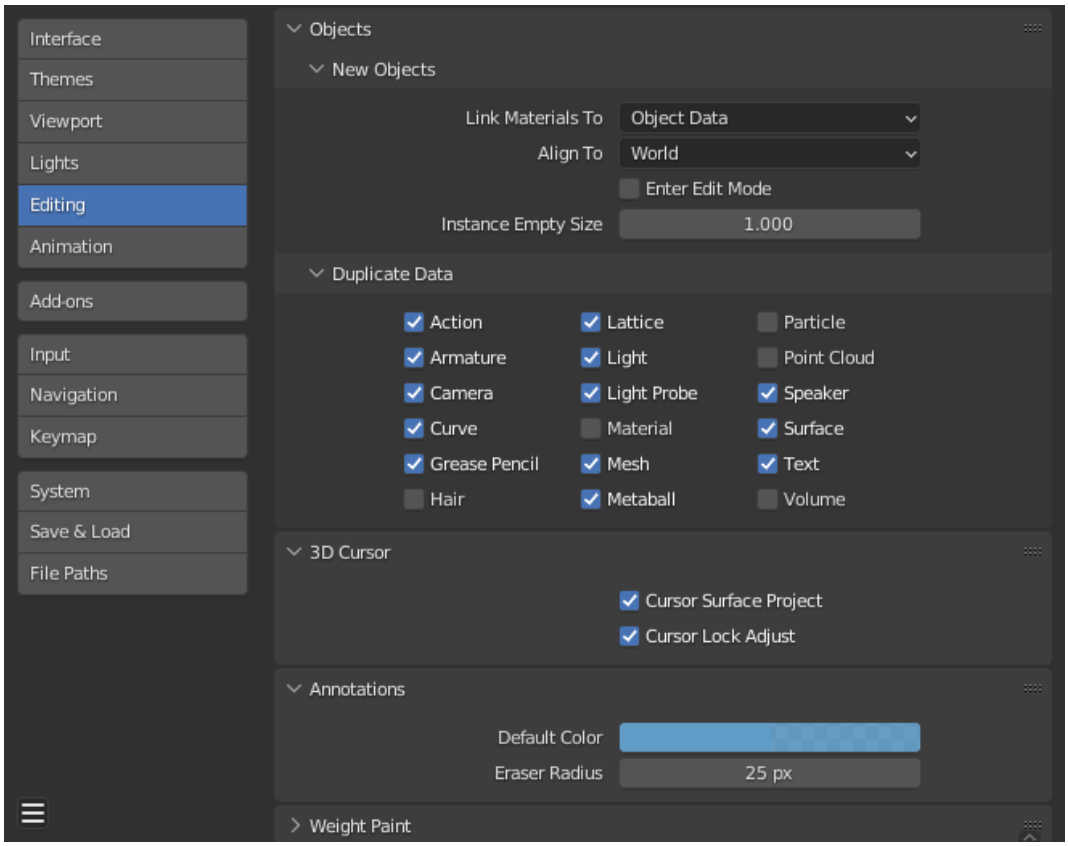
The diffuse pass is multiplied with the base color of the objects and the specular pass is added on top. MatCaps, that only have a diffuse pass tend to look very metallic, with a separate specular pass it is possible to simulate a wider variety of materials.

HDRIs

This panel manages HDRI image files which can be used to light the view when Material Preview or Rendered shading is enabled.

Editing

These preferences control how several tools will interact with your input.



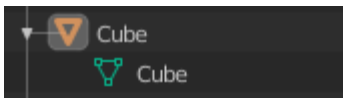
Objects

New Objects

Link Materials To

To understand this option properly, you need to understand how Blender works with Objects. Almost everything in Blender is organized in a hierarchy of data-blocks. A data-block can be thought of as containers for certain pieces of information. For example, the Object data-block contains information about the Object's location, rotation, and scale

while the associated linked Object Data data-block contains information about the mesh.



Example for a mesh.

A material may be linked in two different ways:

Object Data:: Any created material will be created as part of the Object Data data-block.

Object:: Any created material will be created as part of the Object data-block.



A material linked to Object Data (left) and Object (right).

See also

Read more about Blender's Data System.

Align to

World:: New objects align with world coordinates.

View:: New object align with view coordinates.

3D Cursor:: New objects align to the 3D cursor's orientation.

Enter Edit Mode

If selected, Edit Mode is automatically activated when you create a new object.

Instance Empty Size

The display size for empties when a new collection instance is created.

Copy on Duplicate

The checkboxes define what data is copied with a duplicated object and what data remains linked. Any boxes that are checked will have their data copied along with the duplication of the object. Any boxes that are not checked will instead have their data linked from the source object that was duplicated.

For example, if you have *Mesh* checked, then a full copy of the mesh data is created with the new object, and each mesh will behave independently of the duplicate. If you leave the mesh box unchecked then when you change the mesh of one object, the change will be mirrored in the duplicate object.

The same rules apply to each of the checkboxes in the data-block list.

3D Cursor

Cursor – Surface Project
: When placing the cursor by clicking, the cursor is projected onto the surface under the cursor.

Cursor – Lock Adjust
: When the viewport is locked to the cursor, moving the cursor avoids the view *jumping* based on the new offset.

Annotations

Default Color
: The default color for new Annotate layers.

Eraser Radius
: The size of the eraser used with the Annotate Tool.

See also

Read more about Annotations.

Weight Paint

Mesh skin weighting is used to control how much a bone deforms the mesh of a character. To visualize and paint these weights, Blender uses a color ramp (from blue to green, and from yellow to red). Enabling the checkbox will enable an alternate map using a ramp starting with an empty range. Now you can create your custom map using the common color ramp options. For detailed information see the Color ramps page.

Grease Pencil

Manhattan
: The minimum number of pixels the mouse should have moved either horizontally or vertically before the movement is recorded. Decreasing this should work better for curvy lines.

Euclidean
: The minimum distance that mouse has to travel before movement is recorded.

See also

Read more about Grease Pencil.

Text Editor

Auto Close Character Pairs
: Automatically insert the corresponding character to close an expression when typing characters such as quotes, brackets, braces, or parentheses.

Node Editor

Auto-Offset

Automatically offset the following or previous nodes in a chain when inserting a new node. See Auto-Offset for more information.

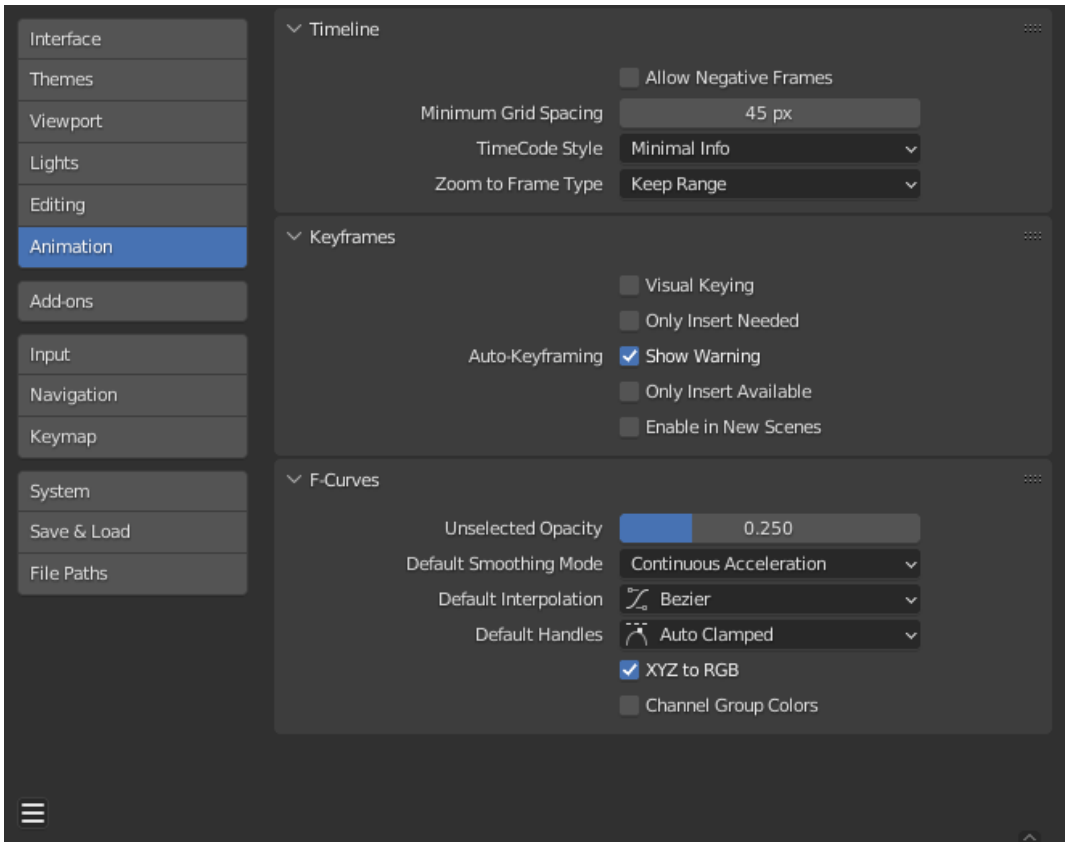
Auto-offset Margin
: Margin to use for offsetting nodes.

Miscellaneous

Sculpt Overlay Color
: Defines a color to be used in the inner part of the brushes circle when in Sculpt Mode, and it is placed as an overlay to the brush, representing the focal point of the brush influence. The overlay color is visible only when the overlay visibility is selected (clicking at the *eye* to set its visibility), and the transparency of the overlay is controlled by the alpha slider located at the *Tool tab* ▸ *Display panel* in the Sidebar.

Animation

The *Animation* section lets you manage settings related to Animation. This includes how editors look and also some different tools properties.



Blender Preferences Animation section.

Timeline

These settings control things in the Timeline.

Allow Negative Frame
: Playback and animations can occur during negative frame ranges.

Minimum Grid Spacing

The minimum number of pixels between grid lines.

Timecode Style
: Format of timecodes displayed when not displaying timing in terms of frames.

  Minimal Info:: Most compact representation, uses '+' as separator for sub-second frame numbers, with left and right truncation of the timecode as necessary.
  SMPTE (Full):: Full SMPTE timecode (format is HH:MM:SS:FF).
  SMPTE (Compact):: SMPTE timecode showing minutes, seconds, and frames only – hours are also shown if necessary, but not by default.
  Compact with Decimals:: Similar to SMPTE (Compact), except that the decimal part of the second is shown instead of frames.
  Only Seconds:: Direct conversion of frame numbers to seconds.

Zoom to Frame Type
: Defines what time range (around the cursor) will be displayed when the *View Frame* `Numpad0` is performed.

  Keep Range:: The currently displayed time range is preserved.
  Seconds:: The number of seconds specified in the *Zoom Seconds* field will be shown around the cursor.
  Keyframes:: The number of animation keyframes defined in the *Zoom Keyframes* field will be shown around the cursor.

Keyframes

These settings control Keyframes which are the building blocks for animations.

Default Key Channels

Which channels to insert keys at when no keying set is active.

Location:: Inset keyframes for an object's Location.
Rotation:: Inset keyframes for an object's Rotation.
Scale:: Inset keyframes for an object's Scale.
Rotation Mode:: Inset keyframes for an object's Rotation Mode.
Custom Properties::
Inset keyframes for Custom Properties.

Only Insert Needed

This will only insert keyframes if the value of the property is different.

Manual:: When keying manually, skip inserting keys that don't affect the animation.
Auto:: Auto-Keying will skip inserting keys that don't affect the animation.

Keyframing – Visual Keying

When an object is using constraints, the object property value does not actually change. *Visual Keying* will add keyframes to the object property, with a value based on the visual transformation from the constraint.

Auto-Keyframing

Enable in New Scenes

Enables *Auto Keyframe* by default for new scenes.

Show Warning

Displays a warning at the top right of the *3D Viewport*, when moving objects, if *Auto Keyframe* is on.

Only Insert Available

This will only add keyframes to channels of F-Curves that already exist.

See also

Learn more about Auto-Keyframing.

F-Curves

These settings control how F-Curves look and their default behavior.

Unselected Opacity
: Controls the opacity of unselected F-Curves against the background of the Graph Editor.

Default Smoothing Mode
: Controls the behavior of automatic curve handles for newly created F-Curves.

Default Interpolation
: Controls the default Interpolation for newly created keyframes.

Default Handles
: Controls the default Handle for newly created F-Curves.

XYZ to RGB
: Color for X, Y, or Z animation curves (location, scale or rotation) is the same as the color for the X, Y, and Z axis.

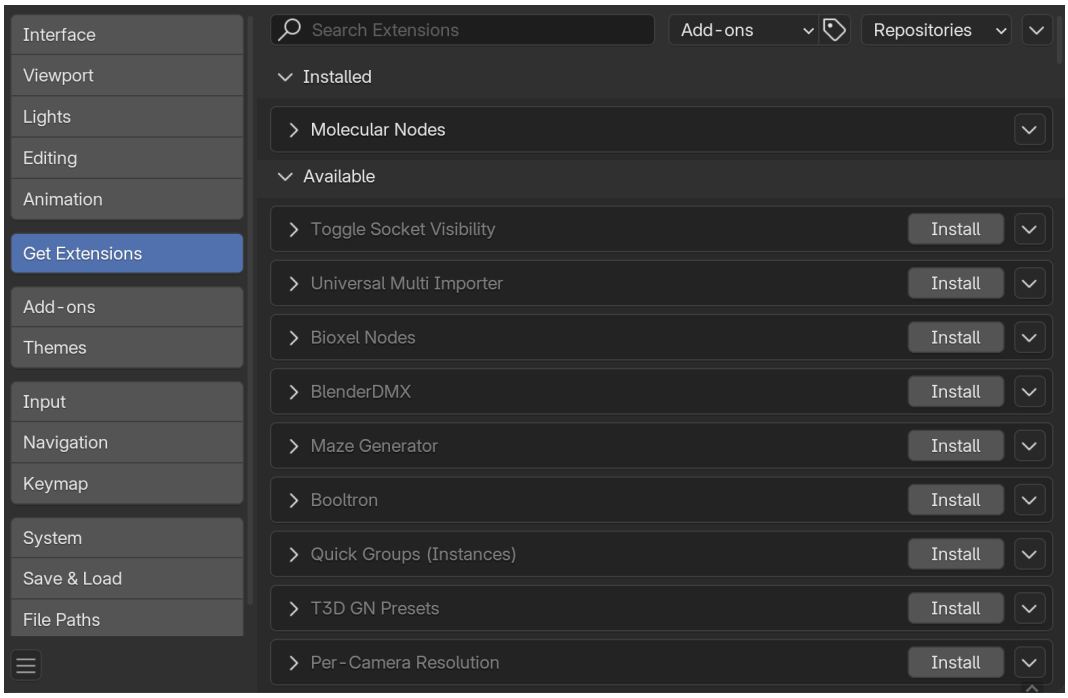
Channel Group Colors
: Display groups and channels with colors matching their corresponding groups.

Only Show Selected F-Curve Keyframes
: Only shows the keyframes markers on the selected curves.

Use High Quality Display
: Display F-Curves using Anti-Aliasing and other effects (disable for a better performance).

Get Extensions

The *Get Extensions* section lets you install and manage extensions preferences.



Blender Preferences Extensions section.

See also

To learn about extensions and how to create them, refer to the Extensions page.

Installing Extensions

There are different ways to install an extension:

Install from the Website

Drag the installation URL into Blender.

Install from Blender
: Search for the extension name and click on Install.

Install from Disk
: Use the drop-down menu in the top right, or drag-and-drop an extension `.zip` package into Blender.

Note

Any installed extension can be removed. This is a permanent change, though. To stop an extension temporarily, it is better to Disable it instead.

Updating Extensions

You need to manually check for available updates. Once an update is found, Blender will let you update any of the available extensions.

The current available version of an extension on the repository will always be considered the latest version.

Enable/Disable

Once an extension is installed it can be disabled (or re-enabled) as part of the user preferences. Some extension types do not support this, and will always be shown as enabled.

Tip

If the Add-on does not activate when enabled, check the Console window for any errors that may have occurred.

Extension Settings

Visit Extensions Platform
: Opens *extensions.blender.org <https://extensions.blender.org/>* in a web browser.

Refresh Remote
: Manually check the online repositories for available updates.

Refresh Local
: Scan extension & legacy add-ons for changes to modules & meta-data (similar to restarting). Any issues are reported as warnings

Install Available Updates
: Update all the extensions that have an update available.

Install from Disk
: Install an extension from a `.zip` package. This is installed to a Local Repository and no updates will be available.

  This can also be used to install legacy Add-ons, for more information see: Installing Legacy Add-ons.

Filter by Type

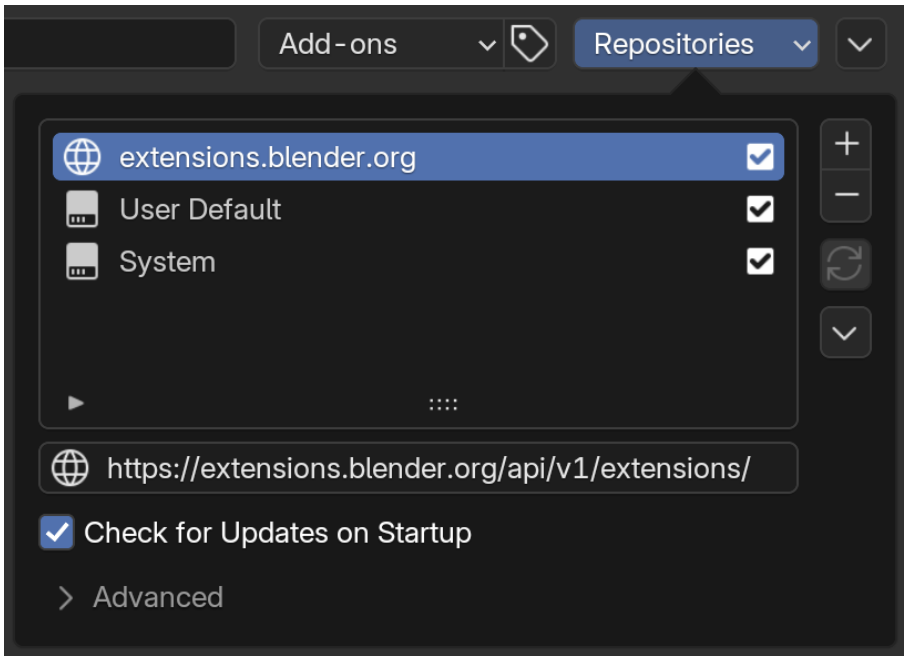
Or show only extensions of a single type:

Add-ons:: Only show add-ons.
Themes:: Only show themes.

Repositories

By default Blender has a Remote Repository pointing towards the Official Blender Extensions Platform and two Local Repositories.

In the cases where more repositories are needed (e.g., to access third party extension platforms), new repositories can be added.



Repositories.

To add new repositories click on the + icon:

Add Remote Repository
: Add a repository from a URL.

Add Local Repository
: Add a repository which will be managed by the user (to be used with Install from Disk).

To remove repositories click on the - icon:

Remove Repository
: Remove an extension repository.

Remove Repository & Files
: Remove a repository and delete all associated files when removing.

These changes are permanent and cannot be reversed.

Remote Repository

Remote repository with support for listing and updating extensions.

Options:

Check for Updates on Startup
: Allows Blender to check for updates upon launch. When updates are available a notification will be visible on the status bar.

Access Token
: Personal access token, may be required by some repositories.

Local Repository

A repository managed manually by the users.

There are two types of local repositories. By default new local repositories are added as User repositories. This is what you want most of the time.

After creating a repository they can be changed in the Advanced options to have a source System. These repositories are intended to bundle extensions with Blender, to make it portable.

Add-ons

The Add-ons section lets you manage secondary scripts, called “Add-ons” that extends Blender’s functionality. Most of the time you can get add-ons as part of the Extensions system.

In this section you can search, install, enable and disable Add-ons.

Tip

If the Add-on does not activate when enabled, check the Console window for any errors that may have occurred.

Filtering Add-ons

Search Add-ons
: Blender comes with some preinstalled Add-ons already, ready to be enabled. But you can also add your own, or any interesting ones you find on the web.

Enabled Add-ons Only
: Shows only enabled add-ons for the current Category.

Add-on Tags
: Add-ons are assigned categories by what areas of Blender they affect.

Add-on Settings

Refresh Local
: Scan extension & legacy add-ons for changes to modules & meta-data (similar to restarting). Any issues are reported as warnings.

Install from Disk

Install an extension from a `.zip` package. This is installed to a Local Repository and no updates will be available.

This can also be used to install legacy Add-ons, for more information see: Installing Legacy Add-ons.

Enabling & Disabling Add-ons

To enable or disable an add-on check or uncheck the box to the right of the add-ons.

The add-on functionality should be immediately available.

Add-on Information

You can click the arrow at the left of the add-on box to see more information, such as its location, a description and a link to the documentation. Here you can also find a button to report a bug specific of this add-on.

Add-on Preferences

Some add-ons may have their own preferences which can be found in the *Preferences* section of the add-on information box.

Some add-ons use this section for example to enable/disable certain functions of the add-on. Sometimes these might even all default to off. So it is important to check if the enabled add-on has any particular preferences.

Installing Legacy Add-ons

To install legacy add-ons, click the *Install from Disk* menu item and select the add-on's `.py` file (if it has only one such file) or its `.zip` file.

The add-on will not be automatically enabled after installation; click the checkbox to do that.

Refresh
: Scans the Add-on Directory for new add-ons.

Tip

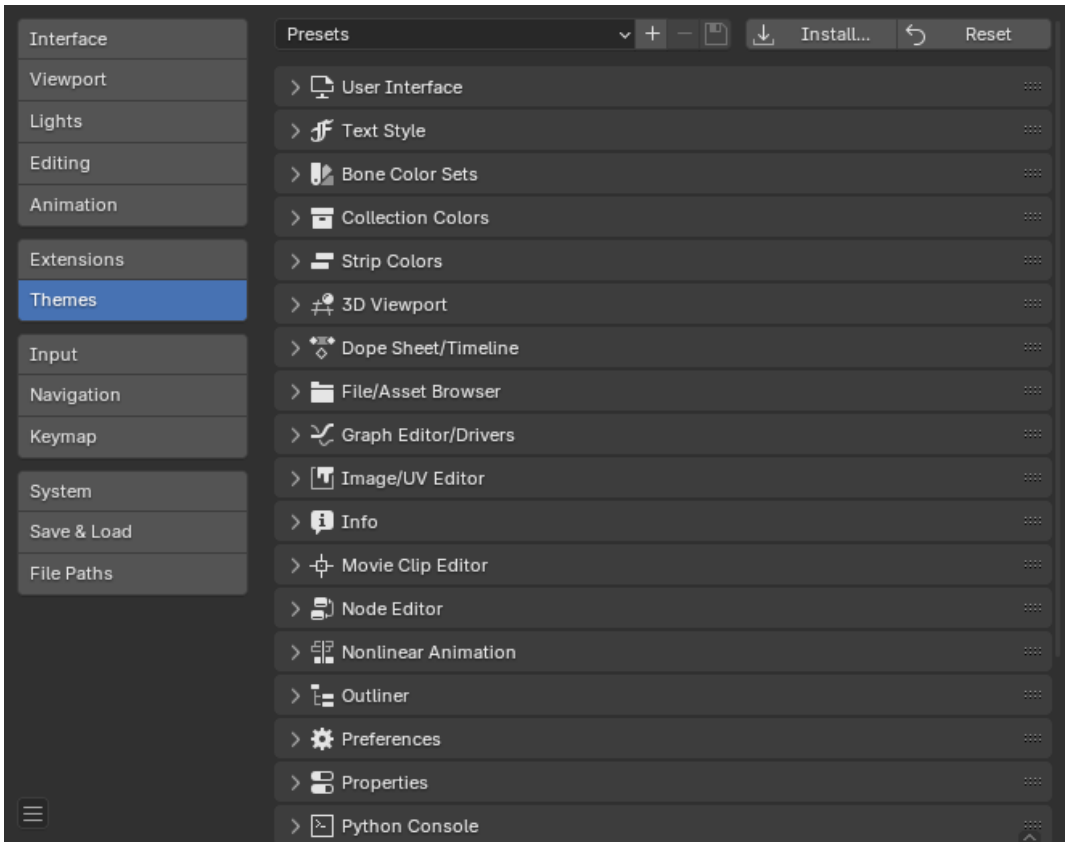
While this screen doesn't allow installing a folder-based addon with loose `.py` files, you can still do so by adding it as a Script Directory:

1. Create an empty directory in a location of your choice (e.g. `my_scripts`).
2. Add a subdirectory under `my_scripts` called `addons` (it *must* have this name for Blender to recognize it).
3. Place your addon folder inside this `addons` folder.
4. Open the *File Paths* section of the *Preferences*.
5. Add a *Script Directories* entry pointing to your script folder (e.g. `my_scripts`).
6. Save the preferences and restart Blender for it to recognize the new add-on location.

The add-ons in this folder will automatically become available; all you need to do is enable them.

Themes

The *Themes* section allows you to customize interface appearance and colors.



The colors for each editor can be set separately by simply selecting the editor you wish to change in the multi-choice list at the left, and adjusting colors as required. Notice that changes appear in real-time on your screen. In addition, details such as the dot size in the *3D Viewport* or the *Graph Editor* can also be changed.

Preset Management

Theme Presets

Select the Theme from a list of predefined Themes.

You add a custom theme to the preset list by `LMB` on the *Add* button `+`.

You remove a custom theme from the preset list by `LMB` on the *Remove* button `−`.

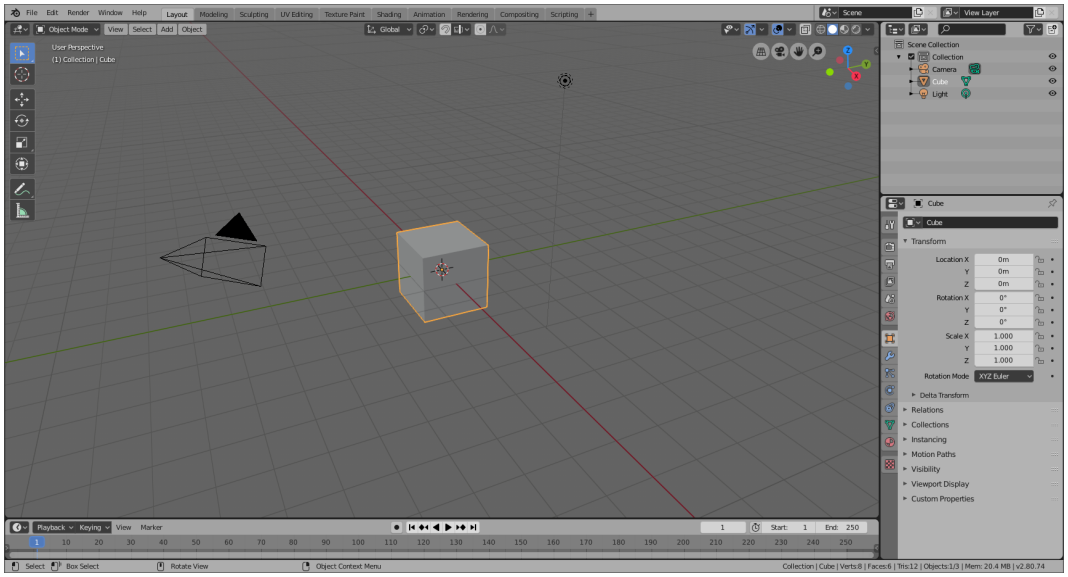
You save a custom theme in the preset list by `LMB` on the *Save* button.

This will save the theme to an XML file in the `./scripts/presets/interface_theme/` subdirectory of one of the configuration directories.

Install
: Load and apply a Blender XML theme file and add it to the list of theme presets.

Reset
: Reset to the default theme colors.

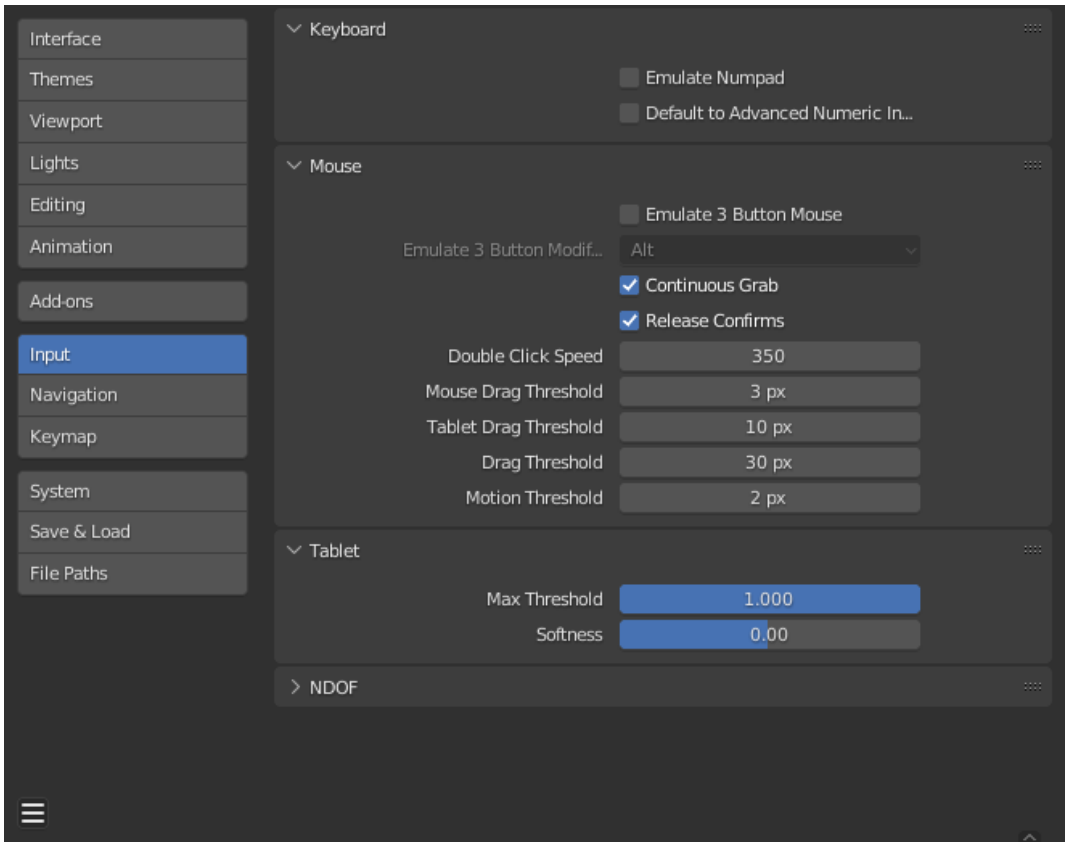
Blender comes bundled with a small selection of themes.



This is an example of the theme *Blender Light*.

Input

In the Input preferences, you can customize how Blender reacts to the mouse and keyboard as well as define your own keymap.



Keyboard

Emulate Numpad

The Numpad keys are used quite often in Blender and are not assigned to the same action as the regular number keys. If you have a keyboard without a Numpad (e.g. on a laptop), you can tell Blender to treat the standard number keys as Numpad keys by checking *Emulate Numpad*.

Default to Advanced Numeric Input

For transform mode, default to Advanced Mode, otherwise Simple Mode is used.

Mouse

Emulate 3 Button Mouse

Blender can be configured to work with pointing devices which do not have an `MMB`. The functionality of the three mouse buttons by holding `Alt-LMB`.

Mouse/Keyboard combinations referenced in this manual can be expressed with the combinations shown in the table. For example:

`MMB` drag becomes `Alt-LMB` drag for example.

Warning

This option prevents certain features from being accessed, since `Alt-LMB` is used for some operations.

- Modifying multiple items values at once (objects, bones… etc).
- Deselecting edge/face rings in Edit Mode.
- Detaching node links.
- Moving the Compositor background image.

Some touchpads support three-finger tap for middle mouse button, which may be an alternative to using this option.

Modifier

The modifier key to press to emulate the middle mouse keybindings. This option is unsupported on Microsoft Windows.

Alt:: Use the `Alt` key to emulate the middle mouse button.

OSKey:: Use the `OSKey` to emulate the middle mouse button.

This has the advantage that it doesn't conflict with existing `Alt-MMB` shortcuts, noted above.

Continuous Grab
: This feature is used to prevent the problem where an action such as moving objects or panning a view, is limited by your screen bounds.

  This is done by warping the mouse within the view.

  Note

  Cursor warping is only supported by *relative* input devices (mouse, trackball, trackpad).

  Graphics tablets, however, typically use *absolute* positioning, this feature is disabled when a tablet is being used.

  This is detected for each action, so the presence of a tablet will not disable *Continuous Grab* for mouse cursor input.

Release Confirms
: Dragging `LMB` on an object will move it. To confirm this (and other) transform, an `LMB` is necessary by default. When this option is activated, the release of `LMB` acts as confirmation of the transform.

Double Click Speed
: The time in milliseconds to trigger a double click.

Mouse Drag Threshold
: The number of pixels that a User Interface element has to be moved before it is recognized by Blender, values below this will be detected as click events.

Tablet Drag Threshold
: The drag threshold for tablet events.

Drag Threshold

The drag threshold for non mouse/tablet events (keyboard or NDOF for example).

This affects Pie Menu on Drag keymap preference.

Motion Threshold
: The number of pixels the cursor must be moved before the movement is registered. This is helpful for tablet pens that are a lot more difficult to keep still, then this could help to reduce stuttering of the cursor position.

  Note

  Unlike the click/drag distinction, this is used to detect small movements for example, picking selection cycles through elements near the cursor. Once the cursor moves past this threshold, selection stops cycling and picks the closest item.

Touchpad

Note

This panel is available on Windows, macOS, and Linux with Wayland.

Multi-touch Gestures
: Use multi-touch gestures for navigation with touchpad, instead of scroll wheel emulation. For more detail on supported gestures, see Configuring Peripherals.

Scroll Direction
: The direction scrolling responds to the scroll gestures.

  Only available on Linux using Wayland.

  Traditional:: Scrolls content down when gestures move up.
  Natural:: Scrolls content up when gestures move up.

Tablet

Tablet API (Windows only)
: Select the native Windows Ink or older Wintab system for pressure sensitivity. Blender automatically selects the API for your operating system and tablet, however in case of problems this can be set manually. You may need to restart Blender for changes to take affect.

Max Threshold
: Amount of pressure required to achieve full intensity.

Softness
: Controls how the softness of the low pressure response onset using a gamma curve.

NDOF

These preferences control how an NDOF device interacts with the 3D Viewport. These preferences can also be accessed using the `NDOFMenu` button on the NDOF device to open a pop-up menu to adjust the settings directly from the 3D Viewport.

Pan Sensitivity
: The overall sensitivity for panning in the 3D Viewport.

Orbit Sensitivity
: The overall sensitivity for orbiting in the 3D Viewport.

Deadzone
: The threshold for the amount of movement needed from the device's rest position for Blender to interrupt that movement.

Navigation
: Navigation style for the viewport.

Free::	Uses the full 6-degrees of freedom.
Orbit::	Orbit about the view center.

Rotation
: Rotation style for the viewport.

Turntable:: Rotates the view keeping the horizon horizontal.
Trackball:: Is less restrictive, allowing any orientation.

Show Navigation Guide
: Display the pivot point and axis during rotation.

Invert Zoom
: Zoom using opposite direction.

Lock Camera Pan/Zoom
: Pan/zoom the camera view instead of leaving the camera view when orbiting.

Pan – Swap Y and Z Axes
: Pan using up/down on the NDOF devices instead of forward/backwards.

Invert Axis Pan
: Reverses the panning axis on the selected axes.

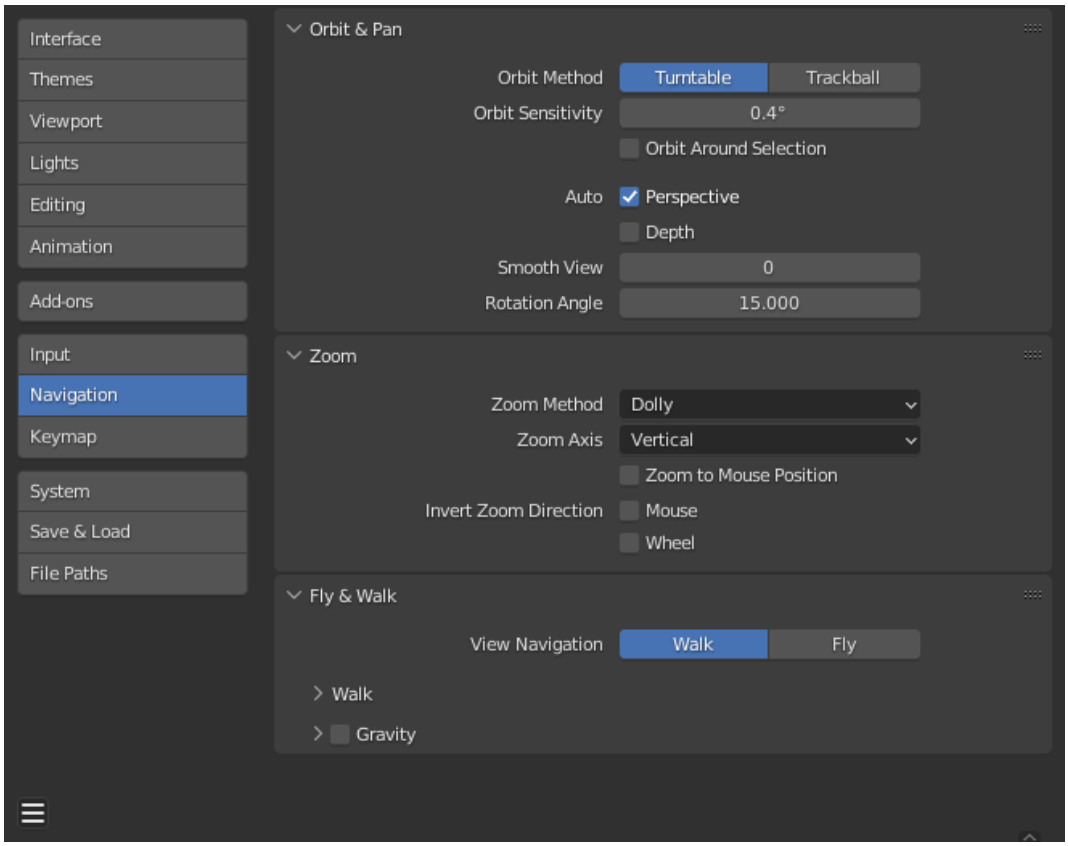
Orbit
: Reverses the orbit axis on the selected axes.

Fly/Walk
: Settings to control how the NDOF device is used while using Walk/Fly Navigation.

  Lock Horizon
  : Keeps the horizontal axis level while flying.

  Helicopter Mode
  : Moves the 3D Viewport up or down when moving the NDOF device up/down.

Navigation



Blender Preferences navigation section.

Orbit & Pan

Orbit Method

Choose you are preferred method of interactively rotating the 3D Viewport.

Turntable:: Rotates the view keeping the horizon horizontal. This behaves like a potter's wheel or record player where you have two axes of rotation available, and the world seems to have a better definition of what is "Up" and "Down" in it.

The drawback to using the *Turntable* style is that you lose some flexibility when working with your objects. However, you gain the sense of "Up" and "Down" which can help if you are feeling disoriented.

Trackball:: Is less restrictive, allowing any orientation.

Orbit Sensitivity

Adjusts the reactivity/speed of orbiting in the 3D Viewport. This setting works differently depending on what *Orbit Method* is used:

- Turntable: *Orbit Sensitivity* controls the amount of rotation per-pixel to control how fast the 3D Viewport rotates.
- Trackball: *Orbit Sensitivity* as a simple factor for how fast the 3D Viewport rotates.

Orbit Around Selection

The selection center becomes the rotation center of the viewport. When there is no selection the last selection will be used.

The method used to calculate the center depends on the the current mode:

- Object mode uses the selections bounding box center.
- Edit & pose mode use the selected elements center.
- Paint modes use the center of the last brush stroke.

Note

While this may seem like ideal behavior, it can be inconvenient for larger objects such as a terrain mesh, where the center is not necessarily a point of interest.

Auto – Perspective

When enabled, the view switches to Perspective when orbiting the view, and to Orthographic when aligning to an axis (Top, Side, Front, Back, etc.).

When disabled, this switching needs to be done manually.

Auto – Depth
: Use the depth under the mouse to improve view pan, rotate, zoom functionality. Useful in combination with *Zoom To Mouse Position*.

Smooth View
: Time (in milliseconds) the animation takes when changing views (Top/Side/Front/Camera…). Reduce to zero to remove the animation.

Rotation Angle
: Rotation step size in degrees, when `Numpad4`, `Numpad6`, `Numpad8`, or `Numpad2` are used to rotate the 3D Viewport.

Zoom

Zoom Method
: Choose your preferred style of zooming in and out, when using interactive zoom.

Scale:: *Scale* zooming depends on where you first click in the view. To zoom out, move the cursor to the area center. To zoom in, move the cursor away from the area center.

Continue:: The *Continue* zooming option allows you to control the speed (and not the value) of zooming by moving away from the initial cursor position. Moving up from the initial click point or to the right will zoom out, moving down or to the left will zoom in. The further away you move, the faster the zoom movement will be. The directions can be altered by the *Vertical* and *Horizontal* radio buttons and the *Invert Zoom Direction* option.

Dolly:: *Dolly* zooming works similarly to *Continue* zooming except that zoom speed is constant.

Zoom Axis
: The axis of the mouse to use for zooming.

Vertical::	Moving up zooms out and moving down zooms in.
Horizontal::	Moving left zooms in and moving right zooms out.

Zoom to Mouse Position
: When enabled, the mouse pointer position becomes the focus point of zooming instead of the 2D window center. Helpful to avoid panning if you are frequently zooming in and out.

> **Tip**
>
> This is useful in combination with Auto Depth to quickly zoom into the point under the cursor.

Invert Zoom Direction – Mouse
: Inverts the Zoom direction for *Dolly* and *Continue* zooming.

Invert Zoom Direction – Wheel
: Inverts the direction of the mouse wheel zoom.

Fly & Walk

View Navigation
: The default mode for interactive first person navigation.

  See Fly/Walk Navigation.

Walk

Reverse Mouse
: Inverts the mouse's Y movement.

Mouse Sensitivity
: Speed factor for when looking around, high values mean faster mouse movement.

Teleport Duration
: Interval of time warp when teleporting in navigation mode.

Walk Speed
: Base speed for walking and flying.

Speed Factor
: The multiplication factor for the speed boost.

Gravity

Simulates the effect of gravity when walking.

View Height
: The distance from the ground floor to the camera when walking.

Jump Height
: The maximum height of a jump.

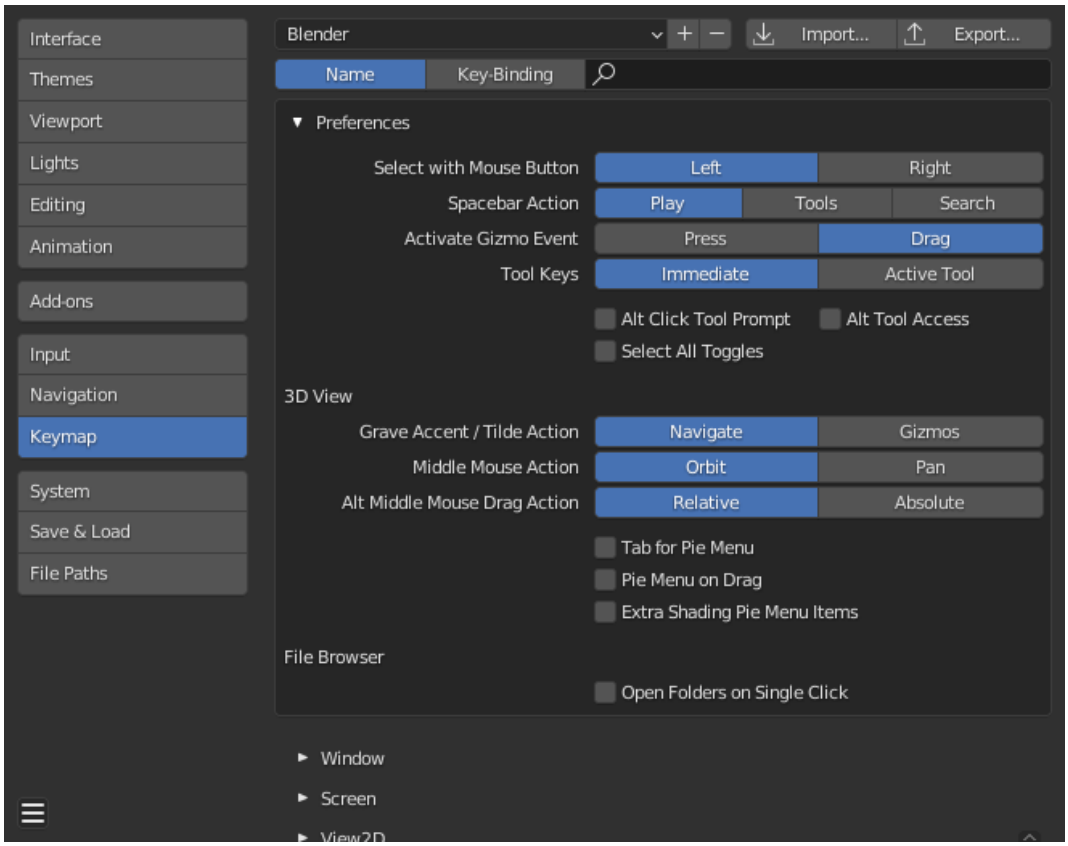
Keymap

The keymap editor lets you adjust your keymap via:

Presets:: Predefined keymaps which come with Blender and can be added to.

Preferences:: Keymaps may define their own preferences to change the functionality or add additional key bindings.

Key Map Items:: You may add/remove/edit individual keymap entries.



Blender Preferences Keymap section.

Preset Management

Keymap Presets

Select the keymap from a list of predefined keymaps.

You add a custom keymap configuration to the preset list by `LMB` on the *Add* button `+`.

You remove a custom keymap configuration from the preset list by `LMB` on the *Remove* button `−`.

Import
: Importing opens a File Browser to select a `.py` file to add to the list of keymap presets.

Export
: Saves the current keymap configuration as a preset others may use.

  All Keymaps
  : When disabled, only the keymaps and categories that have been modified by the user will be exported. In addition, add-ons may register keymaps to their respective functions, however, these keymaps are not exported unless changed by the user. This exported file may be thought of as a *"keymap delta"* instead of a full keymap export.

    When enabled, the entire keymap is written.

Filtering

Filter Type

Name:: Search the keymap item by the operator name it runs.

Key Binding:: Search the keymap item by the key used to activate it.

> **Hint**
> You could for example search with `Ctrl Shift C` for keymap items that use all these keys.

Search
: The text to search (leave blank to disable).

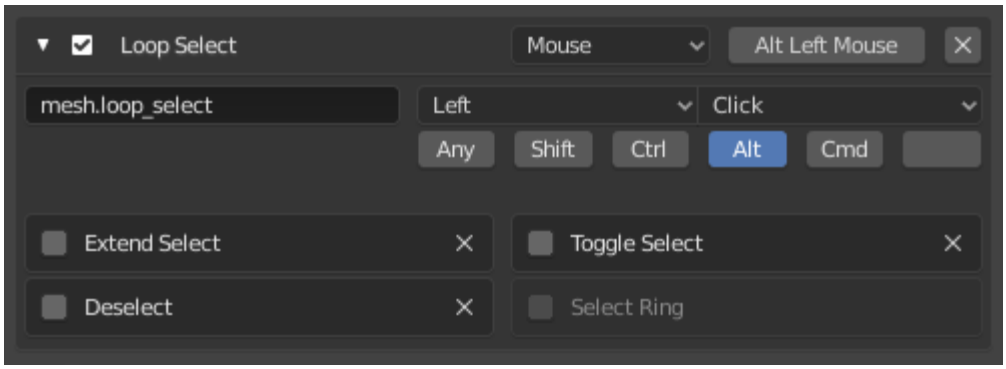
Preferences

Keymaps may define their own preferences, these are predefined adjustments to the keymap you can make without having to manually adjust individual keymap items which can cause problems with newer Blender Versions.

See the default keymap preferences for options available in the default keymap.

Editor

The Keymap editor lets you change the default hotkeys. You can change keymaps for each of Blender's editors.



Keymap editor.

Usage

1. Select the keymap you want to change and click on the white arrows to open up the keymap tree.
2. Select which Input will control the function.
3. Change hotkeys as you want. Just click on the shortcut input and enter the new shortcut.

Active
: Uncheck to disable this keymap item.

Map Type
: **Keyboard::** Single hotkey or key combination.
: **Mouse::** Actions from mouse buttons, tablet or touchpad input.
: **NDOF::** Movement or button from a 3D mouse (NDOF) device.
: **Tweak::** Mouse click and drag *(optionally map drag direction to different actions)*.
: **Text Input::** Use this function by entering a text.
: **Timer::** Used to control actions based on a time period. E.g. by default, *Animation Step* uses "Timer 0", *Smooth View* uses "Timer 1".

Operator ID Name
: The identifier for the operator to call.

> **Hint**
>
> See `bpy.ops` for a list of operators (remove the `bpy.` prefix for the identifier).

Event
: Type
: The key or button that activates this keymap item (depending on the map type).

: Value
: The action (such as press, release, click, drag, etc.), (depending on the map type).

: Modifier
: Additional keys to hold (such as `Ctrl`, `Shift`, `Alt`).

Operator Properties

Changes to the defaults properties this operator is activated with

See also

Keymap Customization for more information on keymap editing.

Restoring

If you want to restore the default settings for a keymap, just click on the *Restore* button at the top right of this keymap.

Tip

Instead of deleting the default keymap to create your custom one, you can just add a new *Preset* for both the mouse and keyboard.

Known Limitations

Blender Versions

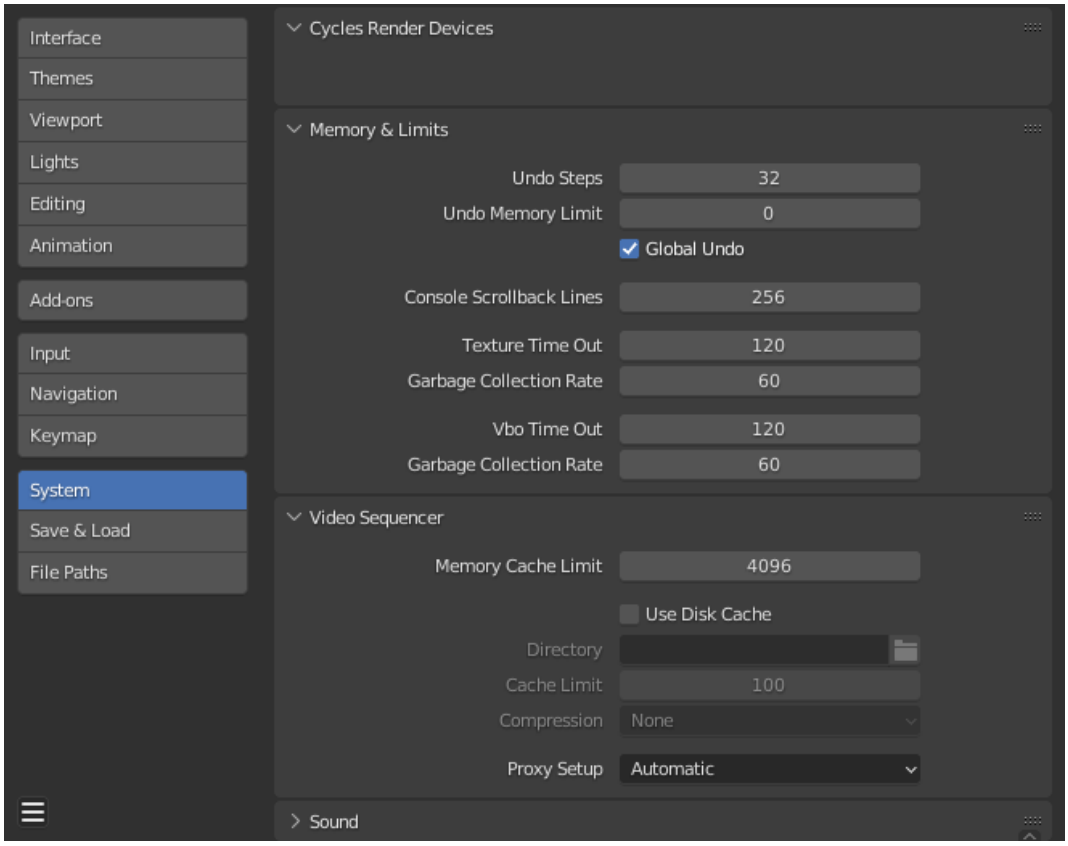
A problem with modifying your own keymap is newer Blender versions key change the way tools are accessed, breaking your customized keymap.

While the keymap can be manually updated, the more customizations you make, the higher the chance of conflicts in newer Blender versions is.

System

The *System* section allows you to set graphics card options, memory limits & sound settings.

If your hardware does not support some of the options described on this page, then they will either not be displayed or be corrected on startup.



Preferences System section.

Cycles Render Device

Changes the computing device the Cycles render engine uses to render images. Cycles can use either the CPU or certain GPUs to render images, for more information see the GPU Rendering page.

None:: When set to *None* or when the only option is *None*: the CPU will be used as the computing device for Cycles.

CUDA:: If the system has a compatible Nvidia CUDA device, it will be available as an option for rendering with Cycles.

OptiX:: If the system has a compatible Nvidia OptiX device, it will be available as an option for rendering with Cycles.

HIP:: If the system has a compatible AMD HIP device, it will be available as an option for rendering with Cycles.

oneAPI:: If the system has a compatible Intel oneAPI device, it will be available as an option for rendering with Cycles.

Metal:: If the system has a compatible Apple Metal device, it will be available as an option for rendering with Cycles.

Distribute Memory Across Devices

Allocates resources across multiple GPUs rather than duplicating data, effectively freeing up space for larger scenes. Note that in order for this option to be available, the GPUs must be connected together with a high bandwidth communication protocol.

Currently only NVLink on Nvidia GPUs is supported.

Embree on GPU

Enables the use of hardware ray tracing on Intel GPUs, providing better overall performance.

Only supported with oneAPI rendering devices..

HIP RT (Experimental)

Speeds up rendering by enabling AMD hardware ray tracing on RDNA2 and above, with shader fallback on older cards. This feature is experimental and some scenes may render incorrectly.

This feature is only available when using a *HIP* render device.

MetalRT
: MetalRT for ray tracing uses less memory for scenes which use curves extensively, and can give better performance in specific cases.

Off::	Disable MetalRT (uses BVH2 layout for intersection queries).
On::	Enable MetalRT for intersection queries.
Auto::	Automatically pick the fastest intersection method.

Operating System Settings

Make this installation your default Blender (MS-Windows & Linux only).

On Linux, if Blender is installed from a package manager such as Snap, file association is handled by the package manager.

Register
: Make the currently in use Blender installation the default for generating thumbnails and the default for opening blend-files.

Unregister
: Remove file association & thumbnailer.

For All Users
: Register Blender for all users, requires escalated privileges.

Linux Registration

Files are setup files under: `/usr/local` for all users, otherwise `~/.local` is used.

- A desktop file & icon is installed so the application is available in launchers.
- A file association for `*.blend` is setup.

- The thumbnailer is installed so blend-file thumbnails will be shown in file managers (**For All Users** only).

Network

Allow Online Access
: Allow Blender to access the internet.

  Add-ons that follow this setting will only connect to the internet if enabled. However, Blender cannot prevent third-party add-ons from violating this rule.

Time Out
: The time (in seconds) that online operations may wait before timing out.

  Use the systems default when zero.

Connection Limit
: The maximum number of simultaneous connections an online operation may make.

  Do not limit the number of connections when zero.

Memory & Limits

Undo Steps
: Number of Undo steps available.

Undo Memory Limit
: Maximum memory usage in Mb (0 is unlimited).

Global Undo
: This enables Blender to save actions done when you are **not** in *Edit Mode*. For example, duplicating objects, changing panel settings or switching between modes.

Warning

While disabling this option does save memory, it stops the Adjust Last Operation panel from functioning, also preventing tool options from being changed in some cases. For typical usage, its best to keep this enabled.

See also

Read more about Undo and Redo options.

Console Scroll-back Lines
: The number of lines, buffered in memory of the console window. Useful for debugging purposes and command-line rendering.

Texture Time Out
: Time since last access of a GL texture in seconds, after which it is freed. Set this to 0 to keep textures allocated.

  Garbage Collection Rate
  : Number of seconds between each run of the GL texture garbage collector.

VBO Time Out
: Time since last access of a GL vertex buffer object (VBO) in seconds after which it is freed (set to 0 to keep VBO allocated).

  Garbage Collection Rate
  : Number of seconds between each run of the GL vertex buffer object garbage collector.

Video Sequencer

Memory Cache Limit

Upper limit of the Video Sequencer and Movie Clip Editor memory cache (in megabytes). For an optimal Clip editor and Sequencer performance, high values are recommended.

Disk Cache
: Writes cached strips to disk which can store a lot more than RAM. To use the Disk Cache, this option must be enabled, the *Disk Cache Directory* and *Disk Cache Limit* set, then save or reopen the existing blend-file.

Directory
: The location on disk to store the cache.

Cache Limit
: Upper limit of the Video Sequencer's disk cache (in gigabytes), setting to zero disables disk cache.

Compression
: The level of compression to compress image in the disk cache. This has a trade off between saving disk space and requiring more processing. The more compression used requires faster disk write/read speeds and more CPU usage.

Proxy Setup
: When and how Proxies are created.

Automatic::	Build proxies for added movie and image strips in each preview size.
Manual::	Set up proxies manually.

See also

Sequencer Cache Properties

Sound

This panel contains the sound settings for live playback within Blender and are only available with a device other than *None*. To control these settings for exporting sound see the Encoding Panel and Audio Panel.

Audio Device

Sets the audio engine to use to process and output audio.

None::	No audio playback support (audio strips can still be loaded and rendered normally).
CoreAudio::	On macOS, CoreAudio is the native audio API. This is the default setting for macOS users and should be preferred.
PulseAudio::	PulseAudio is the most commonly used sound server on modern Linux distributions. If PulseAudio is available, this should be the preferred setting on Linux.
WASAPI::	On Windows, WASAPI is the native audio API introduced with Windows Vista. This is the default setting for Windows users and should be preferred.
Jack::	High quality professional audio engine that needs a properly configured server running on your system. Supports accurate synchronization with other professional audio applications using Jack.
OpenAL::	Available on all platforms in case the native engines do not work. The played back 3D audio might sound different than when rendered.
SDL::	Uses Simple Direct Media Layer API from libsdl.org which supports all platforms. Might be of lower quality and thus should only be used as backup.

Channels

The number of audio source “locations” to output.

Mono::	Output a single audio channel.
Stereo::	Output two audio channels; typically a left and right channel.

4 Channels:: Output a four audio channels.

5.1 Surround:: Output a five audio channels with one LFE channel.

7.1 Surround:: Output a seven audio channels with one LFE channel.

Mixing Buffer

Sets the number of samples used by the audio mixing buffer. Higher buffer sizes can cause latency issues, but if you hear clicks or other problems, try to increase the size.

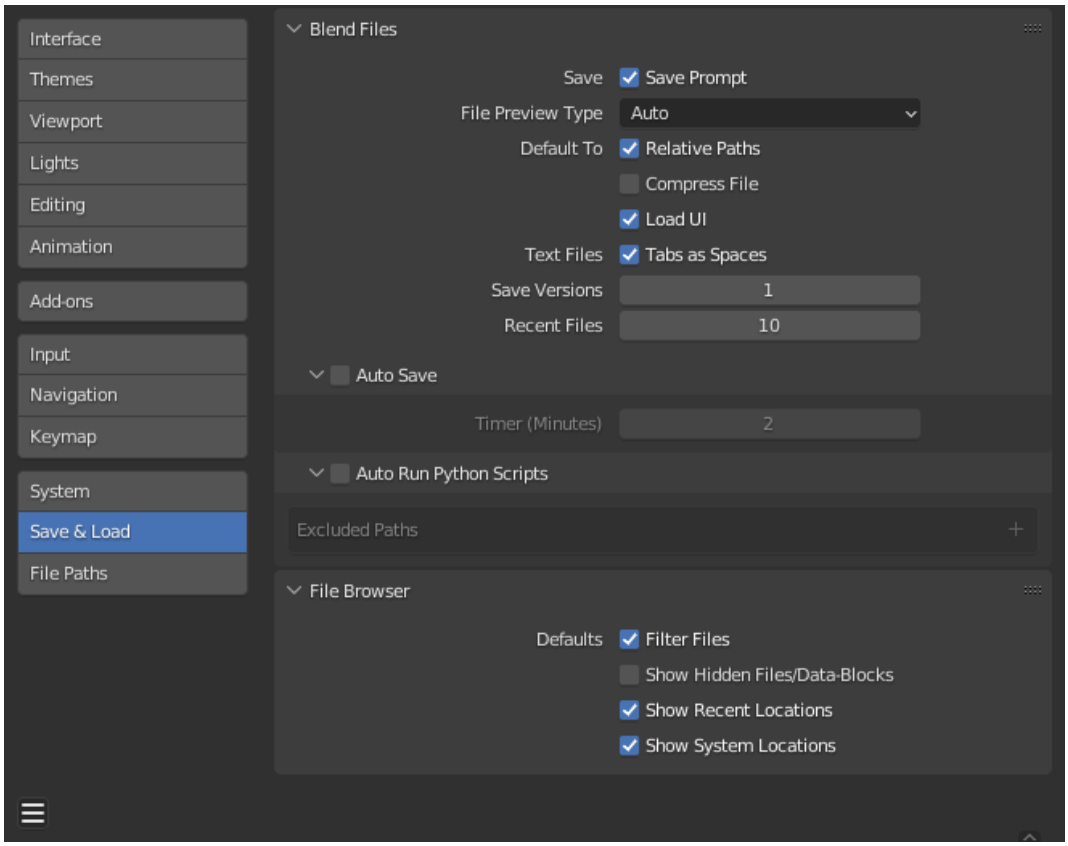
Sample Rate

Sets the audio sampling rate.

Sample Format

Sets the audio sample format.

Save & Load



Preferences Save/Load section.

Blend Files

Save – Save Prompt
: Asks for confirmation before closing or opening a new blend-file if the current file has unsaved changes.

Save Versions
: Number of versions created (for backup) when saving newer versions of a file.

This option keeps saved versions of your file in the same directory, using extensions: `.blend1`, `.blend2`, etc., with the number increasing to the number of versions you specify.

Older files will be named with a higher number. E.g. with the default setting of 2, you will have three versions of your file:

- `*.blend` – last saved.
- `*.blend1` – second last saved.
- `*.blend2` – third last saved.

Recent Files
: Number of files displayed in *File ▸ Open Recent*.

Auto Save
: Enables Auto Save. Tells Blender to *automatically* save a backup copy of your work-in-progress files to the Temporary Directory.

  Timer (Minutes)
  : This specifies the number of minutes to wait between each Auto Save. The default value of the Blender installation is 2 minutes. The minimum is 1, and the Maximum is 60 (save every hour).

File Preview Types
: Select how blend-file preview are generated. These previews are used both in the File Browser and for previews shown in the operating system's file browser.

None::	Do not generate any blend-file previews.
Auto::	If there is no camera in the 3D Viewport a preview using a screenshot of the active Workspace is generated. If a camera is in the scene, a preview of the viewport from the camera view is used.
Screenshot::	Generate a preview by taking a screenshot of the active Workspace.
Camera View::	Generate a preview of a Workbench render from the camera's point of view.

Default To – Relative Paths
: Default value for Relative Paths when loading external files such as images, sounds, and linked libraries. It will be ignored if a path is already set.

Default To – Compress File
: Default value for Compress file when saving blend-files.

Default To – Load UI
: Default value for Load UI when loading blend-files.

Text Files – Tabs as Spaces
: Entering `Tab` in the Text Editor adds the appropriate number of spaces instead of using characters.

Auto Run Python Scripts

Python scripts (including driver expressions) are not executed by default for security reasons. You may be working on projects where you only load files from trusted sources, making it more convenient to allow scripts to be executed automatically.

Excluded Paths
: Blend-files in these folders will *not* automatically run Python scripts. This can be used to define where blend-files from untrusted sources are kept.

See also

Python Security.

File Browser

Show Locations – Recent
: Hide the *Recent* panel of the File Browser which displays recently accessed folders.

Show Locations – System

Hide System Bookmarks in the *File Browser*.

Defaults – Filter Files

By activating this, the file region in the File Browser will only show appropriate files (i.e. blend-files when loading a complete Blender setting). The selection of file types may be changed in the file region.

Defaults – Show Hidden Files/Data-Blocks

Unhide files and data-blocks with names that start with `.` in File Browsers and data IDs.

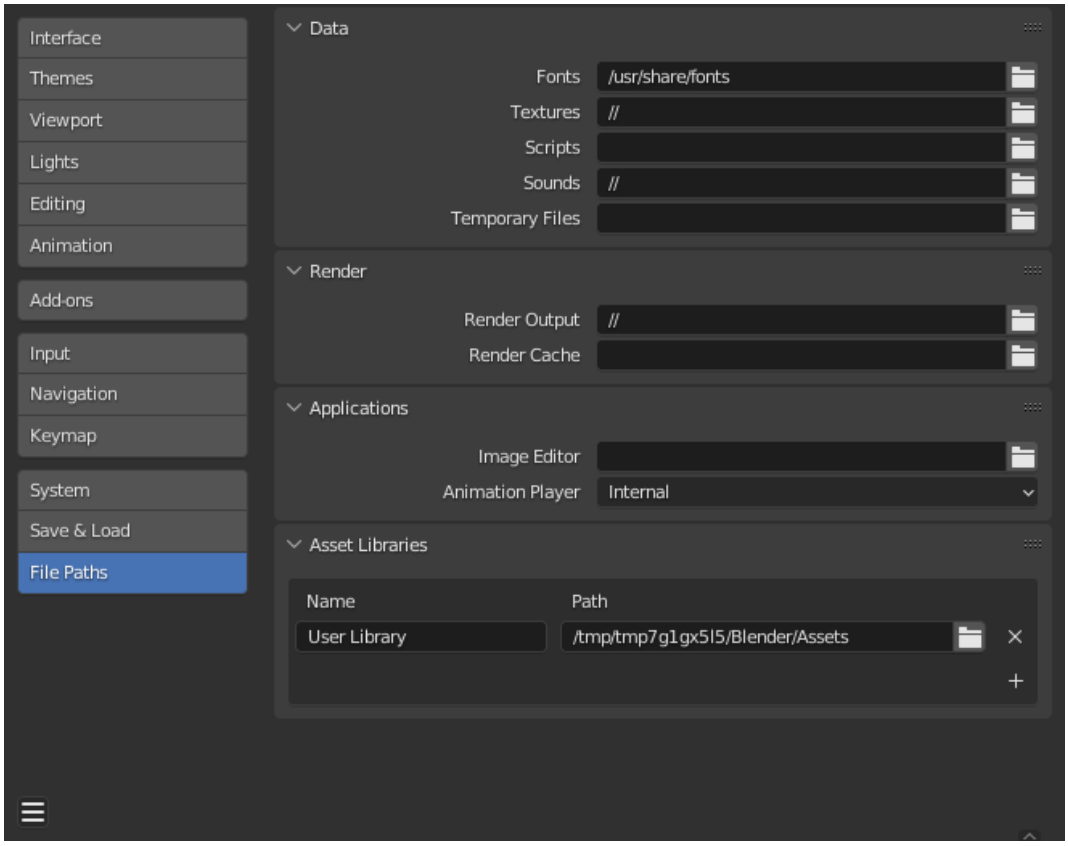
Hint

Data-blocks with names beginning with a `.` can be selected by typing in a search string that also starts with the `.` character, even if this setting is disabled.

File Paths

The *File* section in *Preferences* allows you to configure auto-save preferences and set default file paths for blend-files, rendered images, and more.

Locations for various external files can be set for the following options:



Preferences File Paths section.

Hint

The default path `//` refers to the folder of the currently open blend-file (see Relative Paths for details).

Data

Fonts
: Default location to browse for text object font files.

Textures
: Default location to browse for image textures.

Sounds
: Default location to browse for sound files.

Temporary Files
: The directory for storing temporary save files. The path must reference an existing directory or it will be ignored and the systems temporary directory will be used instead. When left blank, the systems temporary directory will be used (see Temporary Directory for details).

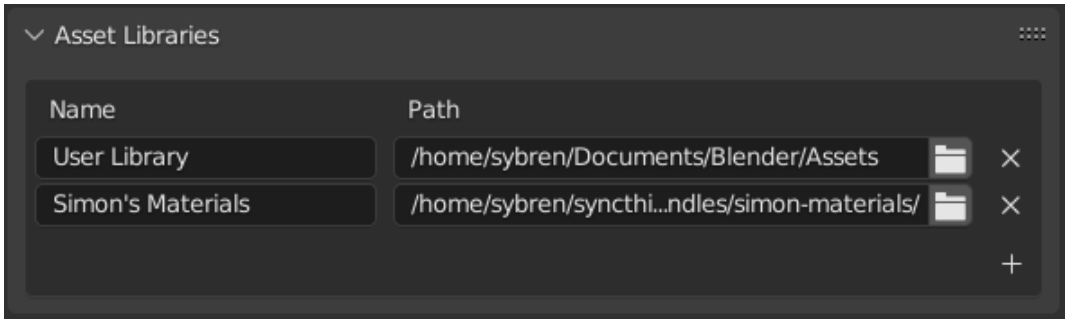
Render

Render Output
: Where rendered images/videos are saved.

Render Cache
: The location where cached render images are stored.

Asset Libraries

Name and on-drive directory paths of asset libraries. To make Blender aware of an asset library, add it to this list. The name is for your reference only, and will appear in asset library selectors. The path should point to the location of the asset library.



Name and Location of asset libraries in the Preferences.

To create a new asset library, just create an empty directory and add it to the List View. Any asset from any blend-file contained in that directory (or subdirectories thereof) will appear in the Asset Browser.

Import Method

Determines how data is managed when an asset is imported, unless overridden by the Asset Browser.

Link:: *Same as File ▸ Link…*
The asset will be linked to the current blend-file, and thus be read-only. Later changes to the asset file will be reflected in all files that link it in.

Append:: *Same as File ▸ Append…*
All of the asset and all its dependencies will be appended to the current file. Dragging a material into the scene three times will result in three independent copies. Dragging an object into the scene three times will also result in three independent copies.
“Dependencies” in this case means everything the asset refers to. For an object, this can be its mesh and materials, but also other objects used by modifiers, constraints, or drivers.
Since the file now has its own copy of the asset, later changes to the asset file will not be reflected in the file it’s appended to.

Append (Reuse Data)::

Specific to the Asset Browser.
The first time an asset is used, it will be appended, including its dependencies, just like described previously. However, Blender will keep track of where it originated, and the next time the asset is used, as much data as possible will be reused. Dragging a material into the scene three times will only load it once, and just assign the same material three times. Dragging an object into the scene three times will create three copies of the object, but all copies will share their mesh data, materials, etc.
Since the file now has its own copy of the asset, later changes to the asset file will not be reflected in the file it's appended to.

Relative Path
: Use relative path when linking assets from this asset library.

Script Directories

Additional locations to search for Python scripts.

Each path can be given a *Name* to signify to purpose of that script directory.

By default, Blender looks in several directories (platform dependent) for scripts. By adding a user script path in the preferences an additional directory is used. This can be used to store your own scripts and add-ons independently of the current Blender version.

You will need to create specific subfolders in this path which match the structure of the `scripts` folder found in Blender's installation directory.

The following subdirectories will be used when present:

`startup/`

Modules in this folder will be imported on startup.

`addons/`
: Legacy add-ons located here will be listed in the add-ons preferences.

`modules/`
: Modules in this folder can be imported by other scripts.

`presets/`
: Presets in this folder will be added to existing presets.

Hint

For add-ons it is now recommended to use a local extension repository if you wish to define additional locations to install and manage them.

To make use of these you will need to define them as extensions.

Note

You have to restart Blender for all changes to the users scripts to take effect.

Applications

Image Editor
: The path to an external program to use for image editing.

Animation Player
: The program used for playing back rendered animations via View Animation.

  By default this is set to *Internal* which uses Blender's built-in animation player.

This has the advantage that all image formats supported by Blender can be played back and no 3rd party application needs to be installed.

Text Editor

Program
: Command to launch the text editor when using Edit Externally, either a full path or a command in `$PATH`. Use the internal editor when left blank.

Arguments
: Defines the specific format of the arguments with which the text editor opens files.

  The supported expansions are as follows:

  - `$filepath`: The absolute path of the file.
  - `$line`: The line to open at (Optional).
  - `$column`: The column to open from the beginning of the line (Optional).
  - `$line0` & `$column0` similar to the above but they start at zero.

  Example: `-f $filepath -l $line -c $column`

Development

Only visible when Developer Extras are enabled.

I18n Branches
: The path to the `/branches` directory of your local SVN translation copy, to allow translating from the UI.

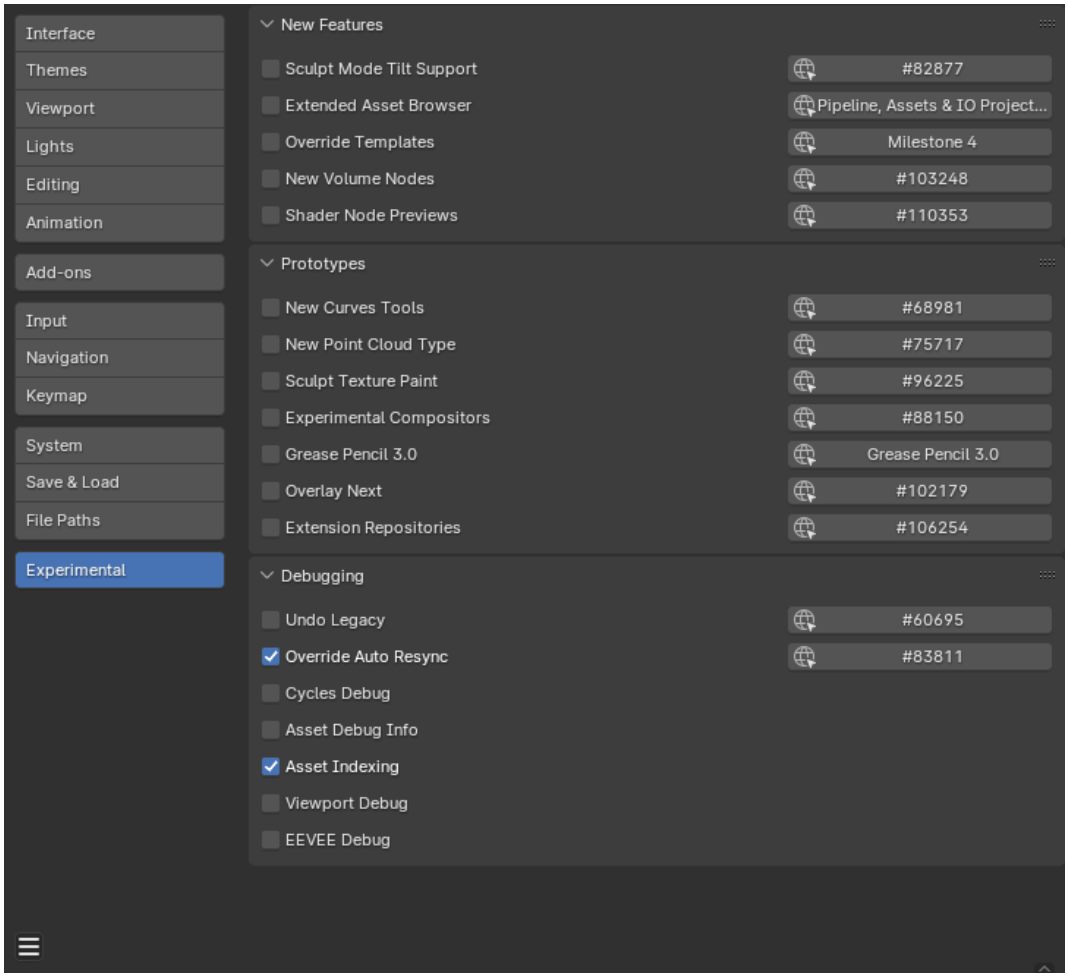
Known Limitations

Permissions on Windows

Be sure that you have the right privileges for running the executable accessing the path defined. On Windows for instance, if the option “Run this program as an administrator” is enabled for the executable, it will lead to a failure to open the editor due to a limitation within the OS User Account Control. Running a program with elevated privileges is potentially dangerous!

Experimental

These preferences are reserved for features that are currently being worked on and are not yet complete. This category can be enabled by enabling Developer Extras. Most experimental features are only available in Daily Builds.



Blender Preferences Experimental section.

Debugging

Undo Legacy

Use legacy undo (slower than the new default one, but may be more stable in some cases).

Override Auto Resync
: Enables library overrides automatic resync detection and process on file load. Disable when dealing with older blend-files that need manual Resync (Enforce) handling.

Cycles Debug
: Show the Cycles rendering debug panel.

Asset Debug Info
: Enable some extra fields in the Asset Browser to aid debugging.

Asset Indexing
: Disabling the asset indexer forces every asset library refresh to completely reread assets from disk.

Viewport Debug
: Enable viewport debugging options for developers in the overlays pop-over.

EEVEE Debug
: Enable EEVEE debuging options for developers.

Scenes & Objects

- Scenes
  - Introduction
  - Scene Properties
- Objects
  - Introduction
  - Object Types
  - Object Origin
  - Selecting
  - Editing
  - Properties
  - Tools
- Collections
  - Introduction
  - Collections
- View Layers
  - Introduction

Scenes

- Introduction
  - Controls
- Scene Properties
  - Scene
  - Units
  - Gravity
  - Simulation
  - Keying Sets
  - Audio
  - Rigid Body World

Introduction

Scenes are a way to organize your work. Each blend-file can contain multiple scenes, which share other data such as objects and materials.

Scene management and library appending/linking are based on Blender's Library and Data System, so it is a good idea to read that manual page first, if you are not familiar with the basics of that system.

Controls

Reference

Menu:: *Topbar ▸ Scene*

You can select and create scenes with the *Scene* data-block menu in the Topbar.



Scene data-block menu.

Scenes
: A list of available scenes.

Add
: New
  : Creates an empty scene with default values.

  Copy Settings
  : Creates an empty scene, but also copies the settings from the active scene into the new one.

  Linked Copy
  : This option creates a new scene with the same settings and contents as the active scene. However, instead of copying the objects, the new scene contains links to the collections in the old scene. Therefore, changes to objects in the new scene will result in the

same changes to the original scene, because the objects used are literally the same. The reverse is also true.

Full Copy

Using this option, nothing is shared. This option creates a fully independent scene with copies of the active scene's contents. Every object in the original scene is duplicated, and a duplicate, private copy of its object-data is made as well.

Note

To choose between these options, it is useful to understand the difference between *Object* and *Object Data*. The choices for adding a scene, therefore, determine just how much of this information will be *copied from* the active scene to the new one, and how much will be *shared* (linked).

(Delete Scene)

Deletes the current scene data-block. Note, there always has to be one scene data-block.

See also

Linking to a Scene

You can link any object from one scene to another.

Scene Properties

Scene

Reference

Panel:: *Properties* ▸ *Scene* ▸ *Scene*

Camera
: Used to select which camera is used as the active camera. You can also set the active camera in the 3D Viewport with `Ctrl-Numpad0`.

Background Scene
: Allows you to use a scene as a background, this is typically useful when you want to focus on animating the foreground for example, without background elements getting in the way.

  This scene can have its own animation, physics simulations, etc, but you will have to select it from the *Scene* data-block menu, if you want to edit any of its contents.

  Background Scenes can themselves have a Background Scene (they're recursively included). So you can always make additions to existing scenes by using them as a background to a newly created scene where your additions are made.

  Tip

  This can also be used in combination with Linking to a Scene, where one blend-file contains the environment, which can be reused in many places.

Active Clip

Selects a Movie Clip that can be used by Motion Tracking Constraints or a camera's Background Images.

Units

Reference

Panel:: *Properties* ▸ *Scene* ▸ *Units*

Unit System
: The unit system to use for user interface controls.

None::	Use units that have with no relation to the real world, practically this is the same as *Metric* just without unit names.
Metric::	Use the metric unit system in this scene.
Imperial::	Use the imperial unit system in this scene.

Unit Scale
: Scale factor to use when converting between internal units and values displayed in the user interface. This can be changed when modeling at microscopic or astronomical scales.

Note

This only influences the values displayed in the user interface and not how things behave internally. For example, physics simulations don't take the unit scale into account.

Separate Units
: When using *Metric* or *Imperial*, display properties as multiple values. For example, `2.285m` will become `2m 28.5cm`.

Rotation
: Unit to use for displaying/editing rotation values.

Degrees:: Use degrees for angles in the user interface.
Radians:: Use radians for angles in the user interface.

Length
: Unit that will be used to display length values.

Adaptive:: The unit used for a specific value depends on the magnitude of the value. For example, some values might be displayed as `23cm` while others are displayed as `10km`.

Meters/Centimeters/Feet:: A fixed unit that will be used for all lengths in the user interface.

Mass
: See *Length*.

Time
: See *Length*.

Temperature
: See *Length*.

Imperial Length Units

Full Name	Short Name(s)	Scale of a Meter
thou	`mil`	0.0000254
inch	`"`, `in`	0.0254
foot, feet	`'`, `ft`	0.3048
yard	`yd`	0.9144

Full Name	Short Name(s)	Scale of a Meter
chain	`ch`	20.1168
furlong	`fur`	201.168
mile	`mi`, `m`	1609.344

Metric Length Units

Full Name	Short Name(s)	Scale of a Meter
micrometer	`um`	0.000001
millimeter	`mm`	0.001
centimeter	`cm`	0.01
decimeter	`dm`	0.1
meter	`m`	1.0
dekameter	`dam`	10.0
hectometer	`hm`	100.0
kilometer	`km`	1000.0

Full Name	Short Name(s)	Scale of a Meter

Gravity

Reference

Panel:: *Properties* ▸ *Scene* ▸ *Gravity*

Options to control global gravity used for physics effects.

See the Physics chapter for more information.

Simulation

Simulation Range
: Use a simulation range that is different from the scene range for Simulation Nodes that do not override the frame range themselves.

Start, End
: The frame at which the simulation starts/ends.

Keying Sets

Reference

Panel:: *Properties* ▸ *Scene* ▸ *Keying Sets*

See Keying Sets.

Audio

Reference

Panel:: *Properties ▸ Scene ▸ Audio*

Options to control global audio settings. To control how sounds is played back from within Blender, see the audio settings in the Preferences.

Volume
: Volume for the scene.

Distance Model
: Changes how the sound attenuation is calculated based on the distance. Most physically correct is the *Inverse* model, but it's also possible to choose a linear and an exponential falloff. The clamped modes limit the volume to be lower than 100% (1.0), that means if the distance is smaller than the reference distance, the volume is always 100%. For an exact description of each option see the OpenAL documentation.

Doppler Speed
: Speed of the sound for the Doppler effect calculations. The typical value is 343.3 m/s in air, in water for example this value is around 1560 m/s.

Doppler Factor
: Controls how strong the Doppler effect is. You can exaggerate or attenuate the change of pitch, but physically correct is a factor of 1.0.

Update Animation Cache
: Updates the audio animation cache. This is useful if you start noticing artifact in the audio.

Rigid Body World

Reference

Panel:: *Properties ▸ Scene ▸ Rigid Body World*

The *Rigid Body World* is a group of rigid body objects, which holds settings that apply to all rigid bodies in this simulation.

See Rigid Body World for more information.

Objects

- Introduction
- Object Types
  - Common Options
- Object Origin
  - Set Origin
- Selecting
  - Selections and the Active Object
  - Select Menu
- Editing
  - Transform
  - Mirror
  - Clear
  - Apply
  - Snap
  - Duplicate
  - Duplicate Linked
  - Join
  - Asset
  - Parenting Objects
  - Modifiers
  - Constraints
  - Track
  - Relations
  - Link/Transfer Data
  - Shading
  - Rigid Body
  - Convert
  - Show/Hide
  - Clean Up
  - Delete
- Properties
  - Transform
  - Relations

    - Collections
    - Instancing
    - Visibility
    - Viewport Display
    - Line Art
- Tools
    - Toolbar
    - Tool Settings
    - Types

Introduction

The geometry of a scene is constructed from one or more objects. These objects can range from lights to illuminate your scene, basic 2D and 3D shapes to fill it with models, armatures to animate those models, to cameras to take pictures or make video of it all.

Each Blender object type (mesh, light, curve, camera, etc.) is composed from two parts: an *Object* and *Object Data* (sometimes abbreviated to "ObData"):

Object
: Holds information about the position, rotation and size of a particular element.

Object Data
: Holds everything else. For example:

  Meshes:: Store geometry, material list, vertex groups, etc.
  Cameras:: Store focal length, depth of field, sensor size, etc.

  Each object has a link to its associated object-data, and a single object-data may be shared by many objects.

Object Types

Reference

Mode::	Object Mode
Menu::	*Add*
Shortcut::	`Shift-A`

New objects can be created with the *Add* menu in the 3D Viewport's header.

Mesh
: Objects composed of vertices, edges and polygonal faces and can be edited extensively with Blender's mesh editing tools. See Mesh Primitives.

Curve
: Mathematically defined objects which can be manipulated with control handles or control points (instead of vertices), to edit their length and curvature. See Curves Primitives.

Surface
: Mathematically defined patches that are manipulated with control points. These are useful for simple rounded forms and organic landscapes. See Surfaces Primitives.

Metaball
: Objects formed by a mathematical function (with no vertices or control points) defining the 3D volume in which the object exists. Meta objects have a liquid-like quality where when two or more metaballs are brought together, they merge by smoothly rounding out the connection, appearing as one unified object. See Meta Primitives.

Text
: Create a two-dimensional representation of a text.

Volume
: Container for OpenVDB files that is generated by other software or Blender's Fluid Simulator.

Grease Pencil
: Objects created by drawing strokes. See Grease Pencil Primitives

Armature
: Used for rigging 3D models to make them pose-able and animatable.

Lattice
: Non-renderable wireframes commonly used for the deformation of other objects with help of the Lattice Modifier.

Empty
: Null objects that are simple visual transform nodes that do not render. They are useful for controlling the position or movement of other objects.

Image
: Empty objects that display images in the 3D Viewport. These images can be used to aid artists in modeling or animating.

  Image Plane
  : Adds a mesh plane with materials and texture from an image file. The dimensions of the plane are calculated to match the aspect of the image file.

Light
: Empty objects that emit light and are used for lighting the scene in renders.

Light Probe
: Used by the EEVEE render engine to record lighting information for indirect lighting.

Camera

This is the virtual camera that is used to determine what appears in the render.

Speaker
: Empty objects that bring a source of sound to the scene.

Force Field
: Empty objects that give simulations external forces, creating movement, and are represented in the 3D Viewport as small control objects.

Collection Instance
: Lets you select from a list of existing collections. Once selected, an empty object will be created, with an instance of the selected collection (collection instancing active).

Common Options

You can change the options of the object in the Adjust Last Operation panel just after creating it:

Type
: You can change the type of some objects after their creation with a selector.

Radius/Size
: Sets the starting size.

Align
: Rotates the new object so that it is aligned in one of the following manners:

World::	Aligns the object to the global space axes, i.e. the object's front faces the negative Y axis (default).
View::	Aligns the object to the view space axes, i.e. the object's front faces the viewport's point of view.
3D Cursor::	Aligns the object to match the rotation of the 3D Cursor.

Location

Objects are placed, by default, at the position of the 3D Cursor. These values let you place the object in an other position.

Rotation

Values let you rotate the object so that default rotation is overridden.

Object Origin

Each object has an origin point. The location of this point determines where the object is located in 3D space. When an object is selected, a small circle appears, denoting the origin point. The location of the origin point is important when translating, rotating or scaling an object. See Pivot Points for more.

The color of the origin changes based on the selection state of the object.

Yellow:: Object is active.
Orange:: Object is selected, but not active.
White:: Object is not linked and not selected.
Turquoise:: Object is linked.
Light Turquoise:: Object is selected, linked, but not active.

Note

Colors are themeable and might appear different. The colors described here are from the default Dark Theme.

Set Origin

Reference

Mode:: Object Mode
Menu:: *Object ▸ Set Origin*

The object origin and geometry can be moved relative to each other and to the 3D cursor.

Type
    Geometry to Origin

Moves the model to the origin and this way the origin of the object will also be at the center of the object.

Origin to Geometry
Moves the origin to the center of the object.

Origin to 3D Cursor
Moves the origin of the model to the position of the 3D cursor.

Origin to Center of Mass
Moves the origin to the calculated center of mass of model (assuming the mesh has a uniform density).

Center
Median Point Center, Bounding Box Center

Tip

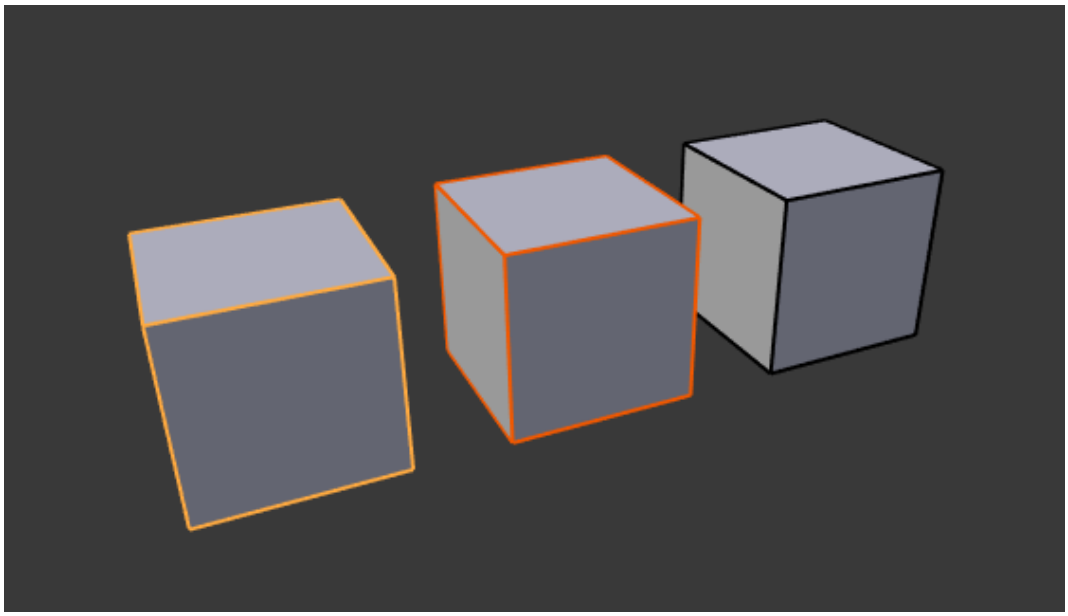
To transform an object's origin directly, enable Affect Only Origins in the *Tool Settings Options*.

Selecting

Selection determines which elements will be the target of our actions. Selections work on the current scene visible objects. Blender has advanced selection methods. Both in *Object Mode* and in *Edit Mode*.

Selections and the Active Object

Blender distinguishes between two different states of selection:



Active object in yellow, selected object in orange, and unselected object in black.

In *Object Mode* the last (de)selected item is called the "Active Object" and is outlined in yellow (the others are orange). There is at most one active object at any time.

Many actions in Blender use the active object as a reference (for example linking operations). If you already have a selection and need to make a different object the active one, simply reselect it with `Shift-LMB`.

All other selected objects are just selected. You can select any number of objects. In order to change a property or to perform an operation on all selected objects (bones, and sequencer strips) hold `Alt`, while confirming.

Select Menu

All

Reference

Mode::	All Modes
Menu::	*Select ▸ All*
Shortcut::	`A`

Select all selectable objects.

None

Reference

Mode::	All Modes
Menu::	*Select ▸ None*
Shortcut::	`Alt-A`

Deselect all objects, but the active object stays the same.

Invert

Reference

Mode::	All Modes
Menu::	*Select ▸ Invert*
Shortcut::	`Ctrl-I`

Toggle the selection state of all visible objects.

Box Select

Reference

Mode::	All Modes
Menu::	*Select ▸ Box Select*
Shortcut::	`B`

Interactive box selection.

Circle Select

Reference

Mode::	All Modes
Menu::	*Select ▸ Circle Select*
Shortcut::	`C`

Interactive circle selection.

Lasso Select

Reference

Mode::	All modes
Menu::	*Select ▸ Lasso Select*
Shortcut::	`Ctrl-Alt-LMB`

See Select Lasso.

Select Active Camera

Reference

Mode::	Object Mode

Menu:: *Select ▸ Select Active Camera*

Selects the active camera, this is especially handy in complex scene.

Select Mirror

Reference

Mode:: All Modes
Menu:: *Select ▸ Select Mirror*

Select the Mirror objects of the selected object, based on their names, e.g. “sword.L” and “sword.R”.

Select Random

Reference

Mode:: Object Mode
Menu:: *Select ▸ Select Random*

Randomly selects unselected objects based on percentage probability. The percentage can be modified in the *Adjust Last Operation* panel. It is important to note that the percentage is the likelihood of an unselected object being selected and not the percentage amount of objects that will be selected.

Select More/Less

Reference

Mode:: Object Mode
Menu:: *Select ▸ More/Less*
Shortcut:: `Ctrl-NumpadPlus`, `Ctrl-NumpadMinus`

Their purpose, based on the hierarchical.

More
: Expand the selection to the immediate parent and children of the selected objects.

Less
: Contrast the selection, deselect objects at the boundaries of parent/child relationships.

Parent
: Deselects the currently selected objects and selects their immediate parents.

Child
: Deselects the currently selected objects and selects their immediate children.

Extend Parent
: Extends the selection to the immediate parents of the currently selected objects.

Extend Child
: Extends the selection to the immediate children of the currently selected objects.

Select All by Type

Reference

Mode::	Object Mode
Menu::	*Select ▸ Select All by Type*

With this tool, it becomes possible to select objects of a certain type in one go. For a description of all object types see Object Types.

Select Grouped

Reference

Mode::	Object Mode
Menu::	*Select* ▸ *Select Grouped*
Shortcut::	`Shift-G`

There are two ways to organize the objects in relation to one another. The first one is *parenting*, and the second is simple *grouping*. These relationships to an artist's advantage by selecting members of respective families or groups. *Select Grouped* uses the active object as a basis to select all others.

Children
: Selects all hierarchical descendants of the active object.

Immediate Children
: Selects all direct children of the active object.

Parent
: Selects the parent of this object if it has one.

Siblings
: Select objects that have the same parent as the active object. This can also be used to select all root level objects (objects with no parents).

Type
: Select objects that are the same type as the active one.

Collection
: Select all objects that are in the same collection as the active one. If the active object belongs to more than one collection, a list will pop up so that you can choose which collection to select.

Object Hooks
: Every hook that belongs to the active object.

Pass
: Select objects assigned to the same Render Pass.

Color
: Select objects with same Object Color.

Keying Set
: Select objects included in the active Keying Set.

Light Type
: Select matching light types.

Select Linked

Reference

Mode::	Object Mode
Menu::	*Select* ▸ *Select Linked*
Shortcut::	`Shift-L`

Selects all objects which share a common data-block with the active object. *Select Linked* uses the active object as a basis to select all others.

Object Data
: Selects every object that is linked to the same Object Data, i.e. the data-block that specifies the type (mesh, curve, etc.) and the build (constitutive elements like vertices, control vertices, and where they are in space) of the object.

Material
: Selects every object that is linked to the same material data-block.

Instanced Collection
: Select every object that is linked to the instanced collection.

Texture
: Selects every object that is linked to the same texture data-block.

Particle System
: Selects all objects that use the same *Particle System*.

Library
: Selects all objects that are in the same Library.

Library (Object Data)
: Selects all objects that are in the same Library and limited to *Object Data*.

Select Pattern

Reference

Mode:: Object Mode
Menu:: *Select* ▸ *Select Pattern…*

Selects all elements whose name matches a given pattern. Supported wildcards: `*` matches everything, `?` matches any single character, `[abc]` matches characters in `abc`, and `[!abc]` match any character not in `abc`. As an example `*house*` matches any name that contains `house`, while `floor*` matches any name starting with `floor`.

Case Sensitive
: The matching can be chosen to be case sensitive or not.

Extend
: When *Extend* checkbox is checked the selection is extended instead of generating a new one.

Editing

- Transform
  - Introduction
  - Transform Control
  - Move
  - Rotate
  - Scale
  - Move/Scale Texture Space
  - Align to Transform Orientation
  - Randomize
  - Align Objects
- Mirror
- Clear
- Apply
- Snap
- Duplicate
- Duplicate Linked
- Join
- Asset
- Parenting Objects
- Modifiers
- Constraints
- Track
- Relations
  - Make Single User
- Link/Transfer Data
  - Link Objects to Scene
  - Link Data
  - Copy UV Maps
  - Transfer Mesh Data
  - Transfer Mesh Data Layout
- Shading
- Rigid Body
- Convert

- Show/Hide
- Clean Up
- Delete

Transform

- Introduction
- Transform Control
  - Numeric Input
  - Axis Locking
  - Precision
- Move
  - Options
- Rotate
  - Options
  - Trackball Rotation
- Scale
  - Options
- Move/Scale Texture Space
- Align to Transform Orientation
- Randomize
- Align Objects
  - Options

Introduction

Transformations refer to a number of operations that can be performed on a selected Object or Mesh that alters its position or characteristics.

Each object can be moved, rotated and scaled in *Object Mode*. However, not all of these transformations have an effect on all objects. For example, scaling a camera has no effect on the render dimensions.

Basic transformations include:

- Move
- Rotate
- Scale

These three transforms are the three big ones. However, more advanced transformations can be found in the Advanced Transformations section.

For making other changes to the geometry of editable objects, you should use *Edit Mode*.

Once you have added a basic object, you remain in *Object Mode*. You can switch between *Object Mode* and *Edit Mode* by pressing `Tab`. The object's wireframe should now appear orange. This means that the object is now selected and active.

Transform Control

Transform controls can be used to modify and control the effects of the available transformations.

- Numeric Input
  - Simple Mode
  - Advanced Mode
- Axis Locking
  - Usage
  - Axis Locking Types
- Precision
  - Usage
  - Snapping
  - Precision

Numeric Input

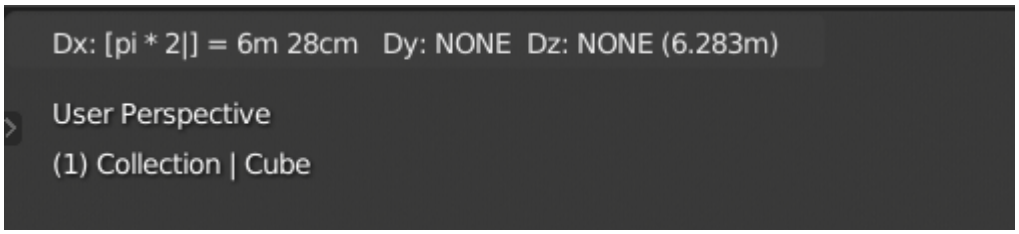
Using the mouse for transformations is convenient, but if you require more precise control, you can also enter numeric values. After pressing the shortcut type a number to indicate the magnitude of the transformation. Then confirm or cancel. E.g. pressing `S 2`, `Return` will double the scale of an object.

Move `G`
: By default and with no other key presses, the translation will occur along the X axis.

Rotation `R`
: The rotation is in clockwise direction for positive values.

Scale `S`
: Scaling works in almost identical fashion to translation. The primary difference is that by default, scaling applies equally to all three axes.

You can see the numbers you enter in the 3D Viewport footer.



Numeric input displayed in the footer.

Tip

Numeric input can also be inputted in the Properties region.

Simple Mode

Blender has two “modes” a simple and an advanced one. Simple mode only accepts simple numbers. You can use basic text editing except selection.

Decimals `Period`
: Decimals can be entered by pressing `Period`.

Negate `Minus`
: Negate the whole value by pressing `Minus`.

Inverse `Slash`
: Hitting `Slash` during number entry switches the number being entered to its reciprocal, e.g. `2 /` results in 0.5 (1/2); `20 /` results in 0.05 (1/20).

Reset `Backspace`
: Hitting `Backspace` after having deleted all leading chars will first reset the edited value to initial state, and on second press, the whole number editing will be canceled, going back to usual transform with mouse.

Next/Previous Component `Tab`, `Ctrl-Tab`
: To enter numeric values for multiple axes, use `Tab` or `Ctrl-Tab`. E.g. To move an object, one unit on all three axes press: `G 1` and `Tab 1` and `Tab 1`.

Non-number Inputs
: You can also combine numeric input with Axis Locking to limit movement to a particular axis or tool specific shortcuts.

Advanced Mode

In advanced mode you can additionally enter expressions and units.

Use `=` or `NumpadAsterisk` to enable advanced mode, and `Ctrl-=` or `Ctrl-NumpadAsterisk` to switch back to simple mode.

It features:

- Units (`cm`, `"`, `deg`, etc.). See unit system.
- Basic operations from Python (`+`, `*`, `/`, `**`, etc.).

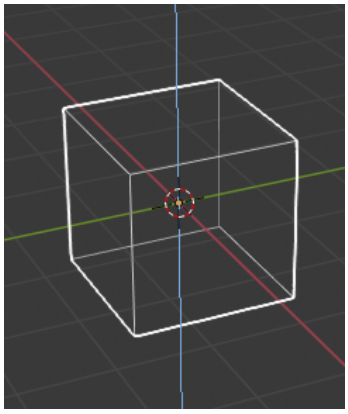
- Math constants and functions (`pi`, `sin`, `sqrt`, etc.). See Python's math module.

You can still use the negate and inverse shortcuts (`Minus`, `Slash`), as well as non-number inputs, but you have to hold `Ctrl` to activate them.

Axis Locking

This option limits the transformation to the specified axis.

Transformations (translation/scale/rotation) in *Object Mode* and *Edit Mode* (as well as extrusions in *Edit Mode*) can be locked to a particular axis relative to the current transform orientation. By locking a transformation to a particular axis you are restricting transformations to a single dimension.



Axis locking.

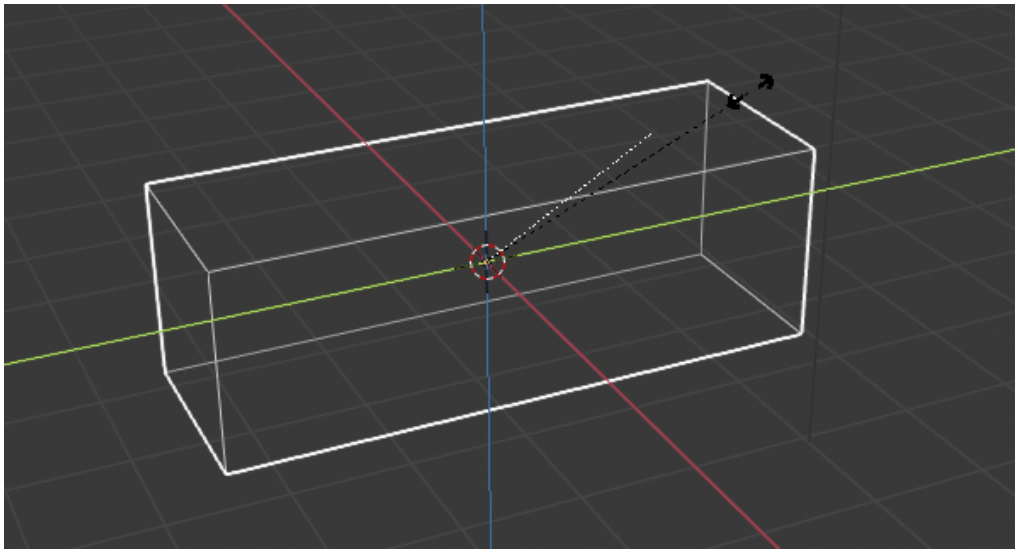
Usage

A locked axis will display in a brighter color than an unlocked axis. For example in the image to the right, the Z axis is shown in light blue as movement is constrained to this axis. This example, can be achieved in two ways:

Hotkey

The axis of movement can be changed at any time during transformation by typing `X`, `Y`, `Z`.

Pointing



Axis constraint in action.

Holding MMB after starting a transformation lets you select an axis to constrain to. A visual option to constrain the translation will be available, showing the three axes in the 3D Viewport space. A dotted white line is used as a pointer. The axis of choice to confirm the operation will depend on the highlighted axis about which the MMB is released.

When you already moved the mouse in the desired direction, pressing MMB will lock to the axis which was pointed at.

Axis Locking Types

Axis Locking

Reference

Mode::	Object and Edit Modes (move, rotate, scale, extrude)
Shortcut::	X, Y, Z or MMB after moving the mouse in the desired direction.

Axis locking limits the transformation to a single axis (or forbids transformations along two axes). An object, face, vertex or other selectable item will only be able to move, scale or rotate in a single dimension.

Plane Locking

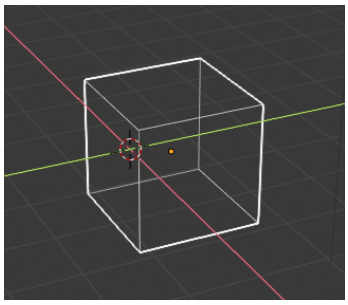
Reference

Mode::	Object and Edit Modes (move, scale)

Shortcut:: `Shift-X`, `Shift-Y`, `Shift-Z` or `Shift-MMB` after moving the mouse in the desired direction.

Plane locking locks the transformation to *two* axes (or forbids transformations along one axis), thus creating a plane in which the element can be moved or scaled freely. Plane locking only affects translation and scaling.

Note that for rotation, both axis and plane locking have the same effect because a rotation is always constrained around one axis. *Trackball* type rotations `R R` cannot be locked at all.



Plane locking.

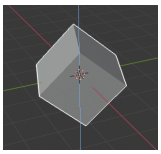
Axis Locking Modes

A single key press constrains movement to the current transform orientation selection. A second key press of the *same* key constrains movement to the corresponding *Global* axis (except if the transform orientation is set to *Global*, in which case the *Local* orientation is used). A third key press of the same key removes constraints.

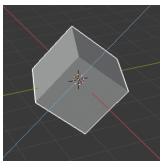
The orientation can be set in the Transform Orientation selector of the 3D Viewport header.

For example, if the current transform orientation is set to *Normal*, pressing `G` to start translation, followed by `Z` will lock translation in the Z direction relative to the *Normal* orientation, pressing `Z` again will lock translation to the Z axis relative to the *Global* orientation. Pressing `Z` again will remove all constraints. The current mode will be displayed in the left-hand side of the 3D Viewport header.

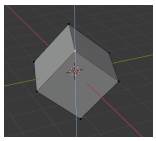
Axis locking modes.



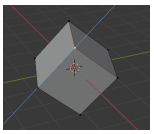
Z axis locking in Global orientation.



Z axis locking in Local orientation.



Z axis locking in Global orientation with vertex selection.



Z axis locking in Normal orientation with vertex selection.

As can be seen in the *Axis locking modes* image, the direction of the transform also takes into account the selection.

Note that using a locked axis does not prevent you from using the keyboard to enter numeric transformation values.

Precision

Reference

Mode::	Object and Edit Modes
Shortcut::	`Ctrl` and/or `Shift`

Holding `Ctrl` during a transform operation (such as move, rotate or scale) will toggle Transform Snapping. When using Increment Snap this allows the transformation to be performed in discrete amounts.

Holding `Shift` during a transform operation will transform the object at 1/10th the speed, allowing much finer control.

The magnitude of the transformation can be viewed in the 3D Viewport header. Releasing `Ctrl` or `Shift` during the transformation will cause the movement to revert back to its normal mode of operation.

Note

The snapping behaviors described on this page **only** apply when Increment Snap is selected.

Tip

It is possible to enable both snapping and precision mode, simply hold `Ctrl` and `Shift`. This has the following effects:

Move
: Changes in 0.1 unit increments, regardless of zoom level.

Rotation
: Changes in 1 unit increments.

Scale
Changes in 0.01 unit increments.

Usage

With Hotkeys

Press `G`, `R` or `S` and then hold either `Ctrl`, `Shift` or `Shift-Ctrl`.

With the Transform Gizmo

Select the gizmo handle then while moving the mouse hold `Ctrl`, `Shift` or `Shift-Ctrl` to activate precision control or snapping.

See also

Read more about the Transform Gizmo.

Tip

Combining with Other Controls

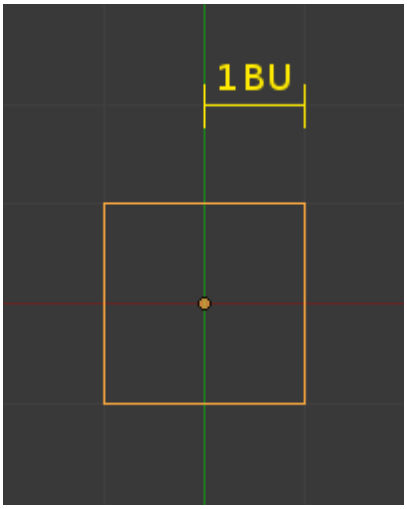
All of the precision controls detailed on the page can be combined with the Axis Locking controls and used with the different Pivot Points.

Snapping

Move

Snapping while moving objects changes the object location in 1 unit increments. While in an aligned view, The increment amount is changed based on the zoom level. For example, at a base zoom level objects are moved in increments of 1 unit (i.e. between the two light gray lines).

Zooming in enough to see the next set of gray lines will snap in increments of 1/10 of a unit. Zooming in further until will snap in increments of 1/100 of a unit and so on until the zoom limit is reached. Zooming out will have the opposite effect and cause movement to happen by increments of 10, 100 units, etc.



One unit (default zoom level).

Rotation

Holding `Ctrl` will cause rotations of 5 degrees.

Scale

Holding `Ctrl` will cause size changes in increments of 0.1 units.

Note

Snapping Modes

Note that when you are Snapping To something other than *Increment*, holding `Ctrl` will cause the selection to snap to that nearest element.

Read more about snapping.

Precision

Holding `Shift` during transformations allows for very fine control that does not rely on fixed increments. Rather, large movements of the mouse across the screen only result in small transformations of the selection.

In rotation mode the selected element will be rotate in 0.01 degree increments.

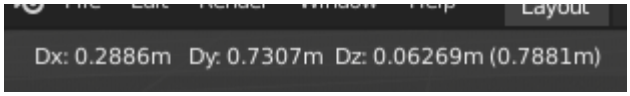
Move

Reference

Mode::	Object Mode, Edit Mode, and Pose Mode
Menu::	*Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Move*
Shortcut::	`G`

In Object Mode, the move option lets you move objects. Translation means changing location of objects. It also lets you move any elements that make up the object within the 3D space of the active 3D Viewport.

Pressing `G` activates "Move" transformation mode. The selected object or element then moves freely according to the mouse pointer's location and camera. To confirm the action, press `LMB`. While moving items, the amount of change along the X, Y, and Z axis is displayed in the header of the 3D Viewport.



Translation Display.

Tip

Moving an object in Object Mode changes the object's origin. Moving the object's vertices/edges/faces in Edit Mode does not change the object's origin.

See also

Using a combination of shortcuts gives you more control over your transformation. See Transform Control.

Options

Move X, Y, Z
: The amount to move the selection on the respected axis.

Orientation
: Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

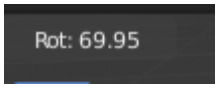
Rotate

Reference

Mode::	Object and Edit Modes
Menu::	*Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Rotate*
Shortcut::	R

Rotation is also known as a spin, twist, orbit, pivot, revolve, or roll and involves changing the orientation of elements (vertices, edges, faces, objects, etc.) around one or more axes or the Pivot Point.

The angle of rotation is displayed in the header of the 3D Viewport.



Rotation values.

See also

Using a combination of shortcuts gives you more control over your transformation. See Transform Control.

Options

Angle
: The amount of rotation.

Axis
: Used to constraint the transformation to one or more axes.

Orientation

Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

Trackball Rotation

Reference

Mode::	Object and Edit Modes
Shortcut::	`R R`

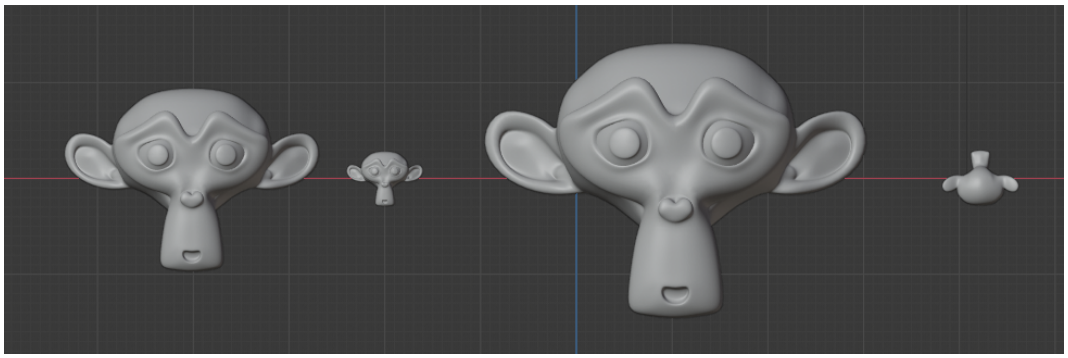
A free rotation mode. Press `R R` to enable Trackball rotation.

Scale

Reference

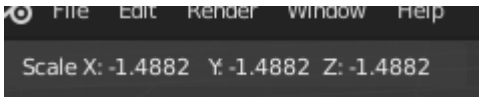
Mode::	Object and Edit Modes
Menu::	*Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Scale*
Shortcut::	S

Scaling means changing proportions of objects. Pressing S will enter the *Scale* transformation mode where the selected element is scaled inward or outward according to the mouse pointer's location. The element's scale will increase as the mouse pointer is moved away from the Pivot Point and decrease as the pointer is moved towards it. If the mouse pointer crosses from the original side of the Pivot Point to the opposite side, the scale will continue in the negative direction and flip the element.



Basic scale usage. From left to right, the panels show: the original object, a scaled down object, a scaled up object and a scale-flipped object.

The amount of scaling will be displayed in the header of the 3D Viewport.



Scale values.

See also

Using a combination of shortcuts gives you more control over your transformation. See Transform Control.

Options

Scale X, Y, Z
: The amount to resize the selection on the respected axis.

Orientation
: Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

Move/Scale Texture Space

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Object* ▸ *Transform* ▸ *Move/Scale Texture Space*

The Move/Scale Texture Space tool transforms the Texture Space of the object, instead of the object or element itself.

Align to Transform Orientation

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Object* ▸ *Transform* ▸ *Align to Transform Orientation*

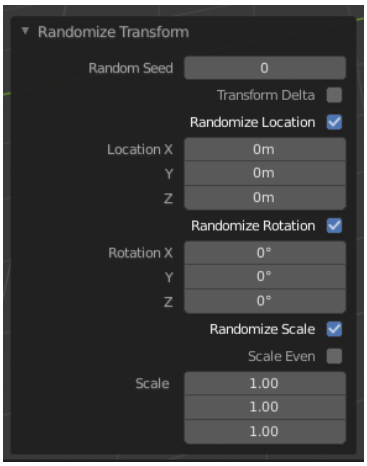
Aligns (rotates) the selected objects so that their local orientation matches the active transform orientation in the Transform orientation panel or the *Orientation* selection in the Transform Adjust Last Operation panels.

Randomize

Reference

Mode:: Object Mode and Edit Mode
Menu:: *Object* ▸ *Transform* ▸ *Randomize Transform*

This tool randomizes the move, rotate, and scale values to an object or multiple objects. When applied on multiple objects, each object gets its own seed value, and will get different transform results from the rest.



Randomize transform options.

Random Seed
: The random seed is an offset to the randomized transformation. A different seed will produce a new result.

Transform Delta
: Randomize Delta Transform values instead of regular transform.

Randomize Location
: Randomize Location values.

Location
: The maximum distances the objects can move along each axis.

Randomize Rotation
: Randomize rotation values.

Rotation
: The maximum angle the objects can rotate on each axis.

Randomize Scale

Randomize scale values.

Scale Even
: Use the same scale for each axis.

Scale
: The maximum scale randomization over each axis.

Align Objects

Reference

Mode:: Object Mode
Menu:: *Object ▸ Transform ▸ Align Objects*

The Align tool is used to align multiple selected objects so they line up on a specified axis.

Options

High Quality
: Uses more precise math to better determine the locations for the objects. In case of positive or negative bounding box alignment, if one or more of the selected objects have any rotation transformations (or delta rotation transformations), it is recommended to check *High Quality* so that their bounding box is calculated with precision for all three global axes.

Align Mode
: The *Align Mode* control will define what part of the objects will be aligned:

  Centers:: The objects centers.
  Positive Sides/Negative Sides:: The positive or negative sides (on the global axes) of their respective bounding boxes.

Relative To
: The *Relative To* control will let us choose to align the objects to:

  Active:: The active object.
  Selection:: The median point of the selection.

3D Cursor:: The current position of the 3D Cursor.
Scene Origin:: The global origin.

Align X, Y, Z

Chooses which axis to align the selected objects on.

Mirror

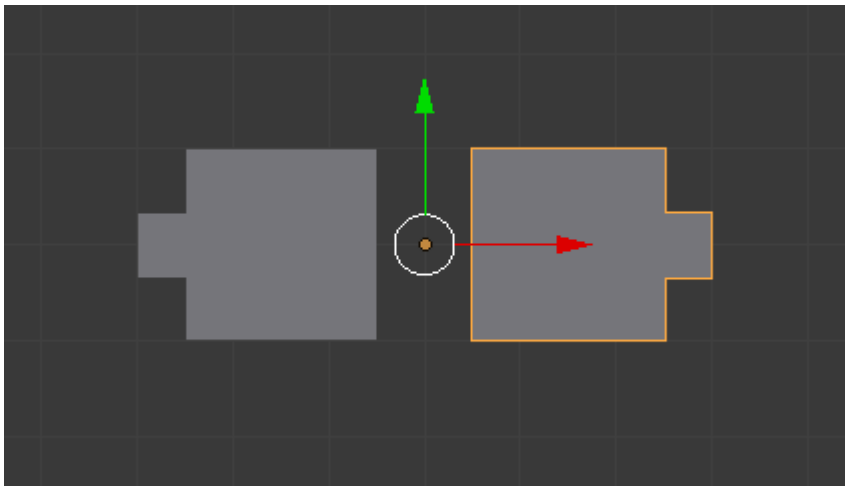
Reference

Mode::	Object and Edit Modes
Menu::	*Object/Mesh ▸ Mirror*
Shortcut::	`Ctrl-M`

Mirroring an object or mesh selection will create a reversed version of the selection. The position of the mirrored version of the selection is determined by the Pivot Point. A common use of mirroring is to model half an object, duplicate it and then use the mirror transform to create a reversed version to complete the model.

Note

Mirrored duplicates can also be created with a Mirror Modifier.

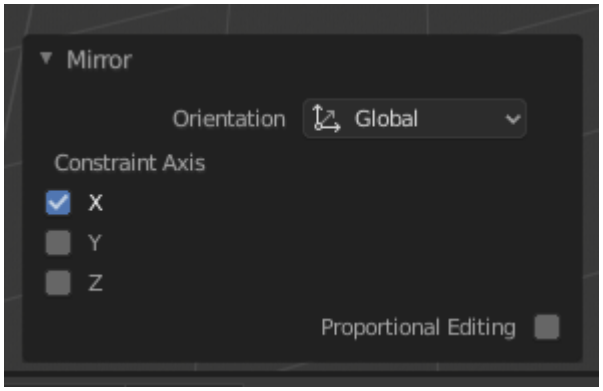


Mirroring a selection.

Usage

To mirror a selection along a particular global axis, press: `Ctrl-M`, followed by `X`, `Y` or `Z`. The image Mirroring a Selection shows the results of this action after a mesh element has been duplicated.

In mesh mode, you can mirror the selection on the currently selected Transform Orientations by pressing the appropriate axis key a second time. For example, if the Transform Orientation is set to *Normal*, pressing: `Ctrl-M`, followed by `X` and then `X` again will mirror the selection along the X axis of the *Normal Orientation*.



Mirror Adjust Last Operation panel.

You can alternatively hold the `MMB` to interactively mirror the object by moving the mouse in the direction of the mirror axis.

Clear

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Clear* ▸ *Location* / *Scale* / *Rotation* / *Origin*
Shortcut::	`Alt-G`, `Alt-S`, `Alt-R`

Clearing transforms resets the transform values. The objects location and rotation values are set to 0, and the scale to 1.

Clear Location `Alt-G`
: Clear (reset) the location of the selection. This will move the selection back to the coordinates (0, 0, 0).

Clear Scale `Alt-S`
: Clear (reset) the scale of the selection. This will change the scale to (1, 1, 1).

Clear Rotation `Alt-R`
: Clear (reset) the rotation of the selection. This will set the rotation of the selection to 0 degrees in each plane.

Clear Origin
: Clears (resets) the offset of the child objects origin from the Parent. This will cause child objects to move to the origin of the parent. The relationship between the parent and child is not affected, you can confirm the relationship is still intact by using the *Outliner* to verify that the child object is still parented.

Options

Clear Delta

Clear the delta transform in addition to clearing the primary transforms. (Appears in the Adjust Last Operation panel.)

Apply

These operations lets you apply several transformations to the selected objects. The object transformation coordinates are transferred to the object data. If the objects have hierarchical descendants, it also applies those transformations to their children.

Transforms

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Apply* ▸ *Location* / *Rotation* / *Scale* / *Rotation & Scale*
Shortcut::	`Ctrl-A`

Applying transform values essentially resets the values of object's location, rotation or scale, while visually keeping the object data in-place. The object origin point is moved to the global origin, the rotation is cleared and scale values are set to 1.

For simple cases you won't notice any difference the 3D Viewport or rendered output, yet modifiers and constraints may depend on object transformation.

Warning

Armature Objects

While applying transformations to armatures is supported, this does **not** apply to their pose location, animation curves or constraints. This tool should be used before rigging and animation.

When applying transforms to an object that shares Object Data between multiple objects, the object must first be made a Single User which can be performed by confirming the pop-up message.

When running *Apply Transform*, the Adjust Last Operation panel lets you choose the combination of transformations to apply.

Options

Location
: Apply (set) the location of the selection. This will make Blender consider the current location to be equivalent to 0 in each plane i.e. the selection will not move, the current location will be considered to be the "default location". The object origin will be set to actual (0, 0, 0) (where the colored axis lines intersect in each view).

Rotation
: Apply (set) the rotation of the selection. This will make Blender consider the current rotation to be equivalent to 0 degrees in each plane i.e. the selection will not rotated, the current rotation will be considered to be the "default rotation".

Scale
: Apply (set) the scale of the selection. This will make Blender consider the current scale to be equivalent to 0 in each plane i.e. the selection will not scaled, the current scale will be considered to be the "default scale".

Rotation and Scale
: Apply (set) the rotation and scale of the selection. Do the above two applications simultaneously.

Apply Properties
: Modify properties such as curve vertex radius, font size and bone envelope according to the applied transformation. (Found in the Adjust Last Operation panel)

Transforms to Deltas

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Apply* ▸ *Location* / *Rotation* / *Scale to Deltas*
Shortcut::	`Ctrl-A`

Converts primary object transformations to delta transforms, any existing delta transforms will be included as well.

- Location to Deltas
- Rotation to Deltas
- Scale to Deltas

All Transforms to Deltas
: Converts all primary transformations to delta transforms.

Animated Transform to Deltas
: Converts the primary transformation animations (of the translation, scale, and, rotation values) to delta transforms.

Options

Reset Values
: Clear primary transform values after transferring to deltas.

Visual Transform

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Apply* ▸ *Visual Transform*
Shortcut::	`Ctrl-A`

Apply (set) the result of a constraint and apply this back to the object's location, rotation and scale.

Visual Geometry as Mesh

Reference

Mode::	Object Mode
Menu::	*Object ▸ Apply ▸ Visual Geometry to Mesh*
Shortcut::	`Ctrl-A`

Apply the visual state of all selected objects (modifiers, shape keys, hooks, etc.) to object data. This is a way to freeze all object data into static meshes, as well as converts non-mesh types to mesh.

For details, see the Convert mesh.

Make Instances Real

Reference

Mode::	Object Mode
Menu::	*Object ▸ Apply ▸ Make Instances Real*

Make Instances Real creates a new object for each instance generated by the selected ones, and removes any direct instancing from those.

In the end, each instance becomes a real object.

Warning

This applies to both direct (from verts or faces…) and indirect (from particle system…) instancing. In case you have tens of thousands of instances (from particles for example), this can significantly slow down

Blender, which does not always deal well with that many objects in a scene.

Options

By default, new objects will be added to the same collection as the one containing their instancer, without keeping any hierarchy relationships. This behavior can be altered with the following options.

Parent

If *Keep Hierarchy* is not set, parents all the generated objects to the former instancer.

Otherwise, parents all the generated objects *which are not already parented* to their respective instancer, or its matching new copy (this is important in case of recursive instancing, see the note below).

Keep Hierarchy

Preserves internal hierarchies (i.e. parent relationships) in the newly generated objects.

Tip

Usually, to get a new hierarchy as close as possible from the instancing one, you'll want to enable both of these options.

Note

Preserving relationships in recursive instancing cases (instancers instancing other instancer objects, etc.) is only supported to some extent currently.

Simple cases (like an empty instancing a collection containing instances of some other collections) will usually work, but more complex cases will fail to fully reproduce the whole instancing hierarchy.

Parent Inverse

Reference

Mode:: Object Mode
Menu:: *Object ▸ Apply ▸ Parent Inverse*

Applies the object's Parent Inverse transform to the object data.

Snap

Reference

Mode::	Object, Edit, and Pose Mode
Menu::	*Object/Object type ▸ Snap*
Shortcut::	`Shift-S`

The *Snap* menu (also available from the 3D header in both *Object Mode* and *Edit Mode Object ▸ Snap* and *Mesh ▸ Snap*). This menu provides a number of options to move the cursor or your selection to a defined point (the cursor, selection or the grid).

Selection to Grid
: Snaps the currently selected object(s) to the nearest grid point.

Selection to Cursor
: Moves each one of the currently selected object(s) to the cursor location.

Selection to Cursor (Offset)
: Places the selection at the position of the 3D cursor. If there are multiple objects selected, they are not moved individually at the cursor position; instead, they are centered around the 3D cursor, maintaining their relative distances.

Selection to Active
: Moves the selection to the origin of the active object.

Cursor to Selected
: Places the cursor to the center of the current selection, unless see below.

Cursor to World Origin
: Places the cursor to the origin of the world (location 0, 0, 0).

Cursor to Grid
Places the cursor to the nearest grid point.

Cursor to Active
Places the cursor to the origin of the *active* (last selected) object.

The *Cursor to Selected* option is also affected by the current Pivot Point. For example:

- With the *Bounding Box Center* pivot point active, the *Cursor to Selected* option will snap the 3D cursor to the center of the bounding box surrounding the objects' origins.
- When the *Median Point* pivot point is selected, *Cursor to Selected* will snap the 3D cursor to the median of the object origins.

Duplicate

Reference

Mode::	Edit and Object Modes
Menu::	*Object ▸ Duplicate Objects*
Shortcut::	`Shift-D`

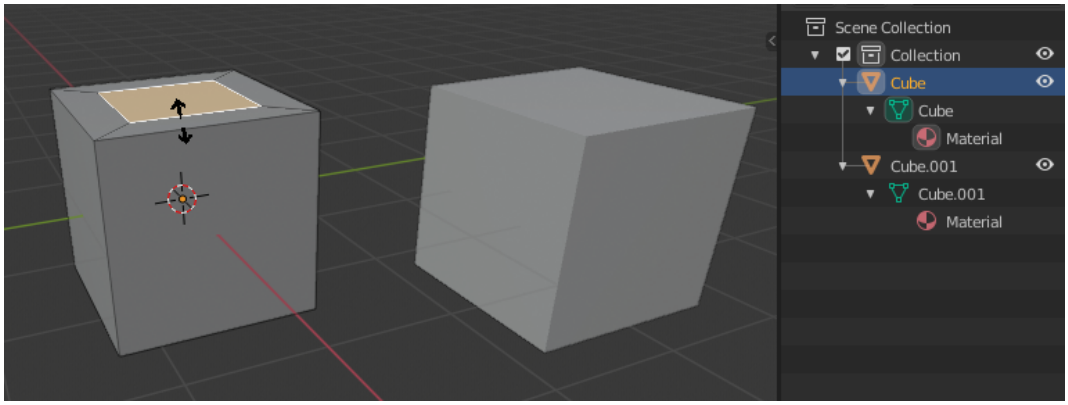
This will create a visually-identical copy of the selected object(s). The copy is created at the same position as the original object and you are automatically placed in move mode. See the examples below.

This copy is a new object, which shares data-blocks with the original object (by default, all the materials, textures, and F-Curves), but which has copied others, like the mesh, for example. That is why this form of duplication is sometimes called “shallow link”, because not all data-blocks are shared; some of them are “hard copied”!

Tip

You can choose which types of data-block will be linked or copied when duplicating in the Preferences.

Examples



The Cube object was duplicated.

The object `Cube` was duplicated, using `Shift-D`. Both these cubes have separate meshes with unique names: `Cube` and `Cube.001`.

- The original left cube is being edited, the duplicated right cube remains unchanged. The mesh data has been copied, not linked.
- Likewise, if one cube is edited in Object Mode, the other cube remains unchanged. The new object's transform properties or data-block is a copy, not linked.
- When the cube was duplicated, it inherited the material of the original cube. The material properties were linked, not copied.

See above if you want separate copies of the data-blocks normally linked.

Duplicate Linked

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Duplicate Linked*
Shortcut::	`Alt-D`

You also have the choice of creating a *Linked Duplicate* rather than a *Duplicate*; this is called a deep link. This will create a new object with **all** of its data linked to the original object. If you modify one of the linked objects in *Edit Mode*, all linked copies are modified. Transform properties (object data-blocks) still remain copies, not links, so you still can rotate, scale, and move freely without affecting the other copies. Reference the Examples for the discussions below.

If the original object was animated, the duplicate will link to the same Action. This means that, even though each object has separate transform properties, they will be set to the same values by the animation system. If this is not desired, make the action a single-user copy in the Action or NLA Editor.

Linked
: In the *Duplicate Objects* Adjust Last Operation panel the *Linked* checkbox is checked unlike with *Duplicate*.

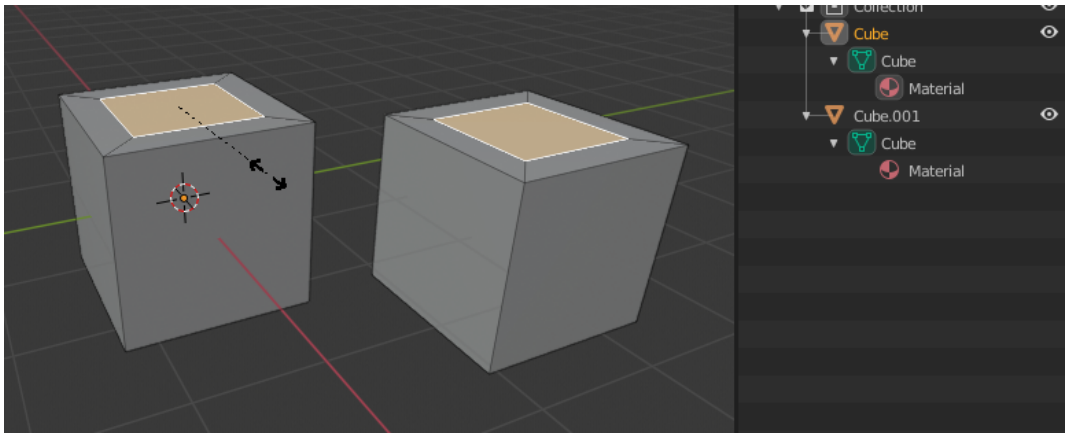
Hint

If you want to make changes to an object in the new linked duplicate independently of the original object, you will have to manually make the object a "single-user" copy by `LMB` the number in the *Object Data* panel of the Properties. (See Data-Block Menu.)

See also

Make Single User for unlinking data-blocks.

Examples



The Cube object was linked duplicated.

The object `Cube` was linked duplicated, using `Alt-D`. Though both these cubes are separate objects with unique names: `Cube` and `Cube.001`, the single mesh named `Cube`, is shared by both.

- As a mesh is edited in *Edit Mode* in one object, the same occurs in the other cube as well. The mesh data are links, not copies.
- In contrast, if one of these two cubes is rotated or scaled in *Object Mode*, the other remains unchanged. The transform properties are copied, not linked.
- As in the previous example, the newly created cube has inherited the material of the original cube. The material properties are linked, not copied.

A common table has a top and four legs. Model one leg, and then make linked duplicates three times for each of the remaining legs. If you later make a change to the mesh, all the legs will still match. Linked duplicates also apply to a set of drinking glasses, wheels on a car… anywhere there is repetition or symmetry.

See also

Linked Library Duplication

Linked Libraries are also a form of duplication. Any object or data-block in other blend-files can be reused in the current file.

Hint

If you want transform properties (i.e. object data-blocks) to be “linked”, see the page on parenting.

Join

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Join*
Shortcut::	`Ctrl-J`

Join merges all selected objects into the last selected *Active* object. All object data is linked to the active object (which must be selected). All objects must be of the same type: mesh, curve, surface or armature. If several curves are joined, each one will keep its subtype (NURBS or Bézier).

Note

Object data has many attributes which may be handled when joining.

Materials, vertex groups, UV and Vertex layers will be merged.

Modifiers, constraints, groups and parent relationships are ignored when joining and will not be applied to the active object.

Asset

Operations for managing the Asset status of an object.

Mark as Asset

See Creating an Asset.

Clear Asset

See Removing Assets.

Clear Asset (Set Fake User)

See Removing Assets.

Parenting Objects

When modeling a complex object, such as a watch, you may choose to model the different parts as separate objects. To make all the parts move as one ("the watch"), you can designate one object as the *parent* of all the other parts. These other parts become its *children*, and any translation, rotation, or scale of the parent will also affects its children.

Contrary to most biological lifeforms, each object or bone in Blender has at most one parent. If an object already has a parent object and you give it another parent then Blender will remove the previous parent relationship. When the plural "parents" is used in this chapter, it references the hierarchy of parents, so the parent, the grandparent, great grandparent, and so on, of an object.

Make Parent

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Parent*
Shortcut::	`Ctrl-P`

To parent objects, select at least two objects (select the child objects first, and select the parent object last), and press `Ctrl-P`. The *Set Parent To* menu will pop up allowing you to select from one of several possible different parenting types. Selecting one of the entries in *Set Parent To* confirms, and the child/children to parent relationship is created. The selected objects will have their 'parent' set to the active object, and as a result will be 'siblings'.

The *Set Parent To* pop-up menu is context-sensitive, which means the number of entries it displays can change depending on what objects are selected when the `Ctrl-P` shortcut is used.

Moving, rotating or scaling the parent will also usually transform the child/children. Yet transforming the child/children of the parent will not affect the parent. In other words, the direction of influence is from parent to child and not child to parent.

Tip

You can "move" a child object back to its parent by clearing its origin.

Type
: Blender supports many different types of parenting, listed below. Besides parenting the selected objects, some types add a Modifier or Constraint to the child objects, with the parent as the target object or activates a parent property i.e. *Follow Path*.

  - Object
  - Armature Deform
  - Bone
  - Curve Deform
  - Follow Path
  - Path Constraint
  - Lattice Deform
  - Vertex
  - Vertex (Triangle)

Keep Transform
: The object's current world transform (so its absolute location, rotation and scale in the world) is computed. The new parent is set, and then the *Parent Inverse* matrix is computed such that after setting the new parent the object is still at its previous world transform.

Hint

Use the Outliner

There is another way to see the parent-child relationship in groups and that is to use the *Outliner* view of the Outliner editor.

Parent Inverse

Blender can assign a parent without moving the child object. This is achieved via a hidden matrix called the *Parent Inverse* matrix, which sits between the transform of the parent and the child.

When objects are parented with `Ctrl-P`, Parent Inverse matrix is updated. Depending on the choice in the Set Parent menu, the object's local location, rotation, and scale are also updated. For more details, see Object Parent.

The Parent Inverse matrix can be cleared by using Clear Parent Inverse.

Note

When setting the parent via the Object Properties panel, the Parent Inverse matrix is always reset. This can cause an unexpected jump in the object's position. To avoid this, use `Ctrl-P` to set the new parent.

Object Parent

Object Parent is the most general form of parenting that Blender supports. It will take selected objects and make the active object the parent object of all the selected objects. Each child object will inherit the transformations of the parent. The parent object can be of any type.

If the object has a preexisting parent, that is cleared first. This moves the object to its own location, rotation and scale, without its parent's influence.

There are three operators that allow you to set an object parent. They differ in the way they compute the Parent Inverse matrix and the local Transform of the object.

Example: Object Parent (Keep Transform)

Object Parent with *Keep Transform* will keep any previous transformations applied to them from the previous parent object.

Assume that we have a scene consisting of three objects, those being two empty objects named “EmptyA” and “EmptyB”, and a Monkey object. Fig. Scene with no parenting. shows the three objects with no parenting relationships active on them.



Scene with no parenting.

If you select the Monkey object by `LMB` click and then `Shift-LMB` click “EmptyA” object and press `Ctrl-P` and finally select *Object* from the *Set Parent To* pop-up menu. This will result in “EmptyA” object being the parent object of the Monkey object. With only “EmptyA” selected rotating/scaling/moving it will result in the Monkey object being altered respectively.

Scale the “EmptyA” object, so that the Monkey becomes smaller and moves to the left a little.



The monkey is the child object of “EmptyA”.

If you select only the Monkey object by `LMB` click and then `Shift-LMB` click “EmptyB” object and press `Ctrl-P` and select *Object* from the *Set Parent To* pop-up menu. This will result in “EmptyB” object being the parent object of the Monkey object. Notice that when you change the parent of the Monkey the scale of the Monkey changed.

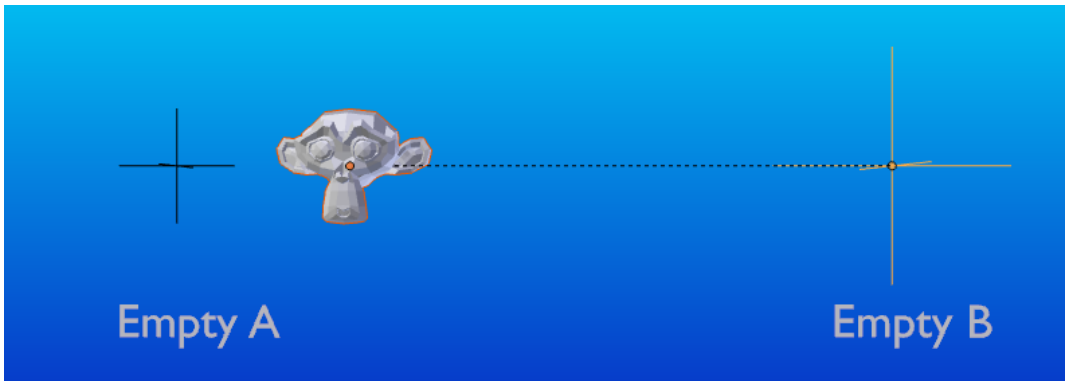


The monkey is the child object of “EmptyB”.

This happens because the Monkey object never had its scale altered directly, the change came about because it was the child of “EmptyA” which had its scale altered. Changing the Monkey’s parent to “EmptyB” resulted in those indirect changes in scale being removed, because “EmptyB” has not had its scale altered.

This is often the required behavior, but it is also sometimes useful that if you change your parent object that the child object keep any previous transformations it got from the old parent object; If instead when changing

the parent object of the Monkey from “EmptyA” to “EmptyB” we had chosen parenting type *Object* and enable *Keep Transform*, the Monkey would keep its scale information it obtained from the old parent “EmptyA” when it is assigned to the new parent “EmptyB”.



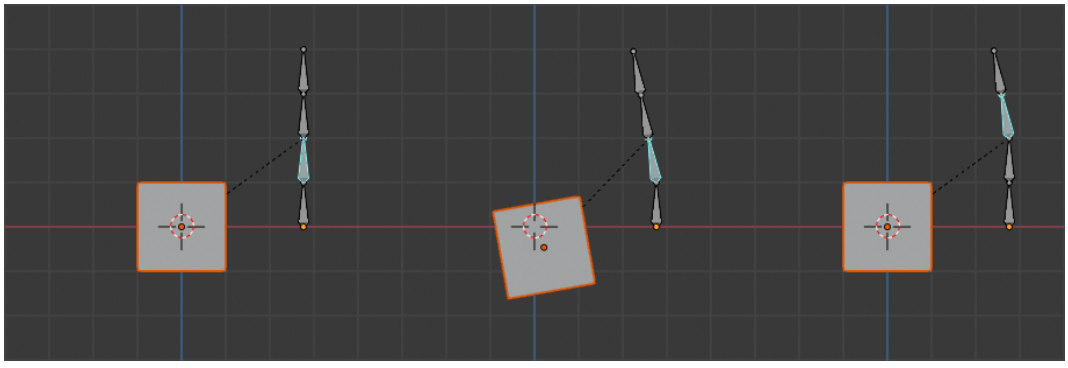
The Object parent with *Keep Transform*.

If you want to follow along with the above description here is the blend-file:

File:Parent - Object (Keep Transform) (Demo File).blend.

Bone Parent

Bone parenting allows you to make a certain bone in an armature the parent object of another object. This means that when transforming an armature the child object will only move if the specific bone is the child object of moves.



Three pictures of armatures with four bones.

In Fig. Three pictures of armatures with four bones. with the 2nd bone being the bone parent of the child object cube. The cube is only transformed if the 1st or 2nd bones are. Notice altering the 3rd and 4th bones has no effect on the cube.

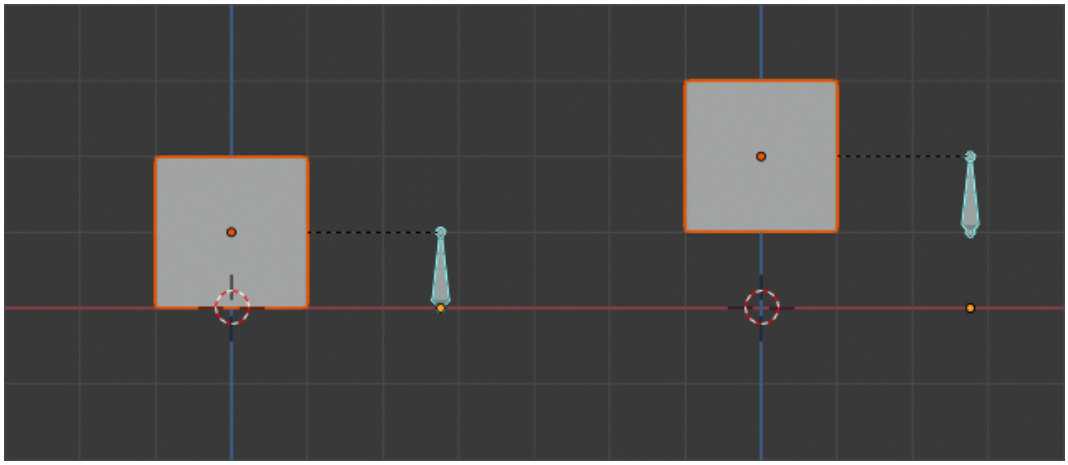
To use bone parenting, you must first select all the child objects you wish to parent to a specific armature bone, then `Shift-LMB` select the armature object and switch it into Pose Mode and then select the specific bone you wish to be the parent bone by `LMB` selecting it. Once done press `Ctrl-P` and select bone from the *Set Parent To* pop-up menu.

Now transforming that bone in Pose Mode will result in the child objects also transforming.

Relative Parenting

Bone relative parenting is an option you can toggle for each bone. This works in the same way as bone parenting with one difference.

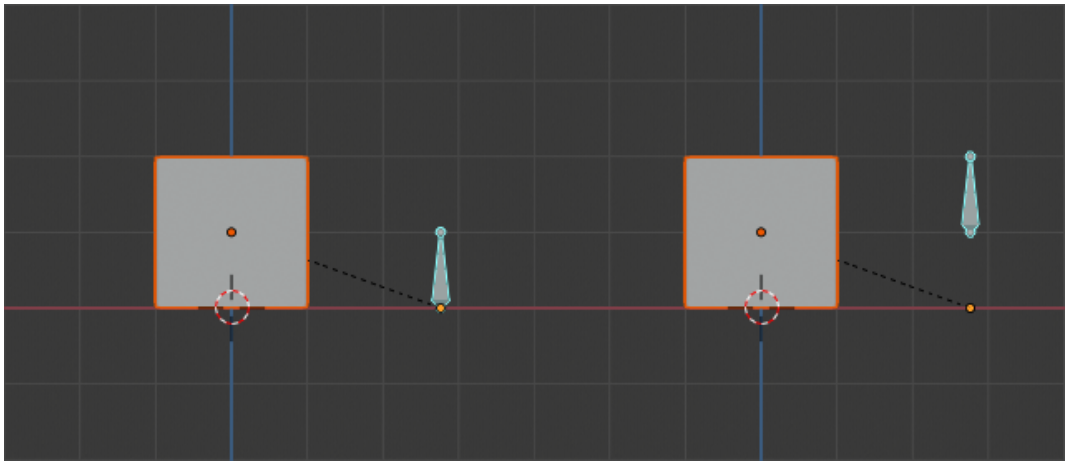
With bone parenting if you have parented a bone to some child objects and you select that bone and switch it into Edit Mode and then move that bone; When you switch back into Pose Mode on that bone, the child object which is parented to that bone will snap back to the location of the bone in Pose Mode.



Single armature bone which has a child object cube parented to it using bone parenting.

In Fig. Single armature bone which has a child object cube parented to it using bone parenting. the 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the child object moves to the new location of the pose bone.

Bone relative parenting works differently; If you move a parent bone in Edit Mode, when you switch back to Pose Mode, the child objects will not move to the new location of the Pose Bone.



Single bone with bone relative parent to a cube.

In Fig. Single bone with bone relative parent to a cube. the 1st picture shows the position of the cube and armature before the bone is moved in Edit Mode. 2nd picture shows the position of the cube and armature after the bone was selected in Edit Mode, moved and switched back into Pose Mode. Notice that the child object does not move to the new location of the pose bone.

Note

When using `Ctrl-P` to set parents, choosing “Bone” or “Bone Relative” will respectively clear and set the bone’s “Relative Parenting” option.

Since “Relative Parenting” is an option that is set per bone, this influences all child objects of that bone at once.

Vertex Parent

For objects of type curve, surface, mesh and lattice, there is the possibility to use one of its vertices or points as the parent of other objects. You can parent an object to a single vertex or a group of three vertices as well; that way the child/children will move when the parent mesh is deformed.

Vertex Parent from Edit Mode

In *Object Mode*, select the child/children and then the parent object. `Tab` into *Edit Mode* and on the parent object select either one vertex that defines a single point, or select three vertices that define an area (the three vertices do not have to form a complete face; they can be any three vertices of the parent object), and then press `Ctrl-P` and confirm.

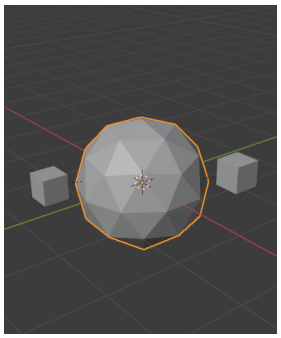
At this point, if a single vertex was selected, a relationship/parenting line will be drawn from the vertex to the child/children. If three vertices were selected then a relationship/parenting line is drawn from the averaged center of the three points (of the parent object) to the child/children. Now, as the parent mesh deforms and the chosen parent vertex/vertices move, the child/children will move as well.

Vertex Parent from Object Mode

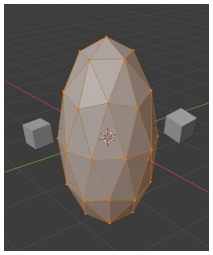
Vertex parenting can be performed from Object Mode, this is done like regular object parenting, press `Ctrl-P` in Object Mode and select *Vertex* or *Vertex (Triangle)*.

The nearest vertices will be used from each object which is typically what you would want.

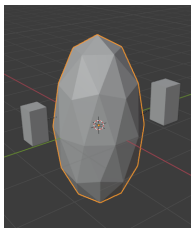
Vertex Parent example.



The small cubes can each be automatically parented to a triad of nearby vertices on the icosphere using the “Vertex (Triangle)” in the set parent context menu.



Reshaping the object in Edit Mode then means each of the cubes follows their vertex parent separately.



Scaling the parent icosphere in Object Mode means the child cubes are also scaled as expected.

The parent context menu item means users can rapidly set up a large number of vertex parent relationships, and avoid the tedious effort of establishing each parent-child vertex relationship separately.

Note

It is in fact a sort of “reversed” hook.

Make Parent without Inverse

Reference

Mode:: Object Mode

Menu:: *Object ▸ Parent ▸ Make Parent without Inverse*

This sets the parent, and then resets the *Parent Inverse* matrix and the object's local location. As a result, the object will move to the location of the parent, but keep its rotation and scale.

Keep Transform
: The object's current world transform (so its absolute location, rotation and scale in the world) is computed. The new parent is set, and then the local transform values are set in such a way that after setting the new parent the object is still at its previous world transform.

Clear Parent

Reference

Mode:: Object Mode
Menu:: *Object ▸ Parent*
Shortcut:: `Alt-P`

You can *remove* a parent-child relationship via `Alt-P`.

Clear Parent
: If the parent in the group is selected, nothing is done. If a child or children are selected, they are disassociated from the parent, or freed, and they return to their *original* location, rotation, and size.

Clear and Keep Transformation
: Frees the children from the parent, and *keeps* the location, rotation, and size given to them by the parent.

  See Non-Uniform Scale which may apply here.

Clear Parent Inverse
: Instead of removing the hierarchical parent-child relationship, this clears the Parent Inverse matrix from the selected objects. With an

Modifiers

Operators for working with an object's Modifiers.

Add Modifier

Reference

Mode:: Object Mode
Menu:: *Object ▸ Modifiers ▸ Add Modifier*

Opens a menu with a list of modifiers, selecting a modifier will add it to the bottom of The Modifier Stack.

Copy Modifiers to Selected Objects

Reference

Mode:: Object Mode
Menu:: *Object ▸ Modifiers ▸ Copy Modifiers to Selected Objects*

Copies all modifiers from the Active object to the other selected objects.

Clear Object Modifiers

Reference

Mode:: Object Mode
Menu:: *Object ▸ Modifiers ▸ Clear Object Modifiers*

Deletes all modifiers from the selected objects.

Constraints

Operators for working with an object's Constraints.

Add Constraint (with Targets)

Reference

Mode::	Object Mode and Pose Mode
Menu::	*Object ▸ Constraint ▸ Add Constraint (with Targets)*

Adds a constraint to the active object. The type of constraint must be chosen from a pop-up menu, though it can be changed later from the *Add Constraint (with Targets)* Adjust Last Operation panel. If there is an other object selected besides the active one, that object will be the constraint target (if the chosen constraint accepts targets).

When using a bone from another armature as the target for a constraint, the tool will look inside the non-active armature and use its active bone, provided that armature is in Pose Mode.

Copy Constraints to Selected Objects

Reference

Mode::	Object Mode and Pose Mode
Menu::	*Object ▸ Constraint ▸ Copy Constraints to Selected Objects*

Copies the active object Constraints to the rest of the selected objects.

Clear Object Constraints

Reference

Mode::	Object Mode and Pose Mode
Panel::	*Object ▸ Constraint ▸ Clear Object Constraints*

Removes all Constraints of the selected object(s).

Track

Reference

Mode::	Object Mode
Panel::	*Object* ▸ *Track*

These tools add a tracking constraint to the selected objects; the target object of the constraint will be the active object, which won't have a constraint added.

- Damped Track Constraint
- Track To Constraint
- Lock Track Constraint

Clear Track

Removes all Damped Track, Track To and Lock Track Constraints from the selected objects.

Clear and Keep Transformation (Clear Track)

Removes all Track Constraint from the selected objects, while keeping the final transform caused by them.

Relations

- Make Single User

Make Single User

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Relations* ▸ *Make Single User*

Makes the selected or all object data-blocks single users, that is, not shared (linked) between other objects in the blend-file.

Additionally, it can also make single-user copies of its dependencies, like meshes, curves, materials, animations…

Type
: These actions work on the selected objects, or on all the objects of the scene.

  All, Selected Objects

Data-blocks
: Lets you, in addition to the menu predefined selection, choose the type of data-blocks individually.

  Object:: Make single user objects.
  Object Data:: Make single user object data.
  Materials:: Make materials local to each data-block.
  Object Animation:: Make the animation of Object Properties data local to each object.
  Object Data Animation:: Make object data (mesh, curve etc.) animation data local to each object.

See also

Making Single User

Link/Transfer Data

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Link/Transfer Data*
Shortcut::	`Ctrl-L`

- Link Objects to Scene
- Link Data
  - Link Object Data
  - Link Materials
  - Link Animation Data
  - Link Collections
  - Link Instance Collection
  - Link Fonts to Text
  - Copy Modifiers
  - Copy Grease Pencil Effects
- Copy UV Maps
- Transfer Mesh Data
  - Vertex Mapping
  - Further Options
- Transfer Mesh Data Layout

Link Objects to Scene

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Link/Transfer Data* ▸ *Link Objects to Scene*

Links the selected objects into a different scene than the current one. The *Link Objects to Scene* in the Adjust Last Operation panel lets you choose between scenes.

This makes the same object exist in more than one scene at once, including its position and animation data. The object's origin will change its color to reflect that.

Link Data

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Link/Transfer Data…*

Links objects between scenes or data-blocks of the active object to all selected objects. In some case (i.e. Object Data, Modifier) the target objects must be of the same type as the active one or capable of receiving the data. If targets already have data linked to them, it will be unlinked first.

Type
: Data-block type to link.

- Link Object Data
- Link Materials
- Link Animation Data
- Link Collections
- Link Instance Collection
- Link Fonts to Text
- Copy Modifiers
- Copy Grease Pencil Effects

See also

Making Single User for unlinking data-blocks.

Link Object Data

Replace assigned Object Data.

Link Materials

Replace assigned Materials.

Link Animation Data

Replace assigned Animation Data.

Link Collections

Replace assigned Collections.

Link Instance Collection

Replace assigned Collection Instance.

Link Fonts to Text

Replace Text object Fonts.

Copy Modifiers

Replace Modifiers.

Copy Grease Pencil Effects

Replace Grease Pencil Effects.

Copy UV Maps

Reference

Mode::	Object Mode
Menu::	*Object ▸ Link/Transfer Data ▸ Copy UV Maps*

The active UV map of the selected objects will be replaced by a copy of the active UV map of the active object. If the selected object doesn't have any UV maps, it is created. Objects must be of type mesh and must have a matching topology.

Transfer Mesh Data

Reference

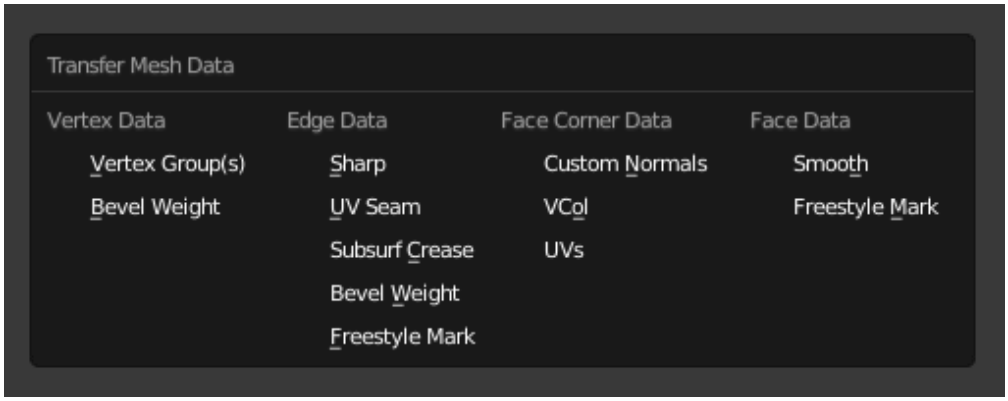
Mode::	Object Mode
Menu::	*Object* ▸ *Link/Transfer Data* ▸ *Transfer Mesh Data*

The *Data Transfer* tool transfers several types of data from one mesh to another. Data types include vertex groups, UV maps, Color Attributes, custom normals… Transfer works by generating a mapping between source mesh's elements (vertices, edges, etc.) and destination ones, either on a one-to-one basis, or mapping several source elements to a single destination one by interpolated mapping.

Transfers data layer(s) from active to selected meshes.

Freeze Operator
: Prevent changes to settings to re-run the operator. This is useful if you are editing several settings at once with heavy geometry.

Data Type
: Which data to transfer.



Data types.

Create Data

Add data layers on destination meshes if needed.

Vertex Mapping
: Method used to map source vertices to destination ones. Because the options change depending on the *Data Type* options are explained in Vertex Mapping below.

Vertex Mapping

Topology

The simplest option, expects both meshes to have identical number of elements, and match them by order (indices). Useful e.g. between meshes that were identical copies, and got deformed differently.

One-To-One Mappings

Those always select only one source element for each destination one, often based on shortest distance.

Vertices
: Nearest Vertex
  : Uses source's nearest vertex.

  Nearest Edge Vertex
  : Uses source's nearest vertex of source's nearest edge.

  Nearest Face Vertex
  : Uses source's nearest vertex of source's nearest face.

Edges
: Nearest Vertices
  : Uses source's edge which vertices are nearest from destination edge's vertices.

  Nearest Edge
  : Uses source's nearest edge (using edge's midpoints).

Nearest Face Edge
: Uses source's nearest edge of source's nearest face (using edge's midpoints).

Face Corners
: A face corner is not a real element by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, …) aspects are used to match them together.

Nearest Corner and Best Matching Normal
: Uses source's corner having the most similar *split* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner and Best Matching Face Normal
: Uses source's corner having the most similar *face* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner of Nearest Face
: Uses source's nearest corner of source's nearest face.

Faces
: Nearest Face
: Uses source's nearest face.

Best Normal-Matching:
: Uses source's face which normal is most similar with destination one.

Interpolated Mappings

Those use several source elements for each destination one, interpolating their data during the transfer.

Vertices
: Nearest Edge Interpolated
: Uses nearest point on nearest source's edge, interpolates data from both source edge's vertices.

Nearest Face Interpolated

Uses nearest point on nearest source's face, interpolates data from all that source face's vertices.

Projected Face Interpolated

Uses point of face on source hit by projection of destination vertex along its own normal, interpolates data from all that source face's vertices.

Edges

Projected Edge Interpolated

This is a sampling process. Several rays are cast from along the destination's edge (interpolating both edge's vertex normals), and if enough of them hit a source's edge, all hit source edges' data are interpolated into destination one.

Face Corners

A face corner is not a real element by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, …) aspects are used to match them together.

Nearest Face Interpolated

Uses nearest point of nearest source's face, interpolates data from all that source face's corners.

Projected Face Interpolated

Uses point of face on source hit by projection of destination corner along its own normal, interpolates data from all that source face's corners.

Faces

Projected Face Interpolated

This is a sampling process. Several rays are cast from the whole destination's face (along its own normal), and if enough of them hit a source's face, all hit source faces' data are interpolated into destination one.

Further Options

Auto Transform

Automatically computes the transformation to get the best possible match between source and destination meshes.

This allows to match and transfer data between two meshes with similar shape, but transformed differently. Note that you'll get best results with exact copies of the same mesh. Otherwise, you'll likely get better results if you "visually" make them match in 3D space (and use *Object Transform*) instead.

Object Transform

Evaluate source and destination meshes in global space.

Only Neighbor Geometry

Source elements must be closer than given distance from destination one.

Max Distance

Maximum allowed distance between source and destination element (for non-topology mappings).

Ray Radius

The starting ray radius to use when Ray Casting against vertices or edges. When transferring data between meshes Blender performs a series of ray casts to generate mappings. Blender starts with a ray with the radius defined here, if that does not detect a hit then the radius is progressively increased until a positive hit or a limit is reached.

This property acts as an accuracy/performance control; using a lower ray radius will be more accurate however, might take longer if Blender has to progressively increase the limit. Lower values will work better for dense meshes with lots of detail while larger values are probably better suited for simple meshes.

Mix Mode

How to affect destination elements with source values.

All
: Replaces everything in destination (note that *Mix Factor* is still used).

Above Threshold
: Only replaces destination value if it is above given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for Boolean values this option fakes a logical AND.

Below Threshold
: Only replaces destination value if it is below given threshold *Mix Factor*. How that threshold is interpreted depends on data type, note that for Boolean values this option fakes a logical OR.

Mix, Add, Subtract, Multiply
: Apply that operation, using mix factor to control how much of source or destination value to use. Only available for a few types (vertex groups, Color Attributes).

Mix Factor
: How much of the transferred data gets mixed into existing one (not supported by all data types).

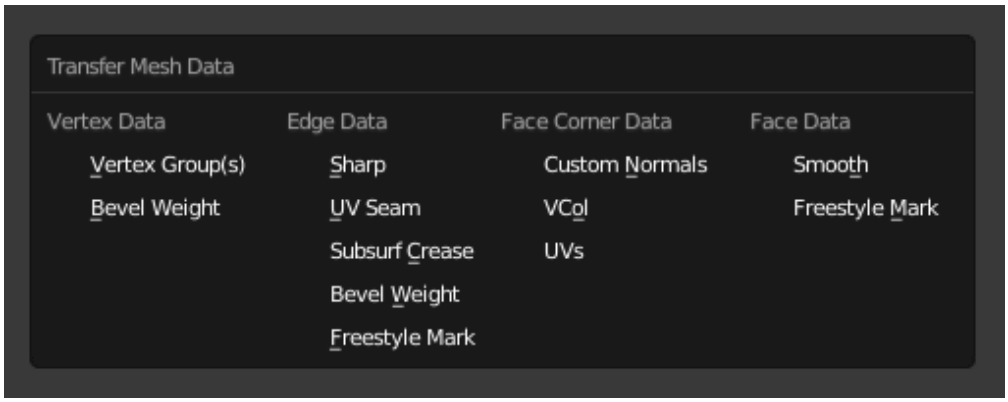
Transfer Mesh Data Layout

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Link/Transfer Data* ▸ *Transfer Mesh Data Layout*

Transfers layout of data layer(s) from active to selected meshes.

Data Type
: Which data to transfer.



Data types.

Exact Match
: Also Delete some data layers from destination if necessary, so that it matches the source exactly.

Source Layers Selection
: Which layers to transfer, in case of multi-layer types.

  Active Layer
  : Only transfer the active data layer.

  All Layers

Transfer all data layers.

Destination Layers Matching

How to match source and destination layers.

By Name

Match target data layers to affect by name.

By Order

Match target data layers to affect by order (indices).

See also

Data Transfer Modifier

Shading

Shade Smooth

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Shade Smooth*

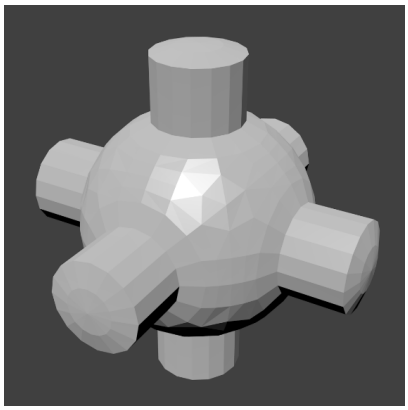
Sets an entire object as smooth or faceted. This forces the assignment of the "smoothing" attribute to each face in the mesh, including when you add or delete geometry.

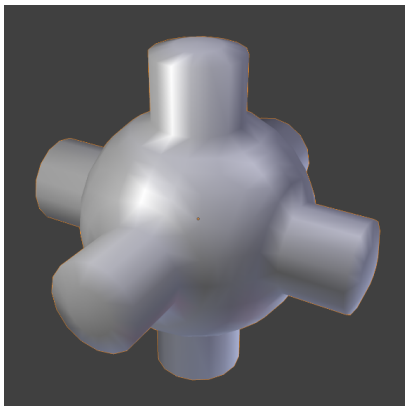
This operator will also remove any Smooth By Angle Modifiers.

Notice that the outline of the object is still strongly faceted. Activating the smoothing features does not actually modify the object's geometry; it changes the way the shading is calculated across the surfaces (normals will be interpolated), giving the illusion of a smooth surface.

Using Shade Flat will revert the shading back (normals will be constant) to that shown in the first image below.

Example mesh flat (left) and smooth-shaded (right). Sample blend-file.





Keep Sharp Edges
: Do not clear sharp edges (which are redundant with objects shaded as flat or smooth). This option is useful to not destroy data in case you want to revert changes later.

Shade Auto Smooth

Reference

Mode::	Object Mode
Menu::	*Object* ‣ *Shade Auto Smooth*

Adds a Smooth By Angle Modifier to automatically set the sharpness of mesh edges based on the angle between the neighboring faces. Note, the modifier will be pinned to be the last modifier.

Auto Smooth
: If disabled, any Smooth By Angle Modifiers are removed.

Angle
: Maximum angle between face normals that will be considered as smooth.

Shade Flat

Rigid Body

Calculate Mass

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	*Object ▸ Rigid Body ▸ Calculate Mass*

Calculate mass values for rigid body objects based on their volume and density. The volume is calculated automatically, the density needs to be given based on the object you want to simulate.

Material Preset
: A list of preset density values for real-world materials, if a material is not given you can research the density and use the *Custom* preset to input the density manually.

Density
: When the *Custom Material Preset* is selected, this is the input density, in kg/m^3, to use.

Convert

Curve

Reference

Mode::	Object Mode
Menu::	*Object ▸ Convert ▸ Curve*

Converts the selected mesh, or text object to a curve object. For mesh objects, only edges belonging to no faces will be taken into account. The resulting curve will be a poly curve type, but can be converted to have smooth segments as described in Set Spline Type.

Mesh

Reference

Mode::	Object Mode
Menu::	*Object ▸ Convert ▸ Mesh*

Converts the selected curve, metaball, surface, or text object to a mesh object. The actual defined resolution of these objects will be taken into account for the conversion. Note that it also keeps the faces and volumes created by closed and extruded curves.

Grease Pencil

Reference

Mode::	Object Mode
Menu::	*Object ▸ Convert ▸ Grease Pencil*

Converts the selected curve, mesh or text object to a Grease Pencil object with strokes matching the curve, mesh, or text; basic materials are also added. When multiple curves, meshes, or texts are selected, they are all converted into the same Grease Pencil object.

Options

Keep Original
: Duplicates the original object before converting it.

Thickness
: Strokes thickness.

Threshold Angle
: Threshold value that determines the strokes end.

Stroke Offset
: Sets offset to separate strokes from filled strokes.

Only Seam Edges
: Convert only edges marked as seam.

Export Faces
: Convert faces as filled strokes.

Trace Image to Grease Pencil

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Convert* ▸ *Trace Image to Grease Pencil*

See Trace Image to Grease Pencil.

Convert to Mesh Plane

Reference

Mode::	Object Mode
Menu::	*Object ▸ Convert ▸ Convert to Mesh Plane*

Converts the selected image empty to a textured mesh plane.

For a description of the options see Import Images as Planes.

Show/Hide

Reference

Mode::	All Modes
Menu::	*Object* ▸ *Show/Hide*

Show Hidden Objects `Alt-H`
: Reveals all hidden objects.

Hide Selected `H`
: Hides all selected objects.

Hide Unselected `Shift-H`
: Hides all unselected objects of the scene.

Clean Up

Clean Vertex Group Weights

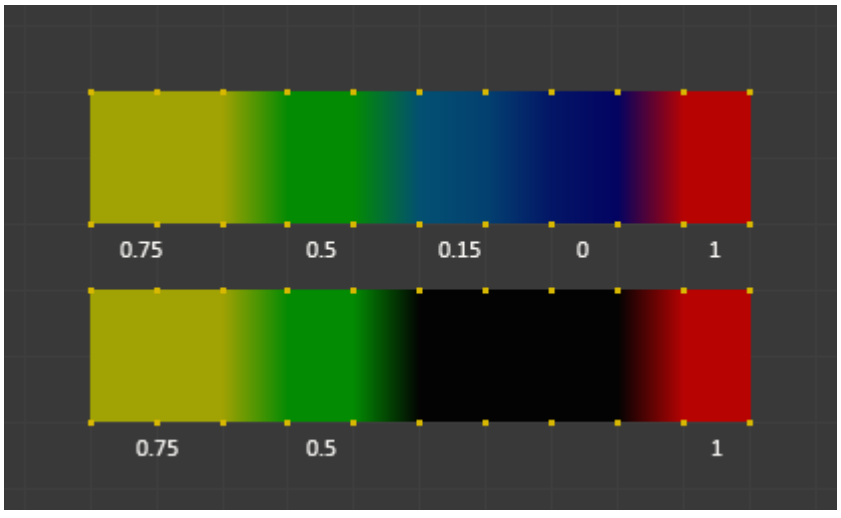
Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Clean Up* ▸ *Clean Vertex Group Weights*

Clean Vertex Group Weights unassigns vertices from Vertex Groups whose weights are below the *Limit*. Removes weights below a given threshold. This tool is useful for clearing your weight groups of very low (or zero) weights.

In the example shown, a cutoff value of 0.2 is used (see operator options below) so all blue parts are cleaned out.

Note, the images use the *Show Zero weights* Active option so that unreferenced Weights are shown in Black.



Clean example.

Subset
: Restrict the tool to a subset. See above The Subset Option for how subsets are defined.

Limit
: This is the minimum weight value that will be kept in the group. Weights below this value will be removed from the group.

Keep Single
: Ensure that the *Clean Vertex Group Weights* tool will not create completely unreferenced vertices (which are vertices that are not assigned to any vertex group), so each vertex will keep at least one weight, even if it is below the limit value!

Limit Total Vertex Groups

Reference

Mode::	Object Mode
Menu::	*Object ▸ Clean Up ▸ Limit Total Vertex Groups*

Reduce the number of weight groups per vertex to the specified Limit. The tool removes lowest weights first until the limit is reached.

Hint

The tool can only work reasonably when more than one weight group is selected.

Subset
: Restrict the tool to a subset. See above The Subset Option for how subsets are defined.

Limit
: Maximum number of weights allowed on each vertex.

Remove Unused Material Slots

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Clean Up* ▸ *Remove Unused Material Slots*

Removes unused material slots.

Delete

Reference

Mode::	Object Mode
Menu::	*Object ▸ Delete*
Shortcut::	`X` or `Delete`

Delete the selected objects from the current scene.

Delete Globally

Reference

Mode::	Object Mode
Menu::	*Object ▸ Delete Globally*
Shortcut::	`Shift-X` or `Shift-Delete`

Delete the selected objects from all scenes, and any other possible usages (like e.g. from a shading node).

Properties

- Transform
  - Delta Transforms
- Relations
- Collections
  - Introduction
  - Collections
- Instancing
  - Vertices
  - Faces
  - Collection
- Visibility
- Viewport Display
- Line Art

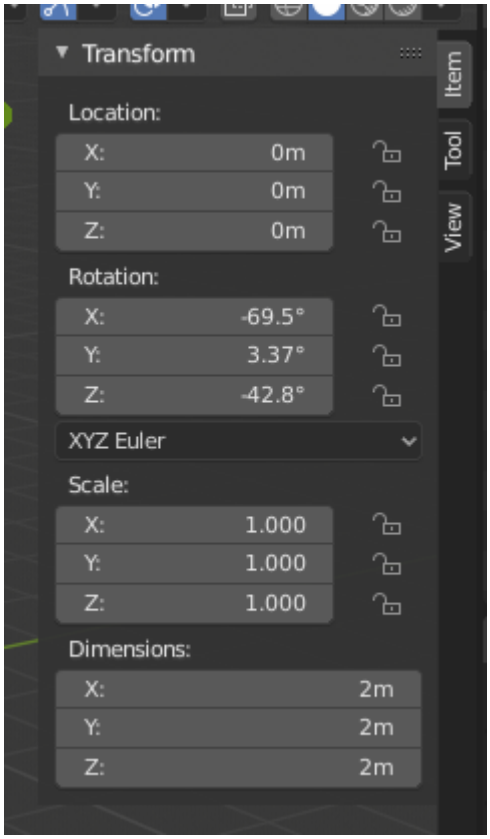
Transform

Reference

Mode::	Object Mode
Panel::	*Properties* ▸ *Object Properties* ▸ *Transform*
Panel::	*3D Viewport* ▸ *Sidebar* ▸ *Transform*

The *Transform* panel in the Sidebar region allows you to view and manually/numerically control the position, rotation, and other properties of an object, in *Object Mode*. Each object stores its position, orientation, and scale values. These may need to be manipulated numerically, reset, or applied. In *Edit Mode*. It mainly allows you to enter precise coordinates for a vertex, or median position for a group of vertices (including an edge/face). As each type of object has a different set of options in its *Transform* panel in *Edit Mode*, see their respective descriptions in the Modeling chapter.

Use this panel to either edit or display the object's transform properties such as position, rotation and/or scaling. These fields change the object's origin and then affect the aspect of all its *vertices* and faces.



Transform Properties.

Location
: The object's origin location in local coordinates.

Rotation
: The object's local orientation, relative to the global axes and its own origin.

Rotation Mode
: Method for calculating rotations, additional information can be found in the manual's appendix.

Euler:: The gizmo handles are aligned to the Euler axis, allowing you to see the discreet XYZ axis underlying the Euler rotation, as well as possible Gimbal Lock.

Axis Angle::

The X, Y, and Z coordinates define a point relative to the object origin. This point and the origin define an axis around the W value defines the rotation.

Quaternion:: X, Y, Z and W correspond to the Quaternion components.

Scale
: The object's relative scale along the local axis (e.g. the *Scale X* value represents the scale along the local X axis). Each object (cube, sphere, etc.), when created, has a scale of one unit in each local direction. To make the object bigger or smaller, you scale it in the desired axis.

Dimensions
: The size of the object's bounding box (aligned with the local axes – think of a cardboard box just big enough to hold the object).

Transform Properties Locking
: When the toggle is locked, the corresponding transformation value can not be changed in any interactive operation. But the value can still be changed using non-interactive operations, like editing the corresponding number field or using Python.

  For example, if you locked the *Location X* property then you cannot use the 3D gizmo to move the object along the global X axis. But you can still move it using the *Location X* number field. Consider the locking feature as a rigid constraint only changeable from the panel.

  To lock a property, click the padlock icon next to the button. The button is unlocked if the icon shows an open padlock, and it is locked if the icon appears as a closed padlock.

Delta Transforms

Reference

Mode:: Object Mode

Panel:: *Properties ▸ Object Properties ▸ Transform ▸ Delta Transforms*

Delta Transforms are simply transformations that are applied on top of the transforms described above. Delta Transforms are particularly useful in animations. For example, you can animate an object with the primary transforms then move them around with Delta Transforms.

Relations

Reference

Mode::	Object Mode
Panel::	*Properties* ‣ *Object Properties* ‣ *Relations*

Parent
: The object to which the selected object is parented to.

Parent Type
: The type of parenting used. See parenting for information on the different types.

Camera Parent Lock
: When the camera is locked to the view, the root parent is transformed rather than the camera. This is useful for camera rigs where you don't want to animate the camera directly.

Tracking Axis
: Axis that points in the "forward" direction. Applies to *Instance Vertices* when *Align to Vertex Normal* is enabled.

Up Axis
: Axis that points in the "upward" direction. Applies to *Instance Vertices* when *Align to Vertex Normal* is enabled.

Pass Index
: Defines the index the object will have in the Object Index render pass. See passes and ID mask for more information.

  Note

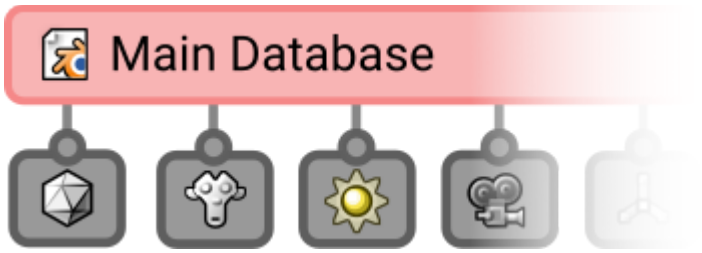
  Volume Objects are not supported.

Collections

- Introduction
  - Collections
  - Naming and Nesting
  - Color Tagging
- Collections
  - Collections Tab
  - Collections Menu
  - Collections Panel

Introduction

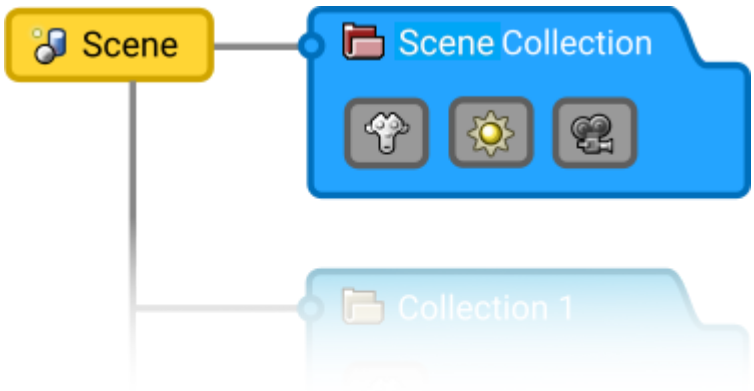
In Blender, objects are not directly part of the scenes. Instead, they all get stored in a main database (basically the blend-file).



The blend-file and its stored data.

From there they are referenced into as many Scenes as you would like to see them.

When they are stored in a scene, they are part of a so-called *scene collection*. So ultimately all the scene objects belong to this special collection.

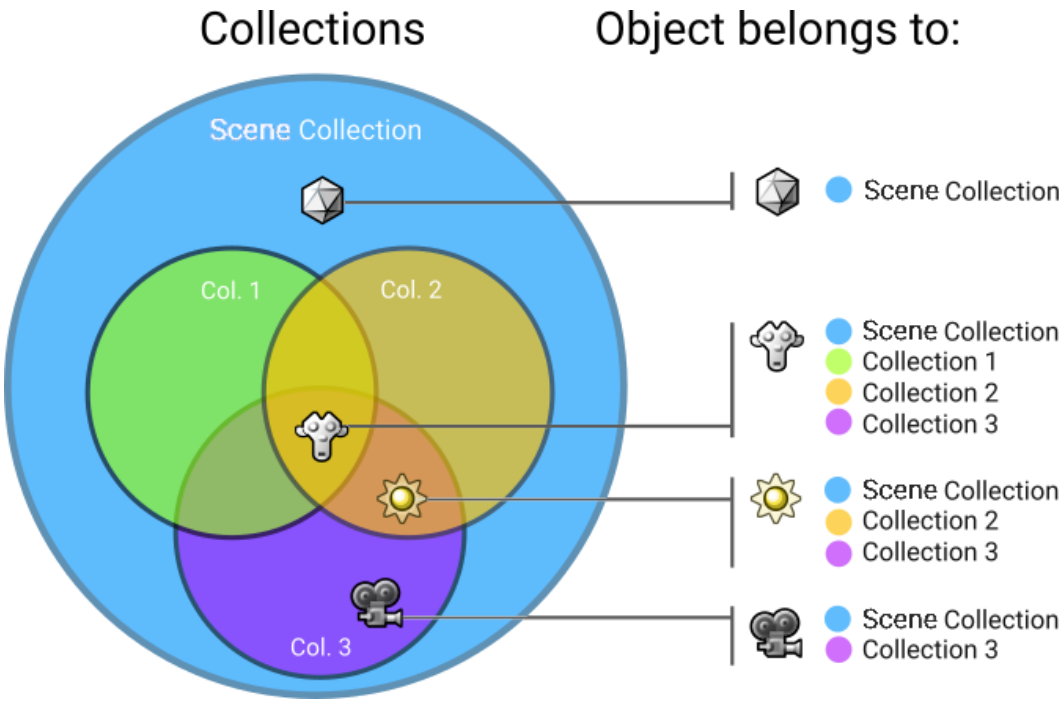


The scene collection.

Collections

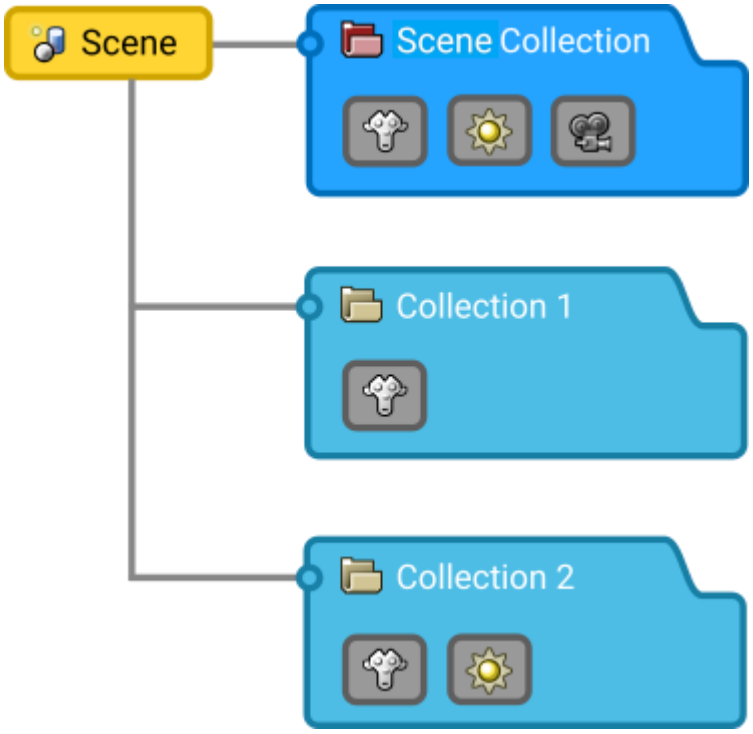
While the *scene collection* contains all the Scene's objects, the user can also make their own collections to better organize these objects.

It works like a Venn diagram, where all the objects are part of the *scene collection*, but can also be part of multiple collections.



Venn diagram.

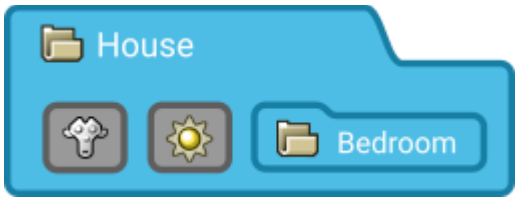
The result is a clear and flexible way to arrange objects together on the Scene level.



Scene organization.

Naming and Nesting

Collections can be named and sorted hierarchically. Just like folders can have subfolders in any operating system, collections can have nested collections too.



Nested collections.

For example: a *house* collection can contain a *bedroom* collection, which in turn contains a *furniture* collection referencing a bed, a cabinet and other objects.

Color Tagging

Collections can have a color group assigned to them; helping organize and group different collections. This color tag is displayed as part of the collection icon in the Outliner and various other menus. The available colors are defined by Blender's interface Theme.

To assign a color to a collection, use the Set Color Tag tool in the Outliner.

Collections

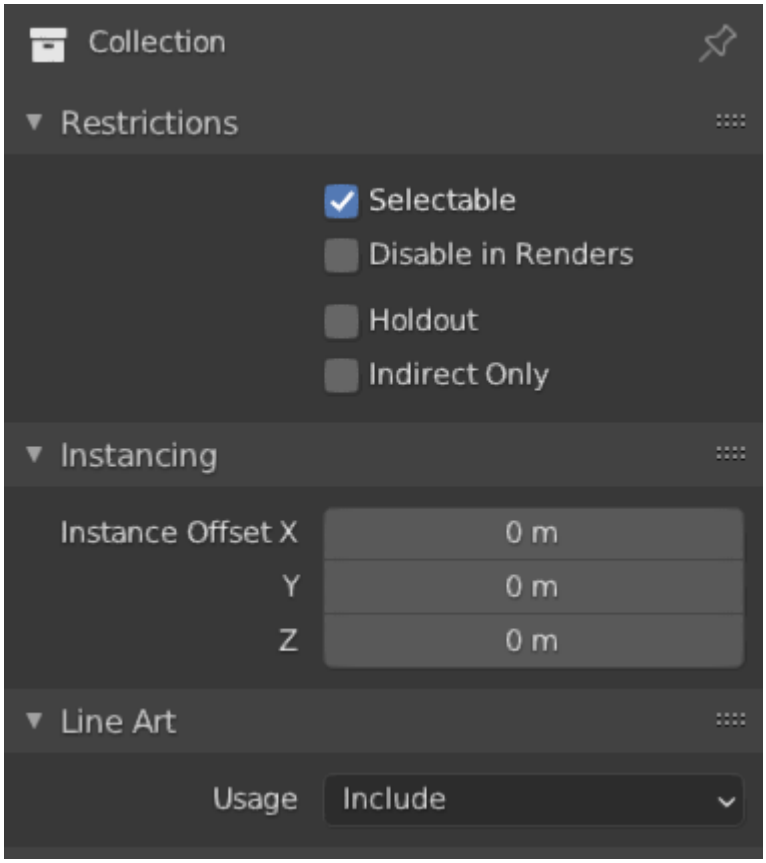
There can be many objects in a scene: A typical stage scene consists of furniture, props, lights, and backdrops. Blender helps you keep everything organized by allowing you to group like objects together. Objects can be grouped together without any kind of transformation relationship (unlike parenting). Collections are used to just logically organize your scene, or to facilitate one-step appending or linking between files or across scenes.

Collections Tab

Reference

Menu:: *Properties ▸ Collection Properties*

Collection properties tab allows convenient access to properties for the active collection.



Collection properties.

Restrictions

Selectable
: Toggles the ability to select the objects from the 3D Viewport. This is useful for if you have placed something in the scene and do not want to accidentally select it when working on something else.

Disable in Renders
: Enables/disables visibility of the collection in renders.

Holdout
: Objects inside this collection will generate a holdout/mask in the active view layer.

Indirect Only

Objects inside this collection will only contribute to the final image indirectly through shadows and reflections.

Instancing

Instance Offset X, Y Z
: Applies a spatial offset of the instanced collections from the original object's origin.

Exporters

Each collection can be exported to a number of various file formats. These exporters are available globally, see Importing & Exporting Files, however, this panel streamlines the process of re-exporting the same asset(s) repeatedly. For example when creating glTF assets for a game and iterating on the look, or when using Blender in a studio pipeline to create USD assets.

The following file formats are supported, see each for the documentation of export parameters:

- Alembic
- Universal Scene Description
- Wavefront OBJ
- Stanford PLY
- FBX
- glTF 2.0

Exporter List
: A list view of all the enabled exporters for the active collection. The selecting an exporter from the list will show it's options in a sub panel below.

Exporters can be added/removed through the + and – icons to right of the Exporter list.

Export All

Exports all exports for the active collection.

Tip

Use *File ▸ Export All Collections* to export all exporters for all collections.

Line Art

Usage
: How the collection is loaded into line art. Child objects of the collection can override this setting if they wish in Object Properties.

Include::	Generate feature lines for this collection.
Occlusion Only::	Objects in the collection will only cause occlusion to existing feature lines and their geometry stay invisible.
Exclude::	Objects in this collection will not be loaded into line art at all.
Intersection Only::	Objects in the collection will only produce intersection lines in the scene and their own geometry stay invisible.
No Intersection::	Include this collection but do not generate intersection lines.
Force Intersection::	Generate intersection lines even with objects that disabled intersection.

Collection Mask
: Use custom intersection mask for faces in this collection. Intersection masks can be used by the Line Art modifier to filter lines. See Collection Masks for more information.

Mask
: Intersections generated by this collection will have this mask value.

Intersection Priority

Assigns an intersection priority value for this collection. The intersection line will be included into the object with the higher intersection priority value.

Custom Properties

Create and manage your own properties to store data in the collection's data block. See the Custom Properties page for more information.

Collections Menu

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Collection*
Shortcut::	`M`, `Shift-M`, `Ctrl-G`, etc.

Move to Collection `M`
: Move selected objects to an existing or new collection.

Link to Collection `Shift-M`
: Add selected objects to a collection, while keeping them in their current collection. This way objects can appear in multiple collections. A new collection can be created in the pop-up.

Create New Collection `Ctrl-G`
: Creates a new collection and adds the selected object(s) to it. The name of the new collection can be specified in the *Create New Collection* Adjust Last Operation panel. This collection is not linked to the active scene.

Remove from Collection `Ctrl-Alt-G`
: Remove the selected objects from a collection. If the object belongs to more than one collection, a pop-up lets you select the collection and an option to remove it from all collections.

Remove from All Collections `Shift-Ctrl-Alt-G`

Remove the selected objects from all collections.

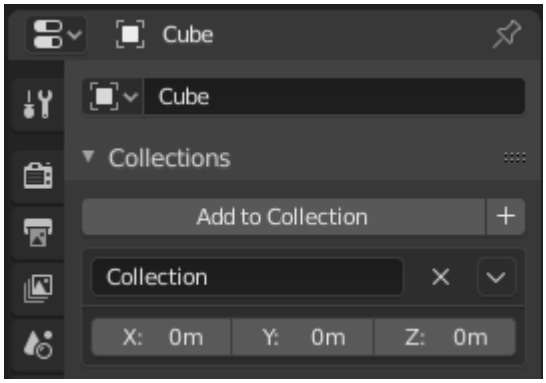
Add Selected to Active Collection `Shift-Ctrl-G`
: Adds the selected objects to the collections to which the active object belongs.

Remove Selected from Active Collection `Shift-Alt-G`
: Causes the selected objects to be removed from the collections to which the active object belongs.

Collections Panel

Reference

Mode:: Object Mode
Panel:: *Object tab ▸ Collections*



Collections panel.

All collections that an object has been assigned to are listed in the Properties *Object tab ▸ Collections panel*.

Add to Collection
: Adds the selected object to a collection. A pop-up lets you specify the collection to add to.

New +
: Creates a new collection and adds the selected object to it.

Name
: To rename a collection, simply click in the collections name field.

(Remove Collection)
: Removes the object from the specified collection.

Specials
: Unlink Collection, Select Collection, Set Offset from Cursor

Offset
: Applies a spatial offset of the instanced collections from the original object's origin.

See also

Appending or Linking Collections

To append a collection from another blend-file, consult this page. In summary, *File ▸ Link/Append Link* Select a blend-file and then the collection.

Tip

Selecting Collections

Collections can be selected, see Select Grouped for more information.

Instancing

Note

Geometry nodes provides a more flexible way to instance objects, with the Instance on Points Node.

Vertices
: This creates an instance of all children of this object on each vertex (for mesh objects only).

Faces
: This creates an instance of all children of this object on each face (for mesh objects only).

Collection
: This creates an instance of the collection with the transformation of the object. Collection instancers can be animated using actions, or can get a Library Override.

- Vertices
  - Usage
- Faces
  - Basic Usage
  - Scale
  - Limitations/Considerations
- Collection
  - Basic Usage
  - Collections and Dynamic Linking
  - Making an Instanced Collection Real

Vertices

Reference

Mode:: Object Mode
Panel:: *Properties* ▸ *Object Properties* ▸ *Instancing*

Instance Vertices allows you to replicate child objects at the location of every vertex of the parent object.

Note

The relative Object Origin position of the parent and child objects determines offset instanced geometry from parent vertex.

Align to Vertex Normal
: Rotates all instanced objects according to the corresponding vertex normals of the parent mesh.

  To change the axis of direction of the instanced objects, select the child object and change the Tracking Axis.

There are actually two approaches to modeling using instanced vertices. They can be used as an arranging tool, allowing you to model geometrical arrangements of objects (e.g. the columns of a Greek temple, the trees in a garden, the desks in a classroom). The object can be of any object type which Blender supports. The second approach is to use them to model an object starting from a single part of it (e.g. the spikes in a club, the thorns of a sea-urchin, the tiles in a wall, the petals in a flower).

Note

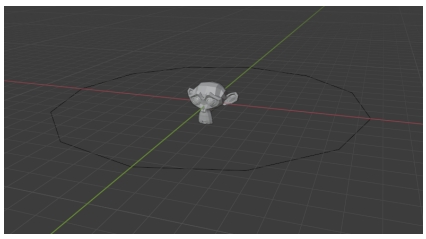
Download Example Blend-File

You can download a file with the examples described on this page. In this blend, the first example, a monkey parented to a circle is on layer 1; while a tentacle parented to an icosphere is on layer 2.

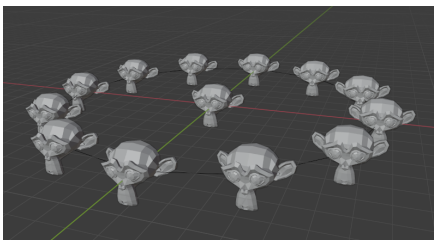
Usage

Instanced Vertices as an Arranging Tool

All you need is a base object (e.g. the *tree* or the *column*) and a pattern mesh with its vertices following the pattern you have in mind. In this section, we will use a simple scene for the following part. We will be using a monkey head located at the origin of the coordinate system as our base object and a circle at the same location as our parent mesh.



A monkey head and a circle.



Instanced monkeys on Vertices.

First, in *Object Mode*, select the base object and `Shift-LMB` to add the circle to the selection (order is very important here), and `Ctrl-P` or *Object* ▸ *Parent* ▸ *Object* to parent the base object to the circle. Now, the circle is the parent of the monkey; if you move the circle, the monkey will follow it.

With only the circle selected, enable *Instancing Vertices*; a monkey head should be placed at every vertex of the circle.

The original monkey head at the center and the parent mesh are still shown in the 3D Viewport but neither will be rendered. If the placement and

rotation of your monkey head are odd, you might need to clear its rotation `Alt-R`, scale `Alt-S`, location `Alt-G`, and origin *Object ▸ Clear ▸ Origin*.

Rearranging

If you now select the base object and modify it in either Object or Edit Mode, all changes will also affect the shape of all instanced objects. You can also select the parent mesh to modify the arrangement of the instances; adding vertices will also add more base objects.

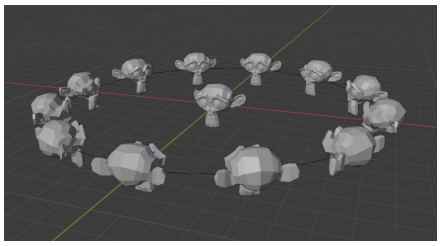
Note that the base objects will inherit changes made to the parent mesh in Object Mode, but not in Edit Mode. So scaling the circle up in Object Mode will enlarge the monkey head, while scaling the circle up in Edit Mode will only increase the distance between the base objects.

Orientation

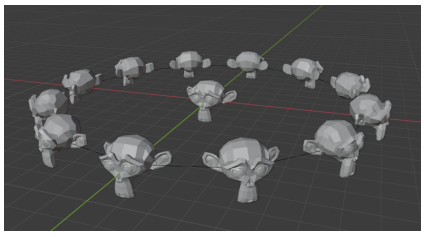
The orientation of the base objects can be controlled by enabling *Align to Vertex Normal* in the *Instancing* panel. This will rotate all base objects according to the vertex normals of the parent mesh.

To change the orientation of the instanced objects, select the base object and change the Tracking Axis.

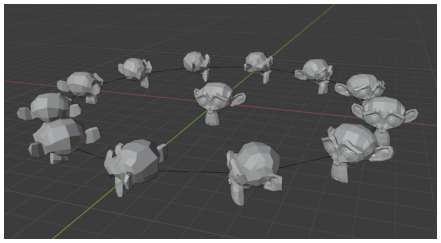
Output of various orientations.



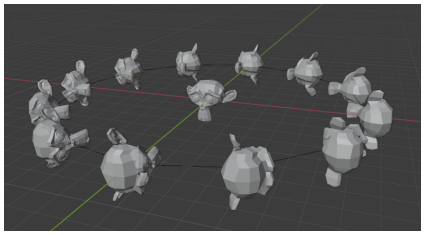
Orientation enabled, orientation +Y.



Negative Y.



Positive X.



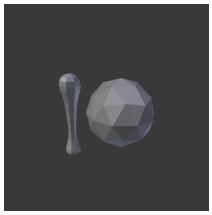
Positive Z, up X.

Note

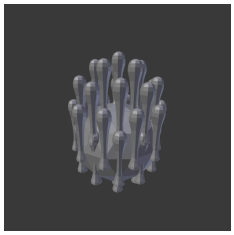
The axes of an object can be made visible in the *Properties ▸ Object Properties ▸ Viewport Display* panel. To display the vertex normals of the parent mesh, enter *Edit Mode* and enable this visualization in the *Display & Shading ▸ Viewport Overlays ▸ Normals* where you can also resize the displayed normals as necessary.

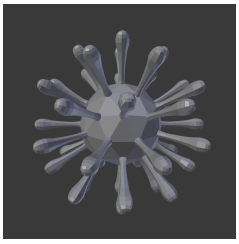
Instanced Vertices as a Modeling Tool

Very interesting models can be made using *Instancing Vertices* and a standard primitive. In this example, a simple tentacle was made by extruding a cube a couple of times. The tentacle object was then parented to an icosphere. With *Align to Vertex Normal* enabled for the parent mesh (the icosphere), the orientation of the base object (the tentacle) was adapted to the vertex normals of the parent mesh (in this case the tentacle was rotated -90° about the X axis in Edit Mode).



A simple tentacle set





to smooth.	Tentacles instanced onto the parent mesh.	*Align to Vertex Normal* enabled to align instanced geometry.

As in the previous example, the shape and proportions of the arrangement can now be tweaked.

To turn all instanced geometry into real objects, select the icosphere and Make Instances Real. To make the icosphere and the tentacle a single object, make sure they are all selected and go to *Object* ▸ *Join*, `Ctrl-J`.

See also

Other duplication methods are listed here.

Faces

Reference

Mode::	Object Mode
Panel::	*Properties* ▸ *Object Properties* ▸ *Instancing*

Instancing Faces is the capability to replicate an object on each face of a parent object. One of the best ways to explain this is through an example illustration.

Scale
: Scales each instance according to the size of its corresponding face.

  Inherit Scale
  : Scale the instance faces objects.

Make Instance Face tool converts linked objects (that share mesh data) into instanced faces. This tool creates the parent object (instancer) with faces where the objects were, then it uses *Instancing Faces* to put instances at the location of every created face.

You can go back from *Instancing Faces* to multiple linked objects using Make Instances Real.

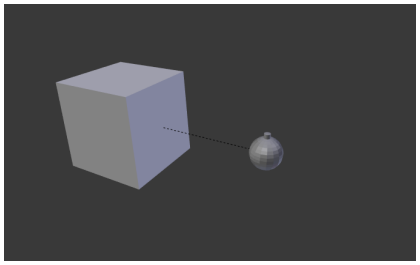
See also

Example blend-file

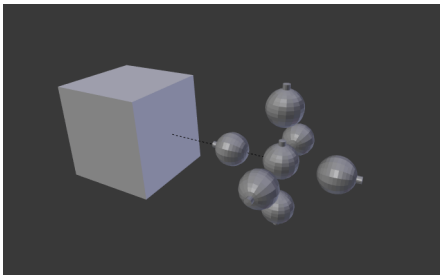
Download the blend-file used for the examples on this page here.

Basic Usage

In this example we will use a UV sphere with an extruded “north pole” as our base object and a cube as our parent mesh. To parent the sphere to the cube, in *Object Mode*, first `LMB` select the sphere, then `Shift-LMB` select the cube (order is very important here), and finally `Ctrl-P` to parent.



A cube and a sphere.



Instancing Faces applied to the cube.

Next, in the *Properties* ▸ *Object Properties* ▸ *Instancing*, select *Faces*. The sphere is instanced, one for each face of the cube.

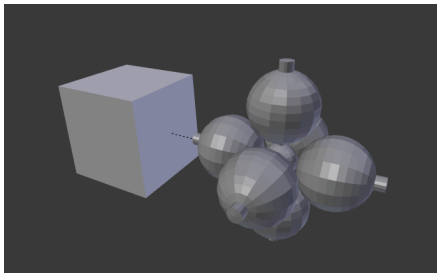
Note

Inherited properties

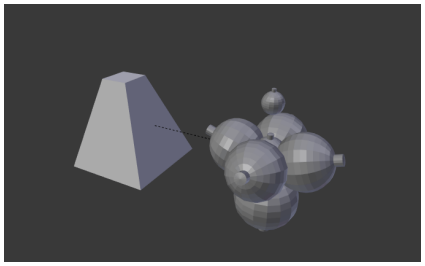
The location, orientation, and scale of the instanced child(ren) matches that of the faces of the parent. So, if several objects are parented to the cube, they will all be instanced once for each face on the cube. If the cube is subdivided, every child will be instanced for each face on the cube.

Both the parent object and original are displayed as editable “templates” in 3D Viewport, but neither is rendered.

Scale



Scale enabled.



Top face of cube scaled down.

By enabling *Scale* for the parent object, the scale of the child objects will be adapted to the size of each face in the parent object.

Thus, by rescaling the face of the parent object, the size of the instanced object will change accordingly.

Limitations/Considerations

The positioning of the instanced geometry relative to the face is dependent upon the position of the child objects relative to the instancer's origin. This can lead to some visual artifacts in the 3D Viewport as the geometry of the original objects overlaps or intersects with the instanced geometry. One workaround is to move the origin of the instancer mesh off of the plane of the faces.

If the geometry of the children is not symmetrical then the orientation of the face (as determined by the order of its vertices) could matter. As of 2.70 Blender does not have tools which allow you to adjust the ordering of the vertices on a face.

However, there is a workflow that lets you control for this. Make a single square and enable the *Instancing Faces* so you can see the instanced geometry in the 3D Viewport. If the orientation is not what you want, rotate the face until it is how you want. Typically you want to do the rotation in Edit Mode, not Object Mode, but this is not a hard rule.

Once you have the orientation correct, Duplicate the face and move the duplicate where you want it. Repeat this process until you have enough faces. Since it is common for these faces to butt up against one another, your geometry will have lots of duplicate vertices. Use the *Merge by Distance* button in the Tools panel.

Demo Video

A short video illustrating this workflow

Collection

Reference

Mode::	Object Mode
Panel::	*Properties* ▸ *Object Properties* ▸ *Instancing* ▸ *Collection*

Instance Collections allows you to create an instance of a collection for each instance of another object. Collections may contain animations, objects with physics simulations and even other nested collections.

Basic Usage

Create a Collection:

- Create a new collection (this can be done via the Outliner).
- Link the objects that need to be instanced as part of the newly created collection.

Create a new Collection Instance:

- *Add* ▸ *Collection Instance*

At this point, an instance of the collection and an empty object will appear. You can duplicate the empty, and the *Instance Collections* settings will be preserved for each empty. This way, you can get multiple copies of linked data very easily.

Collections and Dynamic Linking

See Appending and Linking to understand how to dynamically link data from another blend-file into the current file. You can dynamically link collections from one blend-file to another. When you do so, the linked collection does not appear anywhere in your scene until you create an object controlling where the collection instance appears.

Making an Instanced Collection Real

If you want to make further edits on an instanced collection select the *Instance Collection*. Then call Make Instances Real to convert the collection into regular objects that can be transformed and animated normally.

Note

Note that if the instanced collection was linked from an external file, the Object Data (mesh, materials, textures, transforms) will also still be linked from the original collection. However, the various object's parent-child relationships do not carry over.

Visibility

Reference

Mode::	Object Mode
Panel::	*Properties* ▸ *Object Properties* ▸ *Visibility*

The Visibility panel controls how objects are interacted with in the viewport and in the final render. These visibility options can also be set in the Outliner.

Selectable
: The object is able to be selected in the 3D Viewport.

Show In
: Viewports
  : The object will be displayed in the 3D Viewport.

  Renders
  : The object is able to be in the final render, note that it will still be visible in rendered shading view.

See also

Cycles has additional Visibility properties and also Grease Pencil objects have additional Visibility properties.

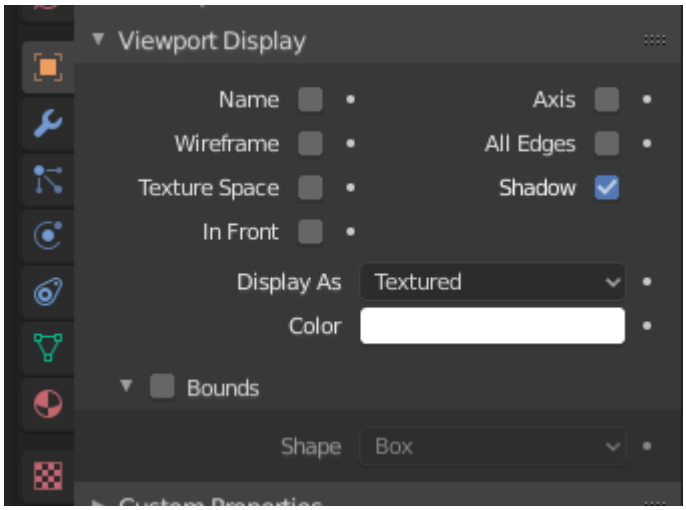
Mask
: Holdout
  : Render objects as a holdout or matte, creating a hole in the image with zero Alpha, to fill out in compositing with real footage or another render.

Viewport Display

Reference

Mode:: Object Mode
Panel:: *Properties* ▸ *Object Properties* ▸ *Viewport Display*

This panel lets you configure display options for the 3D Viewport.



Viewport Display panel.

Show

Name
: Displays the object's name in the 3D Viewport.

Axes
: Displays an object similar to an empty that shows the object's orientation.

Wireframe
: Displays the object's wireframe on top of the solid display.

All Edges

Displays all wireframe edges. This overrides the wireframe threshold that you can set in the 3D Viewport's overlay settings.

Texture Space
: Displays the object's Texture Space.

Shadow
: Allows the object to cast shadows in the viewport.

In Front
: Makes the object display in front of others. Unsupported for instanced objects. Limited support in the *Material Preview* and *Rendered* shading modes (works for e.g. armatures, but not for meshes).

Display As
: Lets you display the object with less detail, going from removing the textures to only showing a bounding box. This can be useful if you have a high-poly object that is slowing down the viewport.

Color
: The object's color in the *Wireframe* and *Solid* viewport shading modes. Used when the viewport's (Wire) Color shading option is set to *Object*.

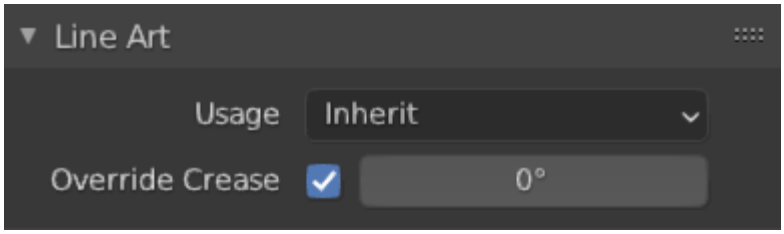
Bounds
: Displays a bounding shape around an object. You can choose between different primitive shapes that might be closer to what the original object looks like.

Line Art

Reference

Mode:: Object Mode
Panel:: *Properties* ▸ *Object Properties* ▸ *Line Art*

The *Line Art* panel is used to enable extra display options for customizing line art rendering for a specific object.



Line Art panel.

Usage
How the object is loaded into line art. This property overrides the parent collection's Line Art usage.

Inherit:: No special loading strategy for line art. Loading of the object is controlled by parent collection's line art settings.
Include:: Force include the object into line art calculation even if its parent collection specifies otherwise.
Intersection Only:: The object will only produce intersection lines in the scene and its own geometry stays invisible.
Occlusion Only:: The object will only cause occlusion to existing feature lines and its geometry stays invisible.
Exclude:: The object will not be loaded into line art at all.
No Intersection:: The object will not generate intersection lines on itself or with other objects in scene.
Force Intersection::

Generate intersection lines even with objects that disabled intersection.

Override Crease

Allows the object to have a different crease value than the global one set in the line art modifier.

Crease

Override crease value for the object.

Intersection Priority

Assigns an intersection priority value for this object. The intersection line will be included into the object with the higher intersection priority value.

Tools

- Toolbar
- Tool Settings
  - Options

Types

- Scale Cage

Toolbar

Tweak
: Select or move.

  Select Box
  : Select objects by dragging a box. All objects that intersect the box will be selected.

  Select Circle
  : Select objects by dragging a circle. All objects that intersect the path of the circle will be selected.

  Select Lasso
  : Select objects by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform
: Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line
: Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Measure
: Measure distances in the scene.

Add Cube
: Interactively add a cube mesh object.

Add Cone
: Interactively add a cone mesh object.

Add Cylinder
: Interactively add a cylinder mesh object.

Add UV Sphere
: Interactively add a UV sphere mesh object.

Add Icosphere
: Interactively add an icosphere mesh object.

Tool Settings

Options

Reference

Mode:: Object Mode and Pose Mode
Header:: *Sidebar* ▸ *Tool* ▸ *Options*

Transform

Affect Only
: Origins `Ctrl-Period`
: Directly transforms the object's origin. This only works for objects with data which can be transformed; i.e. it will not work on object lights.

: When enabled, the object axes are displayed.

: Take care using this option since it transforms the object-data which may cause linked duplicates to be moved unintentionally.

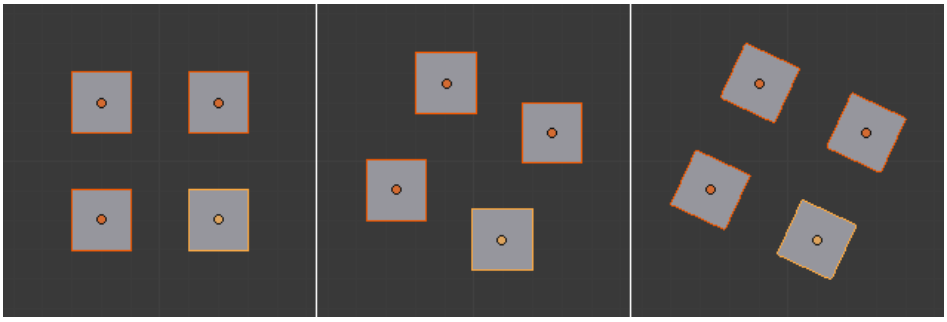
: **Hint**
: Changing the object location and the object-data may impact modifiers, constraints and keyframe animation.

: If you are only temporarily setting the pivot point, use the 3D cursor instead.

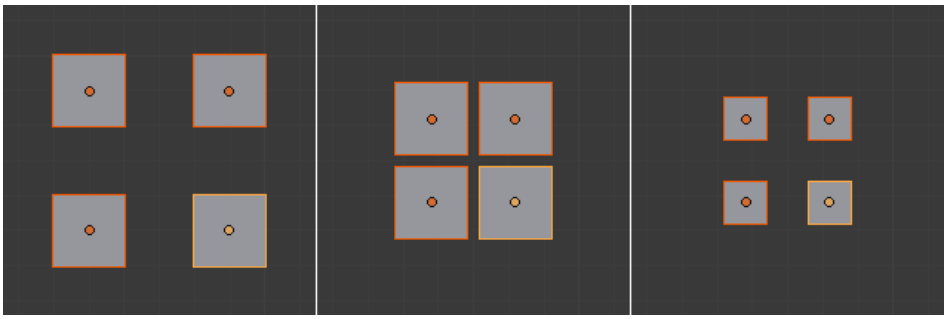
: Locations
: Changes the position of the object's origin relative to another point during transformation. In other words, the pivot point and the origin

cannot share the same location. This will not affect the object local transforms, just its position in world space.

In the examples below, a comparison of the scaling and rotation of objects, when *Location* is enabled (middle) and disabled (right).



Rotation example.



Scaling example.

Parents

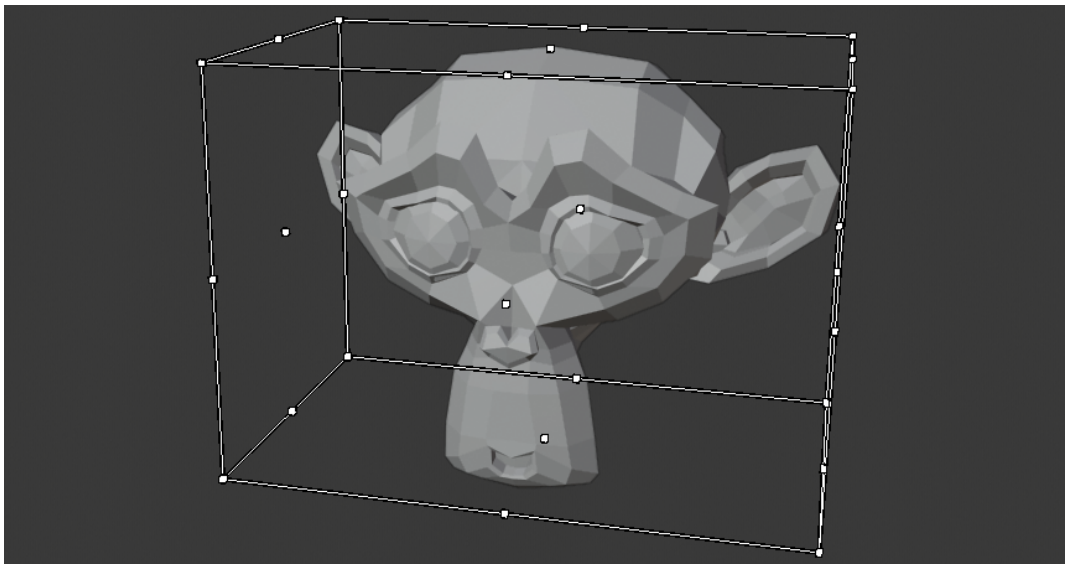
Transforms Parent Objects while leaving their children objects unaffected.

Scale Cage

Reference

Mode::	Object and Edit Modes
Tool::	*Toolbar ▸ Scale ▸ Scale Cage*

The *Scale Cage* tool is a bounding box around the object(s) which scales objects from a particular point or axis. The tool works by selecting a scale point and dragging inwards or outwards to adjust the scale accordingly. The origin for the scale will be from the point on the cube directly opposite from the point selected. Selecting points on the faces of the cube scales along one axis, selecting points on the edges of the cube scales along two axes, and selecting points on the vertices of the cube scales along all three axes.



Scale Cage tool.

Tool Settings

Orientation

Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

Options

Scale X, Y, Z
: The amount to resize the selection on their respected axis.

Orientation
: Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

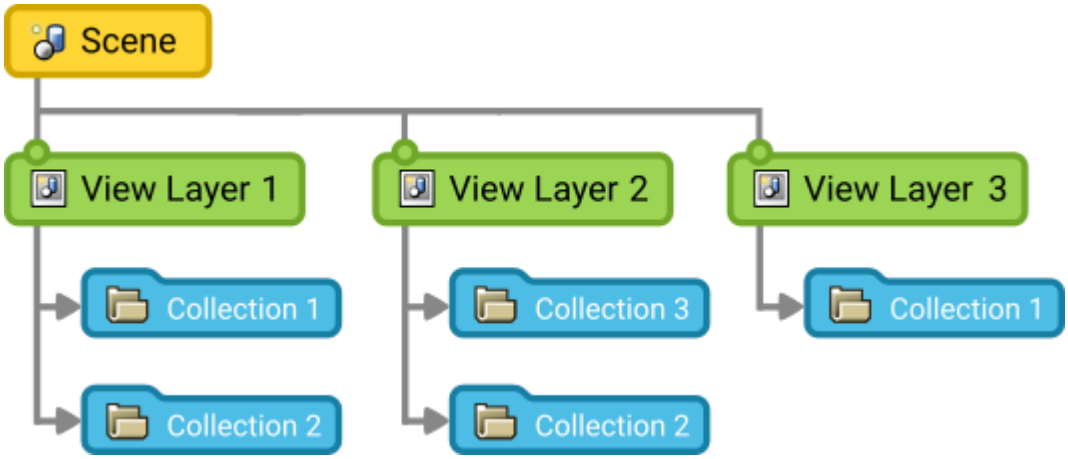
Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

View Layers

- Introduction
  - Outliner

Introduction

The visibility controls are part of *view layers*, designed to help organizing what you want to see or work on.



View layers and collections.

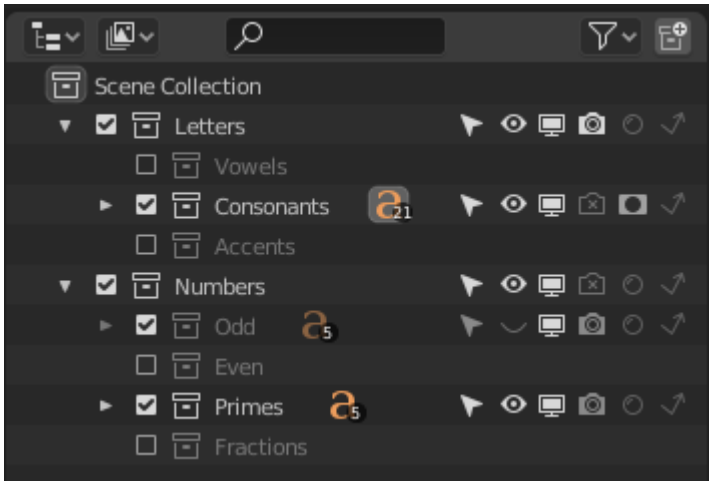
View layers reference to collections, and allow to set their visibility, selectability and other options. A *view layer* can have any collection enabled, and multiple *view layers* can use the same or different collections.

See also

Each view layer can be rendered separately as individual Render Layers to help composite your scene.

Outliner

You can edit the *view layer* collections in the Outliner.



View layer and collections in the Outliner.

There you can enable and disable collections, hide them temporarily, globally, among other options.

See also

Read more about Collections in the Outliner.

Modeling

- Introduction
  - Modes
- Meshes
  - Introduction
  - Structure
  - Primitives
  - Tools
  - Selecting
  - Editing
  - Properties
  - UVs
  - Mesh Analysis
  - Retopology
- Curves
  - Introduction
  - Tools
  - Structure
  - Primitives
  - Selecting
  - Editing
  - Properties
  - Curve Display
- Curves (New)
  - Tools
  - Primitives
  - Selecting
  - Editing
  - Properties
- Surfaces
  - Introduction
  - Toolbar
  - Structure
  - Primitives

    - Selecting
    - Editing
    - Properties
- Metaball
    - Introduction
    - Toolbar
    - Structure
    - Primitives
    - Editing
    - Properties
- Text
    - Introduction
    - Selecting
    - Editing
    - Properties
- Volumes
    - Introduction
    - Properties
- Empties
    - Primitives
    - Editing
    - Properties
    - Usage
- Modifiers
    - Introduction
    - Common Modifier Options
    - Built-In Modifiers
- Geometry Nodes
    - Introduction
    - Inspection
    - Attributes
    - Fields
    - Instances
    - Baking
    - Node-Based Tools
    - Gizmos
    - Node Types

- Transform
  - Introduction
  - Transform Modal Map

Introduction

The creation of a 3D scene needs at least three key components: Models, materials and lights. In this part, the first of these is covered, that being modeling. Modeling is simply the art and science of creating a surface that either mimics the shape of a real-world object or expresses your imagination of abstract objects.

Modes

Depending on the type of object you are trying to model, there are different types of modeling modes. Since modes are not specific to modeling they are covered in different parts of the manual.

Switching between modes while modeling is common. Some tools may be available in more than one mode while others may be unique to a particular mode.

Edit Mode

Edit Mode is the main mode where modeling takes place. Edit Mode is used to edit the following types of objects:

- Meshes
- Curves
- Surfaces
- Metaballs
- Text objects
- Lattice

You can only modify the mesh of the objects you are editing. To modify other objects you can leave Edit Mode, select another object and enter Edit Mode, or use Multi-Object Editing.

Meshes

- Introduction
  - Modeling Modes
- Structure
  - Vertices
  - Edges
  - Faces
  - Normals
  - Topology
- Primitives
  - Common Options
  - Plane
  - Cube
  - Circle
  - UV Sphere
  - Icosphere
  - Cylinder
  - Cone
  - Torus
  - Grid
  - Monkey
- Tools
  - Toolbar
  - Tool Settings
  - Types
- Selecting
  - Introduction
  - Select Mirror
  - Select Random
  - Checker Deselect
  - Select More/Less
  - Select Similar
  - Select All by Trait
  - Select Linked

    - Select Loops
    - Select Sharp Edges
    - Side of Active
    - By Attribute
- Editing
    - Introduction
    - Mesh Operators
    - Vertex Operators
    - Edge Operators
    - Face Operators
    - UV Operators
- Properties
    - Object Data
    - Vertex Groups
    - Geometry Data
- UVs
    - UVs & Texture Space
    - Unwrapping
    - Tools
    - Editing
    - Workflows
    - Using UV Maps
- Mesh Analysis
    - Overhang
    - Thickness
    - Intersections
    - Distortion
    - Sharp Edges
    - Known Limitations
- Retopology
    - Using the Poly Build Tool
    - Remeshing

Introduction

Mesh Modeling typically begins with a Mesh Primitive shape (e.g. circle, cube, cylinder…). From there you might begin editing to create a larger, more complex shape.

Modeling Modes

The 3D Viewport has three principal modes that allow for the creation, editing and manipulation of the mesh models. Each of the three modes has a variety of tools. Some tools may be found in one or more of the modes.

Modes that used for modeling:

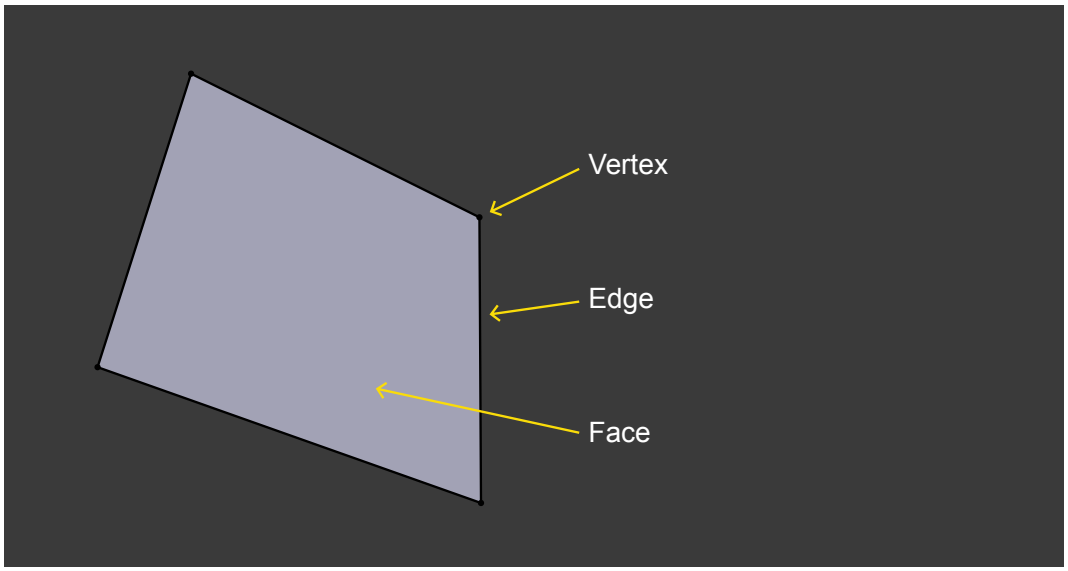
Object Mode
: Supports basic operations such as object creation, joining objects, managing shape keys, UV/color layers.

Edit Mode
: Used for the majority of mesh editing operations.

Sculpt Mode
: Instead of dealing with individual mesh elements, supports sculpting with brushes *(not covered in this chapter)*.

Structure

With meshes, everything is built from three basic structures: *vertices*, *edges* and *faces*.



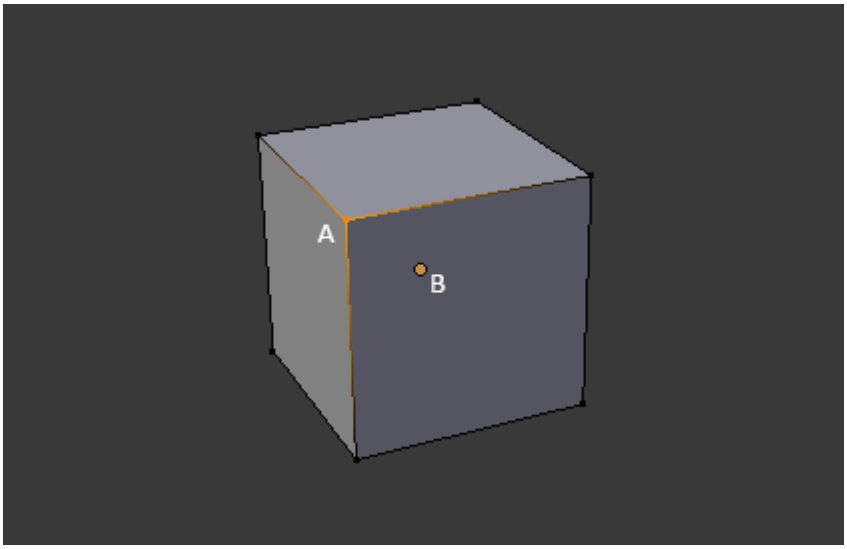
Example of mesh structure.

Vertices

The most elementary part of a mesh is the vertex (vertices plural) which is a single point or position in 3D space. Vertices are represented in the 3D Viewport in Edit Mode as small dots. The vertices of an object are stored as an array of coordinates.

Tip

Do not mistake the object origin for a vertex. It may look similar, but it is bigger and cannot be selected.



The vertex is labeled as “A”; the object’s origin dot is labeled as “B”.

Edges

An edge always connects two vertices by a straight line. The edges are the “wires” you see when you look at a mesh in wireframe view. They are usually invisible on the rendered image. They are used to construct faces.

Faces

Faces are used to build the actual surface of the object. They are what you see when you render the mesh. If this area does not contain a face, it will simply be transparent or nonexistent in the rendered image.

A face is defined as the area between either three (triangles), four (quadrangles) or more (n-gons) vertices, with an edge on every side. The faces are often abbreviated to *tris, quads & n-gons*.

Triangles are always flat and therefore easy to calculate. On the other hand, quadrangles “deform well” and are therefore preferred for animation and subdivision modeling.

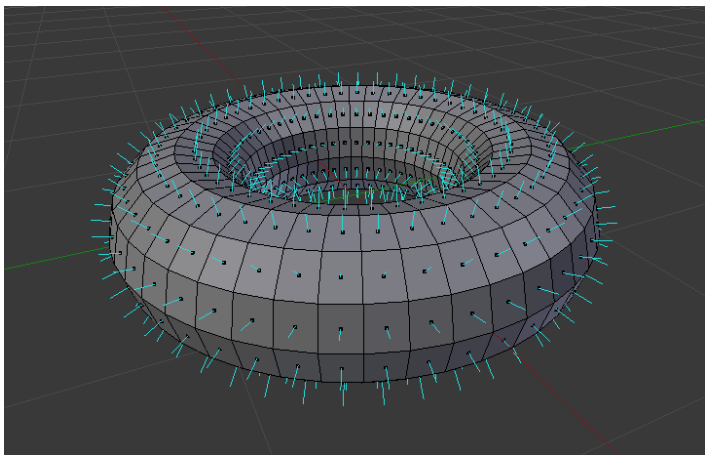
See also

- Why should triangles be avoided for character animation?
- When should N-gons be used, and when shouldn't they?

Normals

In geometry, a normal is a direction or line that is perpendicular to something, typically a triangle or surface but can also be relative to a line, a tangent line for a point on a curve, or a tangent plane for a point on a surface.

Normals help to determine the shading of the mesh among other things.



A visualization of the face normals of a torus.

In the figure above, each blue line represents the normal for a face on the torus. The lines are each perpendicular to the face on which they lie. The visualization can be activated, in Edit Mode, in the Mesh Display Viewport Overlays panel.

Shading

Surface normals play a fundamental role in determining how light interacts with 3D objects and thus greatly influence the shading of those objects.

Normals can be shaded smooth or flat.

When a mesh uses flat shading, the faces are rendered and displayed faces uniformly. This is usually desirable for objects with flat surfaces such as a cube or pyramid.

When a mesh uses smooth shading, the normals are interpolated cross the vertices of a polygonal mesh, smooth transitions between adjacent polygons can be achieved, resulting in a more realistic appearance.

By default face normals have flat shading however, this can be adjusted either for the whole object or per face.

To adjust the the shading of the whole object, use:

- Shade Smooth – To mark the whole object as smooth
- Shade Auto Smooth – To mark portions of the object as smooth

To revert to flat shading, use Shade Flat

The shading of objects can also be adjusted per face, edge, or vertex.

Custom Split Normals

Custom Split Normals is a way to tweak/fake shading by pointing normals towards other directions than the default, auto-computed ones. It is mostly used in game development, where it helps counterbalance some issues generated by low-poly objects (the most common examples are low-poly trees, bushes, grass, etc. and the 'rounded' corners).

Blender supports custom normals on a 'smooth fan' base, defined as a set of neighbor face corners sharing the same vertex and 'linked' by smooth edges. This means you can have normals per face corners, per a set of neighbor face corners, or per vertex.

Tip

The computation of *Custom Split Normals* can be disabled to improve performance. This option can be found in the Simplify Rendering Settings.

Editing Custom Split Normals

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Normals*
Shortcut::	`Alt-N`

There are a number of tools for editing custom split normals. The custom normal mesh edit tools can affect all normals (the default), or only selected ones. To select a custom normal associated with a particular vertex and face:

- Make the element selection mode both Vertex and Face (use `Shift-LMB` to enable the second one).
- Select one or more vertices, then select a face. This can be repeated to select more vertices and a different face and so on. It is easiest to see the effect of these tools if you turn on the Edit Mode Overlays option *Display vertex-per-face normals as lines*.

See also

Editing Normals.

Importing Custom Split Normals

Some tools, particularly those used in CAD, tend to generate irregular geometry when tessellating their objects into meshes (very thin and long triangles, etc.). Auto-computed normals on such geometry often gives bad

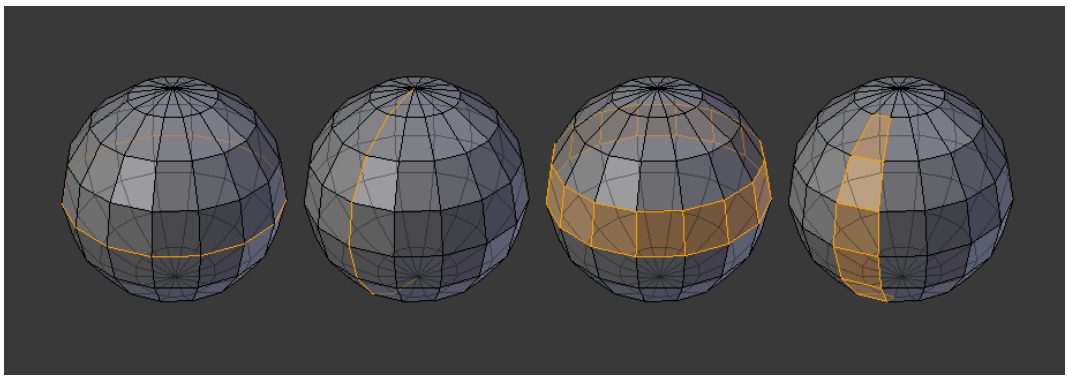
artifacts, so it is important to be able to import and use the normals as generated by the CAD tool itself.

Note

Currently, only the FBX Importer and Alembic Importer are capable of importing custom normals.

Topology

Loops



Edge and face loops.

Edge and *face* loops are sets of faces or edges that form continuous "loops" as shown in Fig. Edge and face loops..

In the image above, loops that do not end in poles are cyclic (1 and 3). They start and end at the same vertex and divide the model into two partitions. Loops can be a quick and powerful tool to work with specific, continuous regions of a mesh and are a prerequisite for organic character animation. For a detailed description of how to work with loops in Blender, see: Select Edge Loops.

Note

Note that loops (2 and 4) do not go around the whole model. Loops stop at so-called poles because there is no unique way to continue a loop from a pole. Poles are vertices that are connected to either three, five, or more edges. Accordingly, vertices connected to exactly one, two or four edges are not poles.

Edge Loops

Loops (1 and 2) in Fig. Edge and face loops. are edge loops. They connect vertices so that each one on the loop has exactly two neighbors that are not on the loop and placed on both sides of the loop (except the start and end vertex in case of poles).

Edge loops are an important concept especially in organic (subsurface) modeling and character animation. When used correctly, they allow you to build models with relatively few vertices that look very natural when used as subdivision surfaces and deform very well in animation.

Take Fig. Edge and face loops. in organic modeling as an example: the edge loops follow the natural contours and deformation lines of the skin and the underlying muscles. The loops are denser in areas that deform more when the character moves, for example at the shoulders or knees.

Further details on working with edge loops can be found in Select Edge Loops.

Face Loops

These are a logical extension of edge loops in that they consist of the faces between two edge loops, as shown in loops (3 and 4) in Fig. Edge and face loops.. Note that for non-circular loops (4) the faces containing the poles are not included in a face loop.

Further details on working with face loops can be found in Face Loop Selection.

Poles

See N-poles & E-poles.

Non-Manifold

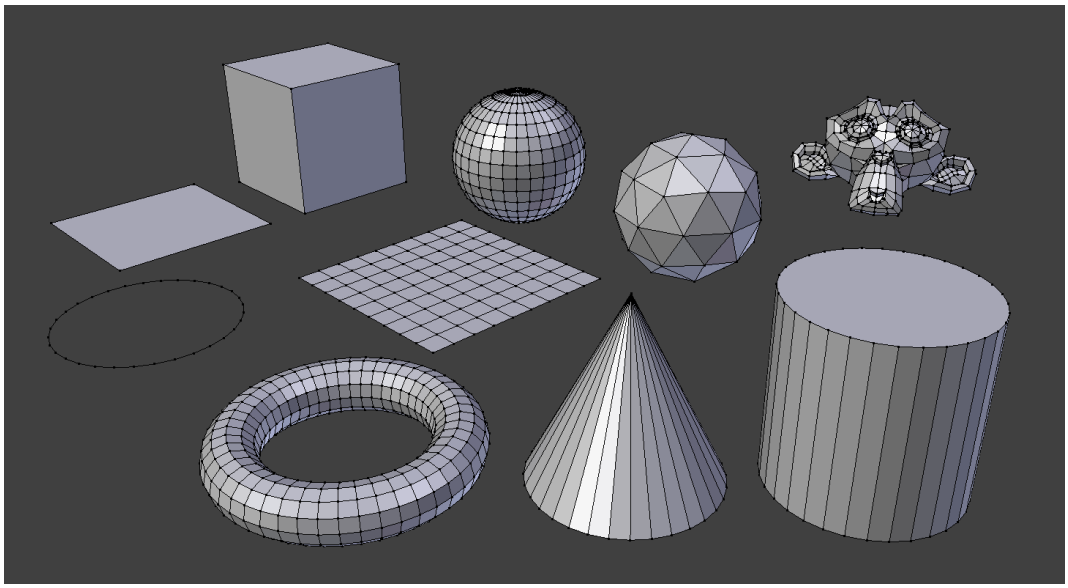
See Non-manifold.

Primitives

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Add* ▸ *Mesh*
Shortcut::	`Shift-A`

A common object type used in a 3D scene is a mesh. Blender comes with a number of "primitive" mesh shapes that you can start modeling from. You can also add primitives in Edit Mode at the 3D cursor.



Blender's standard primitives.

Note

Planar Primitives

You can make a planar mesh three-dimensional by moving one or more of the vertices out of its plane (applies to *Plane*, *Circle* and *Grid*). A simple

circle is often used as a starting point to create even the most complex of meshes.

Common Options

These options can be specified in the Adjust Last Operation panel, which appears when the object is created. Options included in more than one primitive are:

Generate UVs
: Generates a default UV unwrapping of new geometry. This will be defined in the first UV layer (which will get added if needed).

Radius/Size, Align to View, Location, Rotation
: See Common Object Options.

Plane

The standard plane is a single quad face, which is composed of four vertices, four edges, and one face. It is like a piece of paper lying on a table; it is not a three-dimensional object because it is flat and has no thickness. Objects that can be created with planes include floors, tabletops, or mirrors.

See also

Import Images as Planes adds a mesh plane with materials and texture from an image file. The dimensions of the plane are calculated to match the aspect of the image file.

Cube

A standard cube contains eight vertices, twelve edges, and six faces, and is a three-dimensional object. Objects that can be created out of cubes include

dice, boxes, or crates.

Circle

Vertices
: The number of vertices that define the circle or polygon.

Fill Type
: Set how the circle will be filled.

Triangle Fan::	Fill with triangular faces which share a vertex in the middle.
N-gon::	Fill with a single N-gon.
Nothing::	Do not fill. Creates only the outer ring of vertices.

UV Sphere

A standard UV sphere is made out of quad faces and a triangle fan at the top and bottom. It can be used for texturing.

Segments
: Number of vertical segments. Like the Earth's meridians, going pole to pole.

Rings
: Number of horizontal segments. These are like the Earth's parallels.

> **Note**
>
> Rings are face loops and not edge loops, which would be one less.

Icosphere

An icosphere is a polyhedral sphere made up of triangles. Icospheres are normally used to achieve a more isotropical layout of vertices than a UV

sphere, in other words, they are uniform in every direction.

Subdivisions
: How many recursions are used to define the sphere. At level 1 the icosphere is an icosahedron, a solid with 20 equilateral triangular faces. Each increase in the number of subdivisions splits each triangular face into four triangles.

Note

Subdividing an icosphere raises the vertex count very quickly even with few iterations (10 times creates 5,242,880 triangles), Adding such a dense mesh is a sure way to cause the program to crash.

Cylinder

Objects that can be created out of cylinders include handles or rods.

Vertices
: The number of vertical edges between the circles used to define the cylinder or prism.

Depth
: Sets the starting height of the cylinder.

Cap Fill Type
: Similar to circle (see above). When set to none, the created object will be a tube. Objects that can be created out of tubes include pipes or drinking glasses (the basic difference between a cylinder and a tube is that the former has closed ends).

Cone

Objects that can be created out of cones include spikes or pointed hats.

Vertices
: The number of vertical edges between the circles or tip, used to define the cone or pyramid.

Radius 1
: Sets the radius of the circular base of the cone.

Radius 2
: Sets the radius of the tip of the cone. Which will create a frustum (a pyramid or cone with the top cut off). A value of 0 will produce a standard cone shape.

Depth
: Sets the starting height of the cone.

Base Fill Type
: Similar to circle (see above).

Torus

A doughnut-shaped primitive created by rotating a circle around an axis. The overall dimensions can be defined by two methods.

Operator Presets
: Torus preset settings for reuse. These presets are stored as scripts in the proper presets directory.

Major Segments
: Number of segments for the main ring of the torus. If you think of a torus as a “spin” operation around an axis, this is how many steps are in the spin.

Minor Segments
: Number of segments for the minor ring of the torus. This is the number of vertices of each circular segment.

Dimensions Mode

Change the way the torus is defined.

Major/Minor, Exterior/Interior

Major Radius
: Radius from the origin to the center of the cross sections.

Minor Radius
: Radius of the torus' cross section.

Exterior Radius
: If viewed along the major axis, this is the radius from the center to the outer edge.

Interior Radius
: If viewed along the major axis, this is the radius of the hole in the center.

Grid

A regular quadratic grid which is a subdivided plane. Example objects that can be created out of grids include landscapes and organic surfaces.

X Subdivisions
: The number of spans in the X axis.

Y Subdivisions
: The number of spans in the Y axis.

Monkey

This adds a stylized monkey head to use as a test mesh, use Subdivision Surface for a refined shape.

This is intended as a test mesh, similar to:

- Utah Teapot

- Stanford Bunny.

History

This is a gift from old NaN to the community and is seen as a programmer's joke or "Easter Egg". It creates a monkey's head once you press the *Monkey* button. The Monkey's name is "Suzanne" and is Blender's mascot.

Note

In addition to the basic geometric primitives can be added via extensions These are available from the Preferences.

Import Images as Planes

Reference

Menu:: *3D Viewport ▸ Add ▸ Image ▸ Images as Planes*

Imports images and creates planes with them as textures. It automates the process of creating a plane, resizing it to fit the dimensions of the image, and creating a material with the image texture to it. The name of the plane, material, and texture will be derived from the name of the image file.

You can import a single image, multiple images, or an image sequence/movie clip. If you choose a single image, it creates one plane; if you choose multiple images, it creates as many planes as the number of images you selected, either stacked on top of each other or spaced apart. Selecting a movie clip or an image sequence will create a single plane with an animation.

Properties

You can save the current import settings as an Operator Preset.

Options

Relative Path
: Set link to the image file using a relative file path.

Force Reload
: Reload the image file if it already exists as an image data-block.

Animate Image Sequences
: Import sequentially numbered images as an animated image sequence instead of separate planes. They will be imported as a *Clip* texture on a

single plane. The frame range will be automatically set but can be changed later.

Material

Images as Planes sets up a material to display the image. You can set the type of material and related settings before the import.

Shader

Principled:: The material will have a Principled BSDF shader node with default settings as its main component. An Image Texture node linked to the imported image will be connected to the Base Color of the Principled BSDF node.

Shadeless:: A shadeless material is a material that does not respond to light from other objects and always has the same color in any lighting environment. This option creates a material with a node group of a mix between a Diffuse and an Emission shader controlled by a Light Path node.

Emit:: The material will have a Principled BSDF shader node as its main component, but the Color output from the Image Texture node will be linked to the Emission input instead of the Base Color.

Strength

Set the strength of the emission.

Note

Blend Mode and *Shadow Mode* options are specific to the EEVEE renderer. For a detailed explanation of each option, see Material Settings.

Blend Mode

Set the alpha blend mode of the material.

Shadow Mode
: Set the shadow mode of the material.

Show Backface
: Show backside of the transparent part.

Backface Culling
: Hide backside of the plane.

Overwrite Material
: By default, the name of the new material from the name of the imported image. However, if there is already a material with the same name, Blender will append a number to the name of the material to avoid conflict. This *Override Material* option makes it overwrite the existing material of the same name in that case.

Texture

Note

For a detailed explanation of each option, see Image Texture Node.

Interpolation
: Set the method to scale the image.

Extension
: Set how the image is extrapolated past the original bounds.

Alpha
: Use the alpha channel of the image for transparency.

Auto Refresh
: Automatically refresh the images in the viewport on frame changes.

Transform

Images as Planes creates the plane at the 3D Cursor's location. With *Offset Planes*, multiple planes will be placed with distance intervals set in *Offset*, along the axis set in *Local Axis*, beginning at the 3D Cursor's location.

Size Mode

Set how the plane's size will be determined.

Absolute:: The size of the plane will be set based on the height value set in *Height*. The width will be set in direct ratio to the height value. For example, with the default height value of 1 m, an image of 800 × 600 pixels will have a width of 1 / 600 × 800 or 1.33 m.

Height

Set the height of the plane.

Camera Relative:: The size of the plane will be set to fit or fill the camera frame. This will automatically set the *Align* option to *Face Camera*. Make sure to have an active camera in the scene before the import.

Fit:: Scale the plane to fit inside the camera frame while preserving the aspect ratio.

Fill:: Scale the plane so that it fills the entire camera view while preserving the aspect ratio, but some part of the image can spill outside the camera frame.

:DPI:

The size of the plane will be set based on the pixels per inch value set in *Definition*. With the *Unit System* set to *Metric* and the default definition of 600 DPI, an image of 800 × 600 pixels will have a size of 0.0339 × 0.0254 units since 600 pixels are defined as 1 inch (0.0254 m).

Definition

Set the number of pixels to fit in 1 inch.

Dots/BU:: The size of the plane will be set based on the pixels per Blender Unit set in *Definition*. With the default definition value of 600, an image of 800 × 600 pixels will have a size of 1.33 × 1 units.

Definition

Set the number of pixels to fit in 1 Blender Unit.

Align

Set the rotation of the plane.

Main Axis:: The plane will be aligned to a major axis that is best to face the camera's view direction. If there is no camera in the scene, the plane will face toward Z+ (Up) axis.

Face Camera:: Similar to the *Main Axis* but the plane will be rotated to directly face the camera's view direction.

Z- (Down), Y-, X-, Z+ (Up), Y+, X+:: The plane will be rotated to face toward the selected axis.

Track Camera

Add a Locked Track constraint to make the plane always face the camera, even if the camera moves. This option is only available when *Main Axis* or *Face Camera* option is selected in the *Align* menu.

Offset Planes

Place multiple planes with an offset. If disabled, all planes will be created at the same location.

Offset Direction

Choose a local axis (not the global axis) to offset the planes. For example, if you choose *X+*, the planes will be placed along the positive direction of the plane's local X axis.

Distance

Set a distance between each plane.

Tools

- Toolbar
- Tool Settings
  - Options

Types

- Extrude Region
- Extrude Manifold
- Extrude to Cursor
- Loop Cut
- Poly Build
- Spin

Toolbar

Mesh Edit Mode tools:

Select
: Select or move.

  Select Box
  : Select geometry by dragging a box.

  Select Circle
  : Select geometry by dragging a circle.

  Select Lasso
  : Select geometry by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform
: Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line
: Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Measure
: Measure distances in the scene.

Add Cube
: Interactively add a cube mesh object.

Add Cone
: Interactively add a cone mesh object.

Add Cylinder
: Interactively add a cylinder mesh object.

Add UV Sphere
: Interactively add a UV sphere mesh object.

Add Icosphere
: Interactively add an icosphere mesh object.

Extrude Region
: Extrude the selected region together freely or along an axis.

Extrude Manifold
: Extrudes region and dissolves overlapping geometry.

Extrude Along Normals
: Extrude Region along their local normal.

Extrude Individual
: Extrude each individual element along their local normal.

Extrude To Cursor
: Extrude selected vertices, edges or faces towards the mouse cursor.

Inset Faces
: Inset selected faces.

Bevel
: Create a bevel from the selected elements.

Loop Cut
: Create a loop cut along the mesh.

Offset Edge Loop Cut
: Add two edge loops on either side of selected loops.

Knife
: Create a knife cut in the mesh. Press enter to confirm the cut.

Bisect
: Bisect the mesh.

Poly Build
: Create geometry by adding vertices one by one.

Spin
: Create new geometry by extruding and rotating.

Smooth
: Flatten angles of selected vertices.

Randomize
: Randomize selected vertices.

Edge Slide
: Slide edge along a face.

Vertex Slide
: Slide vertex along an edge.

Shrink/Fatten
: Move the selected vertices along their normals.

Push/Pull
: Move the selected elements away from/towards the pivot point.

Shear
: Shear selected elements.

To Sphere
: Move vertices outwards in a spherical shape around object center.

Rip Region
: Rip Polygons and move the result.

Rip Edge
: Extend vertices and move the result.

Tool Settings

Options

Reference

Mode:: Edit Mode
Panel:: *Sidebar ▸ Tool tab ▸ Options panel*

Transform

Correct Face Attributes
: Adjust geometry attributes like UVs and Color Attributes while transforming.

Keep Connected
: Merge attributes connected to the same vertex while using *Correct Face Attributes*.

> **Tip**
>
> Keeping UVs connected is useful for organic modeling, but not for architectural modeling.

Mirror

Mirror allows you to transform vertices symmetrically according to the chosen axis. When you transform an element (vertex, edge or face), if there is its exact axis-mirrored counterpart (in local space), it will be transformed accordingly, through a symmetry along the chosen axis.

Note

The conditions for *Mirror* to work are quite strict, which can make it difficult to use. To have an exact mirrored version of a (half) mesh, it's easier and simpler to use the Mirror Modifier.

Topology Mirror

Note

For *Topology Mirror* to work, at least one of the three *Mirror Axis* must be enabled.

When using any of the three *Mirror Axis* options to work on a mirrored Mesh Geometry, the vertices that are mirrored must be perfectly placed. If they are not exactly positioned in their mirror locations then the *Mirror Axis* will not treat those vertices as mirrored.

Topology Mirror tries to address this problem by determining which vertices are mirrored vertices not only by using their positions but also by looking at how those vertices are related to others in the Mesh Geometry. It looks at the overall topology to determine if particular vertices will be treated as mirrored. The effect of this is that mirrored vertices can be non-symmetrical and yet still be treated as mirrored.

Note

The *Topology Mirror* functionality will work more reliably on mesh geometry which is more detailed. If you use very simple geometry, for example a *Cube* or *UV Sphere*, the *Topology Mirror* option will often not work.

Example

For an example of how to use *Topology Mirror* open up a new Blender scene, then delete the default cube and add a Monkey object to the 3D Viewport.

1. Press `Tab` to put the Monkey object into *Edit Mode*.
2. With all the *Mirror Axis* options disabled move one of the Monkey object's vertices slightly.
3. Then Turn the *X Axis Mirror* on but leave *Topology Mirror* disabled.
4. If you now move that vertex again, the *X Axis Mirror* will not work and the mirrored vertices will not be altered.
5. If you then enable *Topology Mirror* and move the same vertices again, then *X Axis Mirror* should still mirror the other vertex, even though they are not perfectly positioned.

Auto Merge

Reference

Mode::	Edit Mode
Menu::	*Sidebar ▸ Tool ▸ Options ▸ Auto Merge*

When enabled, as soon as a vertex moves closer to another one than the *Threshold* setting, they are automatically merged. This option affects interactive operations only (tweaks made in the Adjust Last Operation panel are considered interactive too). If the exact spot where a vertex is moved contains more than one vertex, then the merge will be performed between the moved vertex and one of those.

Split Edges & Faces
: Detects the intersection of each transformed edge, creating a new vertex in place and sectioning the edge and the face if any.

Threshold
: Defines the maximum distance between vertices that are merged.

UVs

Live Unwrap

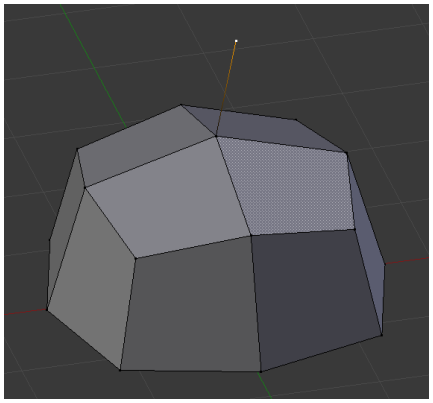
Automatically recalculates the UV unwrapping every time an edge has its seam property changed. Note, this is different than the Live Unwrap option in the UV Editor.

Extrude Region

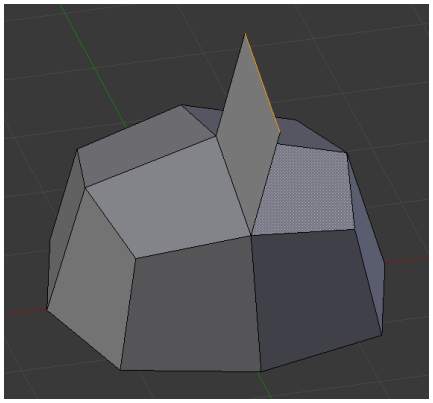
Reference

Mode::	Edit Mode
Tool::	*Toolbar ▸ Extrude Region*
Shortcut::	E

Extrusion tools duplicate vertices, while keeping the new geometry connected with the original vertices. Vertices are turned into edges and edges will form faces.



Single vertex extruded.



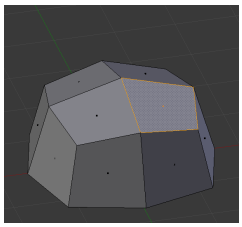
Single edge extruded.

This tool is of paramount importance for creating new geometry. It allows you to create parallelepipeds from rectangles and cylinders from circles, as well as easily creating such things as tree limbs.

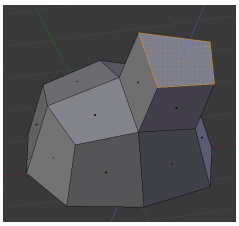
The axis on which vertices and edges are extruded along can be set interactively. Faces are extruded by default along their averaged normal. The extrusion can be limited to a single axis by specifying an axis; see Axis Locking.

The extrude tools differentiate in how the new geometry is connected in itself.

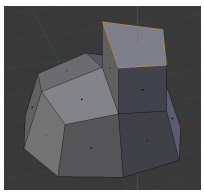
Only the border loop gets extruded. The inner region of the selection gets moved unchanged with the extrusion.



Selected face.



During extrude.



Set to Z axis.

Details

Although the process is quite intuitive, the principles behind *Extrude* are fairly elaborate as discussed below:

- First, the algorithm determines the outside edge loop of the extrude; that is, which among the selected edges will be changed into faces. By default (see below), the algorithm considers edges belonging to two or more selected faces as internal, and hence not part of the loop.
- The edges in the edge loop are then changed into faces.
- If the edges in the edge loop belong to only one face in the complete mesh, then all of the selected faces are duplicated and linked to the newly created faces. For example, rectangles will result in parallelepipeds during this stage.
- In other cases, the selected faces are linked to the newly created faces but not duplicated. This prevents undesired faces from being retained “inside” the resulting mesh. This distinction is extremely important since it ensures the construction of consistently coherent, closed volumes at all times when using *Extrude*.

- When extruding completely closed volumes (like e.g. a cube with all its six faces), extrusion results merely in a duplication, as the volume is duplicated, without any link to the original one.
- Edges not belonging to selected faces, which form an “open” edge loop, are duplicated and a new face is created between the new edge and the original one.
- Single selected vertices which do not belong to selected edges are duplicated and a new edge is created between the two.

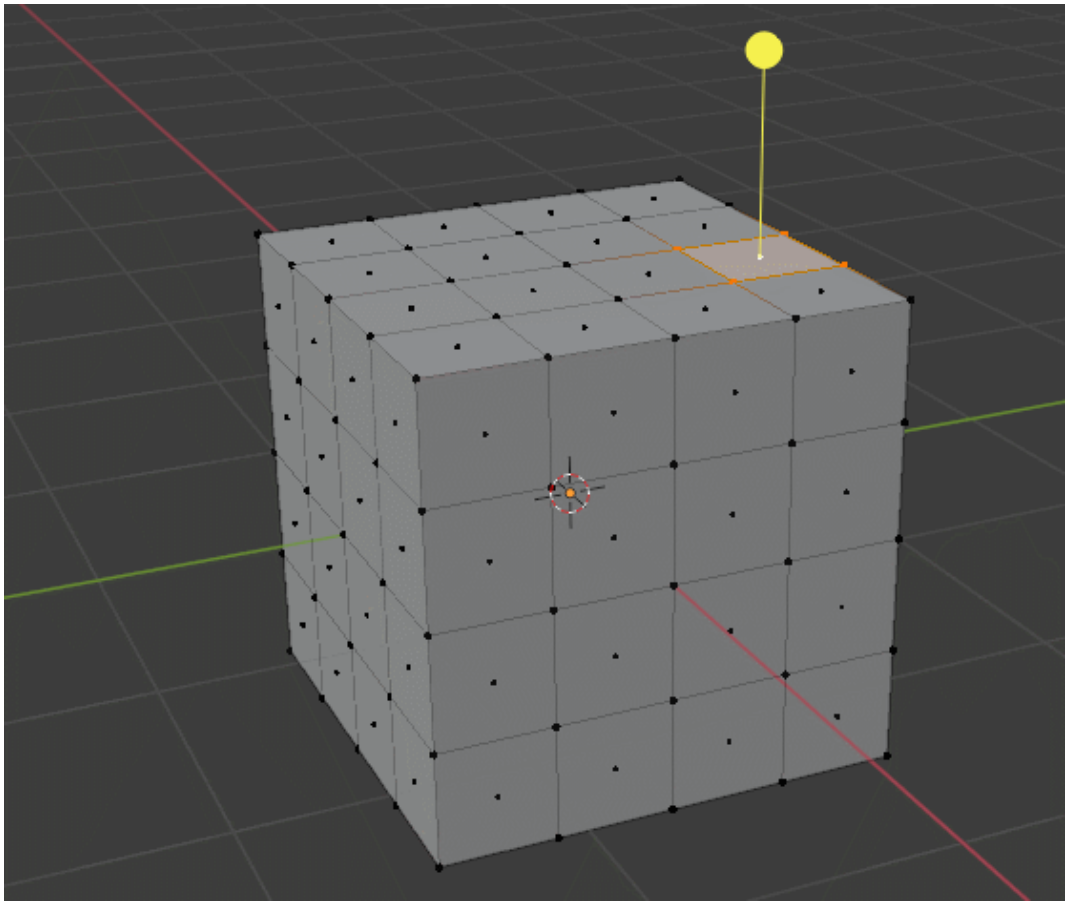
Extrude Manifold

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ‣ *Extrude Manifold*
Menu::	*Mesh* ‣ *Extrude* ‣ *Extrude Manifold*

This tool is very similar to Extrude Faces but enables *Dissolve Orthogonal Edges* by default. This causes the tool to automatically split and remove adjacent faces when extruding inwards.

Example



Extrude Manifold Example.

Extrude to Cursor

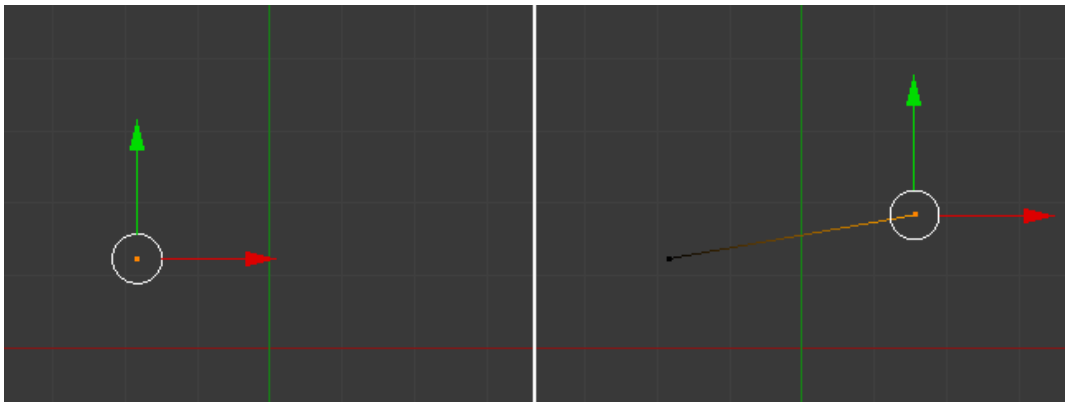
Reference

Mode::	Edit Mode
Shortcut::	`Ctrl-RMB`

Interactively places new vertices with `Ctrl-RMB` at the mouse cursor position.

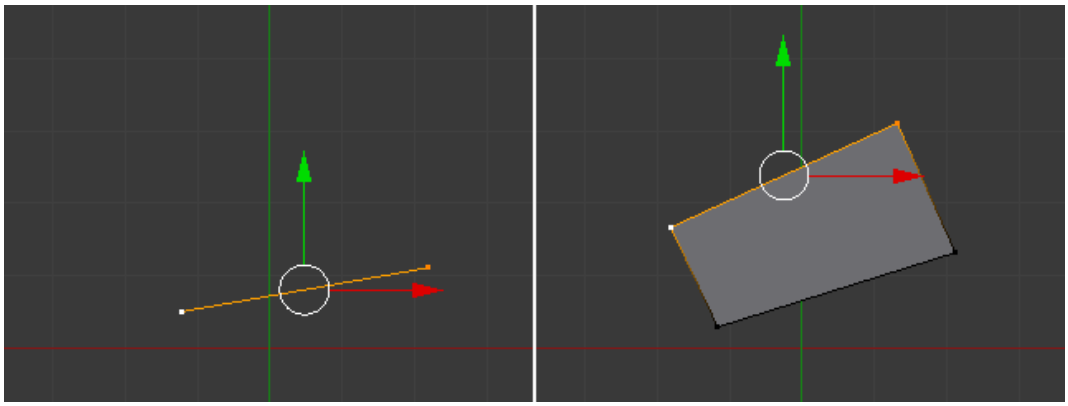
The most basic element, a vertex, can be added with a `Ctrl-RMB` click when no other vertices are selected. Because the camera space (computer screen) is two-dimensional, Blender cannot determine all three vertex coordinates from a single mouse click, so the new vertex is placed at the depth of the 3D cursor.

To create interconnected vertices, you can add a vertex and continuously make subsequent `Ctrl-RMB` operations with the last vertex selected. This will link the last selected vertex with the vertex created at the mouse position with an edge (see Fig. Adding vertices one by one.), and will continuously create and connect new vertices if you continue repeating this operation.



Adding vertices one by one.

Creating Faces



Quad from an Edge with source automatically rotated.

If you have two vertices selected and already connected with an edge, `Ctrl-RMB` click will create a planar face, also known as a quad. Blender will follow your mouse cursor and will use the planar view from your viewport to create those quads.

For `Ctrl-RMB`, Blender will automatically rotate the last selected Edge (the source) for the subsequent operations if you have at least one face created, dividing the angles created between the newly created edge and the last two edges, creating a smooth angle between them. Blender will calculate this angle using the last positive and negative position of the last X and Y coordinates and the last connected unselected edge. If this angle exceeds a negative limit (following a quadrant rule) between the recently created edge and the last two, Blender will wrap the faces. But if you do not want Blender to rotate and smooth edges automatically when extruding from `Ctrl-RMB`, you can also inhibit Blender from rotating sources using the shortcut `Shift-Ctrl-RMB`. In this case, Blender will not rotate the source dividing the angle between those edges when creating a face.

If you have three or more vertices selected, and `Ctrl-RMB` click, you will also create planar faces, but along the vertices selected, following the direction of the cursor. This operation is similar to an extrude operation.

Tip

When adding objects with `Ctrl-RMB`, the extrusions of the selected elements, being vertices, edges and faces with the `Ctrl-RMB`, are viewport dependent. This means, once you change your viewport, for example, from top to left, bottom or right, the extrusion direction will also follow your viewport and align the extrusions with your planar view.

Loop Cut

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Loop Cut*
Shortcut::	`Ctrl-R`

The *Loop Cut* tool is a modal tool version of the Loop Cut and Slide operator. This tool splits a loop of faces by inserting new edge loops intersecting the chosen edge.

Usage

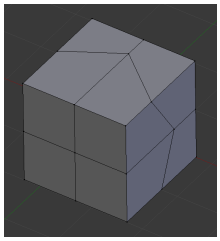
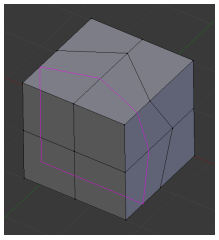
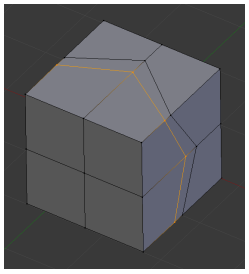
The tool is interactive and has two steps:

1. Pre-Visualizing the Cut

   After the tool is activated, move the cursor over a desired edge. The cut to be made is marked with a magenta colored line as you move the mouse over the various edges. The to be created edge loop stops at the poles (tris and n-gons) where the existing face loop terminates.

2. Perform the Cut

   Once the desired location of the new edge loop is found, the edge loop can be created via `LMB`.

Mesh before inserting edge loop.	Preview of edge loop location.	Interactive placement of edge loop between adjacent loops.

Tool Settings

Number of Cuts
: Increases and decreases the number of cuts to create. These cuts are uniformly distributed in the original face loop, and you will *not* be able to control their positions.

Correct UVs
: Corrects the corresponding UV coordinates, if these exist, to avoid image distortions.

Options

After the modal tool is run the Loop Cut and Slide Options are available in the Adjust Last Operation panel.

Poly Build

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Poly Build*

Poly Build combines several mesh editing tools into one, letting you work more quickly. It's especially useful for retopology.

Tool Settings

Create Quads
: When creating a new triangle that shares an edge with an existing one, automatically dissolves this edge so you're left with a quad.

Controls

Adding Geometry `Ctrl-LMB`
: Creates a new vertex at the mouse cursor, then creates a triangle using this new vertex and the nearest existing edge. If the existing edge already has two neighboring faces, instead creates a new edge using the new vertex and the nearest existing vertex. Holding `Ctrl` will preview the result in blue.

Deleting Geometry `Shift-LMB`
: Dissolves the vertex/deletes the face under the mouse cursor. Holding `Shift` will highlight the target element in red.

Moving Vertices `LMB`
: You can move a vertex by dragging it.

Extruding Edges `LMB`
: You can extrude an edge into a quad by dragging it.

Tip

It is useful to enable Snapping and Auto Merge while tweaking vertices to combine them.

Spin

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Extrude ▸ Spin*
Tool::	*Toolbar ▸ Spin*

The *Spin* tool extrudes (or duplicates it if the selection is manifold) the selected elements, rotating around a specific point and axis.

Use the tool to create the sort of objects that you would produce on a lathe (this tool is often called “lathe” tool or “sweep” tool in the literature, for this reason). In fact, it does a sort of circular extrusion of your selected elements, centered on the 3D cursor, and around the axis perpendicular to the working view…

- The point of view will determine around which axis the extrusion spins…
- The position of the 3D cursor will be the center of the rotation.

Tool Settings

Steps
: Specifies how many copies will be extruded along the “sweep”.

Use Duplicates
: When enabled, will keep the original selected elements as separated islands in the mesh (i.e. unlinked to the result of the spin extrusion).

Axis
: Specifies the axis to use as the pivot of the spin operation.

Options

Steps
: Specifies how many copies will be extruded along the “sweep”.

Angle
: Specifies the angle “swept” by this tool, in degrees (e.g. set it to 180 for half a turn).

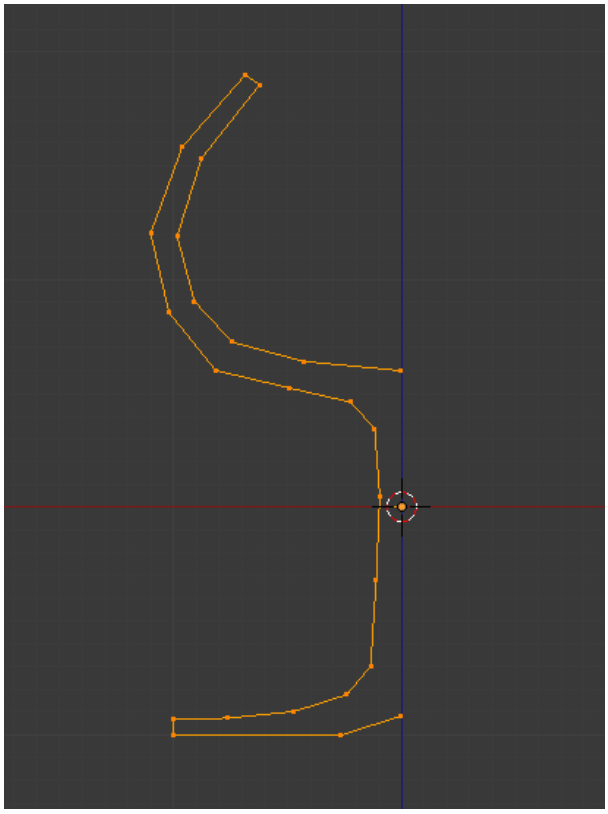
Auto Merge
: Automatically merges the first a last duplicates, if they make a full revolution which results in overlapping geometry.

Flip Normals
: Reverses the Normal’s direction for any resulting geometry.

Center X, Y, Z
: Specifies the center of the spin. By default it uses the cursor position.

Axis X, Y, Z
: Specify the spin axis as a vector. By default it uses the view axis (viewport).

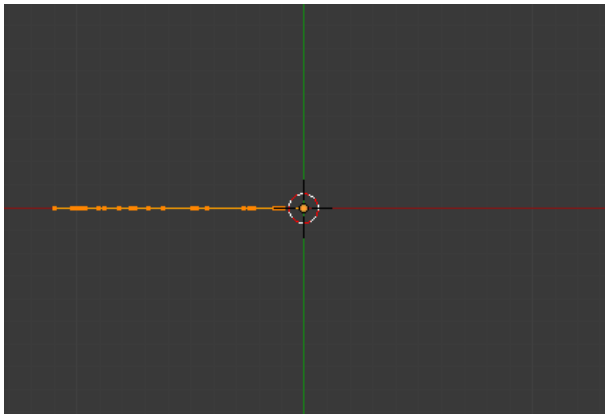
Example



Glass profile.

First, create a mesh representing the profile of your object. If you are modeling a hollow object, it is a good idea to thicken the outline. Fig. Glass profile. shows the profile for a wine glass we will model as a demonstration.

We will be rotating the object around the cursor in the top view, so switch to the top view with `Numpad7`.



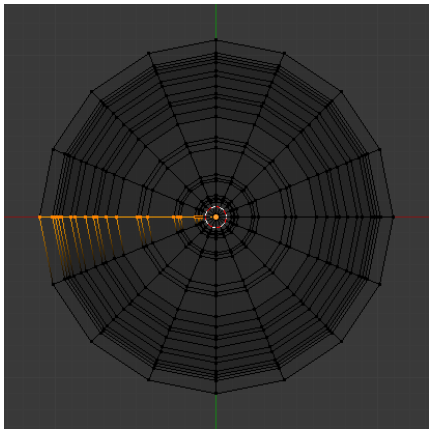
Glass profile, top view in Edit Mode, just before spinning.

Place the cursor along the center of the profile by entering Edit Mode and selecting one of the vertices along the center, and snapping the 3D cursor to that location with *Mesh* ▸ *Snap* ▸ *Cursor to Selection*. (Fig. Glass profile, top view in Edit Mode, just before spinning.) shows the wine glass profile from top view, with the cursor correctly positioned.

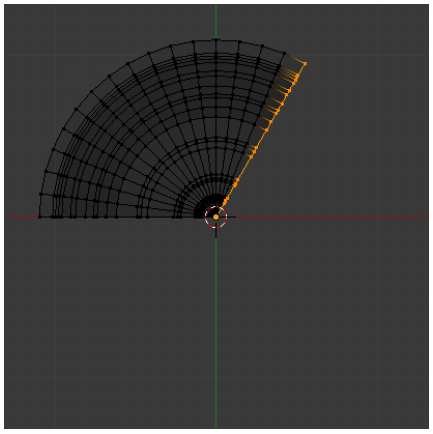
Select all the vertices with `A` and select the *Spin* tool from the Toolbar and use the Gizmo to spin the vertices. Fig. Spun profile. shows the result of a successful spin.

Angle

Spun profile.

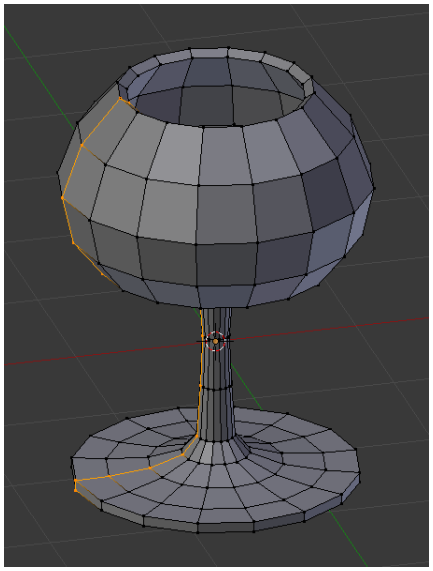


Spun profile using an angle of 360.

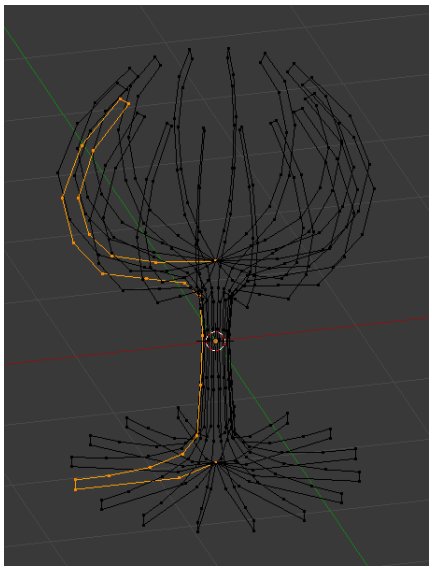


Spun profile using an angle of 120.

Duplicate

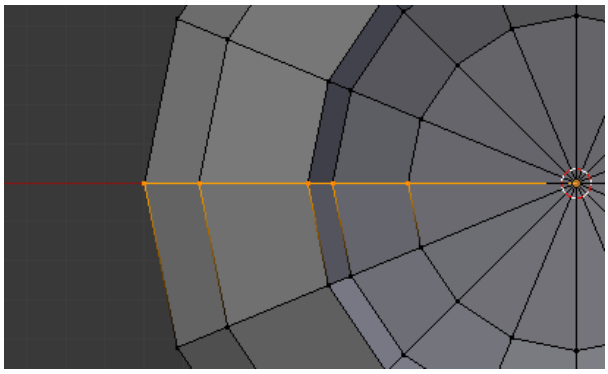


Result of spin operation.



Result of *Duplicate* enabled.

Merge Duplicates



Duplicate vertices.

The spin operation leaves duplicate vertices along the profile. You can select all vertices at the seam with Box select `B` (shown in Fig. Duplicate vertices.) and perform a *Merge by Distance* operation.

Notice the selected vertex count before and after the *Merge by Distance* operation `Vertex count after removing doubles`. If all goes well, the

final vertex count (38 in this example) should match the number of the original profile noted in *Mesh data ▸ Vertex and face numbers*. If not, some vertices were missed and you will need to weld them manually. Or, worse, too many vertices will have been merged.

Note

Merging Two Vertices into One

To merge (weld) two vertices together, select both of them by `Shift–LMB` clicking on them. Press `S` to start scaling and hold down `Ctrl` while scaling to scale the points down to 0 units in the X, Y and Z axis. `LMB` to complete the scaling operation and click *Mesh ▸ Merge ▸ By Distance* to merge the vertices. Alternatively, you can use *Context Menu ▸ Merge Vertices* (or `M`). Then, in the new pop-up menu, choose to merge *By Distance*.

Recalculate Normals

All that remains now is to recalculate the normals to the outside by selecting all vertices, pressing `Alt–N` and validating *Recalculate Normals Outside* in the pop-up menu.

Selecting

- Introduction
  - Selection Modes
  - X-Ray
  - Select Menu
  - Known Issues
- Select Mirror
  - Example
- Select Random
- Checker Deselect
- Select More/Less
  - Select Next/Previous Active
- Select Similar
  - Face Regions
- Select All by Trait
  - Non-Manifold
  - Loose Geometry
  - Interior Faces
  - Faces by Sides
  - Ungrouped Vertices
- Select Linked
  - Linked
  - Shortest Path
  - Linked Flat Faces
- Select Loops
  - Select Edge Loops
  - Select Face Loops
  - Select Edge Rings
  - Select Loop Inner-Region
  - Select Boundary Loop
- Select Sharp Edges
- Side of Active
- By Attribute
  - Usage

Introduction

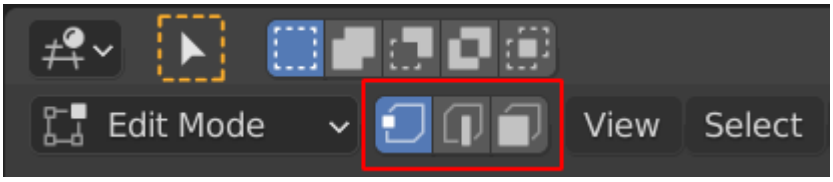
There are many ways to select elements, and it depends on what *Mesh Select Mode* you are in as to what selection tools are available. First we will go through these modes and after that a look is taken at basic selection tools.

Selection Modes

Reference

Mode::	Edit Mode
Menu::	*3D Viewport Header* ▸ *Select Mode*
Shortcut::	`1`, `2`, `3` (`Shift` Multiple Selection Modes, `Ctrl` Expand/Contract Selection).

In *Edit Mode* there are three different selection modes. You can enter the different modes by selecting one of the three buttons in the header.



Edit Mode selection buttons from right to left: Vertex, Edge, Face.

Vertex
: In this mode vertices are shown as points. Selected vertices are displayed in orange, unselected vertices in black, and the active or last selected vertex in white.

Edge
: In this mode the vertices are not shown. Instead the selected edges are displayed in orange, unselected edges black, and the active or last

selected edge in white.

Face
: In this mode the faces are displayed with a selection point in the middle which is used for selecting a face. Selected faces and their selection point are displayed in orange, unselected faces are displayed in black, and the active or last selected face is highlighted in white.

When using these buttons, you can make use of modifier keys, see: Switching Select Mode.

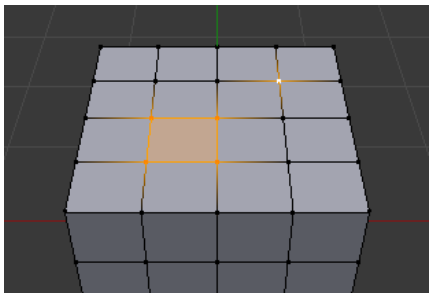
Almost all tools are available in all three mesh selection modes. So you can *Rotate*, *Scale*, *Extrude*, etc. in all modes. Of course rotating and scaling a *single* vertex will not do anything useful (*without* setting the pivot point to another location), so some tools are more or less applicable in some modes.

See Fig. Selection modes. for examples of the different modes.

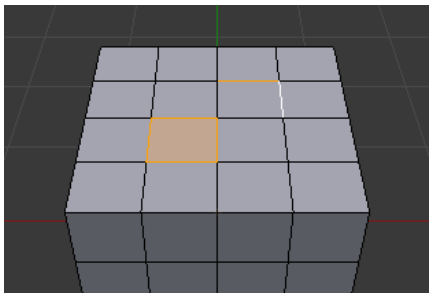
Multiple Selection Modes

By holding `Shift-LMB` when selecting a selection mode, you can enable multiple *Selection Modes* at once. This allows you to quickly select vertices, edges, or faces, without first having to switch mode.

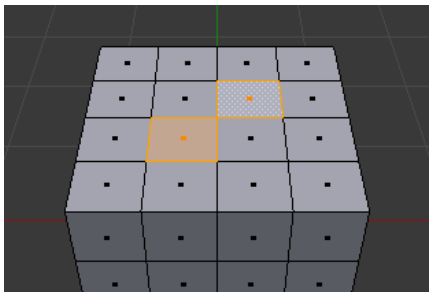
Selection modes.



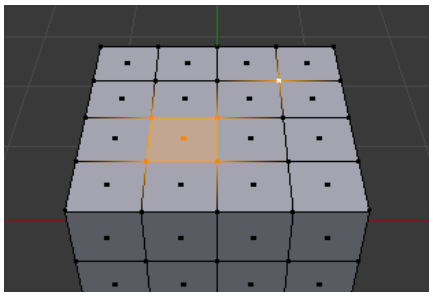
Vertex mode example.



Edge mode example.



Face mode example.

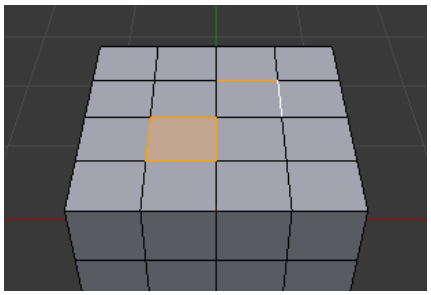


Mixed mode example.

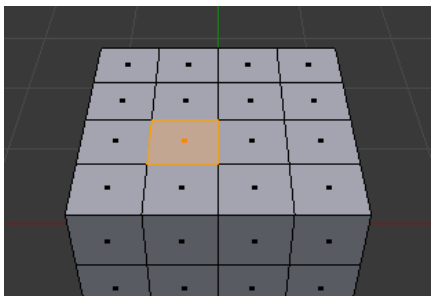
Switching Select Mode

When switching modes in an “ascendant” way (i.e. from simpler to more complex), from *Vertices* to *Edges* and from *Edges* to *Faces*, the selected parts will still be selected if they form a complete element in the new mode.

For example, if all four edges in a face are selected, switching from *Edges* mode to *Faces* mode will keep the face selected. All selected parts that do not form a complete set in the new mode will be unselected.



Edge mode, the initial selection.

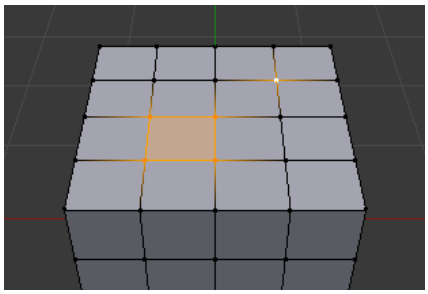


Switching to Face mode.

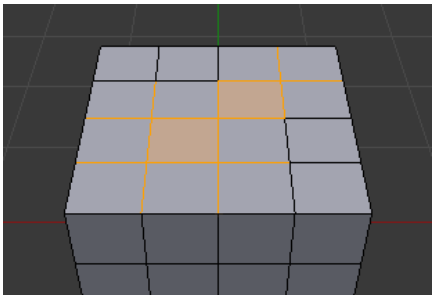
Hence, switching in a “descendant” way (i.e. from more complex to simpler), all elements defining the “high-level” element (like a face) will be selected (the four vertices or edges of a quadrangle, for example).

Expand/Contract Selection

By holding `Ctrl` when selecting a higher selection mode, all elements touching the current selection will be added, even if the selection does not form a complete higher element. Or contracting the selection when switching to a lower mode.



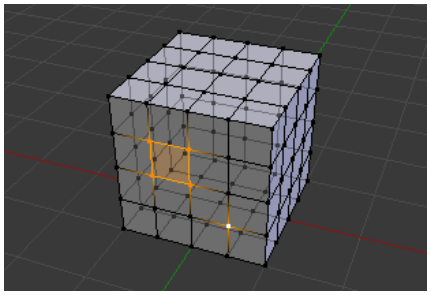
Vertex mode, the initial selection.



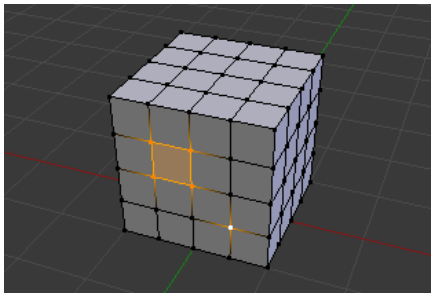
Expanding to Edge mode.

X-Ray

The X-Ray setting is not just for shading, it impacts selection too. When enabled, selection isn't occluded by the objects geometry (as if the object was solid).



X-ray enabled.



X-ray disabled.

Select Menu

All `A`
: Select all.

None `Alt-A`
: Select none.

Invert `Ctrl-I`
: Selects all the geometry that is not selected, and deselect currently selected components.

Box Select `B`
: Interactive box selection.

Circle Select `C`
: Interactive circle selection.

Lasso Select
: Interactive free-form selection.

Select Mirror `Shift-Ctrl-M`
: Select mesh items at the mirrored location across the chosen axis.

Select Random
: Selects a random group of vertices, edges, or faces, based on a percentage value.

Checker Deselect
: Deselect alternate elements relative to the active item.

More/Less

More `Ctrl-NumpadPlus`
: Expands the selection to the adjacent elements of the selection type.

Less `Ctrl-NumpadMinus`
: Contracts the selection from the adjacent elements of the selection type.

Next Active `Shift-Ctrl-NumpadPlus`
: This uses selection history to select the next vertex, edge, or face based on surrounding topology.

Previous Active `Shift-Ctrl-NumpadMinus`
: Select previous just removes the last selected element.

Select Similar `Shift-G`
: Select elements similar to the current selection.

Select All by Trait
: Select geometry by querying its characteristics.

Select Linked
: Select Linked
  : Selects all components that are connected to the current selection.

  Shortest Path
  : Path between two selected elements.

  Linked Flat Faces
  : Select connected faces based on a threshold of the angle between them. This is useful for selecting faces that are planar.

Select Loops
: Edge Loops
  : Select connected edges.

  Face Loops
  : Select connected faces.

  Edge Rings
  : Select connected edge ring.

Sharp Edges
: This tool selects all edges between two faces forming an angle greater than the angle value, where an increasing angle selects sharper edges.

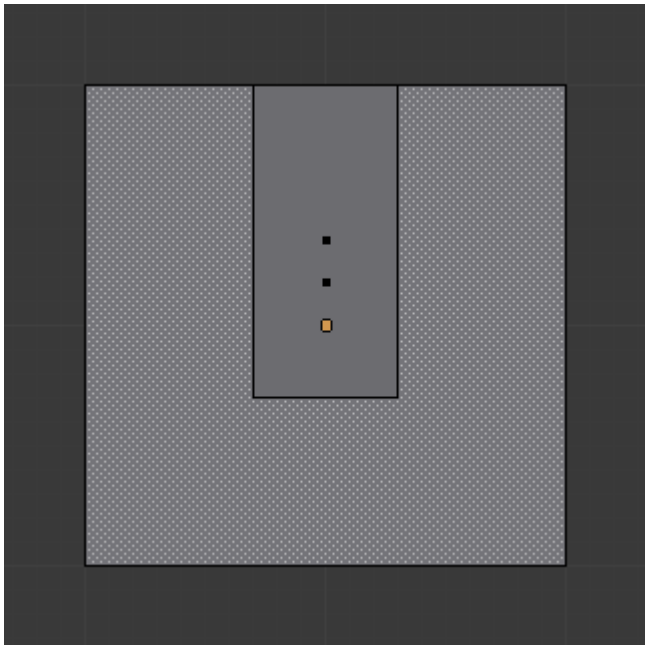
Side of Active
: Selects all vertices on the mesh in a single axis relative to the active vertex. In Vertex selection mode only.

Known Issues

Dense Meshes

Selecting dense meshes with X-Ray disabled, has a limitation where dense meshes may not have all the elements selected. When selecting regions with Box, Circle and Lasso select, vertices may overlap each other causing some vertices not to be selected. This is a limitation with the current selection method, you may workaround this by zooming in or enabling X-Ray.

N-Gons in Face Select Mode



N-gon face having its center dot inside another face.

As already noted, in X-Ray and Wireframe mode faces are marked with a dot in the middle. With n-gons that can lead in certain cases to a confusing display. The example shows the center dot of the U-shaped n-gon being inside of the oblong face inside the “U”. It is not easy to identify which dot belongs to which face (the orange dot in the image is the object origin).

Select Mirror

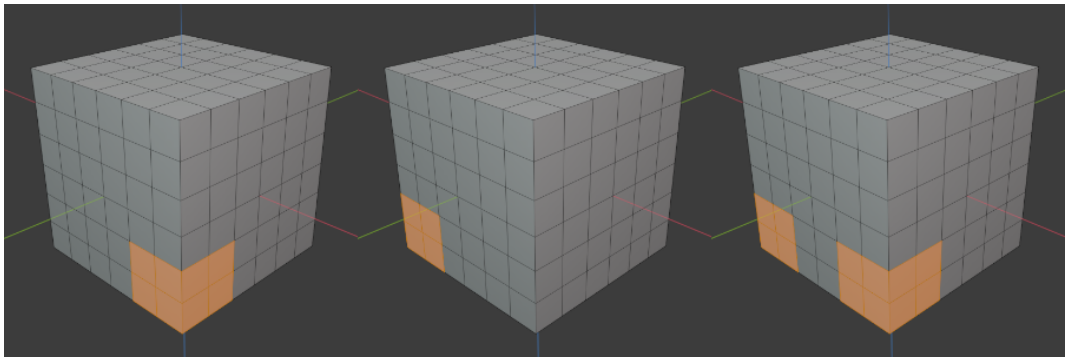
Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Mirror*
Shortcut::	`Shift-Ctrl-M`

Select Mirror flips a selection to the opposite side of the mesh.

Axis
: Choose on which axis the selection will occur. The axis is based on the meshes origin. Therefore, if the origin is not centered within the mesh, the selection will have varying results.

Extend
: The new selection will include the mirrored selection as well as the original.



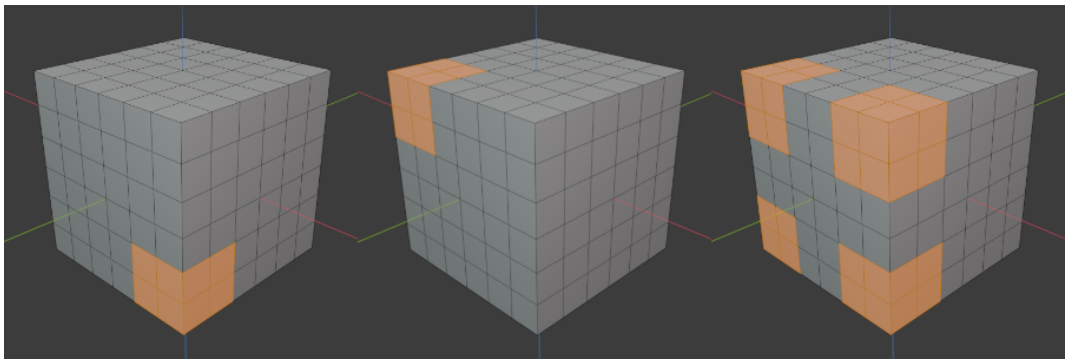
(From left to right) initial selection, after Select Mirror on the X axis, with Extend.

Tip

With *Extend* activated, hold `Shift` while choosing an axis to include more than one axis in the selection. Otherwise, with *Extend* off, the mirror will

take into account two to three axes.

Example



(From left to right) initial selection, mirrored along X and Z axes, with Extend.

Select Random

Reference

Mode::	Object Mode
Menu::	*Select ▸ Select Random*

Randomly selects unselected vertices, edges, or faces based on percentage probability.

Percent
: The likelihood of an unselected mesh element being selected. Note that this is not the percentage amount of elements that will be selected.

Checker Deselect

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Checker Deselect*

This tool applies an alternating selected/deselected checker pattern. This only works if you already have more than one mesh element selected.

Changes the current selection so that only every Nth elements (vertices, edges or faces, depending on the active selection mode) will remain selected, starting from the active one.

In case of islands of selected elements, this tool will affect only the island of the active element (if there is one), or the island of the first element in the order of internal storage (if there is no active element).

Deselected
: The number of deselected elements in each pattern repetition.

Selected
: The number of selected elements in each pattern repetition.

Offset
: Offset from the starting point.

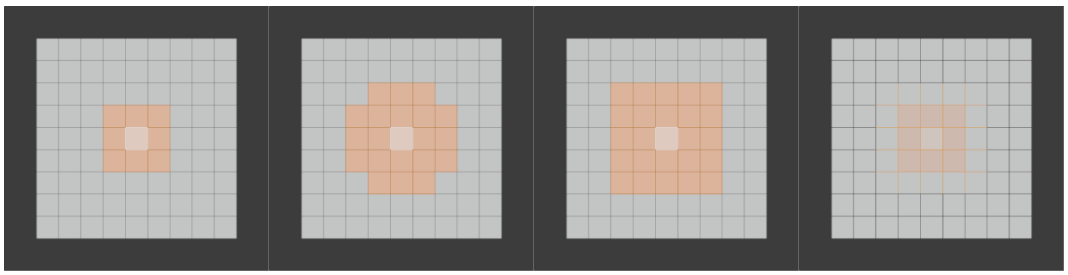
Select More/Less

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select More/Less* ▸ *More*
Shortcut::	`Ctrl-NumpadPlus`
Menu::	*Select* ▸ *Select More/Less* ▸ *Less*
Shortcut::	`Ctrl-NumpadMinus`

With at least one vertex, edge, or face selected, *Select More/Less* expands or shrinks the selection. However, if there is only one selection in any selection mode, *Less* will deselect it.

Face Step
: With *Face Step* on, each use of the tool will affect the size of the selection on a face by face basis. When deactivated, it will be based on either vertices or edges depending on which Selection Mode is active.



(From left to right) initial selection, without Face Step, with Face Step, and in edge selection mode.

Select Next/Previous Active

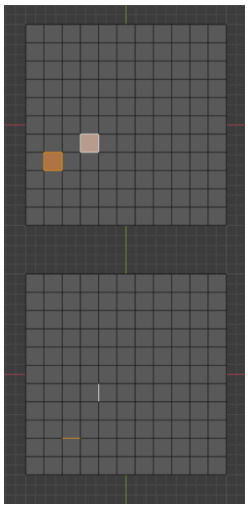
Reference

Mode::	Edit Mode

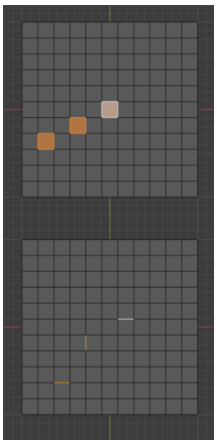
Menu:: *Select ▸ Select More/Less ▸ Next Active*
Shortcut:: `Shift-Ctrl-NumpadPlus`
Menu:: *Select ▸ Select More/Less ▸ Previous Active*
Shortcut:: `Shift-Ctrl-NumpadMinus`

Next Active

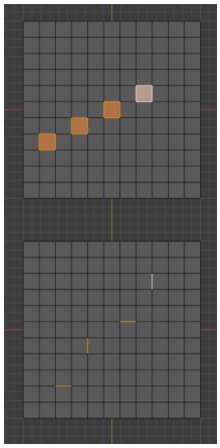
This uses selection history to select the next vertex, edge, or face based on the surrounding topology. Which means that, it will derive the next selection from the previous two selections.



Initial selection.



Using Next Active once.



Using Next Active twice.

Previous Active

Only the last selected element will be removed.

Select Similar

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Similar*
Shortcut::	`Shift-G`

Select geometry that has similar certain properties to the ones selected, based on a threshold that can be set in tool properties after activating the tool. Tool options change depending on the selection mode:

Vertex Selection Mode:

Normal
: Selects all vertices that have normals pointing in similar directions to those currently selected.

Amount of Adjacent Faces
: Selects all vertices that have the same number of faces connected to them.

Vertex Groups
: Selects all vertices in the same vertex group.

Amount of Connecting Edges
: Selects all vertices that have the same number of edges connected to them.

Edge Selection Mode:

Length
: Selects all edges that have a similar length as those already selected.

Direction
: Selects all edges that have a similar direction (angle) as those already selected.

Amount of Faces Around an Edge
: Selects all edges that belong to the same number of faces.

Face Angles
: Selects all edges that are between two faces forming a similar angle, as with those already selected.

Crease
: Selects all edges that have a similar Crease value as those already selected.

Bevel
: Selects all edges that have the same *Bevel Weight* as those already selected.

Seam
: Selects all edges that have the same *Seam* state as those already selected. *Seam* is a mark used in UV texturing.

Sharpness
: Selects all edges that have the same *Sharp* state as those already selected. *Sharp* is a mark used by the Edge Split Modifier.

Face Selection Mode:

Material
: Selects all faces that use the same material as those already selected.

Area
: Selects all faces that have a similar area as those already selected.

Polygon Sides
: Selects all faces that have the same number of edges.

Perimeter
: Selects all faces that have a similar perimeter (added values of its edge lengths).

Normal

Selects all faces that have a similar normal as those selected. This is a way to select faces that have the same orientation (angle).

Co-planar
: Selects all faces that are (nearly) in the same plane as those selected.

Flat/Smooth
: Selects all faces with similar face shading.

Freestyle Face Marks
: Selects all faces with similar Freestyle Face Marks.

Compare
: For quantitative properties, this property selects the type of comparison to between the two numerical values.

Equal::	Select items with the same value as the active item's chosen property.
Greater::	Select items with a larger value as the active item's chosen property.
Less::	Select items with a smaller value as the active item's chosen property.

Threshold
: For quantitative properties, this property controls how close the property's values have to be in the comparison.

Face Regions

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Similar ▸ Face Regions*

Select matching features on a mesh that has multiple similar areas based on the topology.

Select All by Trait

Non-Manifold

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select All by Trait ▸ Non-Manifold*

Selects the Non-manifold geometry of a mesh. This entry is available when editing a mesh, in Vertex and Edge selection modes only.

Extend
: Lets you extend the current selection.

Wire
: Selects all the edges that do not belong to any face.

Boundaries
: Selects edges in boundaries and holes.

Multiple Faces
: Selects edges that belong to three or more faces.

Non Contiguous
: Selects edges that belong to exactly two faces with opposite normals.

Vertices
: Selects vertices that belong to *wire* and *multiple face* edges, isolated vertices, and vertices that belong to non-adjoining faces.

Loose Geometry

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select All by Trait ▸ Loose Geometry*

This selection depends on the currently selected Selection Modes; In vertex and edge selection mode it selects all vertices or edges that do not form part of a face. In face selection mode it selects all faces that do not share edges with other faces.

Interior Faces

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select All by Trait ▸ Interior Faces*

Selects faces where all edges have more than two faces.

Faces by Sides

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select All by Trait ▸ Faces by Sides*

Selects all faces that have a specified number of edges.

Ungrouped Vertices

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select All by Trait ▸ Ungrouped Vertices*

Selects all vertices which are not part of a vertex group.

Select Linked

Linked

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Linked* ▸ *Linked*
Shortcut::	`Ctrl-L`

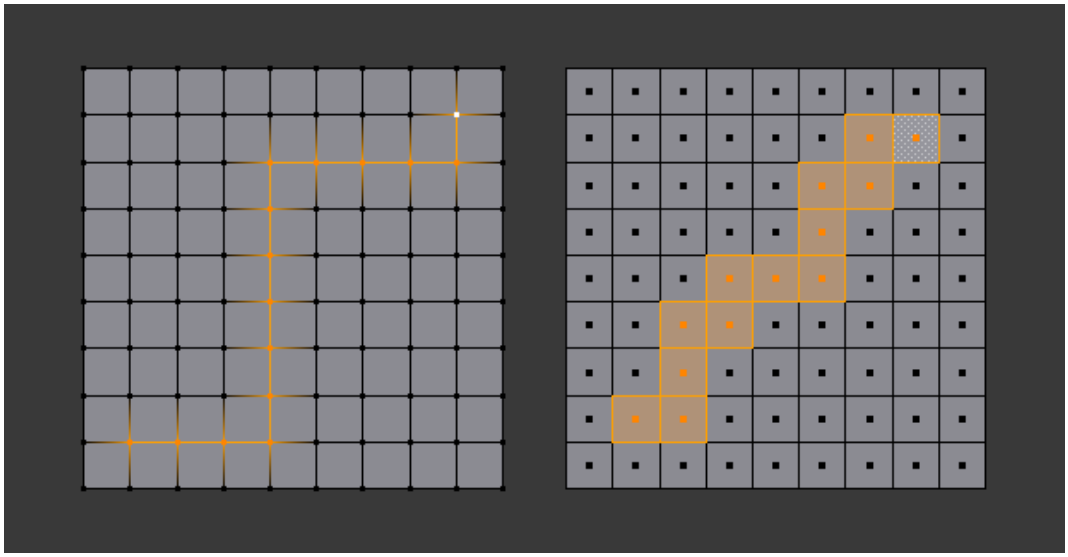
Select geometry connected to already selected elements. This is often useful when a mesh has disconnected, overlapping parts, where isolating it any other way would be tedious. To give more control, you can also enable delimiters in the Adjust Last Operation panel, so the selection is constrained by seams, sharp edges, materials or UV islands.

With *Pick Linked* you can also select connected geometry directly under the cursor, using the `L` shortcut to select or `Shift-L` to deselect linked. This works differently in that it uses the geometry under the cursor instead of the existing selection.

Shortest Path

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Linked* ▸ *Shortest Path*
Shortcut::	`Ctrl-LMB`



Select a face or vertex path with `Ctrl-LMB`.

Selects all geometry along the shortest path from the active vertex, edge, or face to the one which was selected.

Edge Tag (in Edge select mode only)
: This select button indicates what should be done when selecting a vertex path with `Ctrl-LMB`:

  Select
  : Just selects all the edges in the path.

  Tag Seam
  : Marks all edges in the path as seams for UV unwrapping.

  Tag Sharp
  : Marks all edges in the path as sharp for the Edge Split Modifier.

  Tag Crease
  : Marks all edges in the path as creases for the Subdivision Surface Modifier, with weight 1.0.

  Tag Bevel
  : Gives bevel weight 1.0 (for the Bevel Modifier) to all edges in the path.

Tag Freestyle Edge Mark
: Marks all edges in the path as Freestyle edges.

Face Stepping
: Supports diagonal paths for vertices and faces, and selects edge rings with edges.

Topology Distance
: Only takes into account the number of edges of the path and not the length of the edges to calculate the distances.

Fill Region `Shift-Ctrl-LMB`
: Selects all elements in the shortest paths from the active selection to the clicked area.

Checker Deselect Options
: Allows to quickly select alternate elements in a path.

  Deselected
  : The number of deselected elements in the repetitive sequence.

  Selected
  : The number of selected elements in the repetitive sequence.

  Offset
  : Offset from the starting point.

Linked Flat Faces

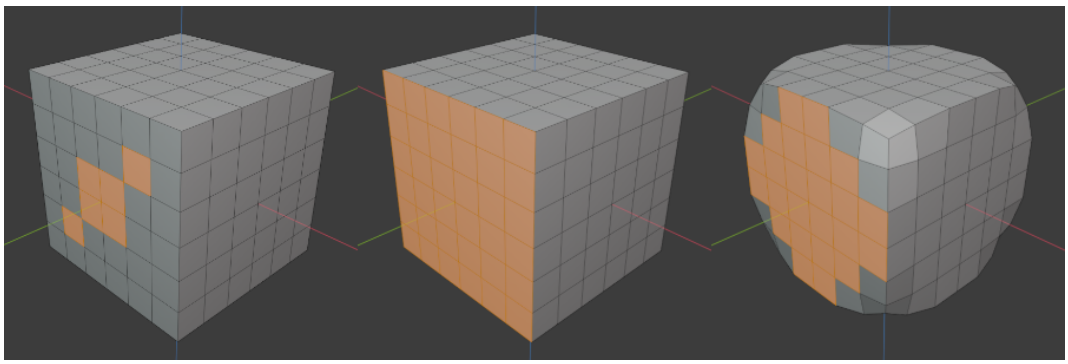
Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Linked ▸ Linked Flat Faces*

Selects all connected faces with a similar angle.

Sharpness

Select connected faces with a reference angle less than the value specified.



Looking at the image above, when at least one face is selected (as seen on the left), *Linked Flat Faces* will select all connecting faces that lie on the same or similar plane (as shown in the middle image). If the corners are smoothed, those faces are no longer lined up with the selected faces. At this point, increasing the *Sharpness* value in the tool options could include the smoothed faces.

Select Loops

Select Edge Loops

Reference

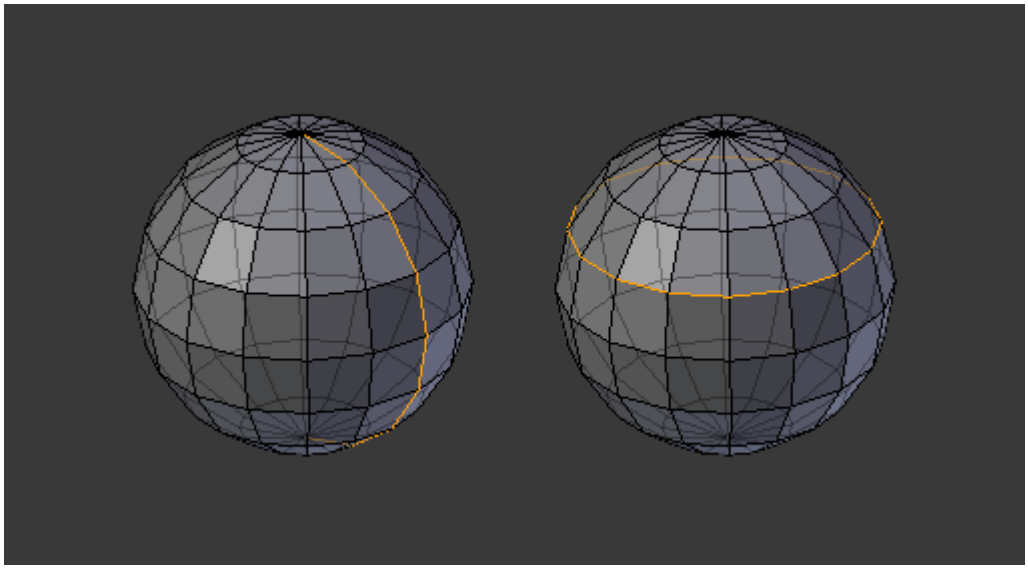
Mode::	Edit Mode (Vertex or Edge select mode)
Menu::	*Select* ▸ *Select Loops* ▸ *Edge Loops*
Shortcut::	`Alt-LMB`, or `Shift-Alt-LMB` for modifying existing selection.

Holding `Alt` while selecting an edge selects a loop of edges that are connected in a line end-to-end, passing through the edge under the mouse pointer. Holding `Shift-Alt` while clicking adds to the current selection.

Note

Vertex mode

In *Vertex* select mode, you can also select edge loops, by using the same shortcuts, and clicking on the *edges* (not on the vertices).



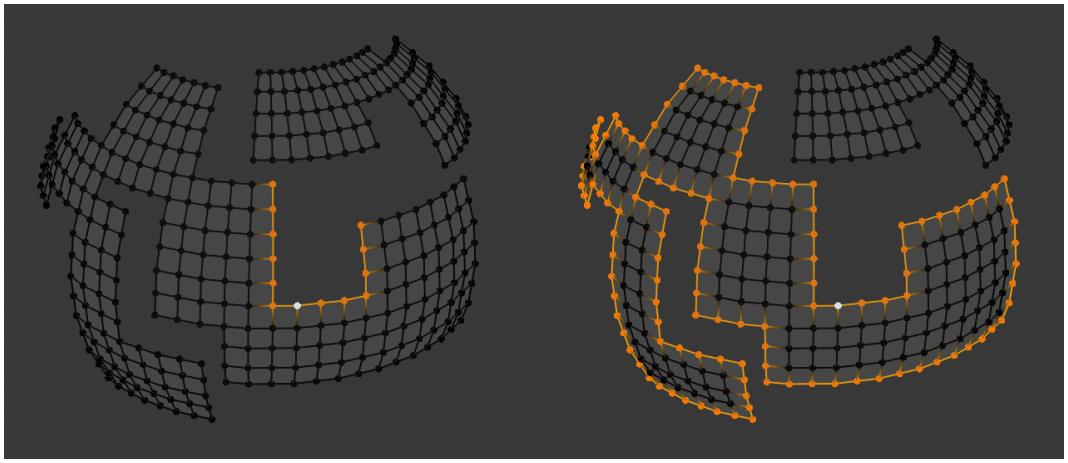
Longitudinal and latitudinal edge loops.

The left sphere shows an edge that was selected longitudinally. Notice how the loop is open. This is because the algorithm hit the vertices at the poles and is terminated because the vertices at the pole connect to more than four edges. However, the right sphere shows an edge that was selected latitudinally and has formed a closed loop. This is because the algorithm hit the first edge that it started with.

Select Edge Loops (All Boundaries)

All boundary edges can be selected by performing a second loop select action on a boundary edge.

This can be useful for selecting boundaries for meshes that include triangles and n-gons, where loop select would not otherwise select the full boundary.



The second loop select action is shown on the right.

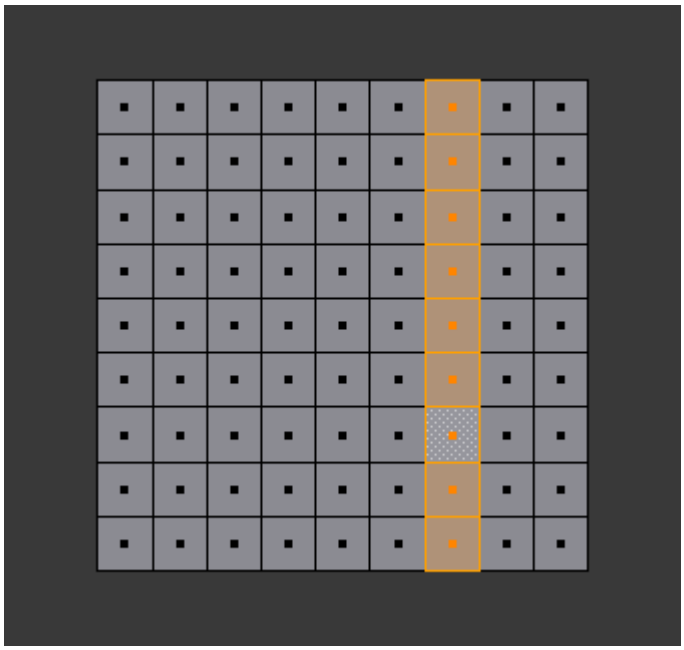
Select Face Loops

Reference

Mode::	Edit Mode (Face or Vertex select modes)
Shortcut::	`Alt-LMB` or `Shift-Alt-LMB` for modifying existing selection.

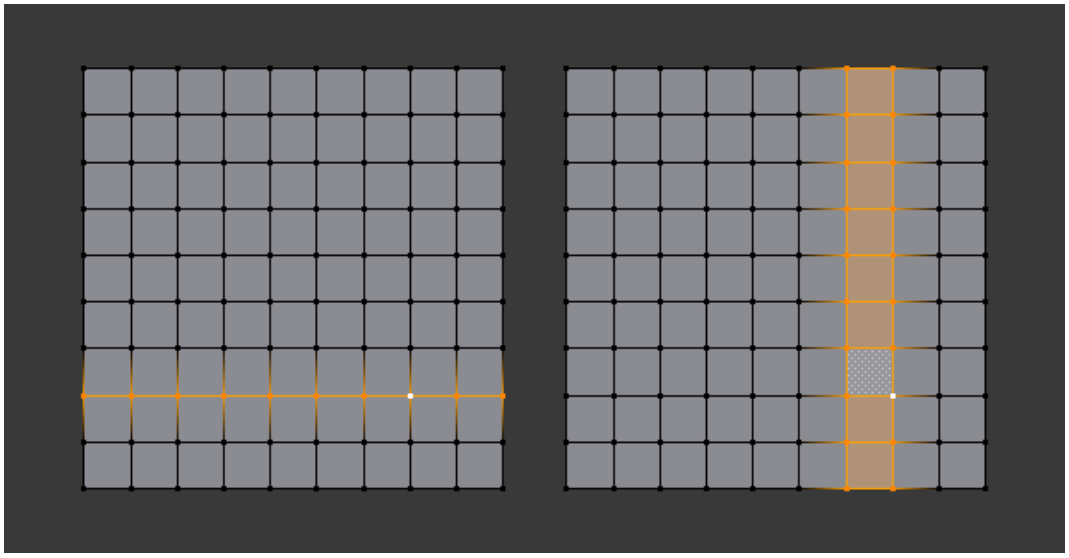
In face select mode, holding `Alt` while selecting an *edge* selects a loop of faces that are connected in a line end-to-end, along their opposite edges.

In vertex select mode, the same can be accomplished by using `Ctrl-Alt` to select an edge, which selects the face loop implicitly.



Face loop selection.

This face loop was selected by clicking with `Alt-LMB` on an edge, in *face* select mode. The loop extends perpendicular from the edge that was selected.



`Alt` versus `Ctrl-Alt` in vertex select mode.

A face loop can also be selected in *Vertex* select mode. Technically `Ctrl-Alt-LMB` will select an *Edge Ring*, however, in *Vertex* select mode, selecting an *Edge Ring* implicitly selects a *Face Loop* since selecting opposite edges of a face implicitly selects the entire face.

Select Edge Rings

Reference

Mode::	Edit Mode
Menu::	*Select* ‣ *Select Loops* ‣ *Edge Rings*
Shortcut::	`Ctrl-Alt-LMB`

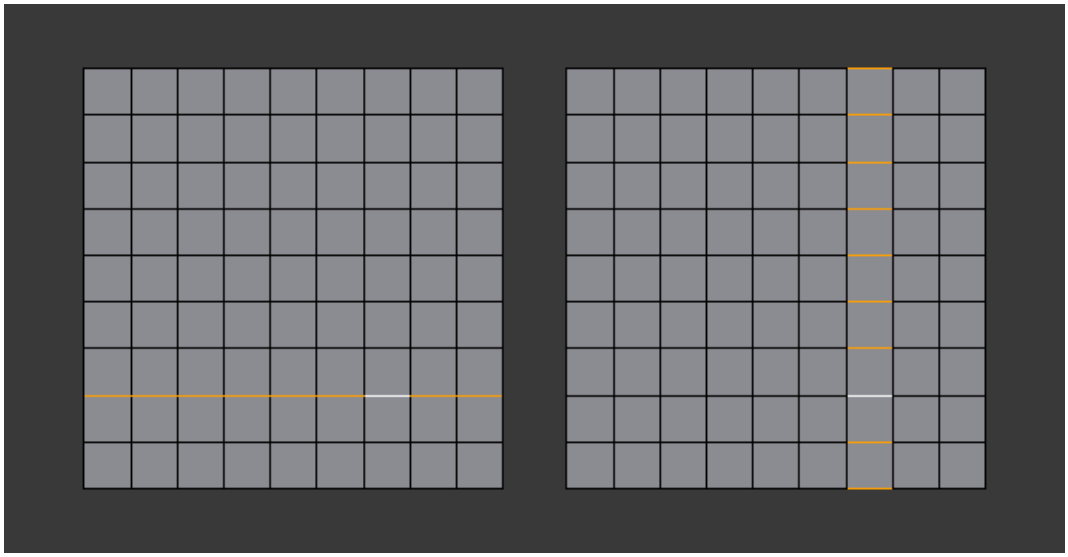
In *Edge* select mode, holding `Ctrl-Alt` while selecting an edge (or two vertices) selects a sequence of edges that are not connected, but on opposite sides to each other continuing along a face loop.

As with edge loops, you can also select edge rings based on current selection, using either *Select* ‣ *Select Loops* ‣ *Edge Rings*.

Note

Vertex mode

In *Vertex* select mode, you can use the same shortcuts when *clicking on the edges* (not on the vertices), but this will directly select the corresponding face loop…



A selected edge loop, and a selected edge ring.

In Fig. A selected edge loop, and a selected edge ring. the same edge was clicked on, but two different “groups of edges” were selected, based on the different tools. One is based on edges during computation and the other is based on faces.

Note

Convert Selection to Whole Faces

If the edge ring selection happened in Edge Select Mode, switching to Face Select Mode will erase the selection.

This is because none of those faces had all its (four) edges selected, just two of them.

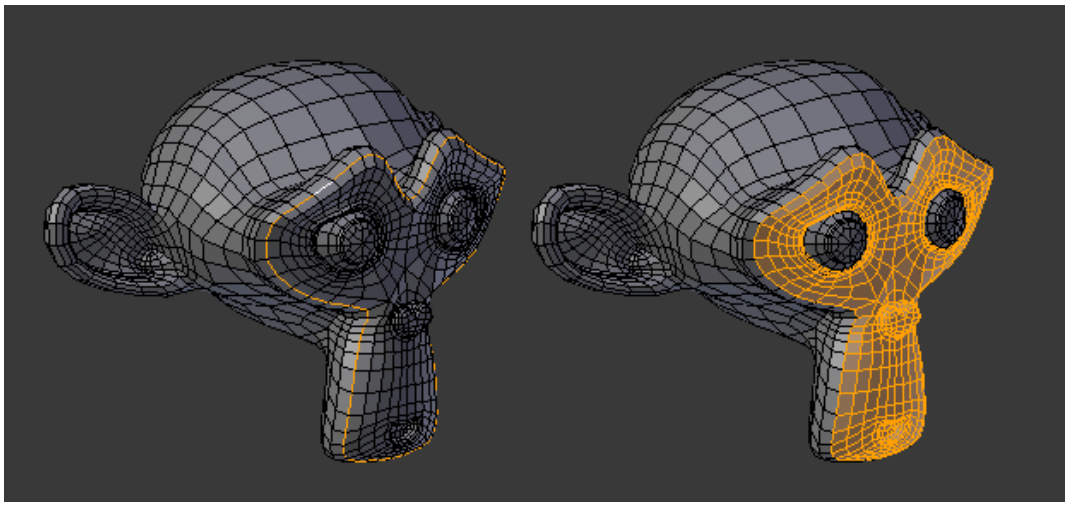
Instead of selecting the missing edges manually or by using `Shift-Alt-LMB` twice, it is easier to first switch to Vertex Select Mode, which will kind of “flood” the selection. A subsequent switch to Face Select Mode will then properly select the faces.

Select Loop Inner-Region

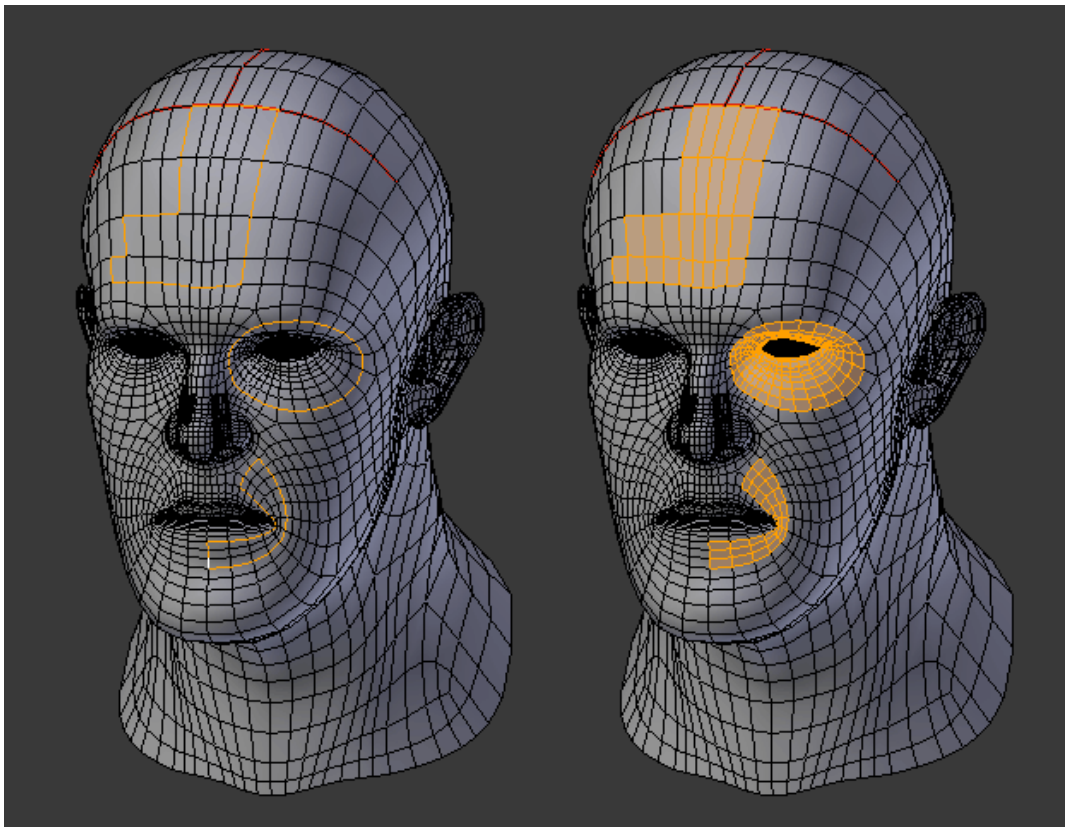
Reference

Mode::	Edit Mode (Edge select mode)
Menu::	*Select* ▸ *Select Loops* ▸ *Select Loop Inner-Region*

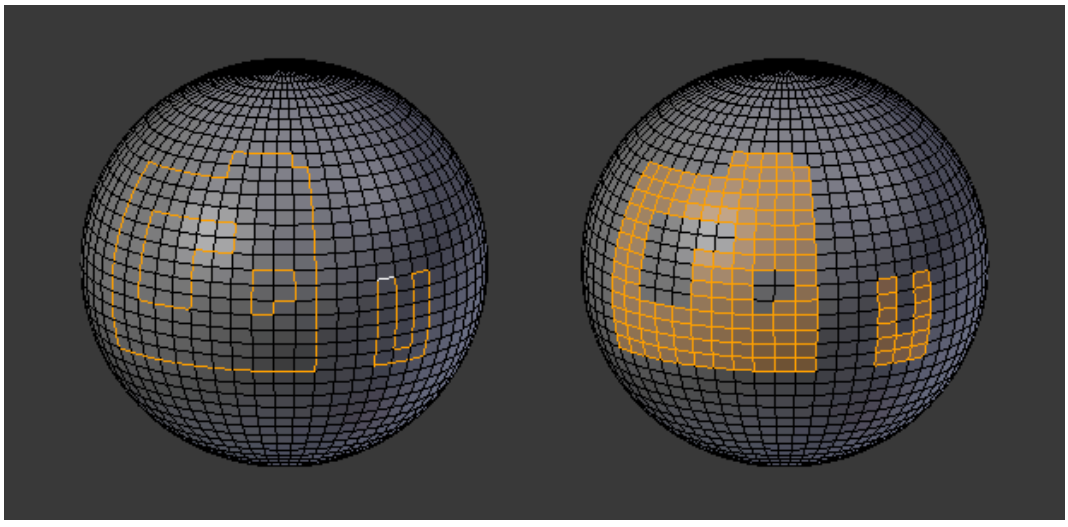
Select Loop Inner-Region selects all faces that are inside a closed loop of edges. While it is possible to use this operator in *Vertex* and *Face* selection modes, results may be unexpected. Note that if the selected loop of edges is not closed, then all connected edges on the mesh will be considered inside the loop.



Loop to Region.



This tool handles multiple loops fine, as you can see.



This tool handles “holes” just fine as well.

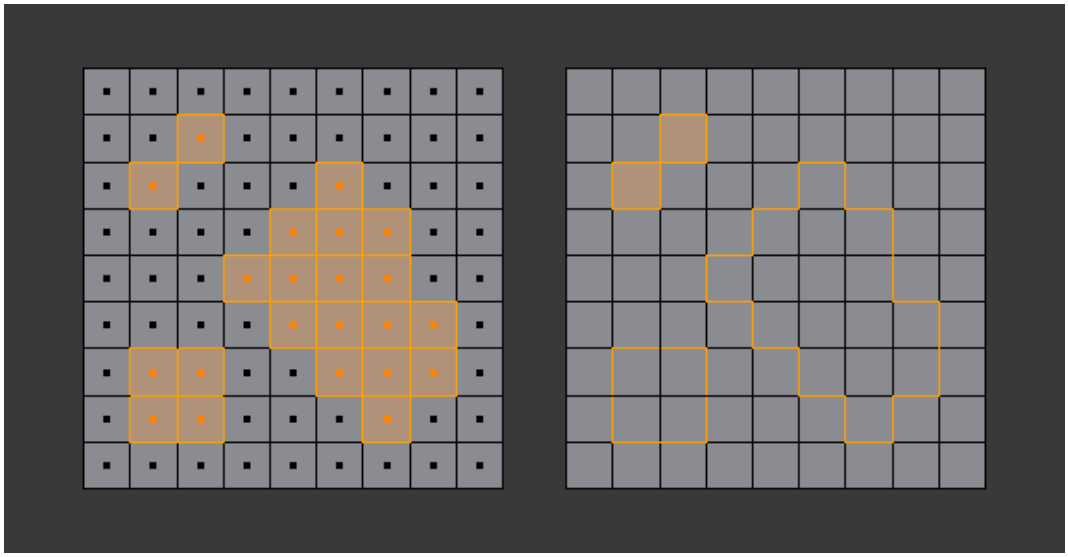
Select Boundary Loop

Reference

Mode::	Edit Mode (Edge select mode)
Menu::	*Select* ▸ *Select Loops* ▸ *Select Boundary Loop*

Select Boundary Loop does the opposite of *Select Loop Inner-Region*, based on all regions currently selected, it selects only the edges at the border (contour) of these islands. It can operate in any select mode, but when in *Face* mode it will switch to *Edge* select mode after running.

All this is much more simple to illustrate with examples:



Select Boundary Loop does the opposite and forces into Edge Select Mode.

Select Sharp Edges

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select Sharp Edges*

Selects edges whose faces intersect at an acute angle. This can be useful to find hard edges in a mesh.

Sharpness
: Determines how sharp of an angle at an edge needs to be in order for it to be selected.

Side of Active

Reference

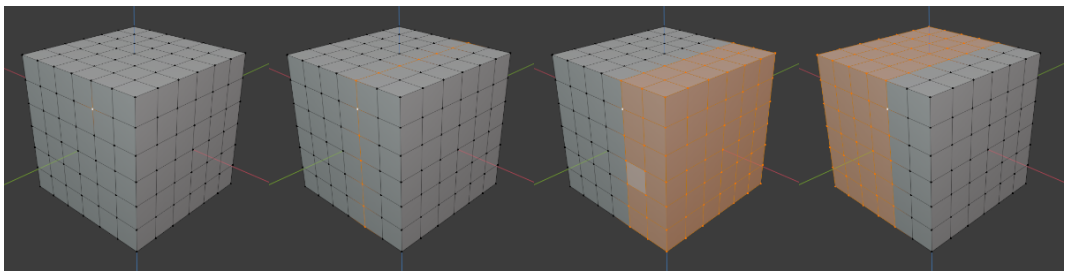
Mode:: Edit Mode
Menu:: *Select ▸ Side of Active*

With an active vertex, this tool will select all vertices in a specified direction. It is similarly to the *Loop Inner-Region* tool in that it will fill select faces within its perimeters, however, it is determined by direction and not by a closed loop.

Axis Mode
: Determines the behavior of the selection. More information on this can be found in Transform Orientations.

Axis Sign
: Positive/Negative Axis
: Depending on which *Axis* is chosen, the selection will encompass the positive or negative axis starting from the active vertex outward.

: Aligned Axis
: Where *Positive and Negative Axis* select all vertices in a given direction, *Aligned Axis* will only select the vertices that are in-line with the active vertex.



While following along the X axis: (from left to right) active vertex, Aligned, Positive, and Negative.

Axis

Chooses the direction of the selection.

Threshold

The amount of influence the selection has outside the original perimeters. The higher the *Threshold* the more vertices will be selected.

By Attribute

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *By Attribute*

Selects vertices, edges, or faces based on the Active Attribute.

Usage

Note

- The active attribute must have a boolean type.
- The active attribute must be on the vertex, edge, or face domain.

1. Select the desired attribute from the Attribute List.
2. Execute the *By Attribute* operator.

Editing

- Introduction
  - Accessing Mesh Operators
- Mesh Operators
  - Transformation
  - Mirror
  - Duplicate
  - Extrude
  - Merge
  - Split
  - Separate
  - Bisect
  - Knife Project
  - Knife Topology Tool
  - Convex Hull
  - Symmetrize
  - Snap to Symmetry
  - Normals
  - Shading
  - Set Attribute
  - Sort Elements
  - Clean Up
  - Deleting & Dissolving
- Vertex Operators
  - Extrude Vertices
  - Extrude to Cursor or Add
  - Bevel Vertices
  - New Edge/Face from Vertices
  - Connect Vertex Path
  - Connect Vertex Pairs
  - Rip Vertices
  - Rip Vertices and Fill
  - Rip Vertices and Extend
  - Slide Vertices

    - Smooth Vertices
    - Laplacian Smooth
    - Blend from Shape
    - Propagate to Shapes
    - Vertex Groups
    - Hooks
    - Make Vertex Parent
- Edge Operators
    - Extrude Edges
    - Bevel Edges
    - Bridge Edge Loops
    - Screw
    - Subdivide
    - Subdivide Edge-Ring
    - Un-Subdivide
    - Rotate Edge
    - Edge Slide
    - Offset Edge Slide
    - Loop Cut and Slide
    - Edge Data
- Face Operators
    - Extrude Faces
    - Extrude Faces Along Normals
    - Extrude Individual Faces
    - Inset Faces
    - Poke Faces
    - Triangulate Faces
    - Triangles to Quads
    - Solidify Faces
    - Wireframe
    - Fill
    - Grid Fill
    - Beautify Faces
    - Intersect (Knife)
    - Intersect (Boolean)
    - Weld Edges into Faces
    - Shade Smooth & Flat

    - Face Data
- UV Operators
    - Unwrap
    - Smart UV Project
    - Lightmap Pack
    - Follow Active Quads
    - Cube Projection
    - Cylinder Projection
    - Sphere Projection
    - Project from View
    - Project from View (Bounds)
    - Reset

Introduction

Blender provides a variety of operators for editing meshes. These operators are used to add, duplicate, move and delete elements.

These are available through the Menus in the 3D Viewport header, and context menus in the 3D Viewport, as well as individual shortcut keys.

Note

All the "transform precision/snap" keys `Ctrl` and/or `Shift` also work for all these advanced operations, but most of them do not have axis locking possibilities, and some of them do not take into account the pivot point and/or transform orientation either.

These transform operators are available in the *Transform* section of the *Mesh* menu in the header. Note that, some of these can also be used on other editable objects, like curves, surfaces, and lattices.

Accessing Mesh Operators

The mesh editing operations are found in various places, and available through shortcuts as well.

Menus

These menus are located in the header. Some of the menus can be accessed with shortcuts:

- `Ctrl-F` brings up the Face operators menu
- `Ctrl-E` brings up the Edge operators menu
- `Ctrl-V` brings up the Vertex operators menu

Mesh Operators

- Transformation
  - Move, Rotate, Scale
  - To Sphere
  - Shear
  - Bend
  - Push/Pull
  - Warp
  - Randomize
  - Shrink/Fatten
  - Skin Resize
- Mirror
  - Axis of Symmetry
  - Pivot Point
  - Transformation Orientations
- Duplicate
- Extrude
  - Extrude Faces
  - Extrude Faces Along Normals
  - Extrude Individual Faces
  - Extrude Manifold
  - Extrude Edges
  - Extrude Vertices
  - Extrude Repeat
  - Spin
- Merge
  - By Distance
- Split
  - Selection
  - Faces by Edges
  - Faces & Edges by Vertices
- Separate
- Bisect
  - Controls

    - Examples
- Knife Project
    - Options
    - Examples
    - Known Limitations
- Knife Topology Tool
    - Usage
    - Tool Settings
    - Controls
    - Known Limitations
- Convex Hull
- Symmetrize
- Snap to Symmetry
- Normals
    - Flip
    - Recalculate
    - Set from Faces
    - Rotate
    - Point to Target
    - Merge
    - Split
    - Average
    - Copy Vectors
    - Paste Vectors
    - Smooth Vectors
    - Reset Vectors
    - Select by Face Strength
    - Set Face Strength
- Shading
- Set Attribute
- Sort Elements
- Clean Up
    - Decimate Geometry
    - Fill Holes
    - Make Planar Faces
    - Split Non-Planar Faces
    - Split Concave Faces

    - Delete Loose
    - Degenerate Dissolve
    - Merge by Distance
- Deleting & Dissolving
    - Delete
    - Dissolve
    - Limited Dissolve
    - Collapse Edges & Faces
    - Edge Loops

Transformation

- Move, Rotate, Scale
- To Sphere
- Shear
- Bend
- Push/Pull
- Warp
- Randomize
- Shrink/Fatten
- Skin Resize

Move, Rotate, Scale

Reference

Mode::	Edit Mode
Tool::	*Toolbar ▸ Move, Rotate, Scale*
Menu::	*Mesh ▸ Transform ▸ Move, Rotate, Scale*
Shortcut::	`G`, `R`, `S`

Once you have a selection of one or more elements, you can move `G`, rotate `R` or scale `S` them, like many other things in Blender, as described in the Manipulation in 3D Space section. To move, rotate and scale selected components, either use the *Move*, *Rotate*, and *Scale* buttons, the transform gizmos, or the shortcuts: `G`, `R`, and `S` respectively.

After moving a selection, the options in the Adjust Last Operation panel allow you to fine-tune your changes, limit the effect to certain axes, turn Proportional Editing on and off, etc. Of course, when you move an element of a given type (e.g. an edge), you also modify the implicitly related elements of other kinds (e.g. vertices and faces).

Pressing `G` twice enters either *Edge Slide* or *Vertex Slide* tool depending on the selection. You also have in *Edit Mode* an extra option when using these basic manipulations: the Proportional Editing.

Transform Panel

Reference

Mode::	Edit Mode
Panel::	*Sidebar region ▸ Transform*

When nothing is selected, the panel is empty. When more than one vertex is selected, the median values is edited and “Median” is added in front of the

labels.

Vertex
: The first controls (X, Y, Z) show the coordinates of the selected vertex or the median point.

Space
: The Space radio buttons let you choose if those coordinates are relative to the object origin (local) or the global origin (global).

  Global, Local

Vertex Data

Bevel Weight
: This vertex property, a value between (0.0 to 1.0), is used by the Bevel Modifier to control the bevel intensity of the vertices, when the *Only Vertices* option is active.

Crease
: This vertex property, a value between (0.0 to 1.0), is used by the Subdivision Surface Modifier to control the sharpness of the vertices in the subdivided mesh.

Edge Data

When an edge is selected, the following options are available. More buttons appear:

Bevel Weight
: This edge property, a value between (0.0 to 1.0), is used by the Bevel Modifier to control the bevel intensity of the edges.

  This property can also be set using the Edge Bevel Weight operator.

Crease

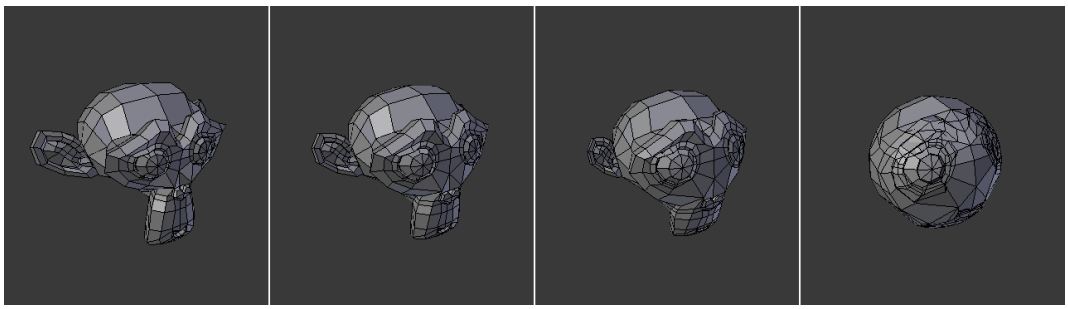
This edge property, a value between (0.0 to 1.0), is used by the Subdivision Surface Modifier to control the sharpness of the edges in the subdivided mesh.

To Sphere

Reference

Mode::	Object and Edit Modes
Menu::	*Mesh* ▸ *Transform* ▸ *To Sphere*
Shortcut::	`Shift-Alt-S`

The *To Sphere* transformation will give the selection spherical qualities. The Fig. Monkey with increasing sphericity. below shows the results of applying the *To Sphere* transformation to the monkey mesh.



Monkey with increasing sphericity.

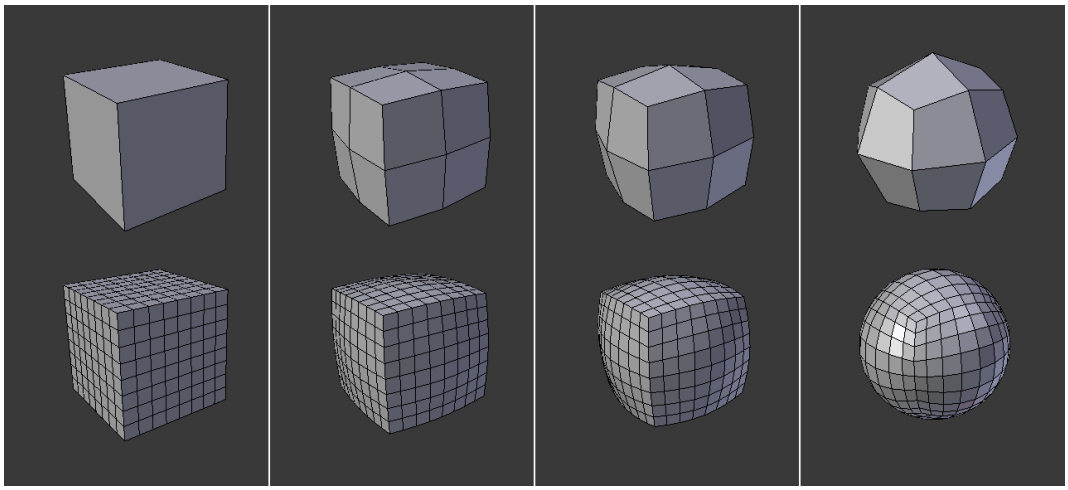
The sequence above shows a monkey mesh with a 0, 0.25 (25%), 0.5 (50%) and 1 (100%) To Sphere transform applied.

Usage



To Sphere Factor.

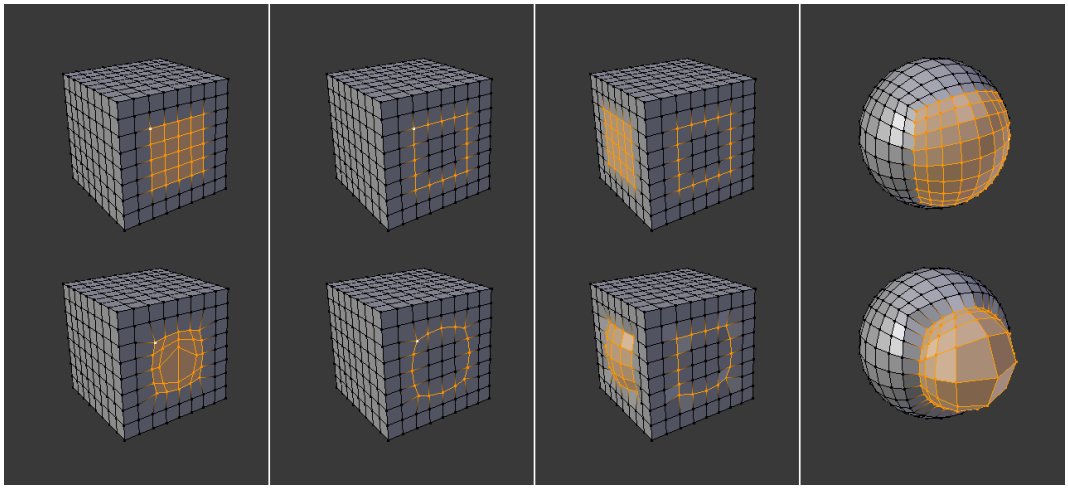
As can be seen in the below image, the result will be smoother and more spherical when there are more mesh elements available to work with.



To Sphere applied to cubes with different subdivision levels.

In this image sequence, To Sphere was applied to the entire cube at levels of 0, 0.25 (25%), 0.5 (50%) and 1 (100%) respectively.

The *To Sphere* transform will generate different results depending on the number and arrangement of elements that were selected (as shown by the below image).



To Sphere applied to different selections.

Shear

Reference

Mode::	Object and Edit Mode
Tool::	*Toolbar* ▸ *Shear*
Menu::	*Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Shear*
Shortcut::	`Shift-Ctrl-Alt-S`

Shearing is a form of movement where parallel surfaces move past one another. During this transform, movement of the selected elements will occur along the horizontal axis of the current view. The axis location will be defined by the Pivot Point. Everything that is "above" this axis will move (shear) in the same direction as your mouse pointer (but always parallel to the horizontal axis). Everything that is "below" the horizontal axis will move in the opposite direction.



Shear Offset Factor.

Tool Settings

Offset
: How far items are shifted from their original location.

Axis
: Defines one axis of the imaginary shearing plane.

Axis Orthographic
: Defines the other axis of the imaginary shearing plane.

Orientation
: See Transform Orientations.

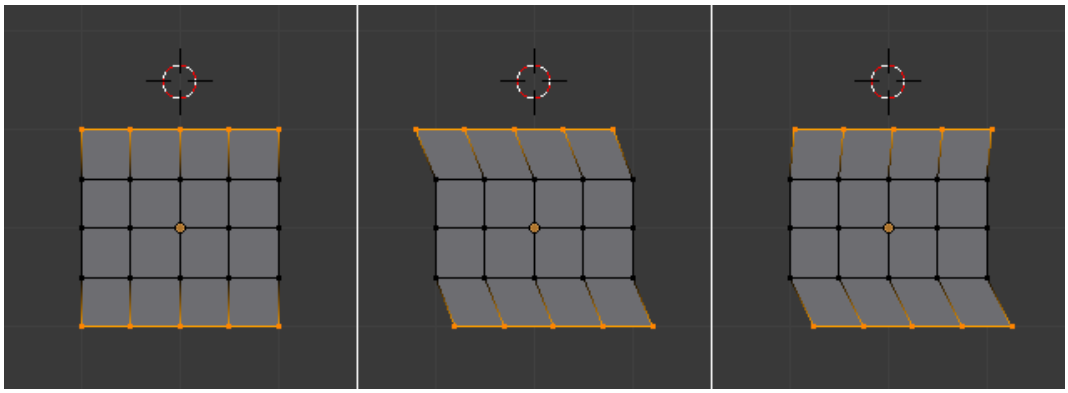
Proportional Editing
: See Proportional Editing.

Warning

The *Axis* and *Axis Orthographic* cannot be the same axis, else the imaginary plane is dimensionless and the objects will disappear into a point.

Usage

See below for the result of using *Shear* on a number of different elements.



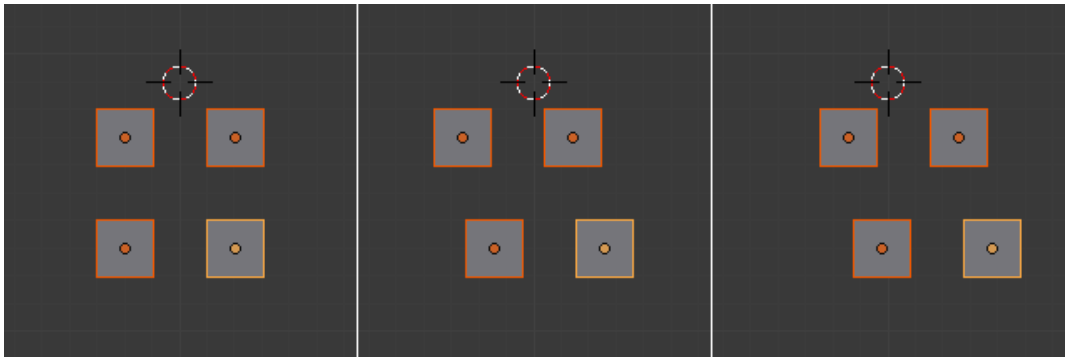
The effects of a Shear transform with different Pivot Points.

The three frames of the image above show the effects of shearing on the selected vertices when the pivot point is altered. In middle frame, the Pivot Point is set to *Median Point* and the mouse was moved to the left during the transform. In right frame, the *Pivot Point* is set to the 3D cursor which is located above the mesh. When the mouse is moved to the left during a *Shear* transform, the selected vertices are moved to the right as they are below the horizontal axis.

Tip

Shear Transform Magnitude

The magnitude of the *Shear* transform applied to the selected elements is directly proportional to the distance from the horizontal axis. i.e. the further it is away from the axis, the greater the movement.



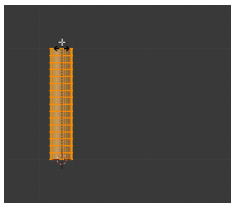
The effects of a Shear transform on objects with different Pivot Points.

Bend

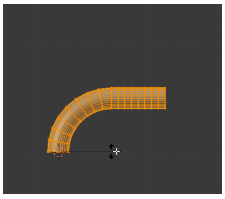
Reference

Mode::	Object and Edit Modes
Menu::	*Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Bend*
Shortcut::	`Shift-W`

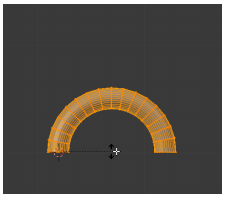
Bend Transform with Clamp on and off.



Before.



Clamp on.



Clamp off.

This tool rotates a line of selected elements forming an arc between the mouse cursor and the 3D cursor.

Usage

The *Bend* tool can be used in any case where you might want to bend a shape in two with a gradual transition between both sides.

This may take a little getting used to, the basics are listed below controls are noted here:

- The initial position of the cursors define the axis to bend on.
- The distance of the mouse cursor to the 3D cursor controls how sharp the bend will be.
- The relative angle of the mouse cursor to the initial axis defines the bend angle.

If this seems overly complicated, it's probably best to try the tool where it becomes quickly apparent how the tool reacts to your input.

Bend Angle
: The amount of rotation.

Radius
: The sharpness of the bend.

Clamp
: Normally the arc turns through a clamped rotation angle with the selected elements extended along a tangent line beyond that (see above left). When the clamp is deactivated, the arc continues around aligning the selected elements into a circle (right).

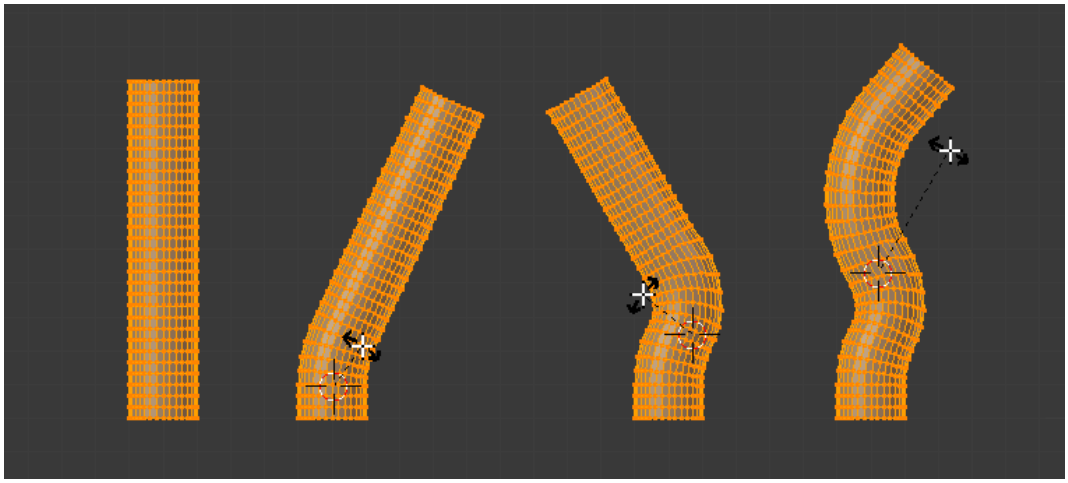
  When off `Alt` all selected elements follow a circle, even when outside the segment between the 3D cursor and the mouse.

Note

Unlike most other transform modes, *Bend* is not effected by *Pivot Point* or *Transform Orientation*, always using the View Plane instead.

Hint

You can turn the bend angle through multiple rotations potentially forming a spiral shape.



Bend Transform example.

Known Limitations

Adjust Last Operation Unsupported

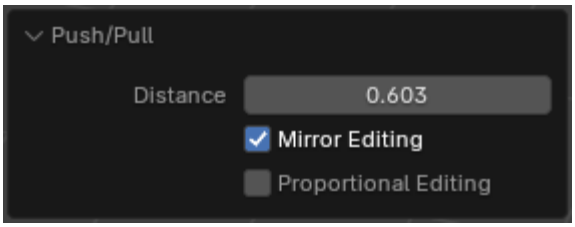
Since the bend tool relies on cursor input, it does not support adjusting the last bend operation.

Push/Pull

Reference

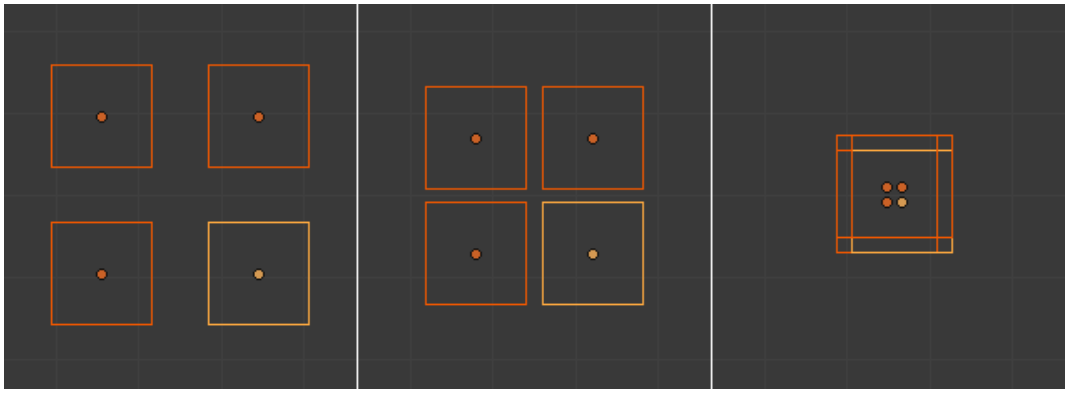
Mode::	Object and Edit Modes
Tool::	*Toolbar* ‣ *Shrink/Fatten* ‣ *Push/Pull*
Menu::	*Object/Mesh* ‣ *Transform* ‣ *Push/Pull*

Moves the selected elements closer to (Push) or further away from (Pull) the pivot point, all by the same distance. You can control this distance by moving the mouse up or down, typing a number, or using the slider in the Adjust Last Operation panel.

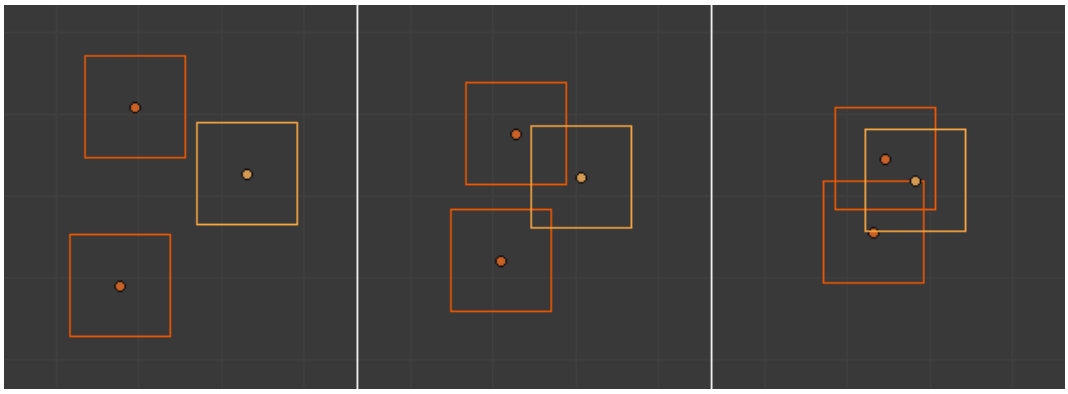


Push/Pull distance.

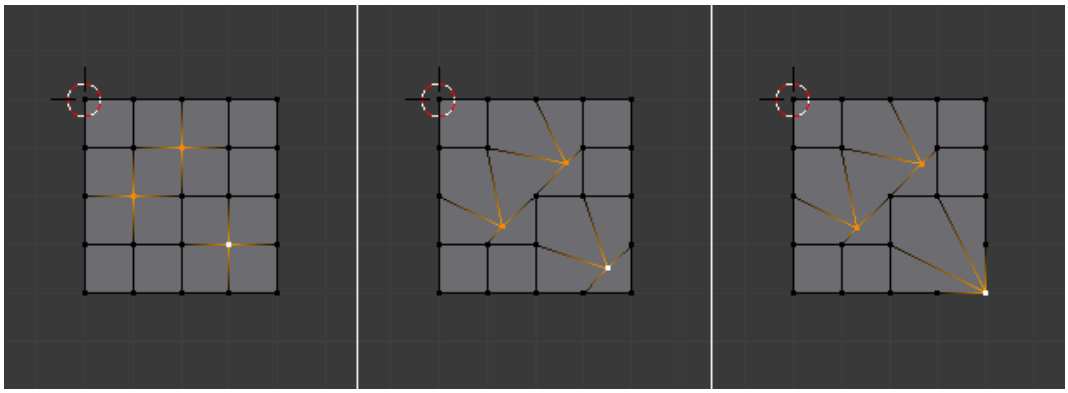
Examples



Equidistant objects being pushed together.



Random objects being pushed together.



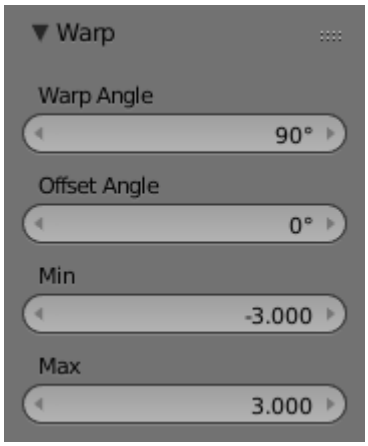
Push (middle) vertices around the 3D cursor compared to Scale (right).

Warp

Reference

Mode:: Edit Modes
Menu:: *Object/Mesh/Curve/Surface* ▸ *Transform* ▸ *Warp*

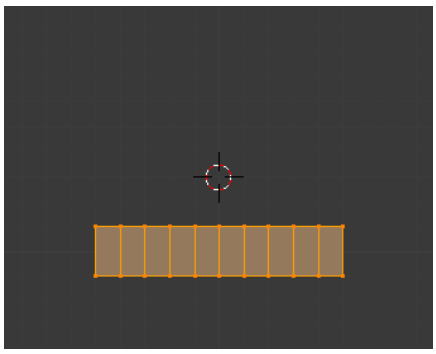
The *Warp* transformation takes selected elements and warps them around the 3D cursor by a certain angle. Note that this transformation is always dependent on the location of the 3D cursor. The Pivot Point is not taken into account. The results of the *Warp* transformation are also view dependent.

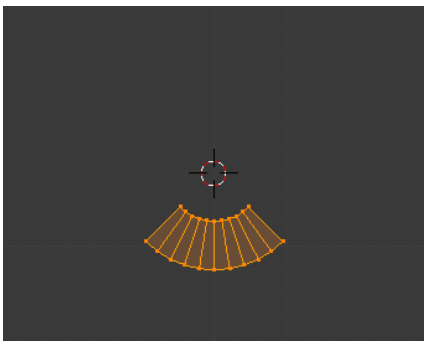


Warp options.

Usage

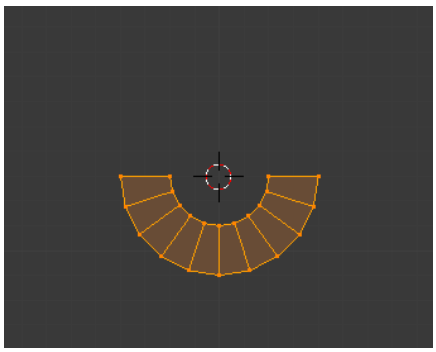
In this example, a plane is warped around the 3D cursor by the indicated number of degrees.

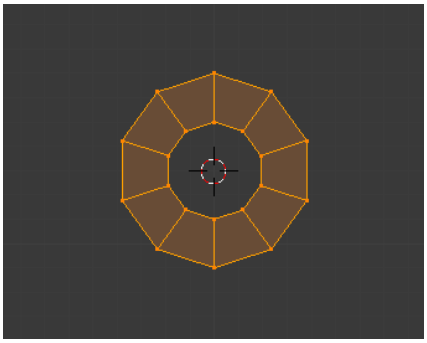




Before.

Warp Angle 90.





Warp Angle 180.

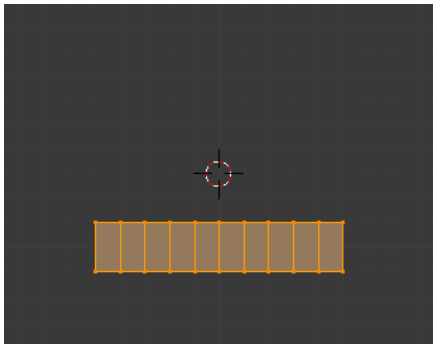
Warp Angle 360.

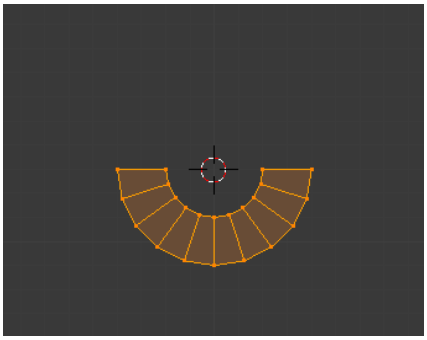
Cursor Position & View

The location of the 3D cursor can be used to alter the results of the *Warp* transformation. As can be seen from the example in this section, the *Warp* radius is dependent on the distance of the cursor from the selected elements. The greater the distance, the greater the radius.

The result of the *Warp* transform is also influenced by your current view. The example in this section shows the results of a 180 degree *Warp* transform applied to the same Plane mesh when in different views.

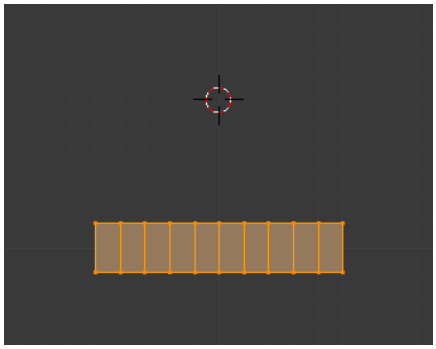
This image shows how the Warp transform is influenced by the location of the cursor.



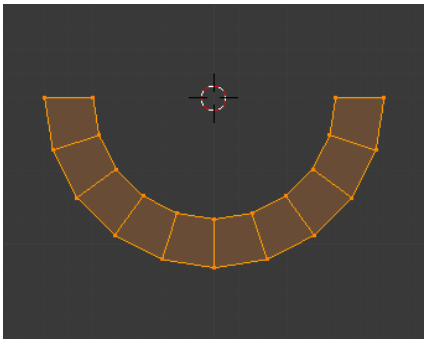


Warp Angle 180.

Before.

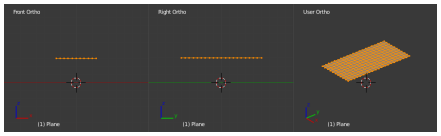


Before.

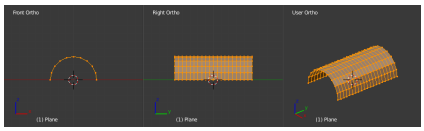


Warp Angle 180.

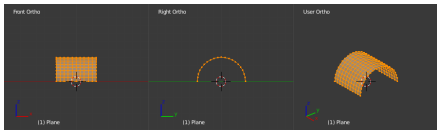
This image shows the influence of the current view.



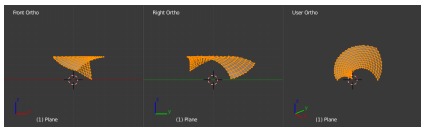
Before.



Warp Angle 180 in XZ view.



Warp Angle 180 in YZ view.



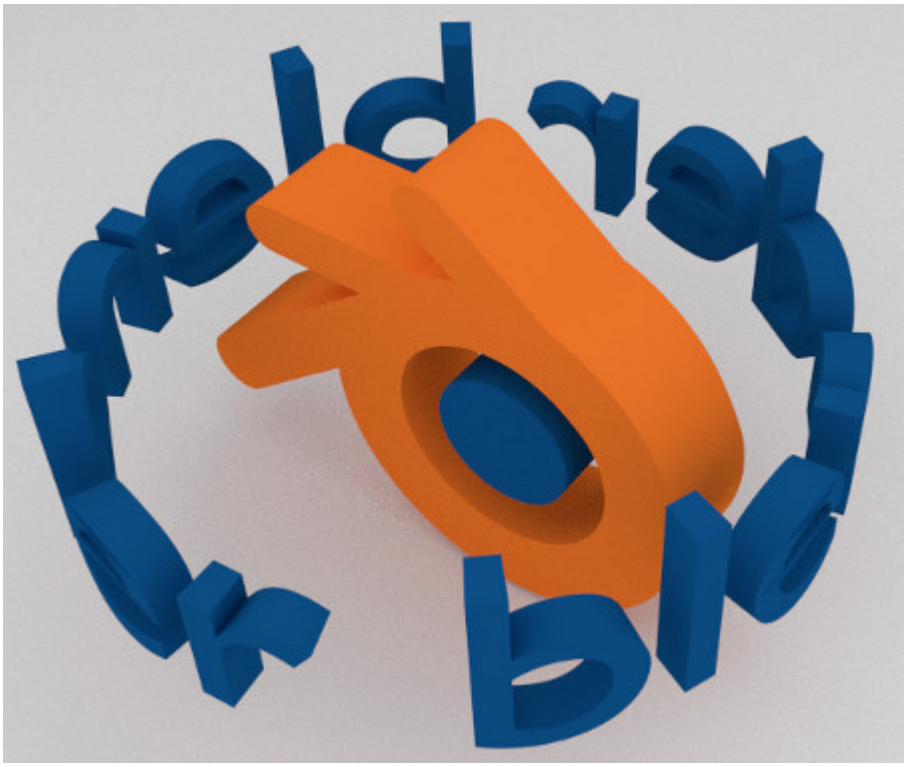
Warp Angle 180 in User view.

Note

Warping text

If you want to warp text, you will need to convert it from a text object to mesh using Convert.

Example



Text wrapped around logo.

This was made by creating the Blender logo and text as separate objects.
The text was converted to a mesh and then warped around the Blender logo.

Randomize

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Transform* ▸ *Randomize*

The *Randomize* tool in Edit Mode allows you to displace the vertices of a mesh along their normal.

Amount
: Distance of the displacement.

Uniform
: Adds a random offset of the amount.

Normal
: Adds a random offset to the displacement normal.

Random Seed
: The random seed is an offset to the random transformation. A different seed will produce a new result.

See also

Object Mode Randomize Transform

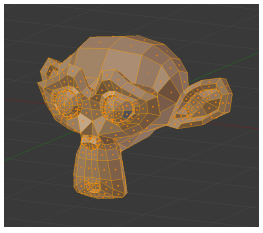
Shrink/Fatten

Reference

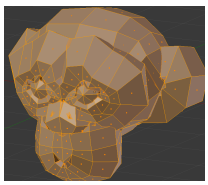
Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Shrink/Fatten*
Menu::	*Mesh* ▸ *Transform* ▸ *Shrink/Fatten*
Shortcut::	`Alt-S`

Moves the selected vertices “inwards” or “outwards” along their normal, all by the same distance. You can control this distance by moving the mouse up or down, typing a number, or using the slider in the Adjust Last Operation panel.

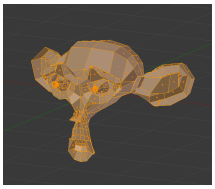
Even Thickness `S`, `Alt`
: Applies a larger offset to vertices that are part of a sharp corner, for a more uniform result. You can toggle this option by pressing `S`, holding `Alt`, or clicking the checkbox in the Adjust Last Operation panel.



Mesh before shrink/fatten.



Inflated using a positive value.



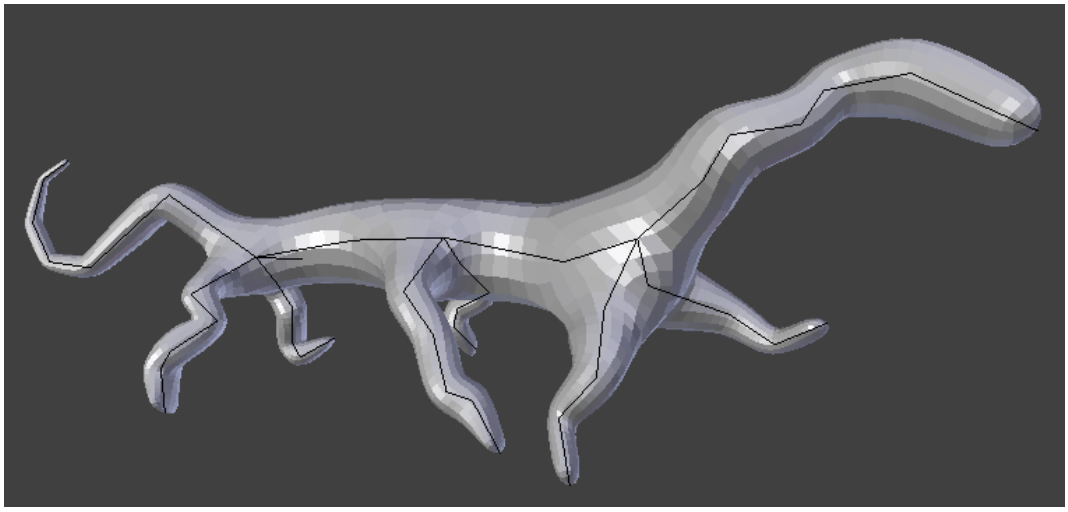
Shrunk using a negative value.

Skin Resize

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Transform* ▸ *Skin Resize*
Shortcut::	`Ctrl-A`

This tool is used to set a skin radius per vertex when using a Skin Modifier. Non-uniform scaling of the X and Y axes is accessible by the usual axis locking with `X` or `Y`. The radius can also be adjusted in the *Transform* panel of the Sidebar, in the 3D Viewport. The mean radius of all vertices can be previewed in the 3D Viewport as a dashed circle around a vertex.



Simple creature, made with only the Skin and Subdivision Surface modifiers.

Mirror

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Mirror*
Shortcut::	`Ctrl-M`

The Mirror tool mirrors a selection across a selected axis.

The Mirror tool in *Edit Mode* is similar to Mirroring in Object Mode. It is exactly equivalent to scaling vertices by -1 around one chosen pivot point and in the direction of one chosen axis, only it is faster/handier.

After this tool becomes active, select an axis to mirror the selection by pressing `X`, `Y`, or `Z`.

You can also interactively mirror the geometry by holding the `MMB` and dragging in the desired mirror direction.

Axis of Symmetry

For each transformation orientation, you can choose one of its axes along which the mirroring will occur.

As you can see, the possibilities are infinite and the freedom complete: You can position the pivot point at any location around which we want the mirroring to occur, choose one transformation orientation and then one axis on it.

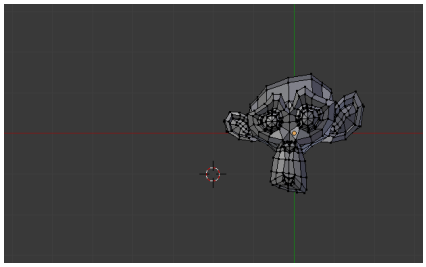
Pivot Point

Pivot points must be set first. Pivot points will become the center of symmetry. If the widget is turned on it will always show where the pivot

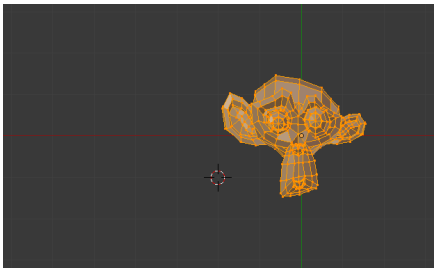
point is.

In Fig. Mirror around the individual origins. the pivot point default to median point of the selection of vertices in *Edit Mode*. This is a special case of the *Edit Mode* as explained on the pivot point page.

Mirror around the individual origins.



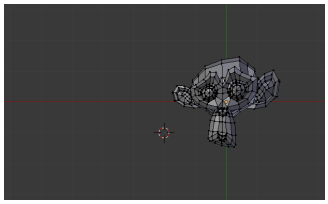
Mesh before mirror.



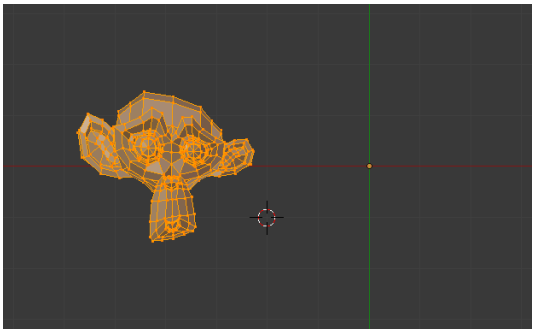
Mesh after mirrored along X axis.

In Fig. Mirror around the 3D Cursor. the pivot point is the *3D Cursor*, the transformation orientation is *Local*, a.k.a. the object space, and the axis of transformation is X.

Mirror around the 3D Cursor.



Mesh before mirror.



Mesh after mirrored along X axis using the 3D cursor as a pivot point.

Transformation Orientations

Transformation Orientations are found on the 3D Viewport header, next to the *Widget* buttons. They decide which coordinate system will rule the mirroring.

Duplicate

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Duplicate*
Shortcut::	`Shift-D`

This tool simply duplicates the selected elements, without creating any connections with the rest of the mesh (unlike extrude, for example), and places the duplicate at the location of the original. Once the duplication is done, only the *new* duplicated elements are selected, and you are automatically placed in move mode, so you can move your copy elsewhere…

In the *Toolbar* are settings for *Vector* offset, *Proportional Editing*, *Duplication Mode*, and *Axis Constraints*.

Note that duplicated elements belong to the same vertex groups as the "original" ones. The same goes for the material indices, the edge's *Sharp* and *Seam* marks, and probably for the other vertex/edge/face properties…

Extrude

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Extrude*
Shortcut::	`Alt-E`

This operators shown in this menu are dependent of what part of a mesh is currently selected. Many of the operators are also available in the Vertex, Edge, abd Face menus.

Extrude Faces

Available when a Face is selected.

See Extrude Faces.

Extrude Faces Along Normals

Available when a Face is selected.

See Extrude Faces Along Normals.

Extrude Individual Faces

Available when a Face is selected.

See Extrude Individual Faces.

Extrude Manifold

Available when a Face is selected.

See Extrude Manifold.

Extrude Edges

Available when a Edge is selected.

See Extrude Edges.

Extrude Vertices

Available when a Vertex is selected.

See Extrude Vertices.

Extrude Repeat

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Extrude ▸ Extrude Repeat*

This tool behaves similar to the Array Modifier, by extruding the selection along the Z axis of the view. If the selection is not Manifold it's extruded the specified number of times.

Offset X, Y, Z
: Distance between the instances.

Steps
: Number of instances.

Scale Offset
: Multiplication factor to increase or decrease the offset.

Spin

See Spin.

Merge

Reference

Mode::	Edit Mode
Menu::	*Mesh* ‣ *Merge*, *Context Menu* ‣ *Merge*
Shortcut::	`M`

This tool allows you to merge all selected vertices to a unique one, dissolving all others. You can choose the location of the remaining vertex in the menu this tool pops up before executing:

At Center
: It will place the remaining vertex at the center of the selection. Available in all select modes.

At Cursor
: It will place the remaining vertex at the 3D Cursor. Available in all select modes.

Collapse
: Every island of selected vertices (connected by selected edges) will merge on its own median center, leaving one vertex per island.

At First
: It will place the remaining vertex at the location of the first one selected. Only available in *Vertex* select mode.

At Last
: It will place the remaining vertex at the location of the last one selected (the active one). Only available in *Vertex* select mode.

Merging vertices of course also deletes some edges and faces. But Blender will do everything it can to preserve edges and faces only partly involved in the reunion.

Note

At First and *At Last* depend on that the selection order is saved: the order is lost, for instance, after changing selection mode.

UVs
: If *UVs* is ticked in the Adjust Last Operation panel, the UV mapping coordinates, if existing, will be corrected to avoid image distortion.

By Distance

Todo.

Split

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Split*
Shortcut::	`Alt-M`

Selection

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Split ▸ Selection*
Shortcut::	`Y`

Splits (disconnects) the selection from the rest of the mesh. The border edge to any non-selected elements are duplicated.

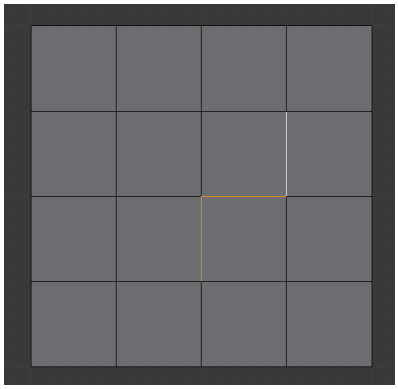
Note that the “copy” is left exactly at the same position as the original, so you must move it `G` to see it clearly…

Faces by Edges

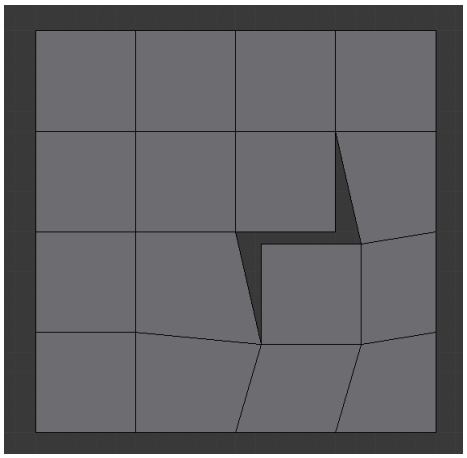
Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Split ▸ Faces by Edges*

Faces by Edges is similar to the *Rip* tool. When two or more touching interior edges, or a border edge is selected, a hole will be created, and the selected edges will be duplicated to form the border of the hole.



Selected edges.



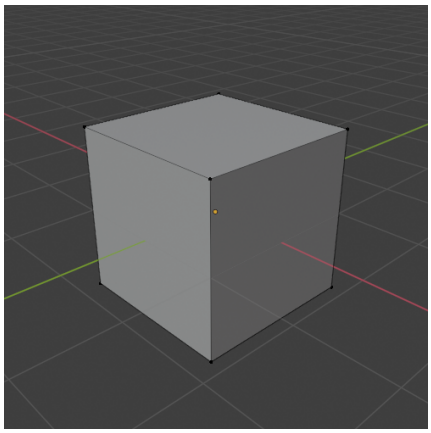
Adjacent face moved to reveal hole left by split.

Faces & Edges by Vertices

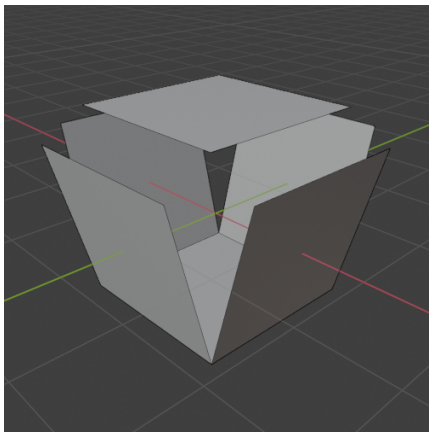
Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Split ▸ Faces & Edges by Vertices*

Faces & Edges by Vertices is similar to *Faces by Edges* except that it also splits the vertices of the adjacent connecting edges. This has the same functionality as manually ripping all faces and edges away from a vertex.



Before.



After (also moving edges away).

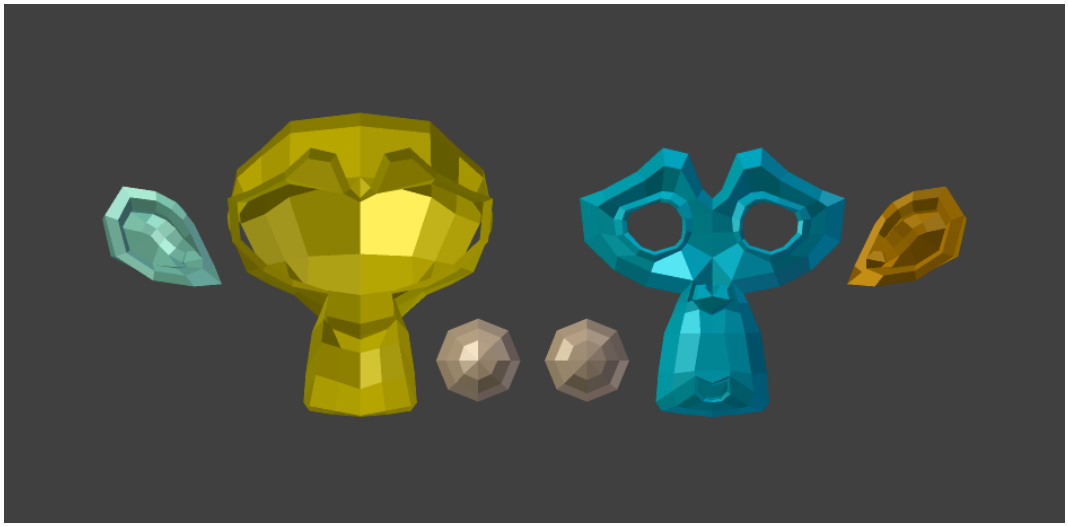
Separate

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Separate*
Shortcut::	P

At some point, you will come to a time when you need to cut parts away from a mesh to be separate.

To separate an object, the vertices (or faces) must be selected and then separated, though there are several different ways to do this.



Suzanne dissected neatly.

Selection
: Separates the selected elements.

By Material
: Separates fragments based on the materials assigned to the different faces.

By Loose Parts
: Creates one object for every independent (disconnected) fragment of the original mesh.

See also

Joining objects.

Bisect

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Knife* ▸ *Bisect*
Menu::	*Mesh* ▸ *Bisect*

The Bisect tool is a quick way to cut a mesh in two along a custom plane.

Use `LMB` click and drag to draw cut line. Once the cut is done the Adjust Last Operation panel gives a few options:

Plane Point, Plane Normal
: The plane can be numerically adjusted for precise values.

Fill
: Cuts can optionally fill in the holes created, with materials, UV maps, and Color Attributes based on the surrounding geometry.

Clear Inner, Clear Outer
: Cuts may remove geometry on one side.

Axis Threshold
: Cut along the straight plane or along the existing geometry below the distance from the plane.

Controls

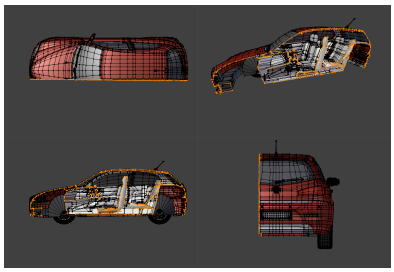
Move `Spacebar`
: Changes the location of the line.

Snap `Ctrl`
: Constrains the rotation of the line to 15 degree intervals.

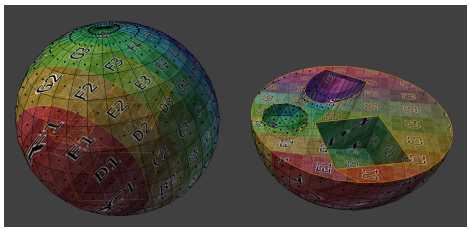
Flip F

Changes the side of the line that is the inner/outer side; this option is useful when using *Clear Inner*, *Clear Outer* and/or *Fill*.

Examples



Example of a common use of bisect.



Example of bisect with the fill option enabled.

Knife Project

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Knife Project*

Knife Project is a non-interactive tool where you can use objects to cookie-cut into one or more meshes rather than hand drawing the line. The outline of the selected objects that *are not* in Edit Mode is projected along the view axis onto the meshes that *are* in Edit Mode, and then cuts into the faces there. Afterwards, the resulting geometry inside the cut gets selected.

Note

The cutting objects must be curves or non-manifold meshes (e.g. flat shapes, loose edges). Select Non-Manifold will highlight the cutting edges of mesh objects.

Keep in mind that Knife Project works from the current view's perspective. For best results, make sure to rotate your view to exactly the position you require before using this tool. Orthographic views such as Right, Front, and Top are commonly used for this.

Hint

3D Viewport Alignment to adjust the projection axis.

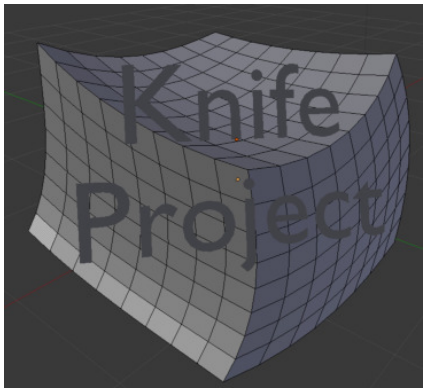
To use Knife Project, select the objects to be cut, switch to *Edit Mode*, select the cutting objects in the Outliner (`Ctrl-LMB`), and choose *Mesh ▸ Knife Project*.

If Blender switches back to *Object Mode* when selecting the cutting objects, make sure that *Edit* ▸ *Lock Object Modes* is checked in the topbar. Alternatively, if you have only one cutting object, you can select it in the viewport with `Ctrl-LMB`.

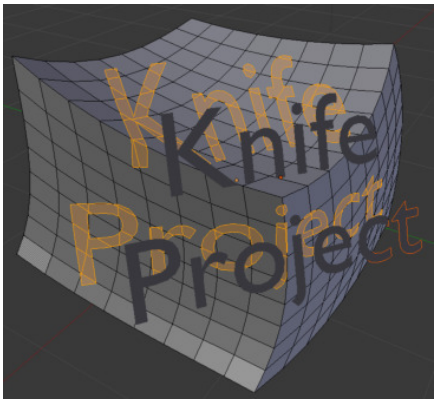
Options

Cut Through
: Projects the cut through the entire mesh, including back faces not currently visible.

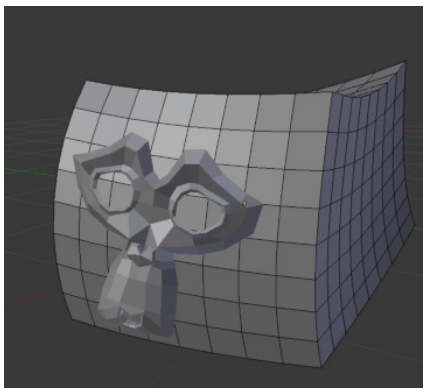
Examples

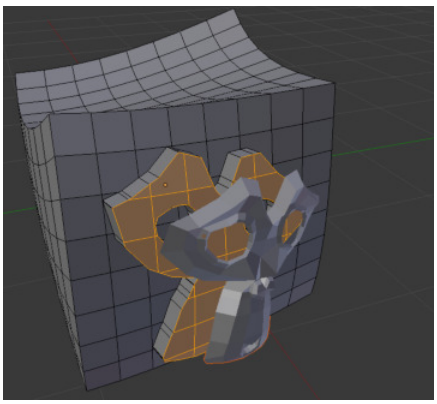


Before projecting from a text object.



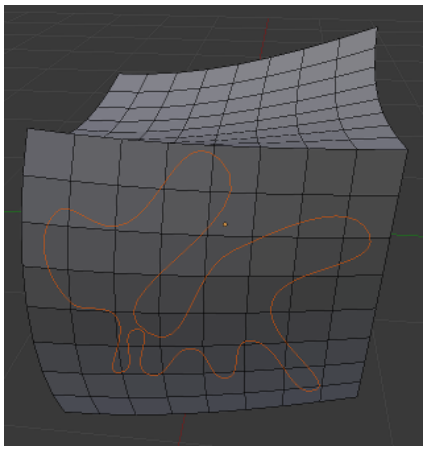
Resulting knife projection.



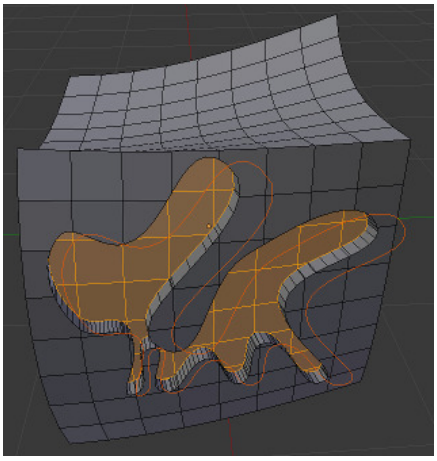


Before projecting from a mesh object.

Resulting knife projection (extruded after).



Before projecting from a 3D curve object.



Resulting knife projection (extruded after).

Known Limitations

When cutting multiple meshes in Edit Mode at once, geometry from these meshes does not occlude separate mesh objects behind them.

Knife Topology Tool

Reference

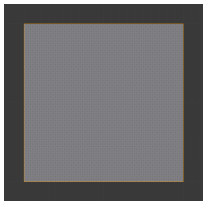
Mode::	Edit Mode
Tool::	*Toolbar ▸ Knife*
Menu::	*Mesh ▸ Knife Topology Tool*
Shortcut::	K

The Knife tool can be used to interactively subdivide (cut up) geometry by drawing lines or closed loops to create holes.

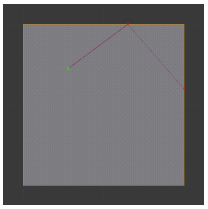
Usage

When using *Knife*, the cursor changes to an icon of a scalpel and the header changes to display options for the tool. You can draw connected straight lines by clicking LMB, marked with small green squares. Red squares are already defined cuts. Surrounding red squares mean that there is a cut already in that position, so no additional vertex will be created (besides the first one).

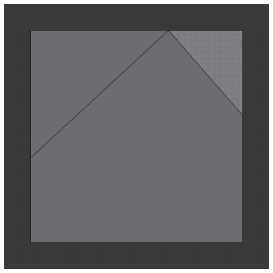
If multiple objects are selected before entering Edit Mode, then knife cuts will affect all of those objects.



Mesh before knife cut.



Knife cut active.



After confirming

knife cut.

Tool Settings

Occlude Geometry
: Only cut geometry visible on screen.

Only Selected `Shift-K`
: Only cuts through selected geometry.

X-Ray
: Show cut points on non-visible geometry too, when *Occlude Geometry* is deactivated.

Measurement
: Which visible measurements to show.

  Distance, Angle, None, Both

Angle Snapping
: Whether or not dragged lines should be constrained to particular angles, and if so, which coordinate system the angle is relative to.

  None, Screen, Relative

Angle Snapping Increment
: When angle snapping is on, the angle will be constrained to a multiple of this angle.

Controls

Confirm `Spacebar` or `Return`
: Confirms the cut. `Return` will leave selected every edge except the new edges created from the cut.

Cancel `Esc`

Cancels the cut.

Draw a Continuous Line `LMB` drag.

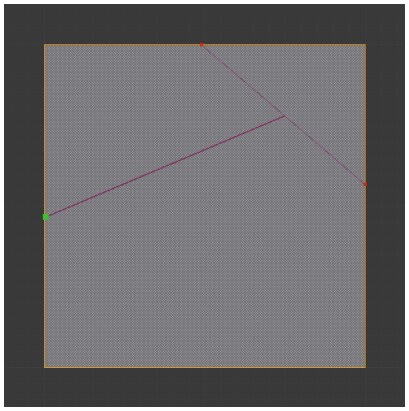
So you can draw a free-hand line over a surface, points will be created at edge intersections.

Close Loop double-click `LMB`

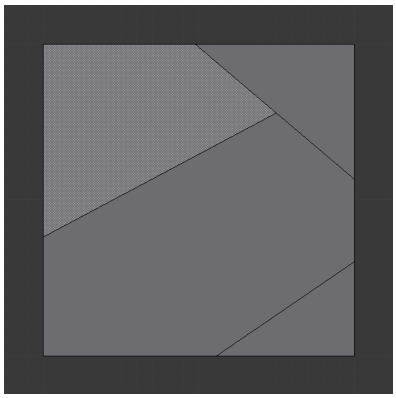
This is a quick way to close the loop you are currently cutting.

New Cut `RMB`

Begins a new cut. This allows you to define multiple distinct cut lines. If multiple cuts have been defined, they are recognized as new snapping points.



Creating multiple cuts.



Result of starting new cuts while in the tool.

Midpoint Snap `Shift`

Hold to snap the cursor to the midpoint of edges, meaning that all cuts will be performed at the exact center of each cut edge.

Ignore Snap `Ctrl`

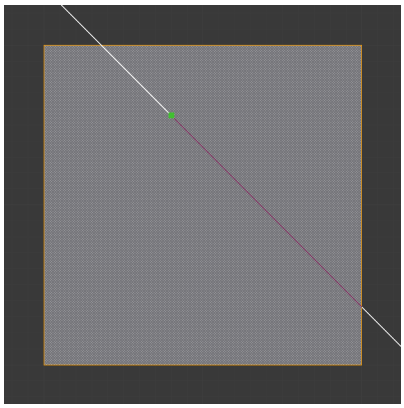
Hold to make the tool ignore snapping, unlike the default where mouse cursor snaps to near edges.

Cut Through: `C`

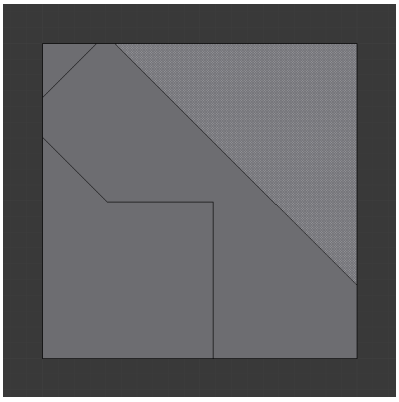
Allow the Cut tool to cut through to occluded faces, instead of only the visible ones.

Angle Constrain `A`

Constrains the cut line to certain degree increments. The increment can be specified in the Tool Settings (see above), or can be typed when angle constraining is active. The default angles are in the plane of the screen, but typing `A` again makes it relative to the last cut edge. If the last cut edge is ambiguous (because the cut was on a vertex), typing `R` cycles through the possible reference edges.



Constraining cut angle.



Result of constraining cut angle.

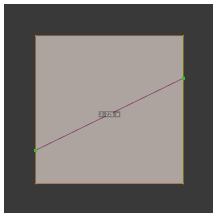
Axis Constrain `X`, `Y`, or `Z`

Constrains the cut line to one of the coordinate system axes. Initially it will be the global axis with the given name, but pressing the same key again switches to the object's local axis system. Additionally, if the scene transformation orientation is set to a custom orientation (e.g. from a face), the constraints will be in that coordinate system.

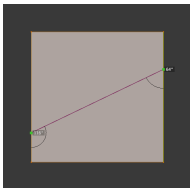
Visible Measurements `S`

Shows measurements of the cuts being made: angles with respect to a mesh edge, lengths, or both. Pressing `S` repeatedly cycles between what can be shown.

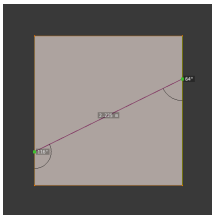
Only Distance, Only Angles, Both, None



Only Distance.



Only Angles.



Both Angles and Distance.

Undo `Ctrl-Z`

: Undoes the previous cut segment. The starting point for the next cut is adjusted accordingly. If a cut is a drag cut, the entire drag cut is undone.

X-Ray Mode `V`

: Toggles whether or not cuts to segments behind the visible geometry are shown.

Known Limitations

Duplicate Vertices

If you experience problems where duplicate vertices are being created by cuts, this is often caused by too large a near/far clipping range.

Try increasing the *Clip Start* to avoid this problem, see Depth Troubleshooting for details.

Unconnected Cuts

Cuts that begin or end in the middle of a face, will be ignored.

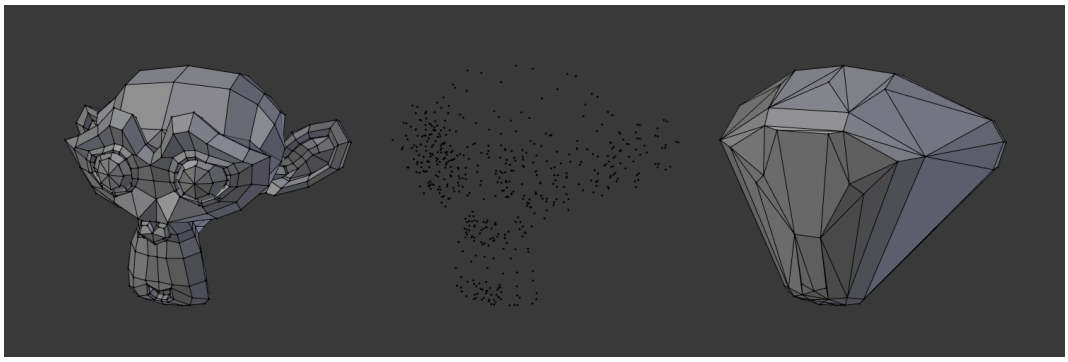
This is constrained by the kinds of geometry Blender can represent.

Convex Hull

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Convex Hull*

The Convex Hull operator takes a point cloud as input and outputs a convex hull surrounding those vertices. If the input contains edges or faces that lie on the convex hull, they can be used in the output as well. This operator can be used as a bridge tool as well.



Input mesh, point cloud, and Convex Hull result.

Delete Unused
: Removes vertices, edges, and faces that were selected, but not used as part of the hull. Note that vertices and edges that are used by other edges and faces not part of the selection will not be deleted.

Use Existing Faces
: Where possible, use existing input faces that lie on the hull. This allows the convex hull output to contain n-gons rather than triangles (or quads if the *Join Triangles* option is enabled).

Make Holes

Delete edges and faces in the hull that were part of the input too. Useful in cases like bridging to delete faces between the existing mesh and the convex hull.

Join Triangles

Joins adjacent triangles into quads. Has all the same properties as the *Tris to Quads* operator (angle limit, compare UVs, etc.).

Max Face Angle, Max Shape Angle, Compare

See Triangles to Quads.

Symmetrize

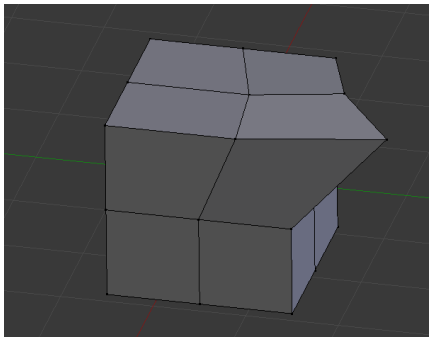
Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Symmetrize*

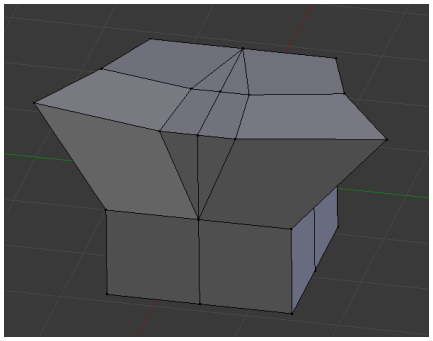
The *Symmetrize* tool is a quick way to make a mesh symmetrical. *Symmetrize* works by cutting the mesh at the pivot point of the object, and mirroring over the geometry in the specified axis, and merges the two halves together (if they are connected). Also the mesh data is copied from one side to the other: e.g. UVs, colors attributes, vertex weights.

Direction
: Specify the axis and direction of the effect. Can be any of the three axes, and either positive to negative, or negative to positive.

Threshold
: The vertices in this range will be snapped to the plane of symmetry.



Mesh before Symmetrize.



Mesh after Symmetrize.

See also

See Mirror for information on mirroring, which allows you to flip geometry across an axis.

Snap to Symmetry

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Snap to Symmetry*

The Snap to Symmetry tool lets you snap a mesh vertices to their mirrored neighbors.

Useful when dealing with meshes which are mostly symmetrical, but have vertices which have been moved enough that Blender does not detect them as mirrored (when X Mirror option is enabled for example).

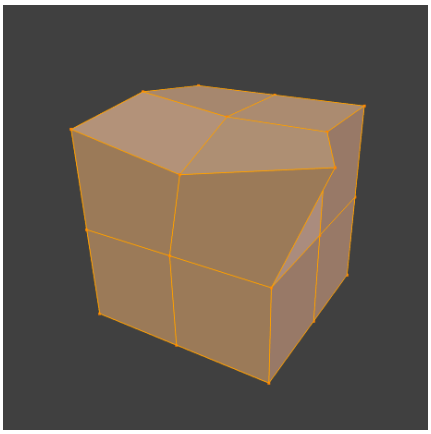
This can be caused by accident when editing without X Mirror enabled. Sometimes models imported from other applications are asymmetrical enough that mirror fails too.

Direction
: Specify the axis and direction to snap. Can be any of the three axes, and either positive to negative, or negative to positive.

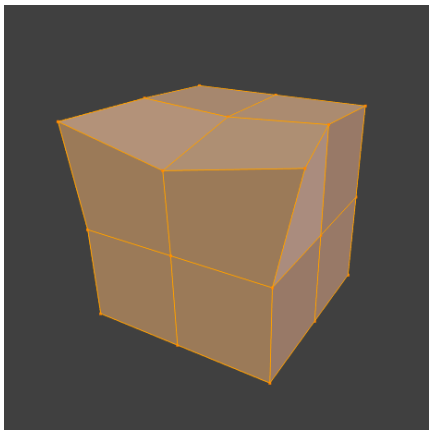
Threshold
: Specify the search radius to use when finding matching vertices.

Factor
: Support for blending mirrored locations from one side to the other (0.5 is an equal mix of both).

Center
: Snap vertices along the center axis to zero.



Before Snap to Symmetry.



After Snap to Symmetry.

Normals

See also

The Normal Edit Modifier can be used to edit normals.

The Weighted Normal Modifier can be used to affect normals by various methods, including *Face Strength* (see below).

You can also copy normals from another mesh using Mesh Data Transfer (operator or modifier).

Flip

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Normals* ▸ *Flip*

This will reverse the normals direction of all selected faces. Note that this allows you to precisely control the direction (**not** the orientation, which is always perpendicular to the face) of your normals, as only the selected faces are flipped.

Recalculate

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Normals* ▸ *Recalculate Outside* and *Mesh* ▸ *Normals* ▸ *Recalculate Inside*
Shortcut::	`Shift-N` and `Shift-Ctrl-N`

These tools will recalculate the normals of selected faces so that they point outside (respectively inside) the volume that the face belongs to. The volume does not need to be closed; inside and outside are determined by the angles with adjacent faces. This means that the face of interest must be adjacent to at least one non-coplanar other face. For example, with a *Grid* primitive, recalculating normals does not have a meaningful result.

Set from Faces

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Set from Faces*

Set the custom normals at corners to be the same as the face normal that the corner is part of.

Rotate

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Rotate*
Shortcut::	`R N`

This is an interactive tool. As you move the mouse around, the selected normals are rotated. You can also invoke the Rotate Normals tool by pressing the Rotate transform key `R`, followed by `N`.

Point to Target

Reference

Mode::	Edit Mode
Menu::	

Mesh ▸ Normals ▸ Point to Target

Shortcut:: `Alt-L`

All selected normals are set to point from their vertex to the target after confirmed by `Return` or `LMB`.

A target is set by the keys:

- The mouse cursor `M`
- The pivot `L`
- The object origin `O`
- The cursor (set at this click) `Ctrl-LMB`
- A mesh item selection (set by this click) `Ctrl-RMB`

Mode
: The tool operation can be modified; if one of the following keys has been previously pressed:

  Align `A`
  : All normals will point in the same direction: from the center of selected points to the target.

  Spherize `S`
  : Each normal will be an interpolation between its original value and the direction to the target.

  Invert `I`
  : The normal directions are reversed from what was specified above.

Reset `R`
: Will reset the custom normals back to what they were when the operation started.

Merge

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Normals ▸ Merge*

Merge all of the normals at selected vertices, making one average normal for all of the faces.

Split

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Normals ▸ Split*

Split the normals at all selected vertices so that there are separate normals for each face, pointing in the same direction as those faces.

Average

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Normals ▸ Average*

Average all of the normals in each fan of faces between sharp edges at a vertex.

Copy Vectors

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Normals ▸ Copy Vectors*

If a single normal is selected, copy it to an internal vector buffer.

Paste Vectors

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Paste Vectors*

Replace the selected normals with the one in the internal vector buffer.

Smooth Vectors

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Smooth Vectors*

Adjust the normals to bring them closer to their adjacent vertex normals.

Reset Vectors

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Reset Vectors*

Put normals back the to default calculation of the normals.

Select by Face Strength

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Normals ▸ Select by Face Strength*

Another way to affect normals is to set a *Face Strength* on the faces of the model. The Face Strength can be either *Weak*, *Medium*, or *Strong*. The idea is that the Weighted Normal Modifier can be set to pay attention to the Face Strength as follows: When combining the normals that meet at a vertex, only the faces with the strongest Face Strength will contribute to the final value.

For example, if three faces meet at a vertex and have the face weights weak, medium, and strong, then only the normal associated with the strong face will be used to set the final result.

Use the submenu to pick one of *Weak*, *Medium*, or *Strong*. Then this tool selects those faces that have the chosen face strength.

Set Face Strength

Reference

Mode::	Edit Mode
Menu::	*Mesh* ▸ *Normals* ▸ *Set Face Strength*

Use the submenu to pick one of *Weak*, *Medium*, or *Strong*. Then this tool changes the Face Strength of currently selected faces to the chosen face strength.

Shading

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Shading*

Alternatively, you can choose which faces to smooth by entering *Edit Mode*, then selecting some faces and picking *Shade Smooth* from the *Face Menu*.

When the mesh is in *Edit Mode*, only the selected faces will receive the "smoothing" attribute. You can set faces as flat (removing the "smoothing" attribute) in the same way by selecting edges and picking the *Shade Flat* from the *Face Menu*.

Set Attribute

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Set Attribute*

Sets the value of the Active Attribute for the selected element. “Active attribute” here means the attribute that is currently selected in the Attributes list.

When the operator is executed a pop-up window will display the attribute’s name and current value. From here, the value field came be adjusted to change the attribute’s value.

See also

Attribute values can be viewed in the Spreadsheet editor.

Sort Elements

Reference

Mode::	Edit Mode
Menu::	*Mesh ▸ Sort Elements…*

This tool (available from the context menu, *Vertices*, *Edges* and *Faces* menus) allows you to reorder the matching selected mesh elements, following various methods. Note that when called from the context menu, the affected element types are the same as the active select modes.

View Z Axis
: Sort along the active view's Z axis, from farthest to nearest by default (use *Reverse* if you want it the other way).

View X Axis
: Sort along the active view's X axis, from left to right by default (again, there is the *Reverse* option).

Cursor Distance
: Sort from nearest to farthest away from the 3D cursor position (*Reverse* also available).

Material
: Sort faces, and faces only, from those having the lowest material's index to those having the highest. Order of faces inside each of those "material groups" remains unchanged. Note that the *Reverse* option only reverses the order of the materials, *not* the order of the faces inside them.

Selected
: Move all selected elements to the beginning (or end, if *Reverse* enabled), without affecting their relative orders. Warning: This option will also affect **unselected** elements' indices!

Randomize

Randomizes indices of selected elements (*without* affecting those of unselected ones). The seed option allows you to get another randomization – the same seed over the same mesh or set of selected elements will always give the same result!

Reverse

Simply reverse the order of the selected elements.

Hint

Enabling the Display Indices Option

Enable the *Developer Extras* Option in *Preferences ▸ Interface ▸ Display* panel, a checkbox will appear in *Display & Shading Menu ▸ Viewport Overlay ▸ Developer ▸ Indices*.

Clean Up

These tools are to help cleanup degenerate geometry and fill in missing areas of a mesh.

Decimate Geometry

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Decimate Geometry*

The Decimate Geometry tool allows you to reduce the vertex/face count of a mesh with minimal shape changes.

Ratio
: Ratio of triangles to reduce to.

Vertex Group
: Use the active vertex group as an influence.

  Weight
  : Strength of the vertex group.

  Invert
  : Inverts the vertex group.

Symmetry
: Maintain symmetry on either the *X*, *Y*, or *Z* axis.

See also

This tool works similar to the Decimate Modifier.

Fill Holes

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Fill Holes*

This tool can take a large selection and detect the holes in the mesh, filling them in.

This is different from the face creation operator in three important respects:

1. Holes are detected, so there is no need to manually find and select the edges around the holes.
2. Holes can have a limit for the number of sides (so only quads or tris are filled in for example).
3. Mesh data is copied from surrounding geometry (UVs, Color Attributes, multi-res, all layers), since manually creating this data is very time-consuming.

Make Planar Faces

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Make Planar Faces*

The *Make Planar Faces* iteratively flattens faces. This can happen with faces over three vertices and it is a common convention that faces should be kept planar.

Factor
: Distance to move the vertices each iteration.

Iterations
: Number of times to repeat the operation.

Split Non-Planar Faces

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Split Non-Planar Faces*

This tool avoids ambiguous areas of geometry by splitting non-flat faces when they are bent beyond a given limit.

Split Concave Faces

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Split Concave Faces*

This tool can be used to convert any Concave Face to convex by splitting the concave into two or more convex faces.

Delete Loose

Reference

Mode::	Edit Mode
Menu::	*Mesh ‣ Clean up ‣ Delete Loose*

This tool removes disconnected vertices and edges (optionally faces).

Degenerate Dissolve

Reference

Mode::	Edit Mode

Menu:: *Mesh ▸ Clean up ▸ Degenerate Dissolve*

This tool collapses / removes geometry which you typically will not want.

- Edges with no length.
- Faces with no areas (faces on a point or thin faces).
- Face corners with no area.

Merge by Distance

Reference

Mode:: Edit Mode
Menu:: *Vertex ▸ Merge by Distance*, *Context Menu ▸ Merge by Distance*

Merge by Distance is a useful tool to simplify a mesh by merging the selected vertices that are closer than a specified distance to each other. An alternative way to simplify a mesh is to use the Decimate Modifier.

Merge Distance
: Sets the distance threshold for merging vertices.

Unselected
: Allows vertices in the selection to be merged with unselected vertices. When disabled, selected vertices will only be merged with other selected ones.

Deleting & Dissolving

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Delete*

These tools can be used to remove components.

Delete

Reference

Shortcut:: `X`, `Delete`

Deletes selected vertices, edges, or faces. This operation can also be limited to:

Vertices
: Delete all vertices in current selection, removing any faces or edges they are connected to.

Edges
: Deletes any edges in the current selection. Removes any faces that the edge shares with it.

Faces
: Removes any faces in current selection.

Only Edges & Faces
: Limits the operation to only selected edges and adjacent faces.

Only Faces
: Removes faces, but edges within the face selection are retained.

Dissolve

Dissolve operations are also accessed from the delete menu. Dissolve will remove the geometry and fill in the surrounding geometry. Instead of removing the geometry, which may leave holes that you have to fill in again.

Removes selected geometry, but without creating holes, effectively turning the selection into a single n-gon. Dissolve works slightly different based on if you have edges, faces or vertices selected. You can add detail where you need it, or quickly remove it where you do not.

Dissolve Vertices

Reference

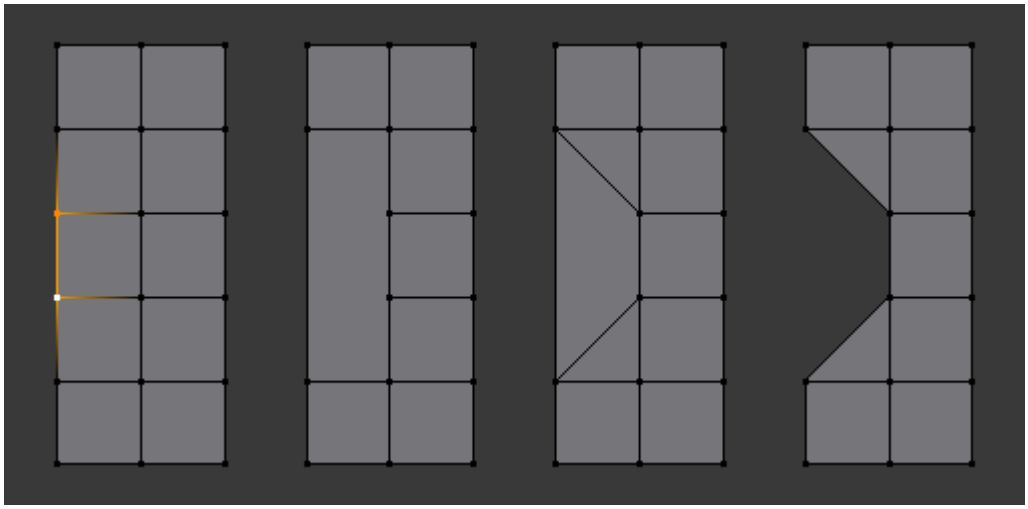
Mode:: Edit Mode
Menu:: *Mesh ‣ Delete ‣ Dissolve Vertices*

Remove the vertex, merging all surrounding faces. In the case of two edges, merging them into a single edge.

Face Split
: When dissolving vertices into surrounding faces, you can often end up with very large, uneven n-gons. The face split option limits dissolve to only use the corners of the faces connected to the vertex.

Tear Boundaries
: Split off face corners instead of merging faces.

Examples



1) Original mesh. 2) Face Split: Off, Tear Boundaries: Off. 3) Face Split: On, Tear Boundaries: Off. 4) Face Split: On/Off, Tear Boundaries: On.

Dissolve Edges

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Delete ▸ Dissolve Edges*

Removes edges sharing two faces (joining those faces).

The options are the same as for the Dissolve Vertices tool.

Dissolve Faces

Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Delete ▸ Dissolve Faces*

Merges regions of faces that share edges into a single face.

Note

This can be accessed quickly using the `F` key, see: Dissolve Existing Faces.

Dissolve (Context-Sensitive)

Reference

Shortcut:: `Ctrl-X`

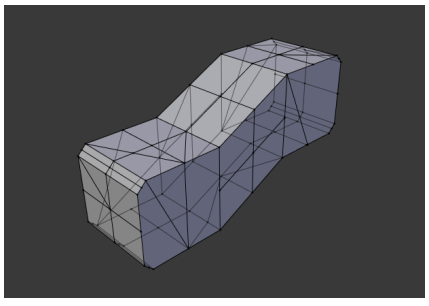
This is a convenient shortcut that dissolves based on the current selection mode (vertex, edge, face).

Limited Dissolve

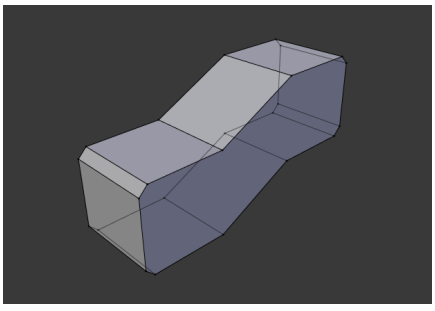
Reference

Mode:: Edit Mode
Menu:: *Mesh ▸ Delete ▸ Limited Dissolve*

This tool can simplify your mesh by dissolving vertices and edges separating flat regions.



Original mesh.



Result of Limited Dissolve.

Max Angle
: Reduces detail on planar faces and linear edges with an adjustable angle threshold.

All Boundaries
: Always dissolve vertices that have two edge users at boundaries.

Delimit
: Prevent faces from joining when they don't share certain properties (material for e.g.).

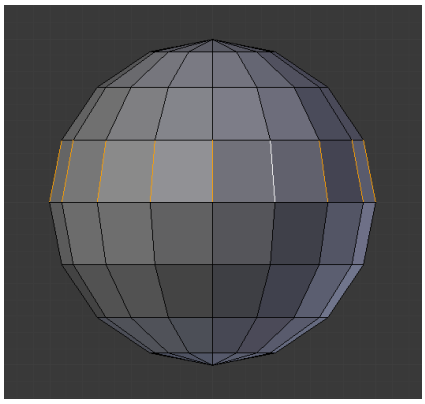
Collapse Edges & Faces

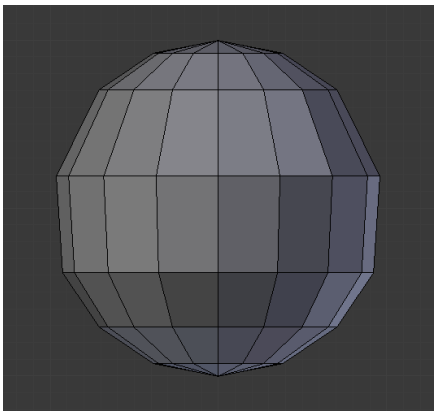
Reference

Mode::	Edit Mode
Menu::	*Mesh* ‣ *Delete* ‣ *Collapse Edges & Faces*
Shortcut::	X, *Collapse Edges & Faces*

Collapse each isolated edge and face region into single vertices, with support for face data such as UVs and vertex colors.

This is useful for taking a ring of edges and collapsing it, removing the face loop it ran through.





Selected edge ring. Edge ring collapsed.

Tip

This can be useful as a general way to remove detail, it has some advantages over:

Delete Vertices:: Leaves holes.
Collapse Vertices:: Doesn't correct UVs, vertex colors, etc.
Dissolve Vertices:: Often creates n-gons.

Edge Loops

Reference

Mode:: Edit Mode (Vertex or Edge select modes)
Menu:: *Mesh ▸ Delete ▸ Edge Loops*
Shortcut:: `X` or `Delete`, *Edge Loops*

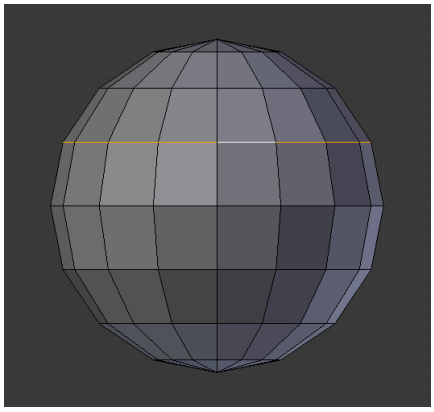
Edge Loop allows you to delete a selected edge loop if it is between two other edge loops. This will create one face loop where two previously existed.

Note

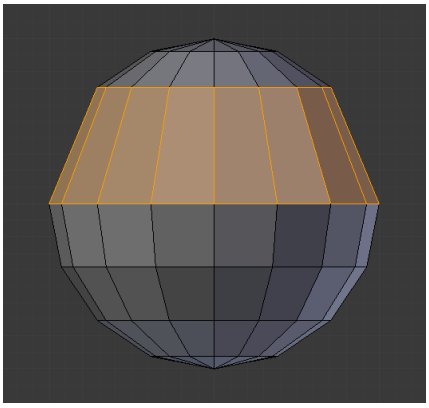
The *Edge Loop* option is very different to the *Edges* option, even if you use it on edges that look like an edge loop. Deleting an edge loop merges the surrounding faces together to preserve the surface of the mesh. By deleting a chain of edges, the edges are removed, deleting the surrounding faces as well. This will leave holes in the mesh where the faces once were.

Example

The selected edge loop on the UV Sphere has been deleted and the faces have been merged with the surrounding edges. If the edges had been deleted by choosing *Edges* from the *Delete* menu there would be an empty band of deleted faces all the way around the sphere instead.



Selected edge loop.



Edge loop deleted.

See also

- Vertex merging.
- Triangles to Quads.
- Un-Subdivide.

Vertex Operators

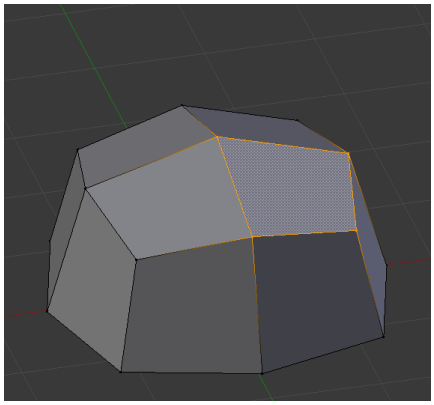
- Extrude Vertices
- Extrude to Cursor or Add
  - Creating Faces
- Bevel Vertices
  - Options
- New Edge/Face from Vertices
  - Methods
- Connect Vertex Path
- Connect Vertex Pairs
- Rip Vertices
  - Examples
  - Limitations
- Rip Vertices and Fill
- Rip Vertices and Extend
- Slide Vertices
- Smooth Vertices
- Laplacian Smooth
- Blend from Shape
- Propagate to Shapes
- Vertex Groups
- Hooks
- Make Vertex Parent

Extrude Vertices

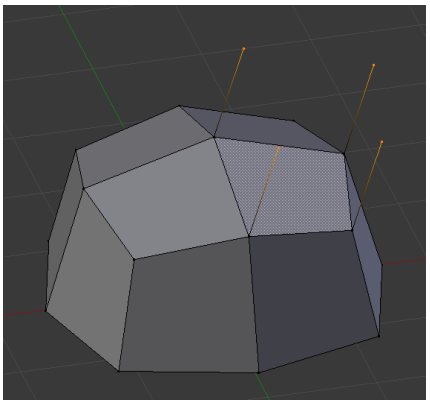
Reference

Mode::	Edit Mode
Menu::	*Vertex* ▸ *Extrude Vertices*, *Mesh* ▸ *Extrude* ▸ *Extrude Vertices*
Shortcut::	E

Extrude vertices as individual vertices.



Vertex selected.



Vertices extrude.

Extrude to Cursor or Add

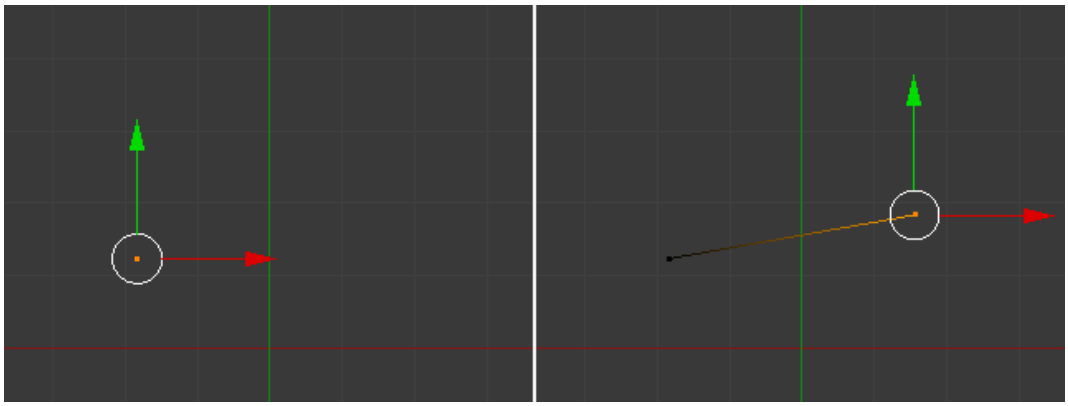
Reference

Mode::	Edit Mode
Shortcut::	`Ctrl-RMB`

Interactively places new vertices with `Ctrl-RMB` at the mouse cursor position.

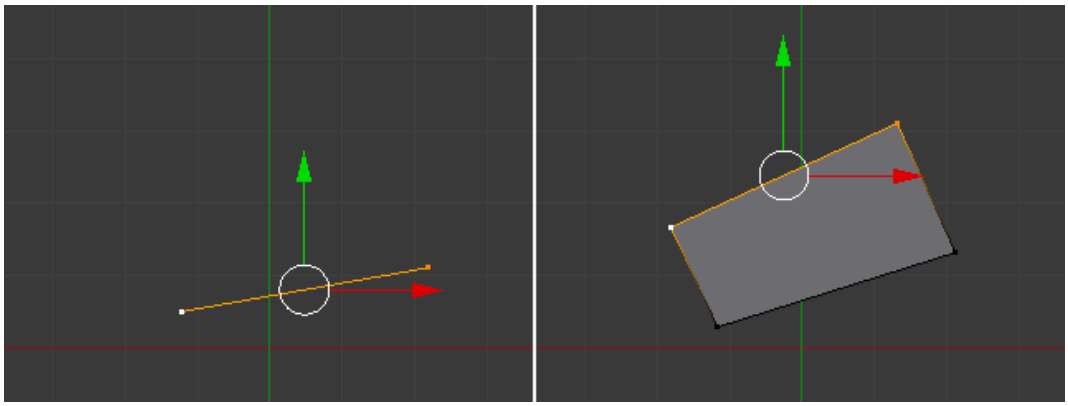
The most basic element, a vertex, can be added with a `Ctrl-RMB` click when no other vertices are selected. Because the camera space (computer screen) is two-dimensional, Blender cannot determine all three vertex coordinates from a single mouse click, so the new vertex is placed at the depth of the 3D cursor.

To create interconnected vertices, you can add a vertex and continuously make subsequent `Ctrl-RMB` operations with the last vertex selected. This will link the last selected vertex with the vertex created at the mouse position with an edge (see Fig. Adding vertices one by one.), and will continuously create and connect new vertices if you continue repeating this operation.



Adding vertices one by one.

Creating Faces



Quad from an Edge with source automatically rotated.

If you have two vertices selected and already connected with an edge, `Ctrl-RMB` click will create a planar face, also known as a quad. Blender will follow your mouse cursor and will use the planar view from your viewport to create those quads.

For `Ctrl-RMB`, Blender will automatically rotate the last selected Edge (the source) for the subsequent operations if you have at least one face created, dividing the angles created between the newly created edge and the last two edges, creating a smooth angle between them. Blender will calculate this angle using the last positive and negative position of the last X and Y coordinates and the last connected unselected edge. If this angle exceeds a negative limit (following a quadrant rule) between the recently created edge and the last two, Blender will wrap the faces. But if you do not want Blender to rotate and smooth edges automatically when extruding from `Ctrl-RMB`, you can also inhibit Blender from rotating sources using the shortcut `Shift-Ctrl-RMB`. In this case, Blender will not rotate the source dividing the angle between those edges when creating a face.

If you have three or more vertices selected, and `Ctrl-RMB` click, you will also create planar faces, but along the vertices selected, following the direction of the cursor. This operation is similar to an extrude operation.

Tip

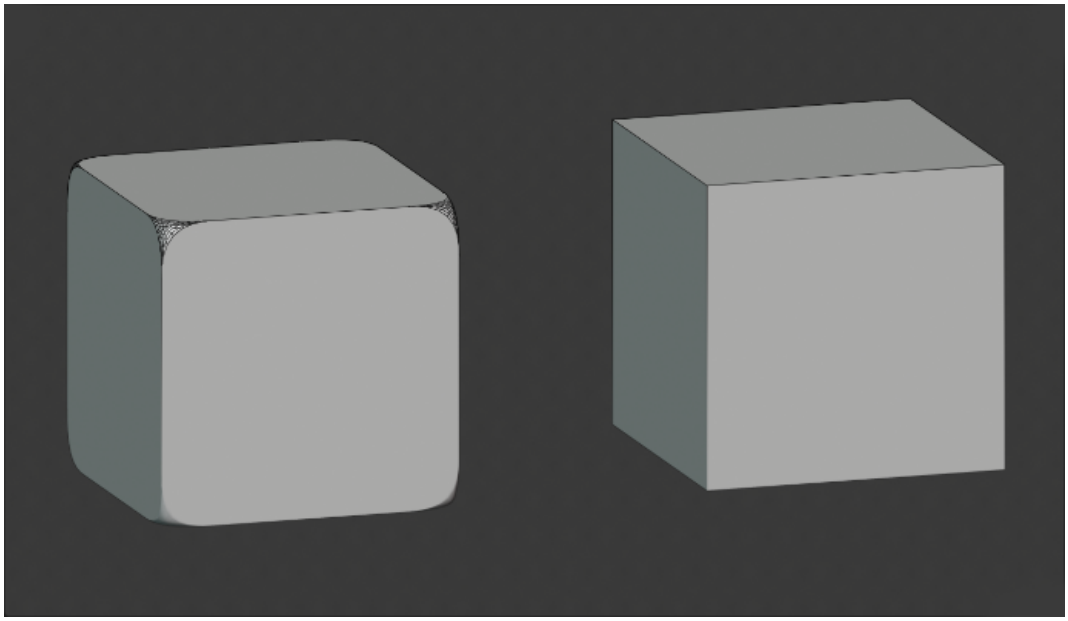
When adding objects with `Ctrl-RMB`, the extrusions of the selected elements, being vertices, edges and faces with the `Ctrl-RMB`, are viewport dependent. This means, once you change your viewport, for example, from top to left, bottom or right, the extrusion direction will also follow your viewport and align the extrusions with your planar view.

Bevel Vertices

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Bevel Edges*
Shortcut::	`Ctrl-B` (Bevel Edges)
Menu::	*Vertex ▸ Bevel Vertices*
Shortcut::	`Shift-Ctrl-B` (Bevel Vertices)

The *Bevel* tool rounds off edges or corners of a mesh at the point of the selected vertices. In "vertex only" mode, the *Bevel Vertices* tool works on selected vertices but the option to switch to *Bevel Edges* is available. By doing so, more vertices are added in order to smooth out profiles with a specified number of *segments* (see the options below for details about the bevel algorithm).



Cubes with and without bevel.

Note

With the Vertex Only option active, some of the other options available will not work. However, they will work with Bevel Edges.

Options

Affect `V`

Vertices:: Only the areas near vertices are beveled, the edges remain unchanged.

Edges:: Bevel the edges, creating intersections at vertices.

Offset `A`

You can change the bevel amount by moving the mouse towards and away from the object, like with transform tools. The exact meaning of the value depends on the *Amount Type* option (see below).

Width Type `M`

Selects how the *Amount* value controls the size of the bevel. According to the selection, the amount is:

Offset:: The relative distance from the new edge to the original.

Width:: The distance between the two new edges formed by the bevel (or the edges on either side of the bevel if there is more than one segment).

Depth:: The perpendicular distance from the original edge to the bevel face.

Percent:: The percentage of the length of adjacent edges that the new edges are slid along.

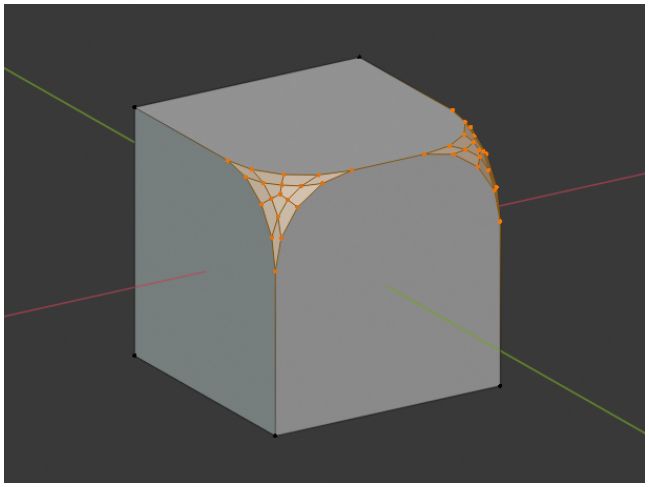
Absolute:: The exact distance along edges adjacent to the beveled edge. A difference from *Offset* is visible when the unbeveled edges attached to beveled edges meet at an angle besides a right angle.

For vertex-only bevels, the Offset and Depth types measure from the original vertex, and the Width type is measured from a new vertex to the center of the new face (as half the amount).

Segments `S`

The number of segments in the bevel can be defined by scrolling the mouse `Wheel` to increase or decrease this value. The greater the number of segments, the smoother the bevel. Or press `S` to change the number with mouse movements, as well as numeric input.

Alternatively, you can manually enter a segment number value while using the tool, or in the Mesh Tool options panel after using the tool.



Bevel with four segments.

Shape `P`

This is a number between 0 and 1 that controls the shape of the profile (side view of a beveled edge). The default value, 0.5, gives a circular arc (if the faces meet at right angles). Values less than that give a flatter profile, with 0.25 being exactly flat, and values less than that giving a concave bevel. Values more than 0.5 give a more convex profile. Similarly as *Segments* it can be set with mouse movements and numeric input after toggling `P`.

Material Index

The *Material* number specifies which material should be assigned to the new faces created by the *Bevel* tool. With the default, -1, the material is inherited from the closest existing face ("closest" can be a bit ambiguous). Otherwise, the number is the slot index of the material to use for all newly created faces.

Harden Normals `H`

When enabled, the per-vertex face normals of the bevel faces are adjusted to match the surrounding faces, and the normals of the surrounding faces are not affected. This will keep the surrounding faces flat (if they were before), with the bevel faces shading smoothly into them. For this effect to work, a mesh must have custom split normals. As a convenience, that option will be enabled for you if it is not already when you enable Harden Normals here.

Clamp Overlap `C`

Limits the width of each beveled edge so that vertices do not overlap with other geometry.

Loop Slide

If there are un-beveled edges along with beveled edges into a vertex, the bevel tries to slide along those edges when possible. Turning the option off can lead to more even bevel widths.

Face Strength Mode

Set *Face Strength* on the faces involved in the bevel, according to the specified mode. This can be used in conjunction with a Weight Normals Modifier (with the *Face Influence* option checked).

None:: Do not set face strength.

New:: Set the face strength of new faces along edges to *Medium*, and the face strength of new faces at vertices to *Weak*.

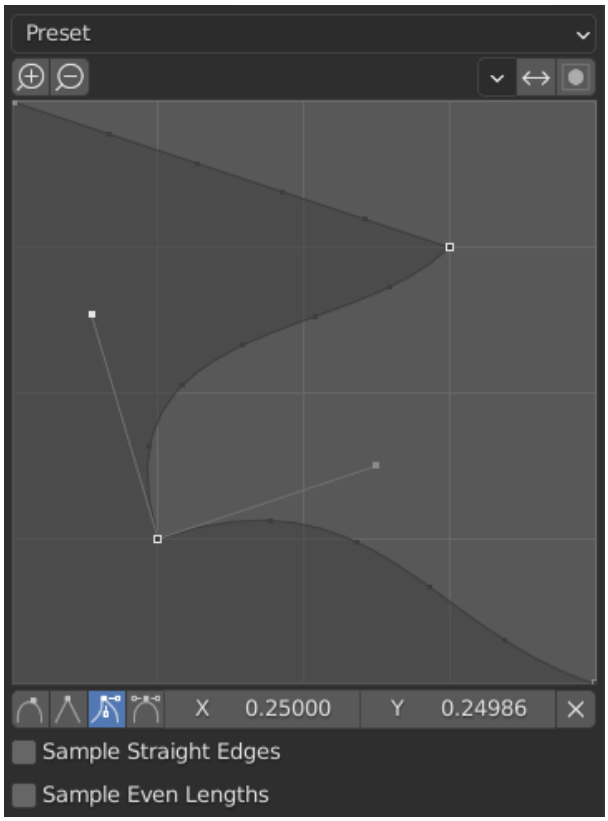
Affected:: In addition to those set for the *New* case, also set the faces adjacent to new faces to have strength *Strong*.

All:: In addition to those set for the *Affected* option, also set all the rest of the faces of the model to have strength *Strong*.

Profile Type `Z`

Superellipse:: Creates a bevel with a uniform concave or convex curve.

Custom::



The custom profile widget.

This widget allows the creation of a user-defined profile with more complexity than with the single profile parameter. The modal tool allows toggling the custom profile, but the shape of the profile is only editable in the options panel after the operation is confirmed.

The profile starts at the bottom right of the widget and ends at the top left, as if it were between two edges intersecting at a right angle. Control points are created in the widget and then the path is sampled with the number of segments from the Bevel modifier.

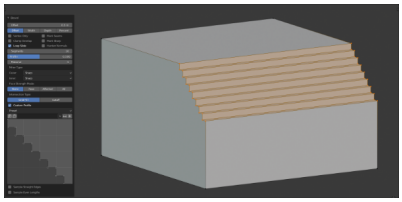
Presets

The *Support Loops* and *Steps* presets are built dynamically depending on the number of segments in the bevel. If the number of segments is changed, the preset will have to be re-applied.

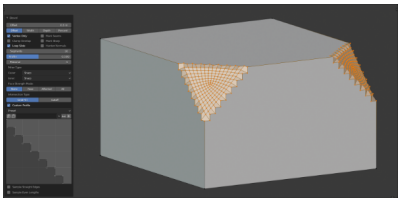
Sampling

Samples will first be added to each control point, then if there are enough samples, they will be divided evenly between the edges. The *Sample Straight Edges* option toggles whether the samples are added to edges with sharp control points on either side. If there aren't enough samples to give each edge the same number of samples, they will just be added to the most curved edges. So it is recommended to use at least as many segments as there are control points.

Bevel with Custom Profile on.



Edge Bevel.



Vertex Bevel.

See also

The Bevel Modifier is a non-destructive alternative to the Bevel tool.

New Edge/Face from Vertices

Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ New Edge/Face from Vertices*
Shortcut::	F

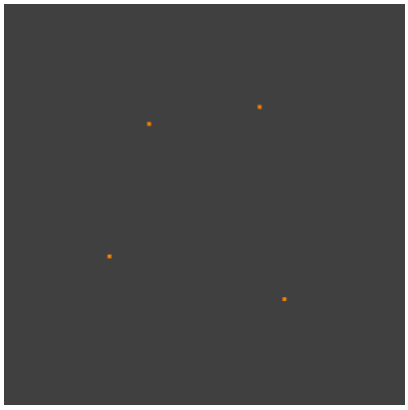
This is a context-sensitive tool which creates geometry by filling in the selection. When only two vertices are selected it will create an edge, otherwise it will create faces.

The typical use case is to select vertices and press F, yet Blender also supports creating faces from different selections to help to quickly build up geometry.

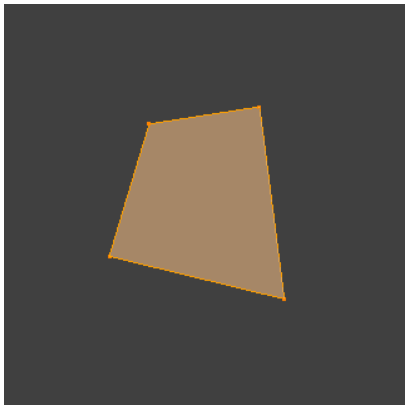
Methods

The following methods are used automatically depending on the context.

Isolated Vertices

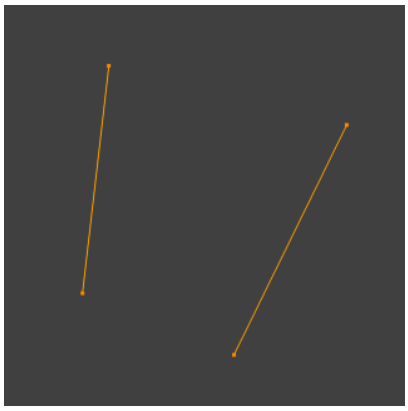


Before.

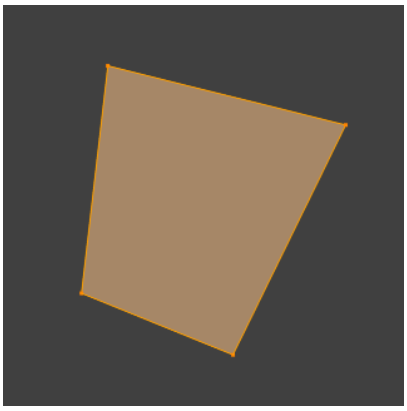


After.

Isolated Edges



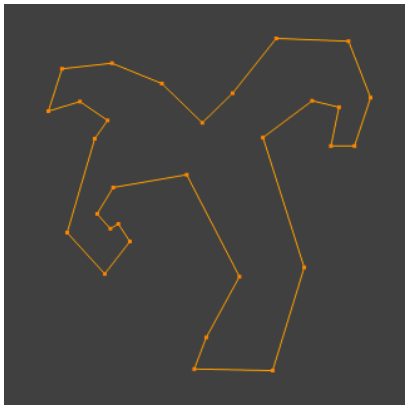
Before.



After.

N-gon from Edges

When there are many edges Blender will make an n-gon. Note that, this does not support holes, to support holes you need to use the Fill Faces tool.



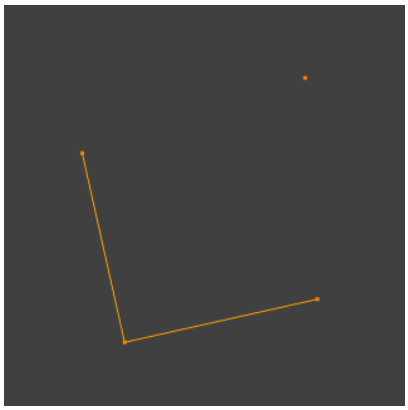
Before.



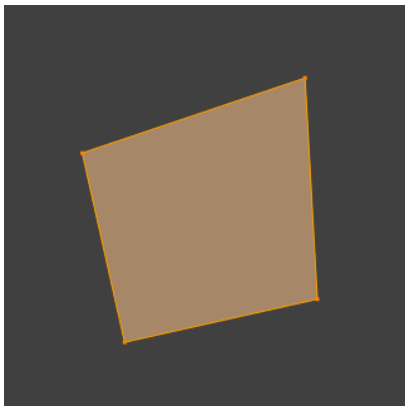
After.

Mixed Vertices/Edges

Existing edges are used to make the face as well as an extra vertex.



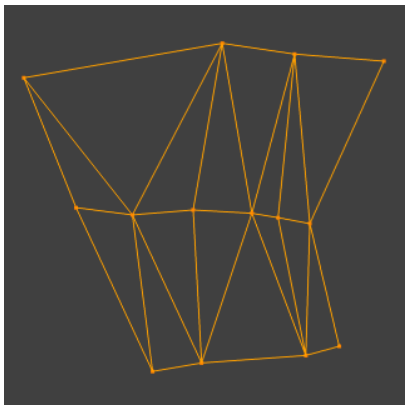
Before.



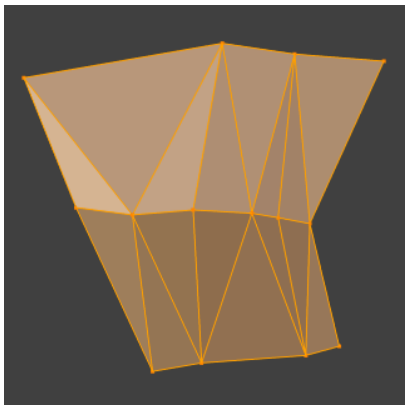
After.

Edge-Net

Sometimes you may have many connected edges without interior faces.



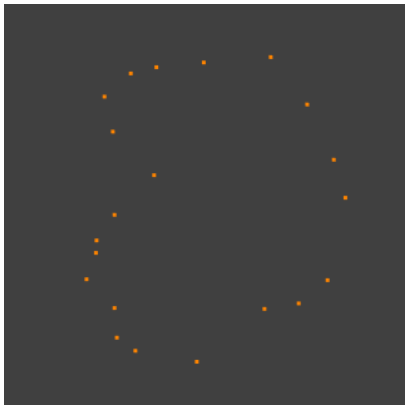
Before.



After.

Point Cloud

When there are many isolated vertices, Blender will calculate the edges for an n-gon.



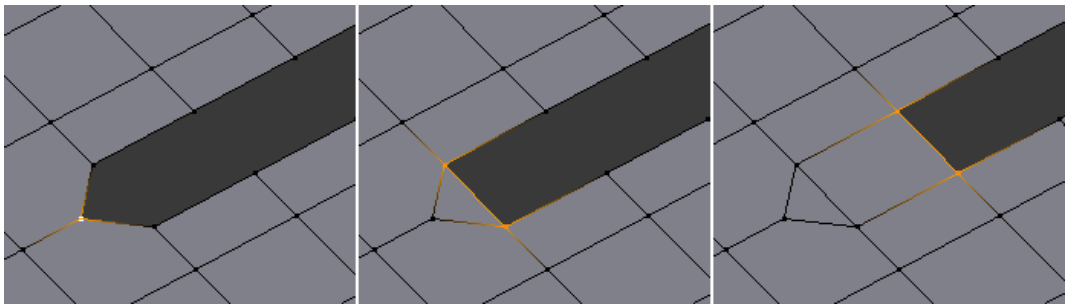
Before.



After.

Single Vertex Selection

With a single vertex selected on a boundary, the face will be created along the boundary, this saves manually selecting the other two vertices. Notice this tool can run multiple times to continue creating faces.



See also

For other ways to create faces see:

- Fill
- Grid Fill
- Bridge Edge Loops

Dissolve Existing Faces

When you have a region of existing faces, creating a face on this selection will remove the shared vertices and edges, creating a single face.

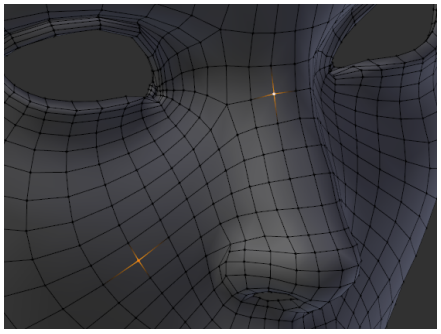
This is simply a convenience for accessing Dissolve Faces.

Connect Vertex Path

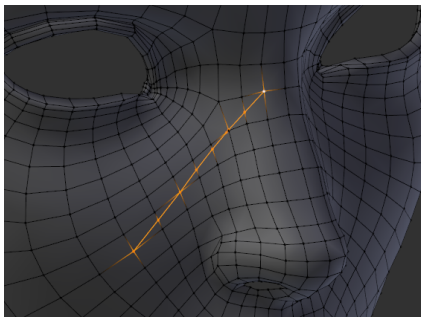
Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Connect Vertex Path*
Shortcut::	`J`

This tool connects vertices in the order they are selected, splitting the faces between them. When there are only two vertices selected, a cut will be made across unselected faces, a bit like the Knife tool; but this is limited to straight cuts across connected faces.

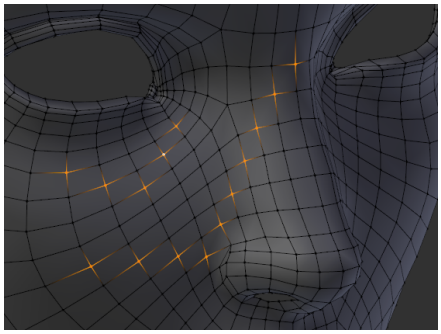


Two disconnected vertices.

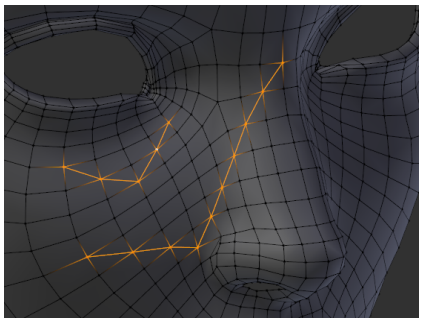


Result of connecting.

Running a second time will connect the first/last endpoints. When many vertices are selected, faces will be split by their selected vertices.



Before.



After.

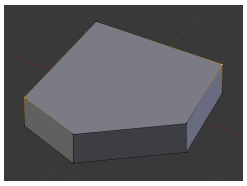
Vertices not connected to any faces will create edges, so this can be used as a way to quickly connect isolated vertices too.

Connect Vertex Pairs

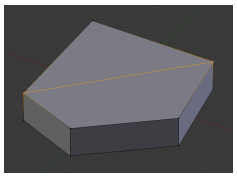
Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Connect Vertex Pairs*

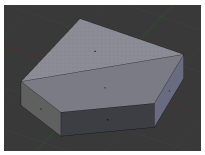
This operator connects selected vertices by creating edges between them and splitting the face. It can be used on many faces at once.



Vertices before connecting.



After connecting vertices.



Resulting face pair.

The main difference between this operator and Connect Vertex Path is that this operator ignores the selection order and connects all selected vertices that share a face.

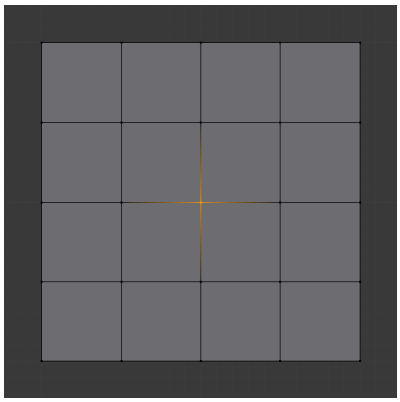
Rip Vertices

Reference

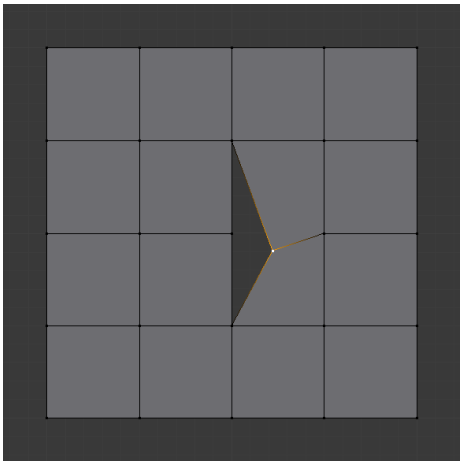
Mode::	Edit Mode
Menu::	*Vertex ▸ Rip Vertices*
Shortcut::	V

Rip creates a “hole” in the mesh by making a copy of selected vertices and edges, still linked to the neighboring non-selected vertices, so that the new edges are borders of the faces on one side, and the old ones, borders of the faces on the other side of the rip.

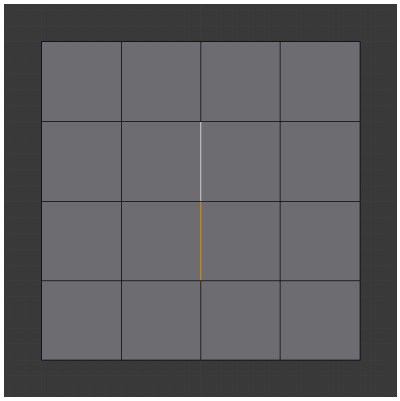
Examples



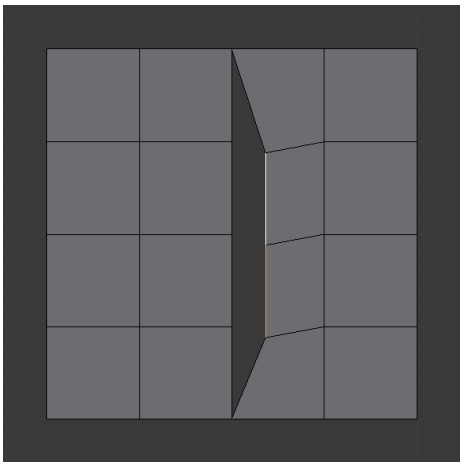
Selected vertex.



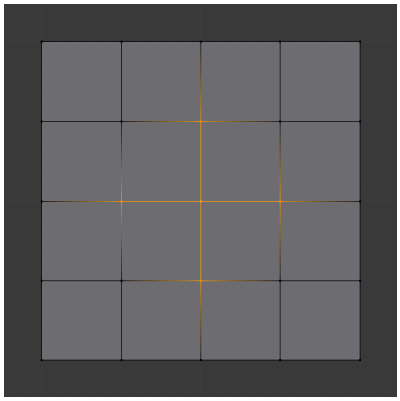
Hole created after using rip on vertex.



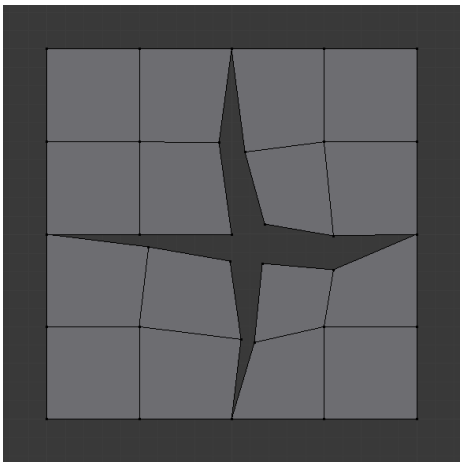
Edges selected.



Result of rip with edge selection.



A complex selection of vertices.



Result of rip operation.

Limitations

Rip will only work when edges and/or vertices are selected. Using the tool when a face is selected (explicitly or implicitly), will return an error message *"Cannot perform ripping with faces selected this way"*. If your selection includes edges or vertices that are not "between" two faces

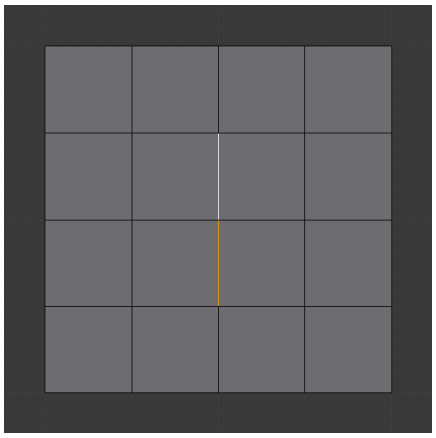
Manifold, it will also fail with the message *"No proper selection or faces include"*.

Rip Vertices and Fill

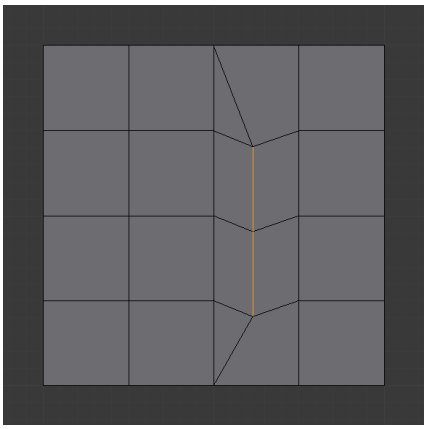
Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Rip Vertices and Fill*
Shortcut::	`Alt-V`

Rip fill works the same as the Rip tool above, but instead of leaving a hole, it fills in the gap with geometry.



Edges selected.



Result of rip fill.

Rip Vertices and Extend

Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Rip Vertices and Extend*
Shortcut::	`Alt-D`

This tool takes any number of selected vertices and duplicate-drags them along the closest edge to the mouse. When extending an edge loop, it extends the vertices at the endpoints of the loop. The behavior is similar to the *Extrude* tool, but it creates an n-gon.

It helps to easily add details to existing edges.

Slide Vertices

Reference

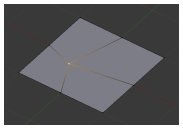
Mode::	Edit Mode
Menu::	*Vertex ▸ Slide Vertices*
Shortcut::	`Shift-V`, or `G` twice

Vertex Slide will transform a vertex along one of its adjacent edges. Use `Shift-V` to activate tool. The nearest selected vertex to the mouse cursor will be the control one. Move the mouse along the direction of the desired edge to specify the vertex position. Then press `LMB` to confirm the transformation.

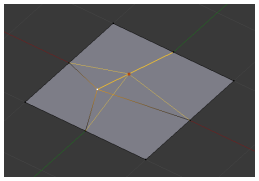
Even `E`
: By default, the offset value of the vertices is a percentage of the edges length along which they move. When Even mode is active, the vertices are shifted by an absolute value.

Flipped `F`
: When Flipped is active, vertices move the same distance from adjacent vertices, instead of moving from their original position.

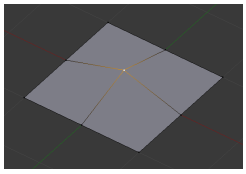
Clamp `Alt` or `C`
: Toggle clamping the slide within the edge extents.



Selected vertex.



Positioning vertex interactively.



Repositioned vertex.

Smooth Vertices

Reference

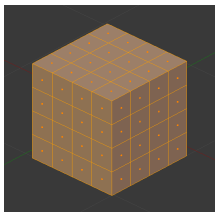
Mode::	Edit Mode
Menu::	*Vertex* ‣ *Smooth Vertices*, *Context Menu* ‣ *Smooth Vertices*

This operator smooths the selected vertices by averaging the angles between the faces.

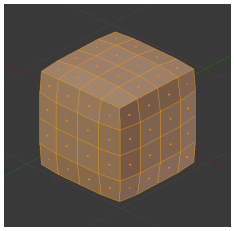
Smoothing
: Smoothing factor.

Repeat
: The number of smoothing iterations.

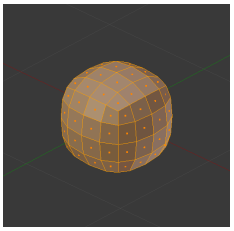
Axes
: Limit the effect to certain axes.



Mesh before smoothing.



Mesh after one smoothing iteration.



Mesh after ten smoothing iterations.

Tip

Using the Smooth tool after subdividing can help create a more organic shape.

See also

Smooth Modifier

The Smooth Modifier, which can be limited to a *Vertex Group*, is a non-destructive alternative to the Smooth operator.

Laplacian Smooth

Reference

Mode::	Edit Mode
Menu::	*Context Menu* ▸ *Laplacian Smooth*

See the Laplacian Smooth Modifier for details.

Laplacian smooth uses an alternative smoothing algorithm that better preserves larger details and this way the overall shape of the mesh. Laplacian smooth exists as a mesh operation and as a non-destructive modifier.

Note

Geometry Smoothing versus Smooth Shading

Do not mistake this tool with the smooth shading options, they do not work the same! This tool modifies the mesh itself, to reduce its sharpness, whereas *Set Smooth* only control the way the mesh is shaded, creating an *illusion* of softness, but without modifying the mesh at all.

Blend from Shape

Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Blend from Shape*

Blend in the shape from a shape key.

Propagate to Shapes

Reference

Mode:: Edit Mode
Menu:: *Vertex ▸ Propagate to Shapes*

Apply selected vertex locations to all other shape keys.

Vertex Groups

Todo.

Hooks

Reference

Mode::	Edit Mode
Menu::	*Vertex* ▸ *Hooks*
Shortcut::	`Ctrl-H`

Adds a Hook Modifier (using either a new empty, or the current selected object) linked to the selection. Note that even if it appears in the history menu, this action cannot be undone in *Edit Mode* – because it involves other objects…

When the current object has no hooks associated, only the first two options will appear on the menu.

Hook to New Object
: Creates a new Hook Modifier for the active object and assigns it to the selected vertices; it also creates an empty at the center of those vertices, which are hooked to it.

Hook to Selected Object
: Does the same as *Hook to New Object*, but instead of hooking the vertices to a new empty, it hooks them to the selected object (if it exists). There should be only one selected object (besides the mesh being edited).

Hook to Selected Object Bone
: Does the same as *Hook to New Object*, but it sets the last selected bone in the also selected armature as a target.

Assign to Hook
: The selected vertices are assigned to the chosen hook. For that to happen, a list of the hooks associated to the object is displayed. All the

unselected vertices are removed from it (if they were assigned to that particular hook). One vertex can be assigned to more than one hook.

Remove Hook

Removes the chosen hook (from the displayed list) from the object. Which means that the specific Hook Modifier is removed from its modifier stack.

Select Hook

Selects all vertices assigned to the chosen hook (from the hook list).

Reset Hook

It's equivalent to the *Reset* button of the specific Hook Modifier (chosen from the hook list).

Recenter Hook

It's equivalent to the *Recenter* button of the specific Hook Modifier (chosen from the hook list).

Make Vertex Parent

Reference

Mode::	Edit Mode
Menu::	*Vertex ▸ Make Vertex Parent*
Shortcut::	`Ctrl-P`

This will parent the other selected object(s) to the vertices/edges/faces selected, as described in parenting.

Edge Operators

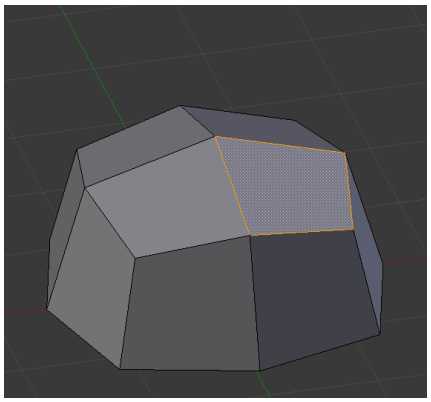
- Extrude Edges
- Bevel Edges
  - Usage
  - Options
  - Examples
- Bridge Edge Loops
  - Examples
- Screw
  - Usage
  - Options
  - Examples
- Subdivide
  - Options
  - Examples
- Subdivide Edge-Ring
- Un-Subdivide
- Rotate Edge
- Edge Slide
  - Usage
- Offset Edge Slide
- Loop Cut and Slide
  - Usage
  - Options
- Edge Data
  - Edge Crease
  - Edge Bevel Weight
  - Mark Seam & Clear Seam
  - Mark Sharp & Clear Sharp
  - Set Sharpness by Angle

Extrude Edges

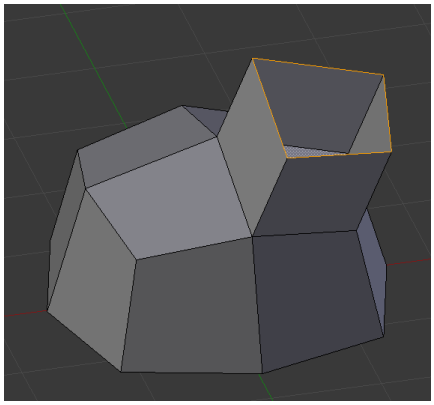
Reference

Mode::	Edit Mode
Menu::	*Edge* ▸ *Extrude Edges Mesh* ▸ *Extrude* ▸ *Extrude Edges*
Shortcut::	`E`

Extrude edges as individual edges.



Edge selected.



Edge extruded.

Bevel Edges

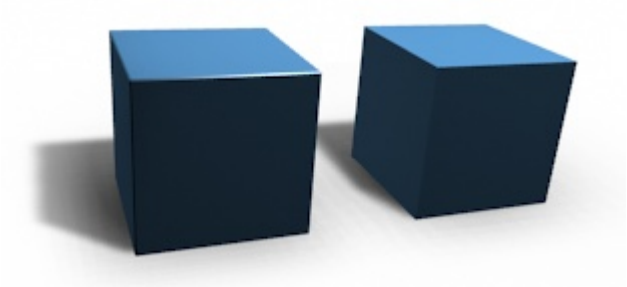
Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Bevel Edges*
Shortcut::	`Ctrl-B` (Bevel Edges)
Menu::	*Vertex ▸ Bevel Vertices*
Shortcut::	`Shift-Ctrl-B` (Bevel Vertices)

The *Bevel* tool allows you to create chamfered or rounded corners on geometry. A bevel is an effect that smooths out edges and corners.

Real world edges are very seldom exactly sharp. Not even a knife blade edge can be considered perfectly sharp. Most edges are intentionally beveled for mechanical and practical reasons.

Bevels are also useful for giving realism to non-organic models. In the real world, the blunt edges on objects catch the light and change the shading around the edges. This gives a solid, realistic look, as opposed to un-beveled objects which can look too perfect.

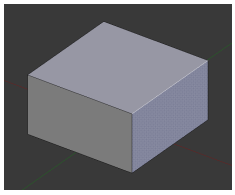


Cubes with and without bevel.

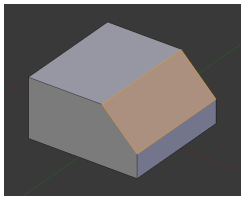
Usage

The *Bevel Edges* tool works only on selected edges with exactly two adjacent faces. It will recognize any edges included in a vertex or face selection as well, and perform the bevel the same as if those edges were explicitly selected. In "vertex only" mode, the *Bevel Vertices* tool works on selected vertices instead of edges, and there is no requirement about having any adjacent faces. The *Bevel* tool smooths the edges and/or "corners" (vertices) by replacing them with faces making smooth profiles with a specified number of *segments* (see the options below for details about the bevel algorithm).

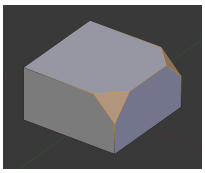
Use `Ctrl-B` or a method listed above to run the tool. Move the mouse to interactively or type a number to specify the bevel offset, and scroll the `Wheel` to increase or decrease the number of segments (see below).



Selected edge before beveling.



Result of bevel (one segment).



Result of bevel (vertex only).

Note

Normal (edge) beveling only works on edges that have exactly two faces attached to them. Vertex beveling has no such restriction.

Options

Affect `V`
: Vertices

Only the areas near vertices are beveled, the edges remain unchanged.

Edges
: Bevel the edges, creating intersections at vertices.

Width Type `M`
: Selects how the *Width* value controls the size of the bevel. According to the selection, the width is:

Offset::	The distance from the new edge to the original.
Width::	The distance between the two new edges formed by the bevel (or the edges on either side of the bevel if there is more than one segment).
Percent::	The percentage of the length of adjacent edges that the new edges slide.
Absolute::	The exact distance along edges adjacent to the beveled edge. A difference from **Offset** is visible when the unbeveled edges attached to beveled edges meet at an angle besides a right angle.

For vertex-only bevels, the Offset and Depth types measure from the original vertex. The Width type is measured from a new vertex to the center of the new face (as half the *Width*).

Width `A`
: You can change the bevel width by moving the mouse towards and away from the object, a bit like with transform tools. The exact meaning of the value depends on the *Width Type* option (see above). As usual, the scaling can be controlled to a finer degree by holding `Shift` to scale in 0.001 steps. `LMB` finalizes the operation, `RMB` or `Esc` aborts the action.

Note

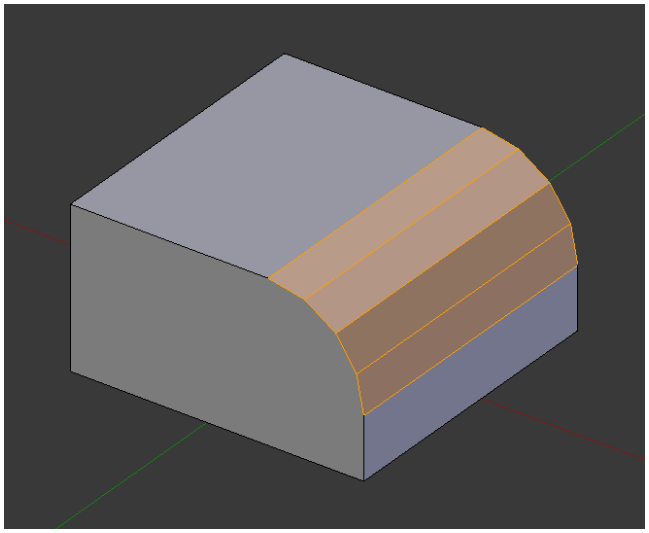
When multiple edges are beveled at the same time, it is sometimes impossible to make the width match the above definition on all edges simultaneously. Bevel tries to compromise in such cases. Sometimes

turning off Loop Slide (see below) can make it easier for Bevel to make the widths as specified.

Segments `S`

The number of segments in the bevel can be defined by scrolling the mouse `Wheel` to increase or decrease this value. The greater the number of segments, the smoother the bevel. Or press `S` to change the number with mouse movements, as well as numeric input.

Alternatively, you can manually enter a segment number value while using the tool, or in the Mesh Tool options panel after using the tool.



Bevel with four segments.

Shape `P`

This is a number between 0 and 1 that controls the shape of the profile (side view of a beveled edge). The default value, 0.5, gives a circular arc (if the faces meet at right angles). Values less than that give a flatter profile, with 0.25 being exactly flat, and values less than that giving a concave bevel. Values more than 0.5 give a more convex profile. Similarly as *Segments* it can be set with mouse movements and numeric input after toggling `P`.

Material Index

The *Material* number specifies which material is assigned to the new faces created by the *Bevel* tool. With the default, -1, the material is inherited from the closest existing face ("closest" can be a bit ambiguous). Otherwise, the number is the slot index of the material to use for all newly created faces.

Harden Normals `H`

When enabled, the per-vertex face normals of the bevel faces are adjusted to match the surrounding faces, and the normals of the surrounding faces are not affected. This will keep the surrounding faces flat (if they were before), with the bevel faces shading smoothly into them. For this effect to work, a mesh must have custom split normals. As a convenience, that option will be enabled for you if it is not already when you enable Harden Normals here.

Clamp Overlap `C`

Limits the width of each beveled edge so that edges cannot cause overlapping intersections with other geometry.

Loop Slide

If there are un-beveled edges along with beveled edges into a vertex, the bevel tries to slide along those edges when possible. Turning the option off can lead to more even bevel widths.

Mark

Seams `U`

If a seam edge crosses a non-seam one and you bevel all of them, this option will maintain the expected propagation of seams.

Sharp `K`

Similar to Mark Seams, but for sharp edges.

Miter Outer `O`

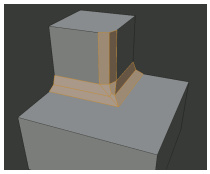
A *miter* is formed when two beveled edges meet at an angle. On the side where the angle is greater than 180 degrees, if any, it is called an *outer miter*. This option specifies the pattern that Blender uses at an outer miter.

Sharp:: Edges meet at a sharp point, with no extra vertices introduced on the edges.

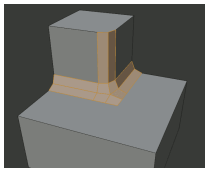
Patch:: Edges meet at a sharp point but in addition, two extra vertices are introduced near the point so that the edges and faces at the vertex may be less pinched together than what occurs in the *Sharp* case.

Arc:: Two vertices are introduced near the intersection, and a curved arc joins them together. The *Spread* slider controls how far the new vertices are from the intersection. The *Profile* curve widget controls the shape of the arc.

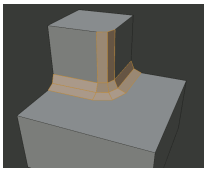
The current choices are shown in this diagram, where the outer miter is along the horizontal surface.



Sharp outer miter.



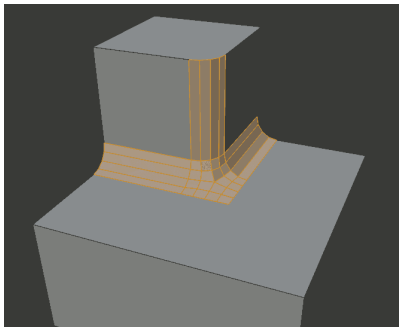
Patch outer miter.



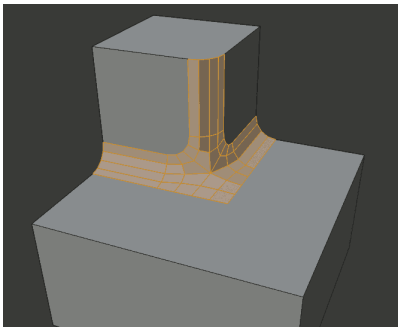
Arc outer miter.

Inner `I`

An *Inner Miter* is formed when the angle between two beveled edges is less than 180 degrees. This option specifies the pattern Blender uses at an inner miter. The options are the same as for Outer Miter, except that *Patch* makes no sense and is therefore omitted. Inner miters are shown in the following diagram, where two inner miters are on the vertical surfaces.



Sharp inner miter.



Arc inner miter.

Spread

The value used to spread extra vertices apart for *Outer* and *Inner Miters*. This option is available when Miter Inner is set to Arc.

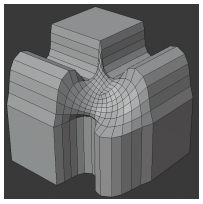
Intersection Type N

When more than two beveled edges meet at a vertex, a mesh is created as a way to complete the intersection between the generated geometry. This option controls the method used to create that mesh.

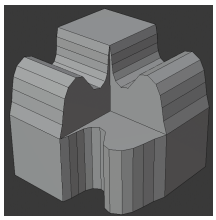
Grid Fill:: The default method for building intersections, useful when a smooth continuation of the bevel profile is desired. Without *Custom Profile* enabled, the curve of the profile continues through the intersection, but with a custom profile it just creates a smooth grid within the intersection's boundary.

Cutoff:: Creates a cutoff face at the end of each beveled edge coming into the vertex. This is most useful for custom profiles when the new intersection is too complex for a smooth grid fill.
With a three way intersection, when the inner corners of the cutoff profiles faces meet at the same location, no center face is created.
The direction of the cutoff faces depends on the original vertex's normal.

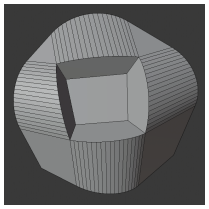
Intersection method options.



Grid fill intersection method.



Three way cutoff intersection where the inner vertices are merged.



Cutoff intersection method with a center face.

Face Strength

Set *Face Strength* on the faces involved in the bevel, according to the specified mode. This can be used in conjunction with a Weight Normals Modifier (with the *Face Influence* option checked).

None:: Do not set face strength.

New:: Set the face strength of new faces along edges to *Medium*, and the face strength of new faces at vertices to *Weak*.

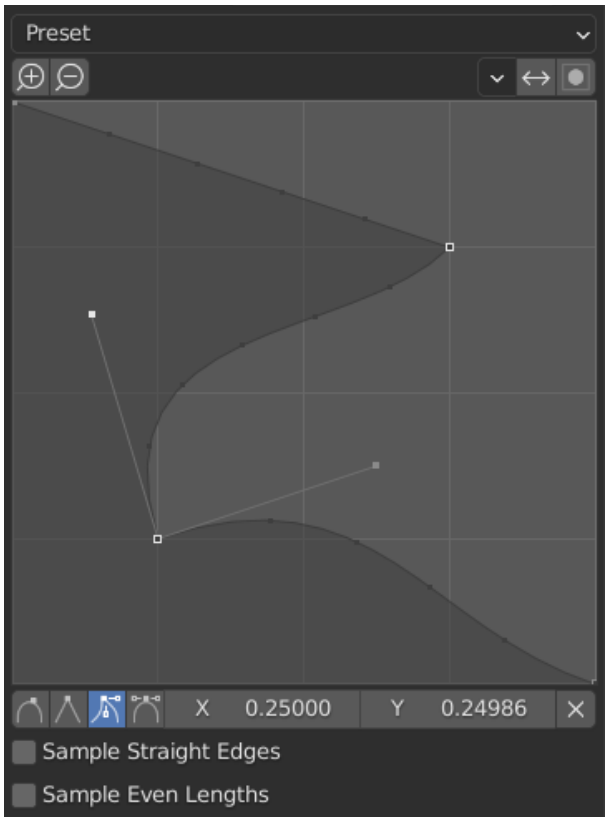
Affected:: In addition to those set for the *New* case, also set the faces adjacent to new faces to have strength *Strong*.

All:: In addition to those set for the *Affected* option, also set all the rest of the faces of the model to have strength *Strong*.

Profile Type Z

Superellipse:: Creates a bevel with a uniform concave or convex curve.

Custom:: This Curve Widget allows the creation of a user-defined profile with more complexity than with the single profile parameter. The modal tool



The custom profile widget.

allows toggling the custom profile, but the shape of the profile is only editable in the options panel after the operation is confirmed.

The profile starts at the bottom right of the widget and ends at the top left, as if it were between two edges intersecting at a right angle. Control points are created in the widget and then the path is sampled with the number of segments from the Bevel modifier.

Note

The *Profile* curve widget stays active when miters are enabled because it still controls the shape of the miter profiles.

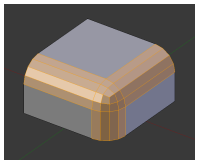
Presets

The *Support Loops* and *Steps* presets are built dynamically depending on the number of segments in the bevel. If the number of

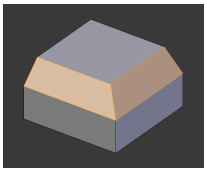
segments is changed, the preset will have to be re-applied.

Sampling
: Samples will first be added to each control point, then if there are enough samples, they will be divided evenly between the edges. The *Sample Straight Edges* option toggles whether the samples are added to edges with sharp control points on either side. If there aren't enough samples to give each edge the same number of samples, they will just be added to the most curved edges. So it is recommended to use at least as many segments as there are control points.

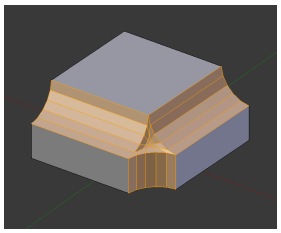
Examples



Result of beveling multiple edges.



Another example of beveling multiple edges.



An example using Profile=0.150.

See also

The Bevel Modifier is a non-destructive alternative to the Bevel tool.

Bridge Edge Loops

Reference

Mode:: Edit Mode
Menu:: *Edge ▸ Bridge Edge Loops*

Bridge Edge Loops connects multiple edge loops with faces.

Connect Loops
- **Open Loop::** Loops connected with open ends.
- **Closed Loop::** Tries to connect to a circular loop (where the start and end are merged).
- **Loop Pairs::** Connects each even count of loops individually.

Merge
: Merges edge loops rather than creating a new face.

Merge Factor
: Which edge loop the edges are merged to, a value of 0.5 will merge at a half-way point.

Twist
: Determines which vertices in both loops are connected to each other.

Number of Cuts
: The number of intermediate edge loops used to bridge the distance between two loops.

Interpolation
: Linear, Blend Path, Blend Surface

Smoothness
: Smoothness of the *Blend Path* and *Blend Surface*.

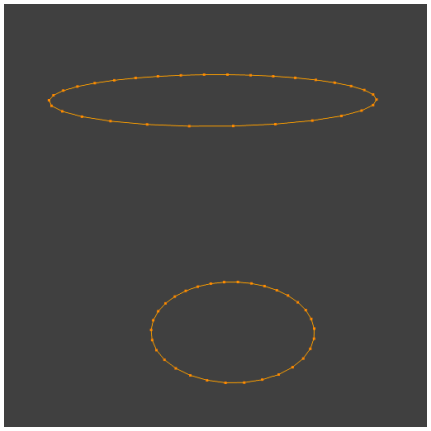
Profile Factor

How much intermediary new edges are shrunk/expanded.

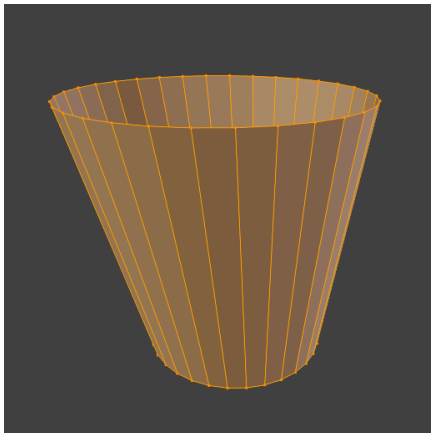
Profile Shape
: The shape of the new edges. See the Proportional Editing page for a description of each option.

Examples

Simple example showing two closed edge loops.



Input.

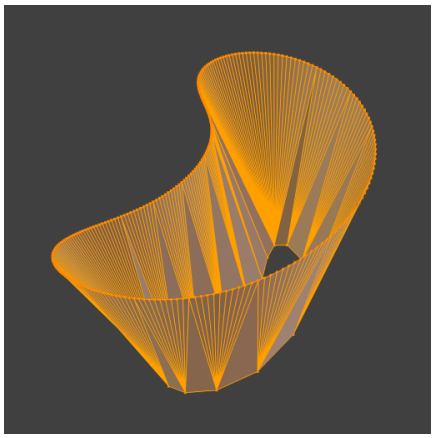


Bridge result.

Example of the Bridge tool between edge loops with different numbers of vertices.

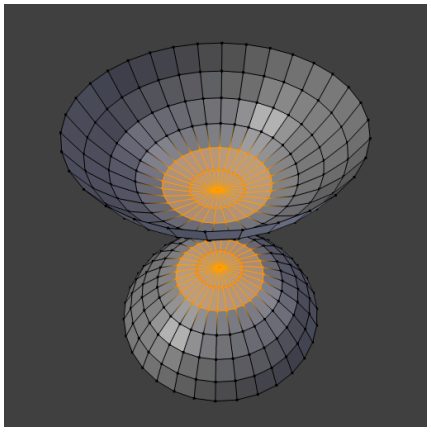


Input.

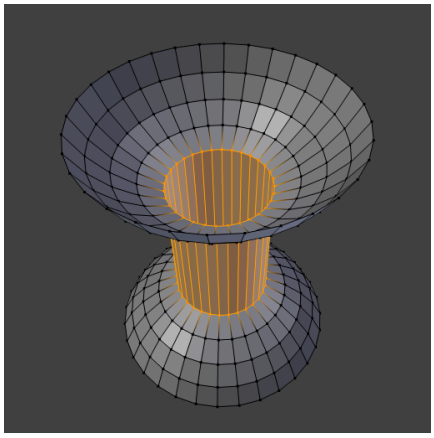


Bridge result.

Example using the Bridge tool to cut holes in face selections and connect them.

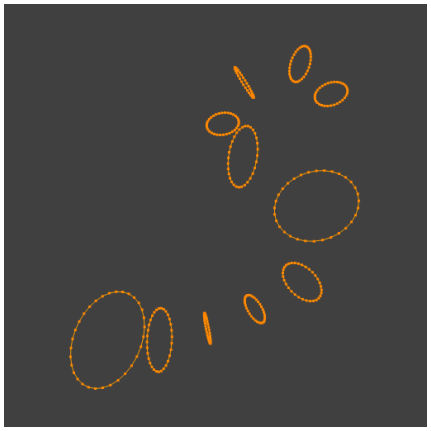


Input.

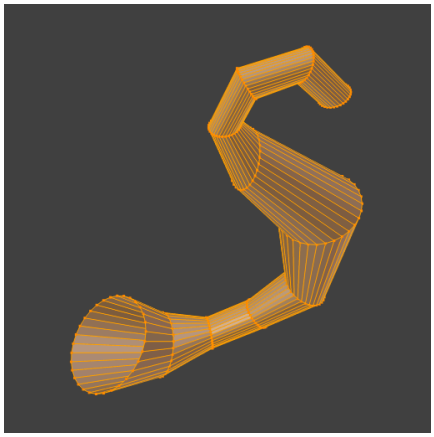


Bridge result.

Example showing how Bridge tool can detect multiple loops and connect them in one step.

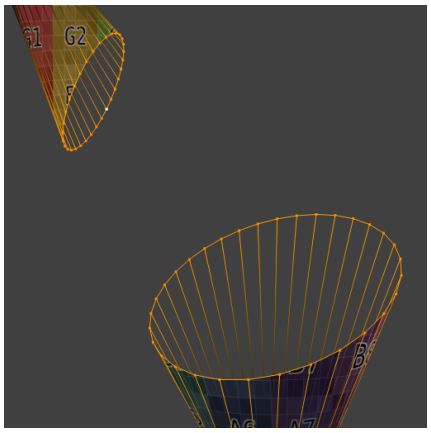


Input.

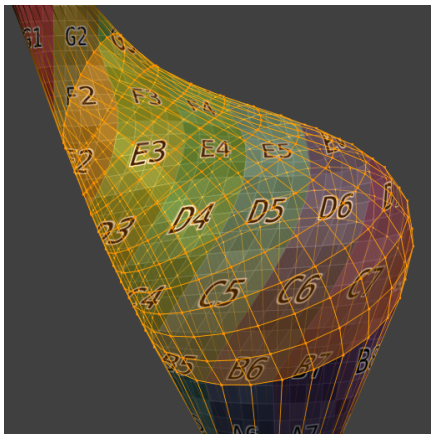


Bridge result.

Example of the subdivision option and surface blending with UVs.



Input.



Bridge result.

Screw

Reference

Mode::	Edit Mode
Menu::	*Edge* ▸ *Screw*

The *Screw* operator extrudes geometry along a helix. You can use it to create screws, springs, sea shells and so on.

While it's similar to the Screw Modifier, there are some important differences:

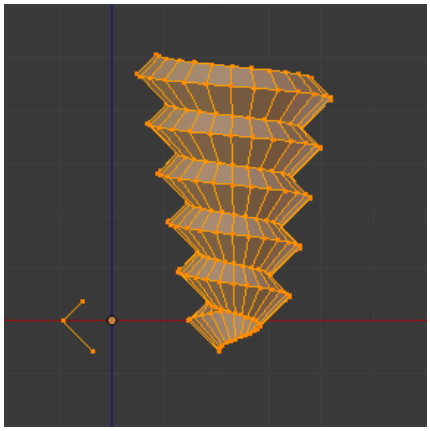
Screw operator	**Screw modifier**
Works in world space.	Works in object space.
Extrudes only the selected geometry.	Extrudes all geometry.
The centerpoint can be specified manually.	The centerpoint is always the object's origin.
One revolution is always 360°.	The angle can be chosen freely.
The height offset of each revolution is calculated automatically based on the geometry.	The height offset must be specified manually.

Screw operator	Screw modifier
Each revolution can also have a radial offset away from/towards the central axis (again determined by the geometry).	The radius stays constant.

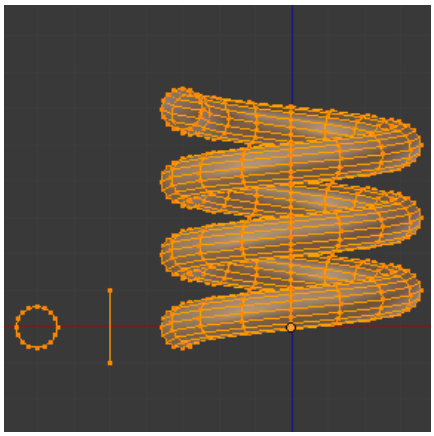
As described above, the Screw operator automatically determines the height offset and radial offset to apply after each revolution. It does this by looking for the endpoints of an open profile – a series of connected edges that don't form a closed loop. The geometry is extruded so that the profile's top vertex in one revolution will coincide with the bottom vertex in the next.

The most common use case is extruding such an open profile. You're not limited to this, however. As long as there is one open profile in the selection – even just a single loose edge – you can also extrude closed profiles and even geometry with faces.

You can see some examples below.



Wood Screw.



Spring.

Usage

First, make sure you have an open profile in your mesh. If you want to extrude anything else (such as one or more circles), you should create an open profile next to it.

Once that's done, enter *Edit Mode* and select the geometry you want to extrude, making sure to include exactly one open profile. If you have fewer or more, the operator will fail with the error "You have to select a string of connected vertices too."

Make sure the 3D Cursor is at the centerpoint around which you want the geometry to turn. Also make sure that the vertical axis on your screen matches the direction of the axis around which it should turn. The most common example is turning around the global Z axis: for that, you'd place the 3D Cursor at the world origin and switch to an orthographic side view.

Now you're ready to run the operator: open the *Edge* menu (by clicking it in the 3D Viewport's header or pressing `Ctrl-E`) and click *Screw*.

You can change the number of steps and turns in the Adjust Last Operation panel.

If you only created an open profile to guide the extrusion (not because you wanted its geometry), you can select its extruded faces by hovering over one and pressing `L`, then delete them by pressing `X` or `Delete`.

Options



Screw panel (in Edit Mode).

Steps
: The number of extrusions to be done for each 360° turn.

Turns
: The number of turns to create.

Center X, Y, Z
: The worldspace coordinates of the centerpoint around which to spin the geometry. Initially this is the location of the 3D Cursor.

Axis X, Y, Z
: The direction vector around which to spin the geometry. Initially this is the screen-space vertical axis (so the global Z axis when in a side view or the global Y axis when in top view). By inverting the *Axis*, you can flip between going clockwise and counterclockwise.

Tip

You can use the Align View to Active menu item to align the viewport, and thereby the *Axis*, to a certain item in the scene.

Notice that the *Axis* only determines how the geometry spins "horizontally" around the centerpoint. It doesn't determine how the geometry moves "vertically." Instead, the geometry always moves by a distance and direction given by the endpoints of the open profile, and goes downward in the object's local space.

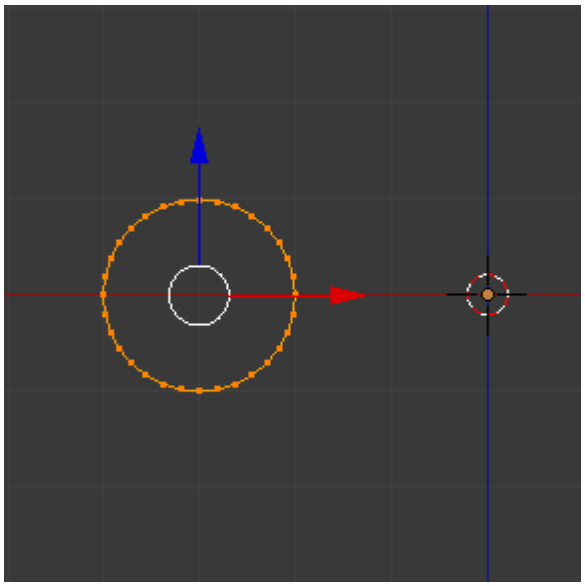
Examples

Creating a Spring

First, let's create a circle to serve as the spring's cross section:

1. Open Blender and delete the default Cube.

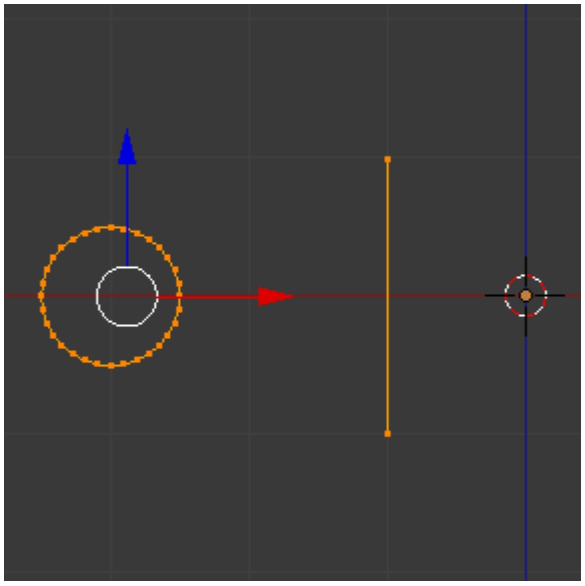
2. Add a circle by pressing `Shift-A` and selecting *Mesh* ▸ *Circle*.
3. Set the *Location X* property of this new object to -3 and its *Rotation X* property to 90°.
4. Enter *Edit Mode* by pressing `Tab`.
5. Switch to the *Front Orthographic* view by pressing `Numpad1`.



Extrusion profile created.

Next, let's create a vertical line to specify the distance between spring loops:

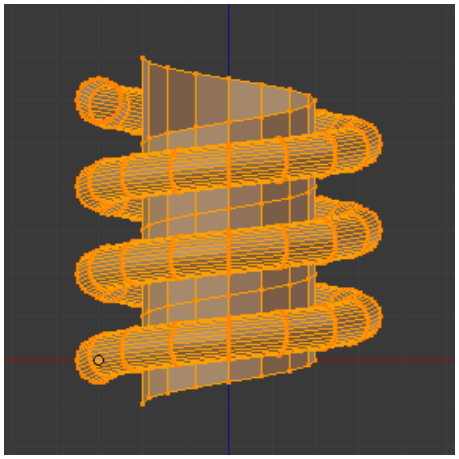
1. Deselect all vertices by clicking on an empty space or pressing `Alt-A`.
2. Click `Ctrl-RMB` twice to create two vertices connected by an edge.
3. Select both vertices and press `S X 0 Return` to ensure they have the same X coordinate. (This is necessary to keep the spring's radius constant.)



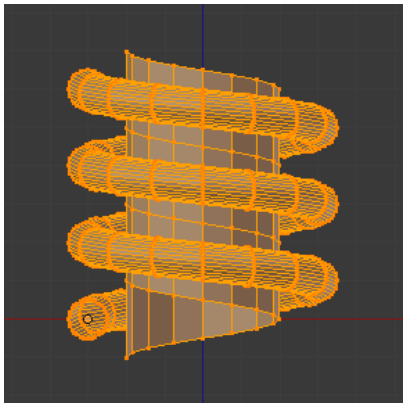
Guide profile created.

Now, we're ready to create the spring:

1. Select both the circle and the line by pressing `A`.
2. Click *Edge* ▸ *Screw*.
3. Adjust the *Steps* and *Turns* to your liking.
4. Try changing *Axis Z* to -1 and see that this makes the spring turn the other way.

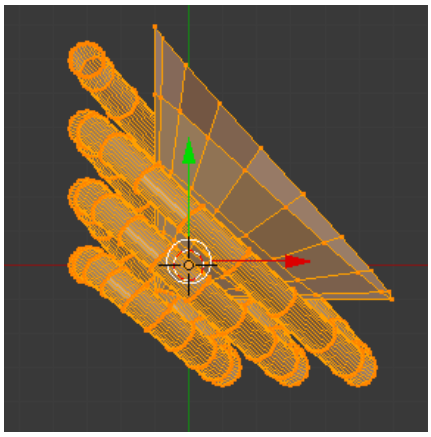


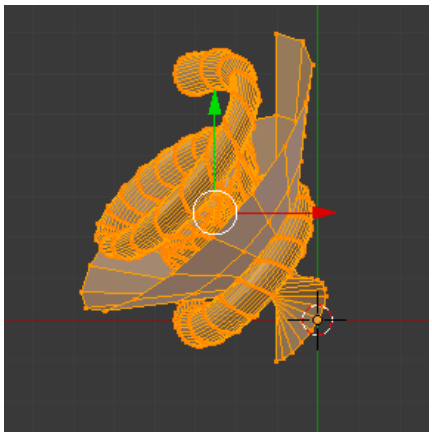
Counterclockwise direction.



Flipped to Clockwise direction.

You can get some interesting results by making the *Axis* diagonal (e.g. keeping *Axis Z* at -1 and setting *Axis X* to 1). Notice that each individual spring loop is inclined by 45°, but that after each loop, we still go down vertically (in the direction of the guide line).

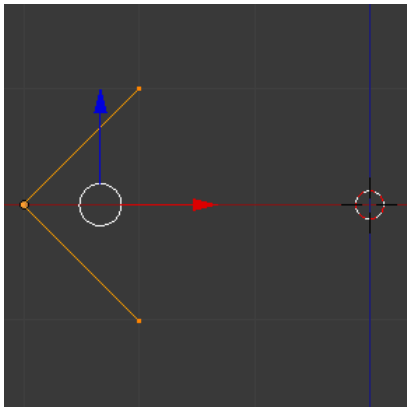




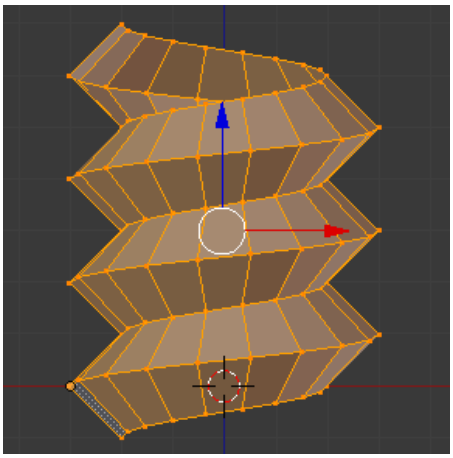
Creating a Screw Spindle

The Screw operator is perfectly suited for creating helixes without any gaps between the turns.

1. Open Blender and enter *Edit Mode* for the default cube.
2. Delete all vertices by pressing `X` or `Delete`.
3. Switch to the *Front Orthographic* view by pressing `Numpad1`.
4. Click `Ctrl-RMB` three times to create a profile like the one below.
5. Select the two vertices closest to the global Z axis and press `S X 0 Return` to ensure they have the same X coordinate.
6. Select all three vertices and click *Edge ▸ Screw*.
7. Adjust the *Steps* and *Turns* to your liking.



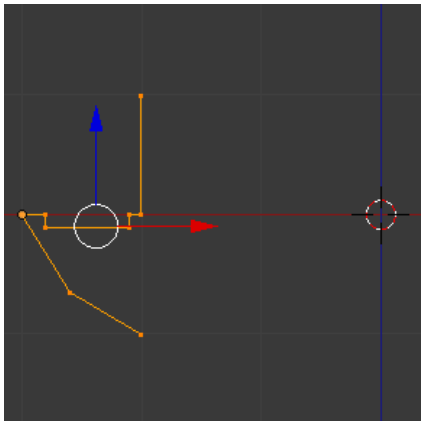
Profile for a screw spindle.



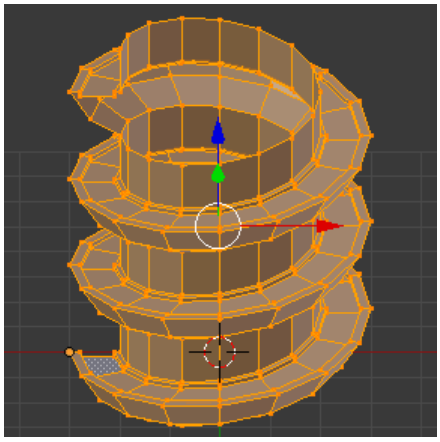
Result after running the Screw operator.

You can also create more interesting shapes, like this spiral “staircase”:

Ramp.



Profile.

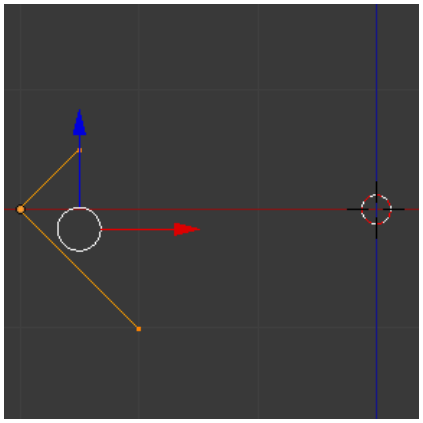


Generated mesh.

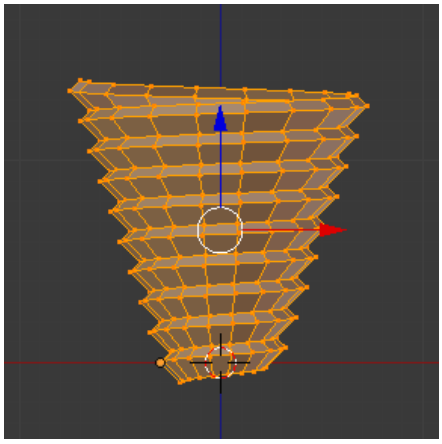
Creating a Screw Tip

Until now, we’ve always made sure that the first and last vertex of the profile have the same X coordinate, thereby keeping the radius of the helix

constant. However, nothing stops you from using different X coordinates and having the helix shrink/expand along its height.



Profile.



Generated.

Subdivide

Reference

Mode::	Edit Mode
Menu::	*Edge* ▸ *Subdivide*

Subdividing splits selected edges and faces by cutting them in half or more, adding new vertices, and subdividing accordingly the faces involved. It adds resolution to the mesh by divide faces or edges into smaller units.

This process follows a few rules, depending on the settings:

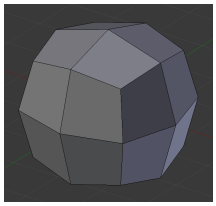
- When only one edge of a face is selected (Triangle mode), triangles are subdivided into two triangles, and quads, into three triangles.
- When two edges of a face are selected:
  - If the face is a triangle, a new edge is created between the two new vertices, subdividing the triangle in a triangle and a quad.
  - If the face is a quad, and the edges are neighbors, there have *three* possible behaviors to divide the quad, depending on the setting of *Quad Corner Type* (see below for details).
  - If the face is a quad, and the edges are opposite, the quad is just subdivided in two quads by the edge linking the two new vertices.
- When three edges of a face are selected:
  - If the face is a triangle, this means the whole face is selected and it is then subdivided in four smaller triangles.
  - If the face is a quad, first the two opposite edges are subdivided as described above. Then, the "middle" edge is subdivided, affecting its new "sub-quad" as described above for only one edge.
- When a face of a Quad is selected, the face is subdivided into four smaller quads.
- When a face of an N-gon is selected, the individual edges will be subdivided but the face will stay unsubdivided.

Options

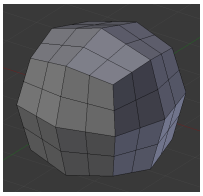
These options are available in the *Tool* panel after running the tool:

Number of Cuts
: Specifies the number of cuts per edge to make. By default this is 1, cutting edges in half. A value of 2 will cut it into thirds, and so on.

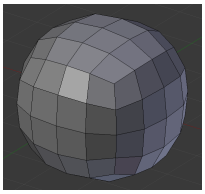
Smoothness
: Displaces subdivisions to maintain approximate curvature. The effect is similar to the way the Subdivision Surface Modifier might deform the mesh.



Mesh before subdividing.



Subdivided with no smoothing.



Subdivided with smoothing of 1.

Quad/Tri Mode
: Forces subdivide to create triangles or quads instead of n-gons (see examples below). This mode doesn't allow the use of *Straight Cut* on quad corners.

Quad Corner Type
: Controls the way quads with only two adjacent selected edges are subdivided.

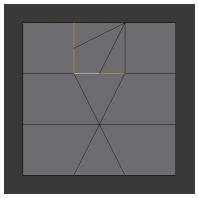
Fan:: The quad is subdivided in a fan of four triangles, the common vertex being the one opposite to the selected edges.

Inner Vertices::

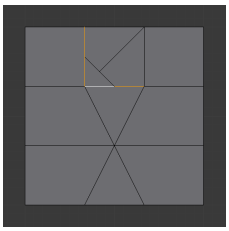
The selected edges are subdivided, then an edge is created between the two new vertices, creating a small triangle. This edge is also subdivided, and the "inner vertex" thus created is linked by another edge to the one opposite to the original selected edges. All this results in a quad subdivided in a triangle and two quads.

Path:: First an edge is created between the two opposite ends of the selected edges, dividing the quad in two triangles. Then, the same goes for the involved triangle as described above.

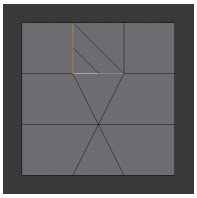
Straight Cut:: The selected edges are subdivided, then an edge is created between the two new vertices, creating a small triangle and n-gon.



Fan cut type.



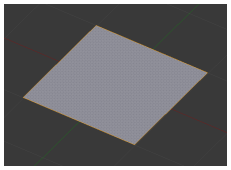
Inner Vertices cut type.



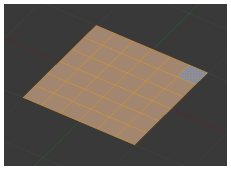
Path cut type.

Fractal

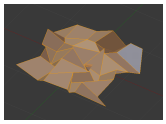
Displaces the vertices in random directions after the mesh is subdivided.



Plane before subdivision.



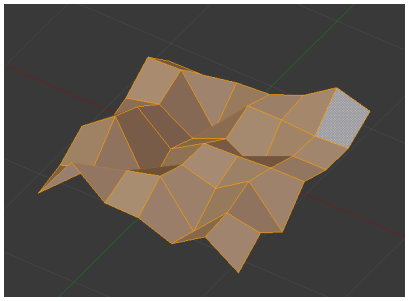
Regular subdivision.



Same mesh with fractal added.

Along Normal

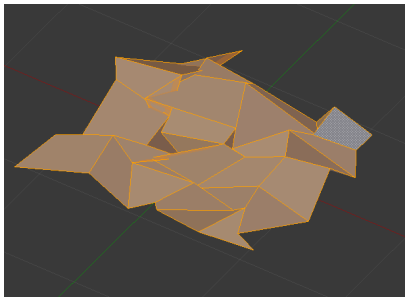
Causes the vertices to move along their normals, instead of random directions.



Along normal set to 1.

Random Seed

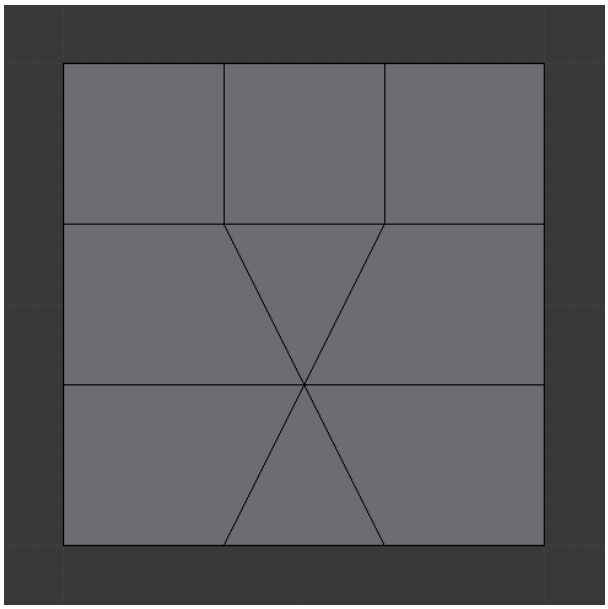
Changes the random seed of the *Fractal* noise function, producing a different result for each seed value.



Same mesh with a different seed value.

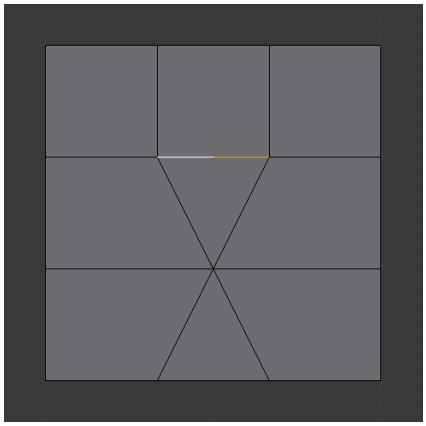
Examples

Below are several examples illustrating the various possibilities of the *Subdivide* and *Subdivide Multi* tools. Note the selection after subdivision.

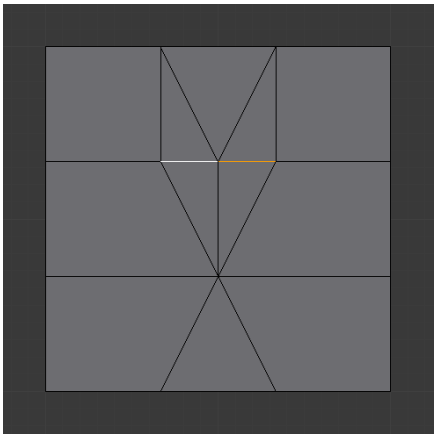


The sample mesh.

One Edge

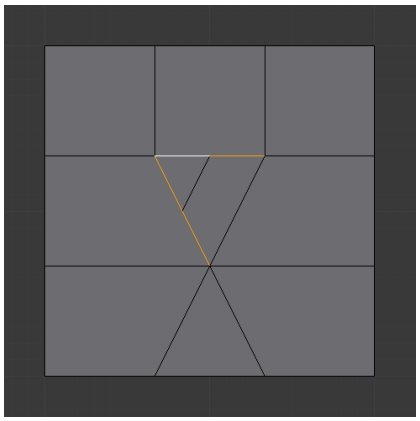


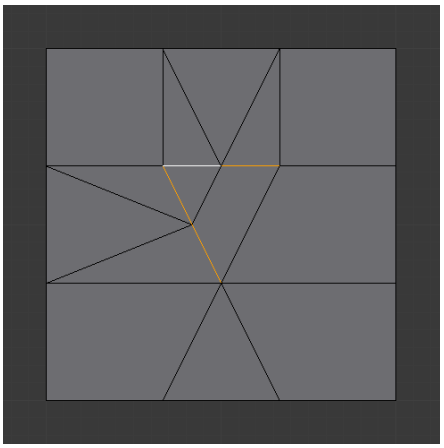
One Edge.



Quad/Tri Mode.

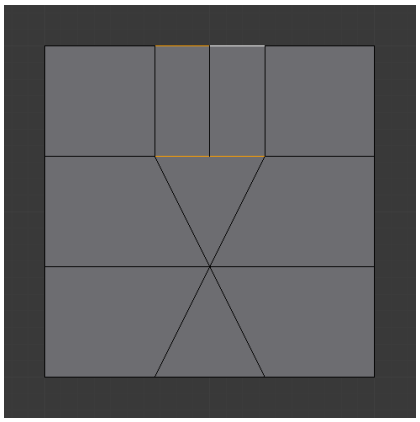
Two Tri Edges

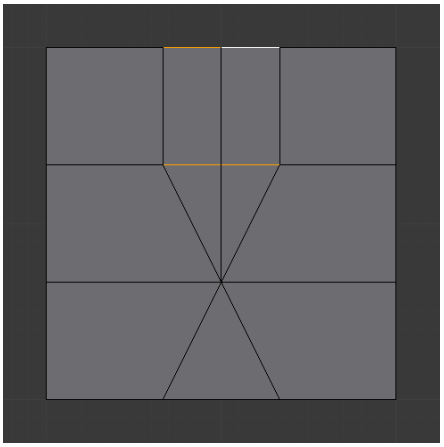




Quad/Tri Mode.

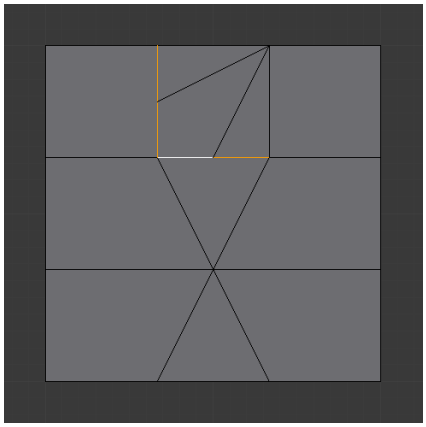
Two Opposite Quad Edges



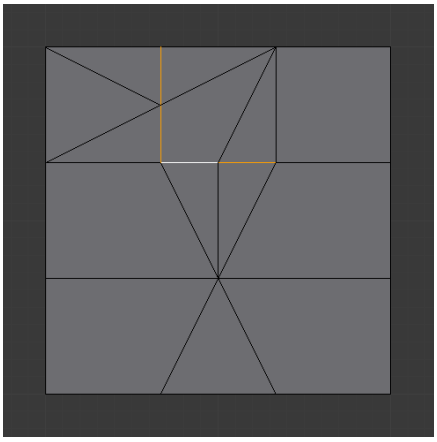


Quad/Tri Mode.

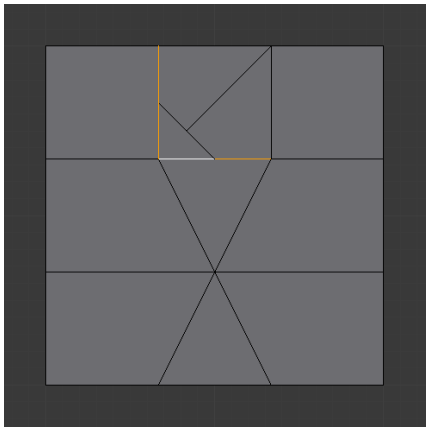
Two Adjacent Quad Edges



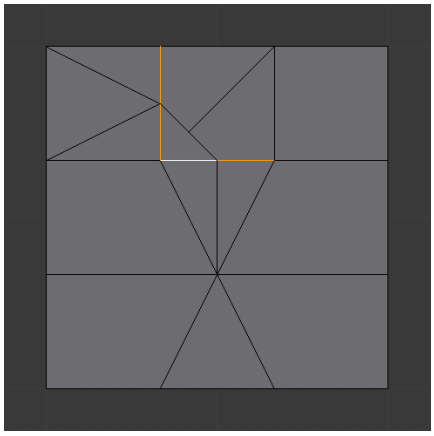
Fan cut type.



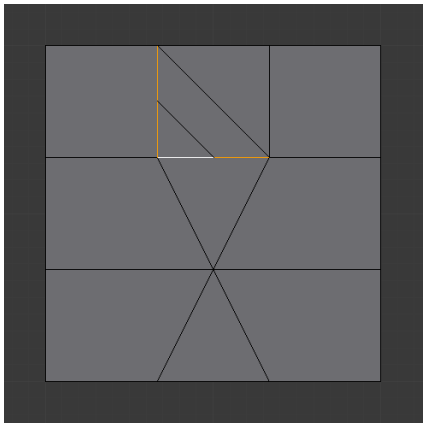
Quad/Tri Mode.

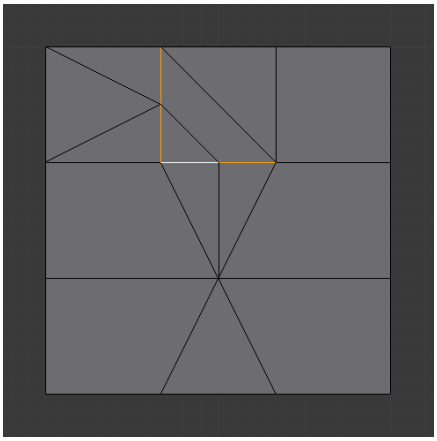


Inner vertices cut type.



Quad/Tri Mode.

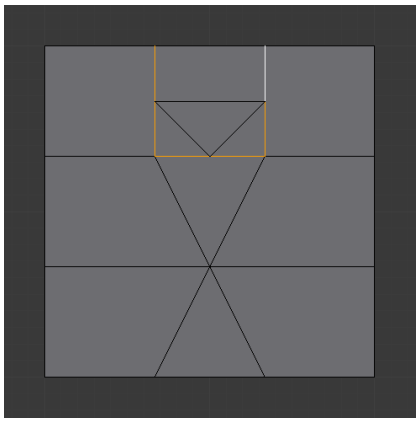


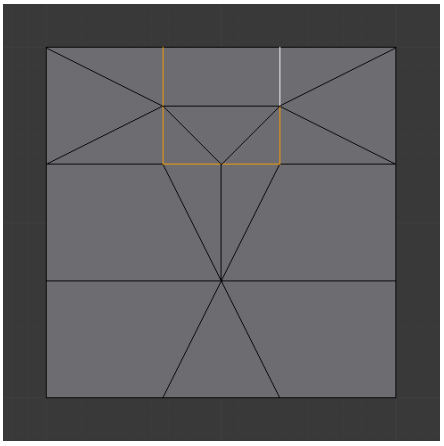


Path cut type.

Quad/Tri Mode.

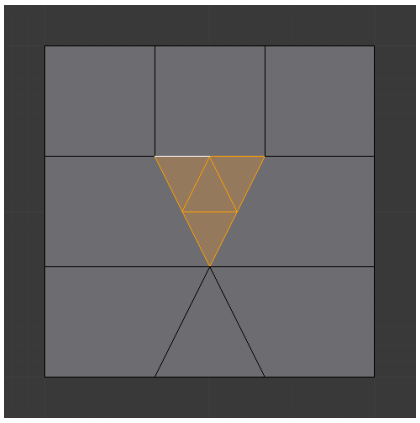
Three Edges

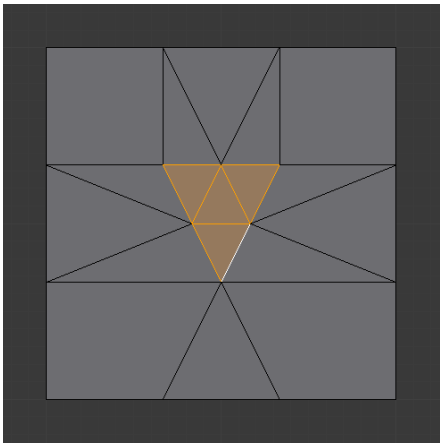




Quad/Tri Mode.

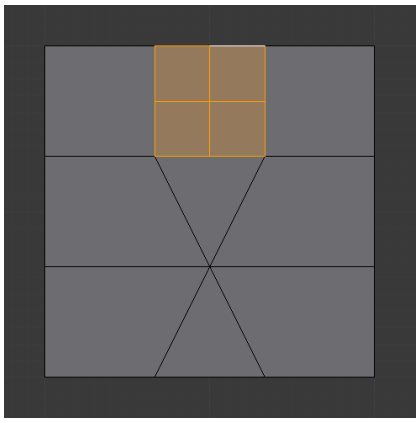
Tri

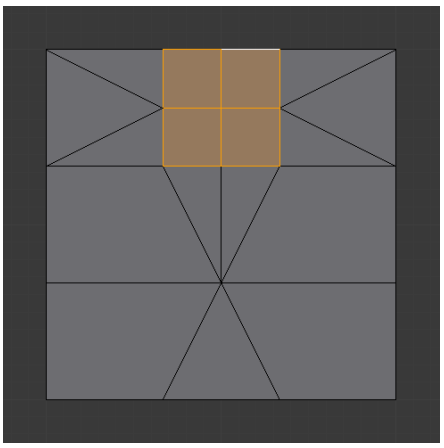




Quad/Tri Mode.

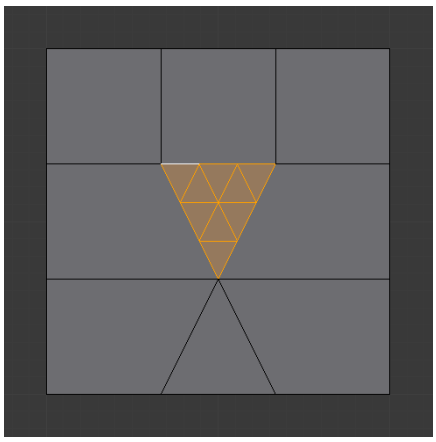
Quad/Four Edges



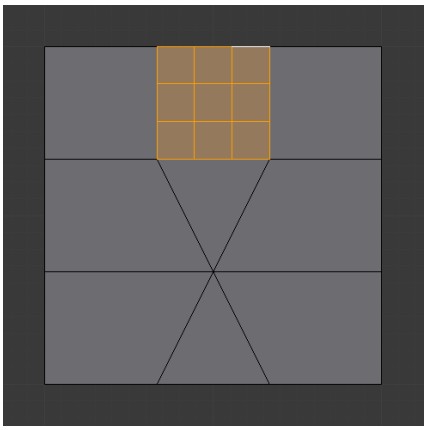


Quad/Tri Mode.

Multiple Cuts



Triangle with two cuts.



Quad with two cuts.

Subdivide Edge-Ring

Reference

Mode::	Edit Mode
Panel::	*Edge ‣ Subdivide Edge-Ring*

Take an edge ring, and subdivide with interpolation options.

Options
: See Bridge Edge Loops.

Un-Subdivide

Reference

Mode:: Edit Mode
Menu:: *Edge ▸ Un-Subdivide*

Un-subdivide functions as the reverse of subdivide by attempting to remove edges that were the result of a subdivide operation. If additional editing has been done after the subdivide operation, unexpected results may occur.

Iterations
: How many subdivisions to remove.

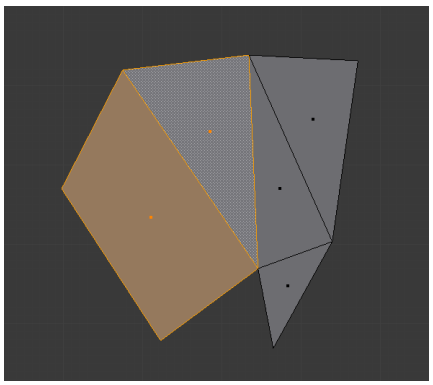
Rotate Edge

Reference

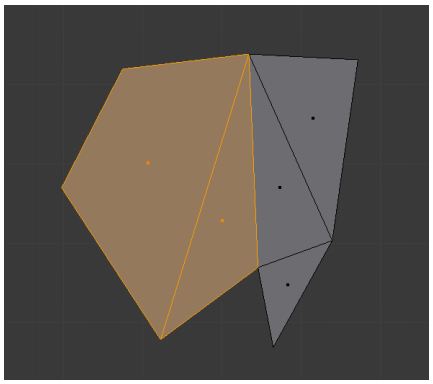
Mode::	Edit Mode
Menu::	*Edge ▸ Rotate Edge CW / Rotate Edge CCW*

Rotating an edge clockwise (CW) or counterclockwise (CCW) spins an edge between two faces around their vertices. This is very useful for restructuring a mesh's topology.

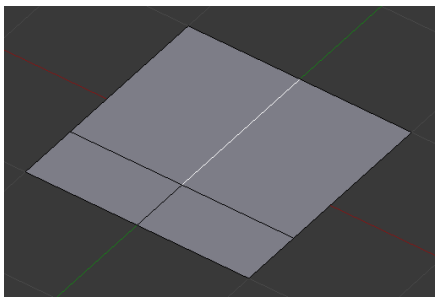
The tool operates on selected edges or the shared edge between selected faces.

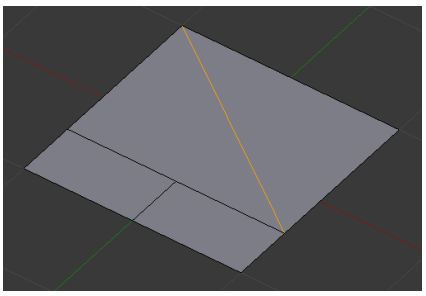


Two adjacent faces selected.



Selected edge rotated.





Selected edge. Edge, rotated CW.

Warning

To rotate an edge based on faces you must select adjacent face pairs, otherwise Blender notifies you with an error message, *"Could not find any selected edges that can be rotated"*. Using either *Rotate Edge CW* or *Rotate Edge CCW* will produce exactly the same results as if you had selected the common edge.

Edge Slide

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Edge Slide*
Shortcut::	`G`, `G`

Slides one or more edges across adjacent faces with a few restrictions involving the selection of edges (i.e. the selection *must* define a valid loop, see below).

Even `E`
: Forces the edge loop to match the shape of the adjacent edge loop. You can flip to the opposite vertex using `F`.

Flipped `F`
: When Even mode is active, this flips between the two adjacent edge loops the active edge loop will match.

Clamp `Alt` or `C`
: Toggle clamping the slide within the edge extents.

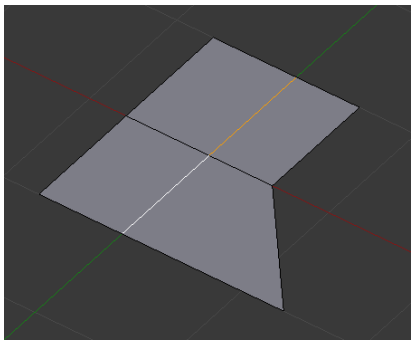
Factor
: Determines the amount of slide performed. Negative values correspond to slides toward one face, while positive ones, refer to the other one. It is also displayed in the 3D Viewport footer.

Mirror Editing
: Lets you propagate the operation to the symmetrical elements of the mesh (if present, in local X direction).

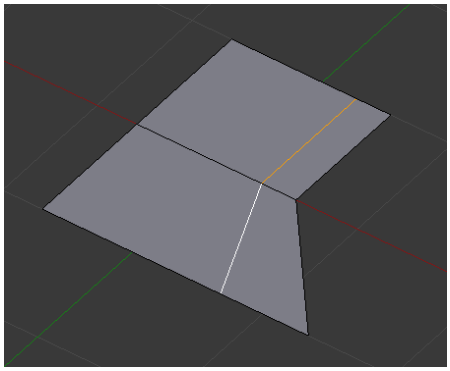
Correct UVs
: Corrects the corresponding UV coordinates, if these exist, to avoid image distortions.

Usage

By default, the position of vertices on the edge loop move as a percentage of the distance between their original position and the adjacent edge loop, regardless of the edges' lengths.



Selected edge loop.

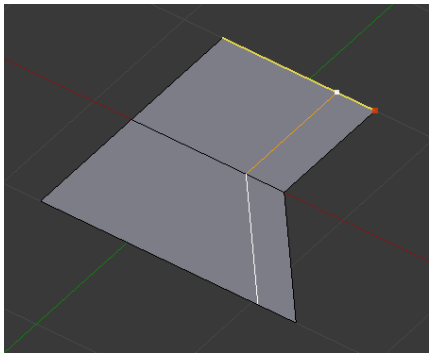


Repositioned edge loop.

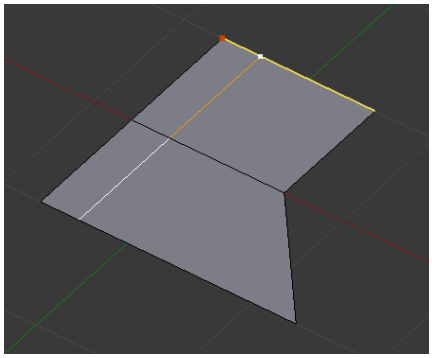
Even Mode

Even mode keeps the shape of the selected edge loop the same as one of the edge loops adjacent to it, rather than sliding a percentage along each perpendicular edge.

In *Even* mode, the tool shows the position along the length of the currently selected edge which is marked in yellow, from the vertex that has an enlarged red marker. Movement of the sliding edge loop is restricted to this length. As you move the mouse the length indicator in the header changes showing where along the length of the edge you are.



Even Mode enabled.



Even Mode with Flip enabled.

Moving the mouse moves the selected edge loop towards or away from the start vertex, but the loop line will only move as far as the length of the currently selected edge, conforming to the shape of one of the bounding edge loops.

Limitations & Workarounds

There are restrictions on the type of edge selections that can be operated upon. Invalid selections are:

Loop Crosses Itself
: This means that the tool could not find any suitable faces that were adjacent to the selected edge(s). An example that shows this is selecting two edges that share the same face. A face cannot be adjacent to itself.

Multiple Edge Loops
: The selected edges are not in the same edge loop, which means they do not have a common edge. You can minimize this error by always selecting edges end-to-end or in a “chain”. If you select multiple edges just make sure they are connected. This will decrease the possibility of getting looping errors.

Border Edges

When a single edge was selected in a single-sided object. An edge loop cannot be found because there is only one face. Remember, edge loops are loops that span two or more faces.

A general rule of thumb is that if multiple edges are selected they should be connected end-to-end such that they form a continuous chain. This is *literally* a general rule because you can still select edges in a chain that are invalid because some of the edges in the chain are in different edge loops.

Offset Edge Slide

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Offset Edge Slide*
Shortcut::	`Shift-Ctrl-R`

Add two edge loops on either side of selected loops.

Cap Endpoint
: Extends the loop by creating triangles at endpoints.

Factor
: Location of the loop relative to the center loop and the outside edge loops.

Even
: Only available for single edge loops. This matches the shape of the edge loop to one of the adjacent edge loops. (See Edge Slide tool for details.)

Flipped
: When Even is enabled, this flips the target edge loop to match. (See Edge Slide tool for details.)

Clamp
: Clamp within the edge extents.

Correct UVs
: Correct UV coordinates when transforming.

Loop Cut and Slide

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Loop Cut and Slide*
Shortcut::	`Ctrl-R`

Loop Cut and Slide splits a loop of faces by inserting a new edge loop intersecting the chosen edge.

Usage

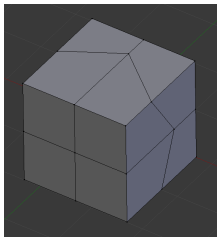
The tool is interactive and has two steps:

1. Pre-Visualizing the Cut

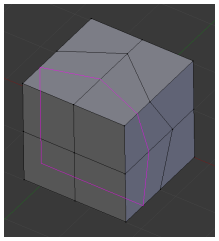
   After the tool is activated, move the cursor over a desired edge. The cut to be made is marked with a magenta colored line as you move the mouse over the various edges. The to be created edge loop stops at the poles (tris and n-gons) where the existing face loop terminates.

2. Sliding the new Edge Loop

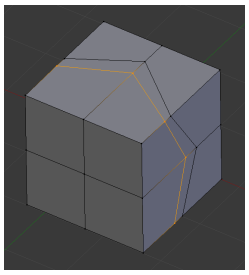
   Once an edge is chosen via `LMB`, you can move the mouse along the edge to determine where the new edge loop will be placed. This is identical to the Edge Slide tool. Clicking `LMB` again confirms and makes the cut at the pre-visualized location, or clicking `RMB` forces the cut to exactly 50%. This step is skipped when using multiple edge loops (see below).



Mesh before inserting edge loop.



Preview of edge loop location.



Interactive placement of edge loop between adjacent loops.

Options

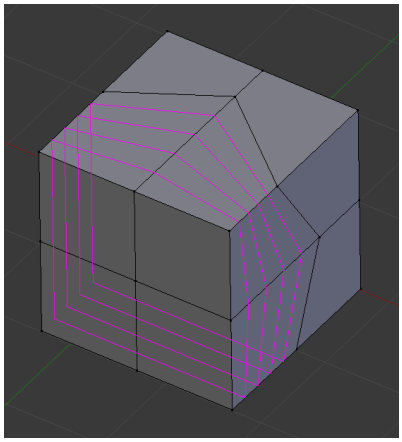
These options are available while the tool is in use, and later in the Adjust Last Operation panel.

Number of Cuts `Wheel` or `PageUp` / `PageDown`

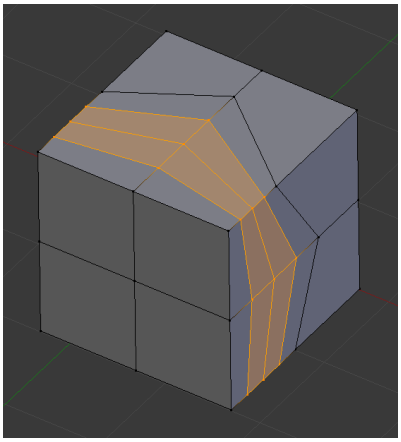
After activating the tool, but before confirming initial loop location, you can increase and decrease the number of cuts to create, by entering a number with the keyboard, scrolling `Wheel` or using `PageUp` and `PageDown`.

Note

When creating multiple loops, these cuts are uniformly distributed in the original face loop, and you will *not* be able to control their positions.



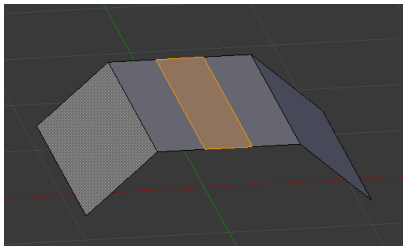
Preview of multiple edge loops.



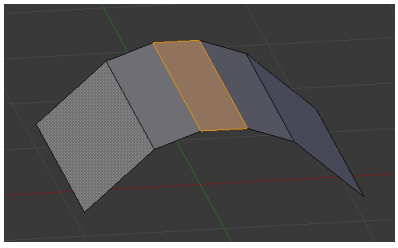
Result of using multiple cuts.

Smoothness `Alt-Wheel`

Smoothing causes edge loops to be placed in an interpolated position, relative to the face it is added to, causing them to be shifted outwards or inwards by a given percentage, similar to the *Subdivide Smooth* tool. When not using smoothing, new vertices for the new edge loop are placed exactly on the preexisting edges. This keeps subdivided faces flat, but can distort geometry, particularly when using Subdivision Surfaces. Smoothing can help maintain the curvature of a surface once it is subdivided.



Added edge loops without smoothing.



Same edge loops, but with smoothing value.

Falloff

Falloff type for *Smoothness*, changes the shape of the profile.

Factor
: Position of the edge loop relative to the surrounding edge loops.

Even `E`
: Only available for single edge loops. This matches the shape of the edge loop to one of the adjacent edge loops. (See Edge Slide tool for details.)

Flip `F`
: When Even is enabled, this flips the target edge loop to match. (See Edge Slide tool for details.)

Clamp
: Clamp within the edge extents.

Correct UVs
: Corrects the corresponding UV coordinates, if these exist, to avoid image distortions.

Edge Data

Edges can have several different properties that affect how certain other tools affect the mesh.

Edge Crease

Reference

Mode::	Edge Mode
Menu::	*Edge ▸ Edge Crease*
Shortcut::	`Shift-E`

This operator interactively sets the Edge Crease amount by moving the mouse (or typing a value with the keyboard). Selecting more than one edge will adjust the mean (average) crease value. A negative value will subtract from the actual crease value, if present. To clear the crease edge property, enter a value of -1.

Edge Bevel Weight

Reference

Mode::	Edge Mode
Menu::	*Edge ▸ Edge Bevel Weight*

Sets the value for the `bevel_weight_edge` attribute, a value between (0.0 to 1.0).

This attribute is used by the Bevel Modifier to control the bevel intensity of the edges.

This operator enters an interactive mode (a bit like transform tools), where by moving the mouse (or typing a value with the keyboard) you can set the bevel weight of selected edges. If more than one edge is selected, this operator alters the average weight of the edges.

See also

Vertex Bevel Weight

Mark Seam & Clear Seam

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Mark Seam/Clear Seam*

These operators set or unset this mark for selected edges. Seams are a way to create separations, "islands", in UV maps. See the UV Mapping section for more details.

Mark Sharp & Clear Sharp

Reference

Mode::	Edit Mode
Menu::	*Edge ▸ Mark Sharp/Clear Sharp*

Signifies the selected edge(s) as being "sharp". This edge attribute can either be set (mark) or unset (clear).

This influences the rendering of Normals to appear flat if smooth shading is enabled for the connecting face or object. This attribute can also be used by many modifiers or operators to mask their effect.

Internally, this uses the sharp edge attribute.

Set Sharpness by Angle

Reference

Mode:: Edit Mode
Menu:: *Edge ▸ Set Sharpness by Angle*

Sets the sharp edge attribute based on the angle between neighboring faces.

Angle
: Maximum angle between face normals that will be considered as smooth.

Extend
: Add new sharp edges without clearing existing sharp edges.

Face Operators

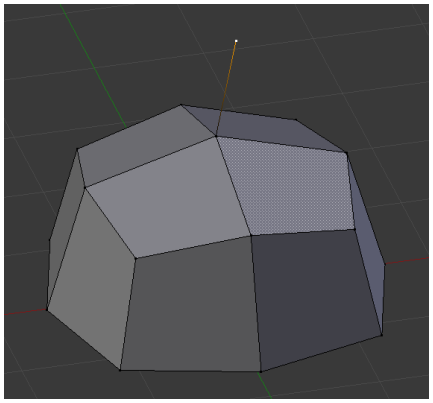
- Extrude Faces
- Extrude Faces Along Normals
- Extrude Individual Faces
- Inset Faces
  - Options
- Poke Faces
- Triangulate Faces
- Triangles to Quads
- Solidify Faces
- Wireframe
- Fill
- Grid Fill
- Beautify Faces
- Intersect (Knife)
- Intersect (Boolean)
- Weld Edges into Faces
- Shade Smooth & Flat
  - Shade Smooth
  - Shade Flat
- Face Data
  - Rotate Colors
  - Reverse Colors
  - Rotate UVs
  - Reverse UVs
  - Flip Quad Tessellation
  - Mark Freestyle Face
  - Clear Freestyle Face

Extrude Faces

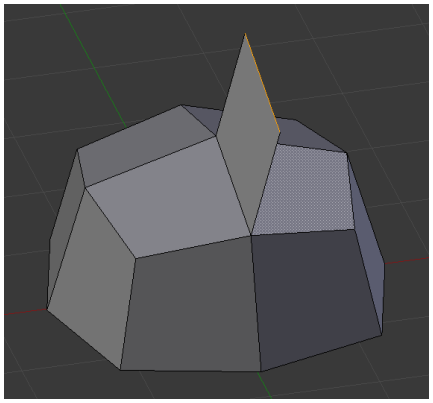
Reference

Mode::	Edit Mode
Menu::	*Face* ▸ *Extrude Faces*, *Mesh* ▸ *Extrude* ▸ *Extrude Faces*
Shortcut::	E

Extrude Faces duplicate faces, while keeping the new geometry connected with the original vertices.



Single vertex extruded.

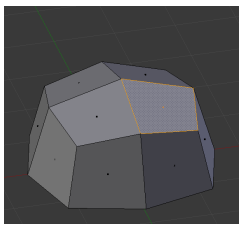


Single edge extruded.

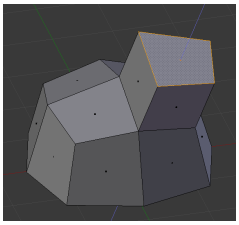
This tool is of paramount importance for creating new geometry. It allows you to create parallelepipeds from rectangles and cylinders from circles, as well as easily creating such things as tree limbs.

The axis on which faces are extruded along can be set interactively. Faces are extruded by default along their averaged normal. The extrusion can be limited to a single axis by specifying an axis; see Axis Locking.

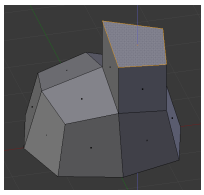
The extrude tools differentiate in how the new geometry is connected in itself. Only the border loop gets extruded. The inner region of the selection gets moved unchanged with the extrusion.



Selected face.



During extrude.



Set to Z axis.

Flip Normals
: Only the *normals* of the new faces created from the extrusion will be flipped.

Dissolve Orthogonal Edges
: Removes and connects edges whose faces form a flat surface and intersect new edges.

Orientation
: Aligns the transformation axes to a specified orientation constraint. See Transform Orientations for more information.

Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

> **Note**
>
> Even with the *Proportional Size* set to it's minimum, it will extrude the selected face as well as the new geometry and they will be layered on top of each other.

Extrude Faces Along Normals

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Extrude Along Normals*
Menu::	*Face* ▸ *Extrude Faces Along Normals*, *Mesh* ▸ *Extrude* ▸ *Extrude Faces Along Normals*
Shortcut::	E

Extrusion and offset will be locked in to only move along the local normals of the selected mesh.

Flip Normals
: Only the *normals* of the new faces created from the extrusion will be flipped.

Dissolve Orthogonal Edges
: Removes and connects edges whose faces form a flat surface and intersect new edges.

Offset
: Amount to move geometry along the normals.

Offset Even
: The length of the new edges will be uniform.

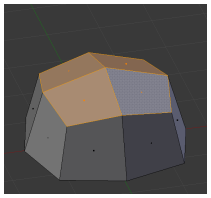
Proportional Editing
: The extruded face will affect nearby geometry. See Proportional Editing for a full reference.

Extrude Individual Faces

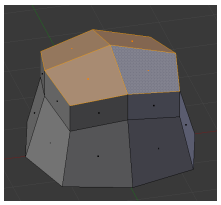
Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Extrude Individual*
Menu::	*Face* ▸ *Extrude Individual Faces*, *Mesh* ▸ *Extrude* ▸ *Extrude Individual Faces*

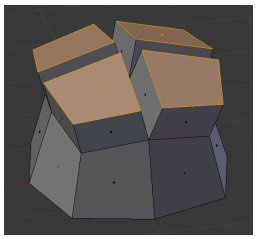
This tool allows you to extrude a selection of multiple faces as individuals, instead of as a region. The faces are extruded along their own normals, rather than their average. This has several consequences: first, “internal” edges (i.e. edges between two selected faces) are no longer deleted (the original faces are).



Selection of multiple faces.



Extruded using extrude region.



Extruded using Extrude Individual.

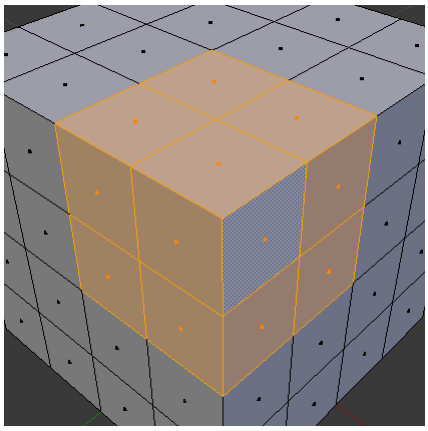
Inset Faces

Reference

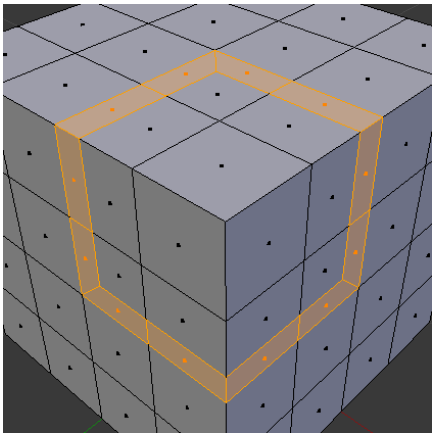
Mode::	Edit Mode
Tool::	*Toolbar ▸ Inset Faces*
Menu::	*Face ▸ Inset Faces*
Shortcut::	`I`

This tool takes the currently selected faces and creates an inset of them, with adjustable thickness and depth. Think of it as like creating an edge loop, but relative to the selected edges, even in complex meshes.

The tool is modal, such that when you activate it, you may adjust the thickness with your mouse position. You may also adjust the depth of the inset during the modal operation by holding `Ctrl`.



Selection to inset.



Result of inset with *Select Outer* enabled.

Options

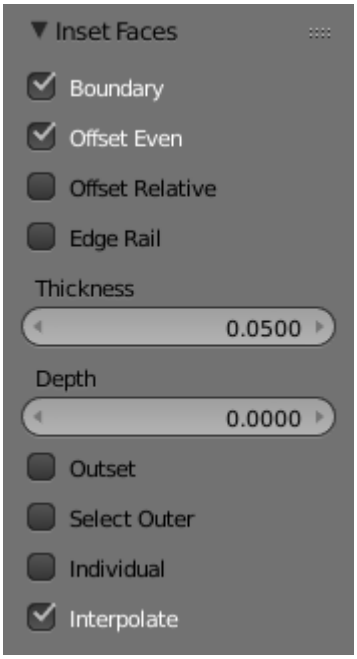
Boundary `B`
: Determines whether open edges will be inset or not.

Offset Even
: Scale the offset to give a more even thickness.

Offset Relative
: Scale the offset by lengths of surrounding geometry.

Edge Rail
: Created vertices slide along the original edges of the inner geometry, instead of the normals.

Thickness
: Set the size of the offset.



Inset operator options.

Depth `Ctrl`
: Raise or lower the newly inset faces to add depth.

Outset `O`
: Create an outset rather than an inset. Causes the geometry to be created surrounding selection (instead of within).

Select Outer
: Toggle which side of the inset is selected after the operation.

Individual `I`
: By default the Inset tool operates on the region around selected faces, but with this option each selected face can be inset on its own.

Interpolate
: Interpolate mesh data: e.g. UVs, Color Attribute's colors, weights, etc.

Poke Faces

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Poke Faces*

Splits each selected faces into a triangle fan, creating a new center vertex and triangles between the original face edges and new center vertex. The *Offset* can be used to make spikes or depressions.

Poke Offset
: Offset the new center vertex along the face normal.

Offset Relative
: Multiply the Offset by the average length from the center to the face vertices.

Poke Center
: Computes the center of a face.

  Weighted Median:: Using the mean average weighted by edge length.
  Median:: Using the mean average.
  Bounds:: Uses center of bounding box.

Triangulate Faces

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Triangulate Faces*
Shortcut::	`Ctrl-T`

This tool converts each of the selected faces (whether it be quads or n-gons) to triangular faces. See the Triangulate Modifier.

Triangles to Quads

Reference

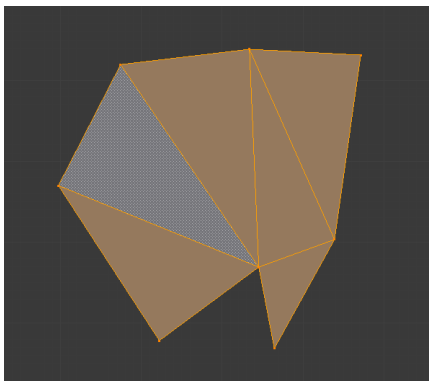
Mode::	Edit Mode
Menu::	*Face ▸ Triangles to Quads*
Shortcut::	`Alt-J`

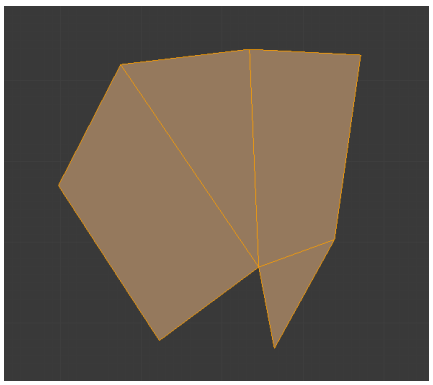
This tool converts the selected triangles into quads by taking adjacent triangles and removing the shared edge to create a quad, based on a threshold. This tool can be applied on a selection of multiple triangles.

This means you can select the entire mesh and convert triangles that already form square shapes – to be converted into quads, without having to concern yourself with individual faces.

Alternatively you can force this operation selecting a pairs of faces (see hint below for other ways of joining).

To create a quad, this tool needs at least two adjacent triangles. If you have an even number of selected triangles, it is also possible not to obtain only quads. In fact, this tool tries to create most even rectangular quads from the given triangles, which means some triangles could remain.





Before converting tris to quads.	After converting tris to quads.

Max Angle
: This value, between (0 to 180), controls the threshold for this tool to work on adjacent triangles. With a threshold of 0.0, it will only join adjacent triangles that form a perfect rectangle (i.e. right-angled triangles sharing their hypotenuses). Larger values are required for triangles with a shared edge that is small, relative to the size of the other edges of the triangles.

Compare UVs
: Prevents the union of triangles that are not also adjacent in the active UV map.

Compare Color Attributes
: Prevents the union of triangles that do not have matching Color Attributes.

Compare Sharp
: Prevents the union of triangles that share an edge marked as sharp.

Compare Materials
: Prevents the union of triangles that do not have the same material assigned.

Hint

When isolated groups of faces are selected, they can be combined with Create Face or Dissolve Faces; this is not limited to quads.

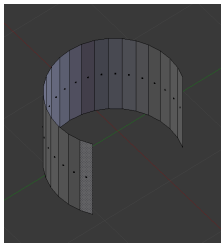
Solidify Faces

Reference

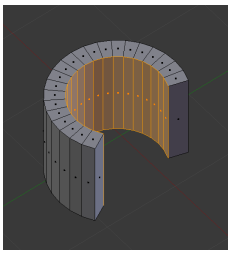
Mode:: Edit Mode
Menu:: *Face ▸ Solidify Faces*

This takes a selection of faces and solidifies them by extruding them uniformly to give volume to a Non-manifold surface. This is also available as a Modifier. After using the tool, you can set the offset distance in the Adjust Last Operation panel.

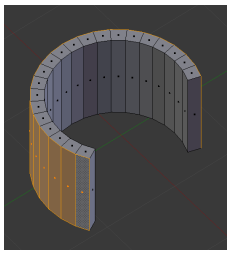
Thickness
: Amount to offset the newly created surface. Positive values offset the surface inward relative to the normals direction. Negative values offset outward.



Mesh before solidify operation.



Solidify with a positive thickness.



Solidify with a negative thickness.

Wireframe

Reference

Mode::	Edit Mode
Menu::	*Face* ▸ *Wireframe*

The Wireframe tool makes a wireframe from faces by turning edges into wireframe tubes, similar to the Wireframe Modifier.

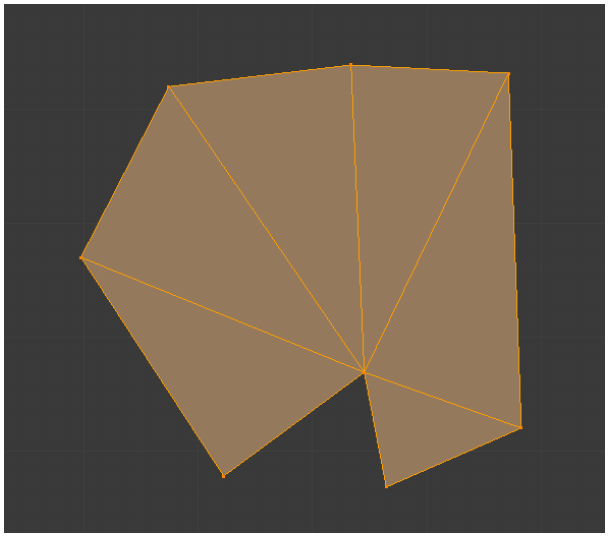
Fill

Reference

Mode::	Edit Mode
Menu::	*Face* ▸ *Fill*
Shortcut::	`Alt-F`

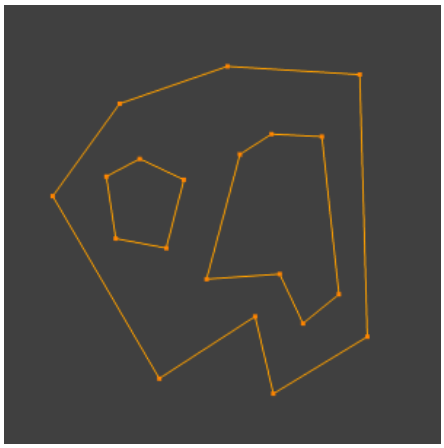
The *Fill* option will create *triangular* faces from any group of selected edges or vertices, as long as they form one or more complete perimeters.

Beauty
: Arrange the new triangles nicely.



Filled using fill.

Note, unlike creating n-gons, *Fill* supports holes.



A closed perimeter of edges with holes.



Filled using fill.

Grid Fill

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Grid Fill*

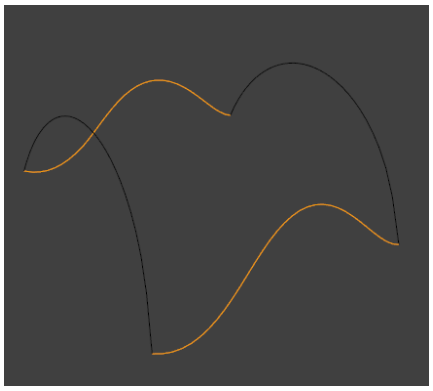
Grid Fill uses a pair of connected edge loops or a single, closed edge loop to fill in a grid that follows the surrounding geometry.

The best predictable result can be achieved if you select two opposite edge loops with an equal number of vertices. When a single, closed edge loop is selected, the Span/Offset options allows you to adjust the way two opposite edge loops are detected from one closed edge loop.

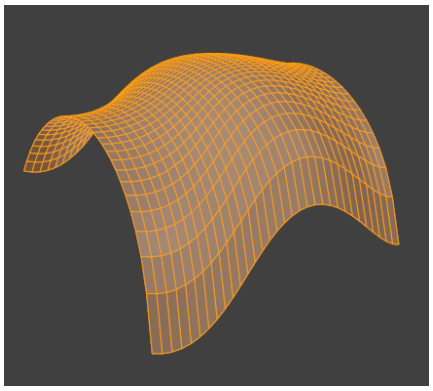
Span
: Specifies the number of columns in the grid.

Offset
: Defines the vertex that is considered to be the corner of the grid, by default, it's the active vertex. The Offset allows you to rotate the grid lines.

Simple Blending
: Use a simple interpolation algorithm to generate grid vertices from boundary loops, which does not tries to maintain the shape, useful for flat surfaces or times when keeping the shape gives odd results.



Input.



Grid Fill result.

Beautify Faces

Reference

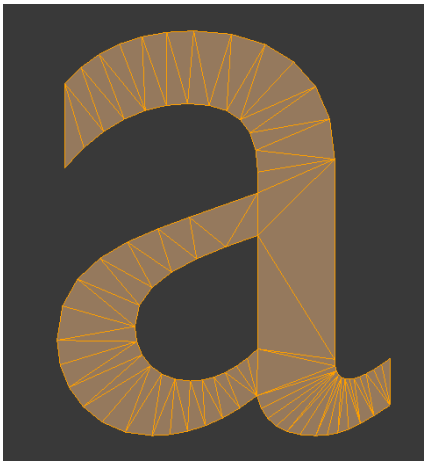
Mode:: Edit Mode
Menu:: *Face* ▸ *Beautify Faces*

Beautify Faces works only on selected existing faces. It rearrange selected triangles to obtain more “balanced” ones (i.e. less long thin triangles).

Max Angle
: An angle delimiter option to limit edge rotation to flat surfaces.



Text converted to a mesh.



Result of Beautify Faces.

Intersect (Knife)

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Intersect (Knife)*

The Intersect tool lets you cut intersections into geometry. It is a bit like the Boolean tool, but, does not calculate interior/exterior geometry. Faces are split along the intersections, leaving new edges selected.

Source

Self Intersect:: Operate on the overlapping geometry of the mesh.

Selected/Unselected:: Operate between the selected and unselected geometry.

Separate Mode

All:: Splits the geometry at the new edge.

Cut:: Keep each side of the intersection separate without splitting the faces in half.

Merge:: Merge all the geometry from the intersection.

Solver

Algorithm used to calculate the intersections.

Fast:: Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.

Merge Threshold

Tolerance for close faces to be considered touching. It may be useful to increase this when some intersections aren't detected that should be and when extra geometry is being

created because edges aren't detected as overlapping.

Warning
A threshold approaching the size of faces may cause very slow calculation, in general keep this value small.

Exact:: Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Fast Solver*.

Intersect (Boolean)

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Intersect (Boolean)*

Performs Boolean operations with the selection on the unselected geometry. While the Boolean Modifier is useful for non-destructive edits, access to these operations with a tool in Edit Mode can be useful to quickly perform edits.

Boolean Operation

Intersect:: Opposite of *Difference* (everything *inside* of the target mesh is kept).
Union:: The target mesh is added to the modified mesh.
Difference:: The target mesh is subtracted from the modified mesh (everything *outside* of the target mesh is kept).

Solver

Algorithm used to calculate the Boolean intersections.

Fast:: Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.

Merge Threshold

Tolerance for close faces to be considered touching. It may be useful to increase this when some intersections aren't detected that should be and when extra geometry is being created because edges aren't detected as overlapping.

Warning

A threshold approaching the size of faces may cause very slow calculation, in general keep this value small.

Exact:: Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Fast Solver*.

Swap

Changes the order of the operations when using *Difference* to determine which side is kept.

Self Intersection

Correctly calculates cases when one or both operands have self-intersections, this involves more calculations making it slower.

Weld Edges into Faces

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Weld Edges into Faces*

A tool to split selected faces by loose wire edges. This can be used in a similar way to the Knife tool, but the edges are manually setup first.

Shade Smooth & Flat

The appearance of the mesh edges are determined to be evened out or well defined within the 3D Viewport and render. In Edit Mode, individual faces can be selected to determine which faces are smoothed or flattened.

Note

Both Shade Smooth and Flat are also available in Object Mode and function the same way.

Shade Smooth

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Shade Smooth*

Using interpolated vertex normals, the mesh faces will blur at the edges and appear smooth.

Shade Flat

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Shade Flat*

Face normals are displayed evenly, because of this all the edges of the selected mesh will be easily visible.

Tip

Use the Edge Split modifier and Smooth by Angle Smooth to balance between smooth surfaces and sharp edges.

Face Data

Rotate Colors

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Face Data ▸ Rotate Colors*

Rotates the Color Attribute's colors inside faces either clockwise or counterclockwise.

Reverse Colors

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Face Data ▸ Reverse Colors*

Flips the direction of Color Attribute's colors inside the selected faces.

Rotate UVs

Reference

Mode:: Edit Mode
Menu:: *Face ▸ Face Data ▸ Rotate UVs*

See Rotate UVs.

Reverse UVs

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Face Data ▸ Reverse UVs*

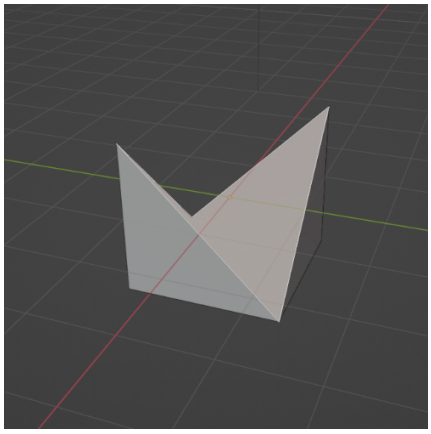
See Reverse UVs.

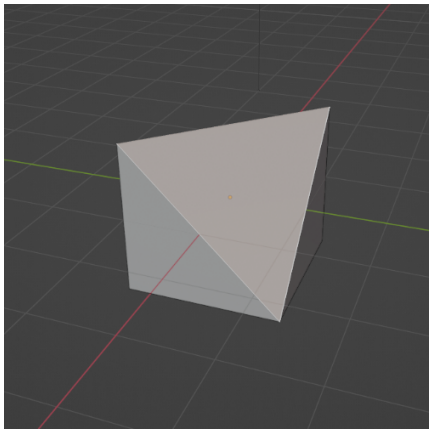
Flip Quad Tessellation

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Face Data ▸ Flip Quad Tessellation*

Internally, all quads are Tessellated into 2 triangles, this operator swaps which way the quad is split into triangles.





Mark Freestyle Face

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Face Data ▸ Mark Freestyle Face*

Todo.

Clear Freestyle Face

Reference

Mode::	Edit Mode
Menu::	*Face ▸ Face Data ▸ Clear Freestyle Face*

Todo.

UV Operators

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header ▸ UV*
Shortcut::	U

Blender offers several ways of mapping UVs. The simpler projection methods use formulas that map 3D space onto 2D space, by interpolating the position of points toward a point/axis/plane through a surface. The more advanced methods can be used with more complex models, and have more specific uses.

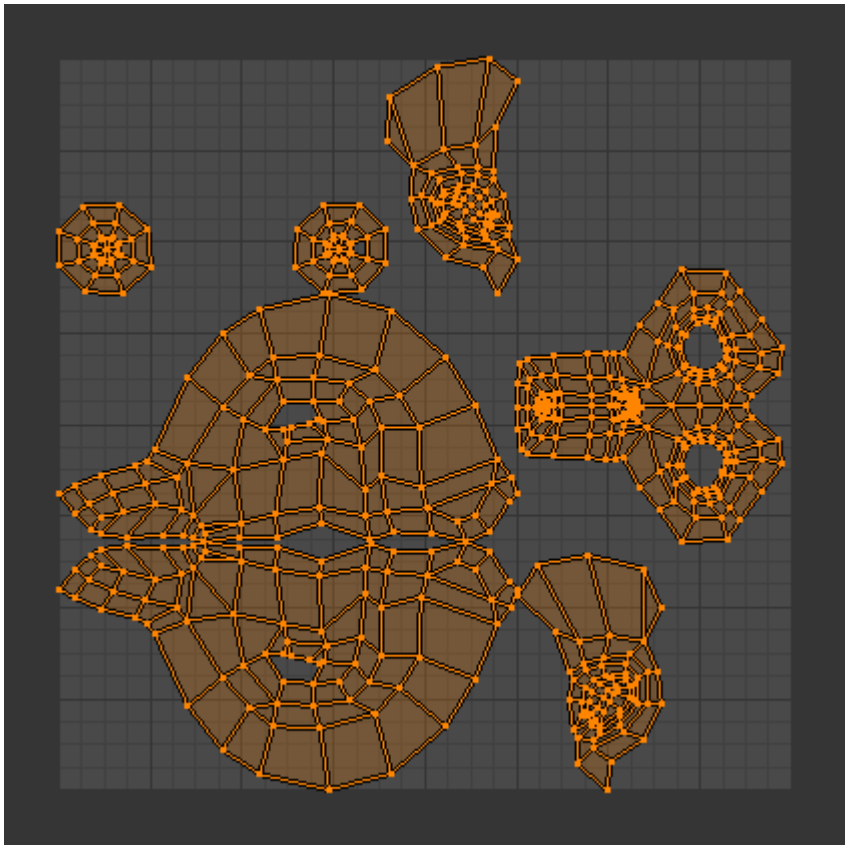
Unwrap

Reference

Editor::	3D Viewport and UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Unwrap*
Shortcut::	U

Flattens the mesh surface by cutting along seams. Useful for organic shapes.

Begin by selecting all the faces you want to unwrap. In the 3D Viewport, select *UV ▸ Unwrap* or U and select *Unwrap*. You can also do this from the UV Editor with *UV ▸ Unwrap* or U. This method will unwrap all faces and reset previous work. The UVs menu will appear in the UV Editor after unwrapping has been performed once.



Result of unwrapping Suzanne.

This operation unwraps the faces of the object to provide the “best fit” scenario based on how the faces are connected and will fit within the image, and takes into account any seams within the selected faces. If possible, each selected face gets its own different area of the image and is not overlapping any other faces UVs. If all faces of an object are selected, then each face is mapped to a part of the image.

Tip

A face’s UV image texture only has to use *part* of the image, not the *whole* image. Also, portions of the same image can be shared by multiple faces. A face can be mapped to less and less of the total image.

Options

The Adjust Last Operation panel allows fine control over how a mesh is unwrapped:

Method

Angle Based:: Uses Angle Based Flattening (ABF). This method gives a good 2D representation of a mesh.

Conformal:: Uses Least Squares Conformal Mapping (LSCM). This usually results in a less accurate UV mapping than Angle Based, but performs better on simpler objects.

Minimum Stretch:: Uses Scalable Locally Injective Mapping (SLIM). This tries to balance minimizing area distortion and minimizing angle distortion.

Fill Holes

Activating Fill Holes will prevent overlapping from occurring and better represent any holes in the UV regions.

Correct Aspect

Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to the Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Use Subdivision Surface

Map UVs taking vertex position after Subdivision Surface Modifier into account.

Iterations

The Minimum Stretch method is iterative, where each iteration reduces the distortion more. This option says how many iterations to use before stopping.

Allow Flips

When using the Minimum Stretch method this option allows faces to flip, which sometimes results in less distortion when there are pins.

Importance Weights

The Minimize Stretch method has a feature that allows a user-specified vertex group to control the relative amount of area used by different parts of the unwrapped map. Vertices with higher weights will mark portions of the mesh whose adjacent UV map faces should be stretched larger than smaller-weight areas. When this option is chosen, there are two additional options to control this:

Attribute:: The name of the vertex group with the weights to be used.
Factor:: A global factor to multiply all the weights. A bigger number will result in a more exaggerated difference between high-weight and low-weight areas.

Margin Method
The method to use when calculating the empty space between islands.

Scaled:: Use scale of existing UVs to multiply margin.
Add:: Simple method, just add the margin.
Fraction:: Precisely specify the fraction of the UV unit square for margin. (Slower than other two methods.)

Margin
The scale for the empty space between islands.

Smart UV Project

Reference

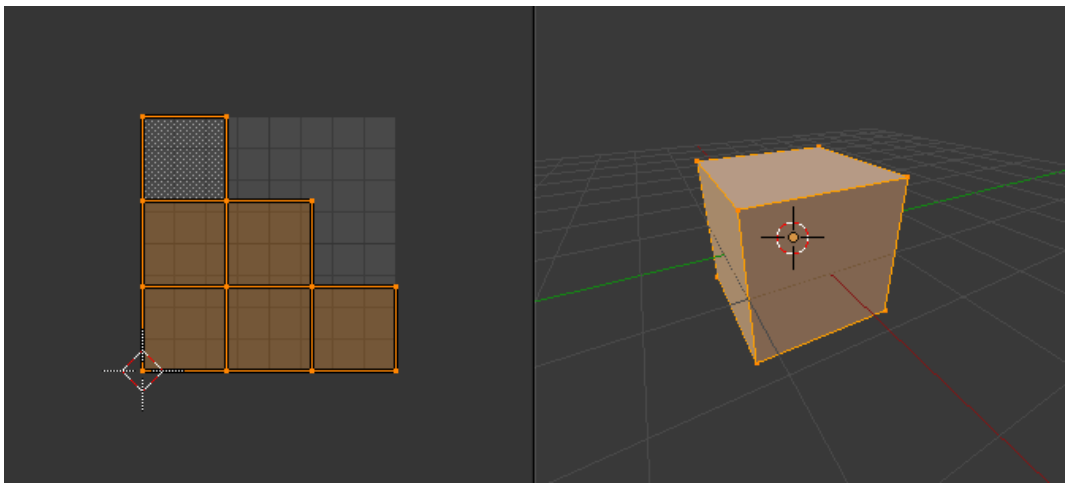
Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *UV ▸ Smart UV Project*
Shortcut:: U

Smart UV Project, cuts a mesh based on an angle threshold (angular changes in your mesh). This gives you fine control over how automatic

seams are created. It is a good method for simple and complex geometric forms, such as mechanical objects or architecture.

This algorithm examines the shape of your object, the selected faces and their relation to one another, and creates a UV map based on this information and settings that you supply.

In the example below, the Smart Mapper mapped all of the faces of a cube to a neat arrangement of three sides on top, three sides on the bottom, for all six sides of the cube to fit squarely, just like the faces of the cube.



Smart UV project on a cube.

For more complex mechanical objects, this operator can quickly and easily create a regular and straightforward UV layout.

Options

The Adjust Last Operation panel allows fine control over how a mesh is unwrapped:

Angle Limit
: This controls how faces are grouped: a higher limit will lead to many small groups but less distortion, while a lower limit will create fewer groups at the expense of more distortion.

Margin Method
: The method to use when calculating the empty space between islands.

Scaled:: Use scale of existing UVs to multiply margin.
Add:: Simple method, just add the margin.
Fraction:: Precisely specify the fraction of the UV unit square for margin. (Slower than other two methods.)

Rotation Method
Axis-aligned:: Rotated to a minimal rectangle, either vertical or horizontal.
Axis-aligned (Horizontal):: Rotate islands to be aligned horizontally.
Axis-aligned (Vertical):: Rotate islands to be aligned vertically.

Island Margin
: This controls how tightly the UV islands are packed together. A higher number will add more space between islands.

Area Weight
: Weight projection's vector by faces with larger areas.

Correct Aspect
: Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to the Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Scale to Bounds
: If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Lightmap Pack

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *UV ▸ Lightmap Pack*
Shortcut:: U

Lightmap Pack takes each of a mesh's faces, or selected faces, and packs them into the UV bounds. Lightmaps are used primarily in realtime rendering, where lighting information is baked onto texture maps, when it is needed to use as much UV space as possible. It has several options that appear in the Adjust Last Operation panel:

Options

Selection
Selected Faces:: Only unwraps the selected faces.
All Faces:: Unwraps the whole mesh.

Share Texture Space
This is useful if mapping more than one mesh. It attempts to fit all of the objects' faces in the UV bounds without overlapping.

New UV Map
If mapping multiple meshes, this option creates a new UV map for each mesh. See UV Maps.

Pack Quality
Pre-packing before the more complex Box packing.

Margin
This controls how tightly the UV islands are packed together. A higher number will add more space between islands.

Follow Active Quads

Reference

Editor:: 3D Viewport

Mode:: Edit Mode
Menu:: *UV ▸ Follow Active Quads*
Shortcut:: U

Extrapolate UV's based on the active quad by following continuous face loops, even if the mesh face is irregularly-shaped.

Note

For a clean 90-degree unwrap it's typically best to first make sure the quad a rectangle in UV space.

Otherwise any distortion in the active UV is extended which doesn't result in a useful grid-layout.

Note

The resulting unwrap is not clamped within the UV bounds, you may wish to scale down the active quad's UV's so the result is in a usable range.

Options

Edge Length Mode

Method to space UV edge loops.

Even:: Space all UVs evenly, where the shape of the quad in the 3D viewport is ignored.

Length:: Each face's UV's are calculated based on the edge length.
While this minimizes distortion, adjacent loops may become disconnected.

Length Average:: Average space UVs edge length of each loop.

This has the benefit of minimizing distortion, while keeping UV's connected.

Cube Projection

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*UV ▸ Cube Projection*
Shortcut::	U

Cube Projection maps the mesh onto the faces of a cube, which is then unfolded. It projects the mesh onto six separate planes, creating six UV islands. In the UV editor, these will appear overlapped, but can be moved. See Editing UVs.

Options

Cube Size
: Set the size of the cube to be projected onto.

Correct Aspect
: Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to the Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Clip to Bounds
: Any UVs that lie outside the (0 to 1) range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds
: If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Cylinder Projection

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*UV ▸ Cylinder Projection*
Shortcut::	U

Normally, to unwrap a cylinder (tube) as if you slit it lengthwise and folded it flat, Blender wants the view to be vertical, with the tube standing "up". Different views will project the tube onto the UV map differently, skewing the image if used. However, you can set the axis on which the calculation is done manually.

Options

Direction

View on Poles::	Use when viewing from the top (at a pole) by using an axis that is straight down from the view.
View on Equator::	Use if view is looking at the equator, by using a vertical axis.
Align to Object::	Uses the object's transform to calculate the axis.

Align

How to determine rotation around the pole.

Polar ZX::	Polar 0 is on the X axis.
Polar ZY::	Polar 0 is on the Y axis.

Pole

How to handle faces at the poles.

Pinch::	UVs are pinched at the poles.
Fan::	UVs are fanned at the poles.

Preserve Seams

Separate projections by islands isolated by seams.

Radius
: The radius of the cylinder to use.

Correct Aspect
: Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to a Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Clip to Bounds
: Any UVs that lie outside the (0 to 1) range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds
: If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Sphere Projection

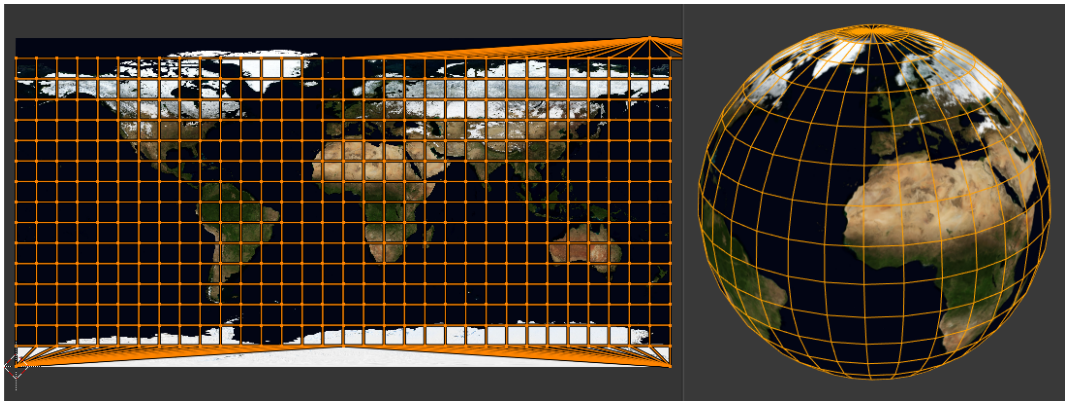
Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*UV ▸ Sphere Projection*
Shortcut::	U

Spherical mapping is similar to cylinder but the difference is that a cylindrical mapping projects the UVs on a plane toward the cylinder shape, while a spherical map takes into account the sphere's curvature, and each latitude line becomes evenly spaced. *Sphere Projection* is useful for spherical shapes, like eyes, planets, etc.

Recall the opening cartographer's approaching to mapping the world? Well, you can achieve the same here when unwrapping a sphere from different points of view. Normally, to unwrap a sphere, view the sphere with the

poles at the top and bottom. After unwrapping, Blender will give you an equirectangular projection; the point at the equator facing you will be in the middle of the image. A polar view will give a very different but common projection map. Using an equirectangular projection map of the earth as the UV image will give a good planet mapping onto the sphere.



Using an equirectangular image with a Sphere Projection.

Options

Direction
: Direction of the sphere.

View on Poles::	Use when viewing from the top (at a pole) by using an axis that is straight down from the view.
View on Equator::	Use if view is looking at the equator, by using a vertical axis.
Align to Object::	Uses the object's transform to calculate the axis.

Align
: Select which axis is up.

Polar ZX::	Polar 0 is on the X axis.
Polar ZY::	Polar 0 is on the Y axis.

Pole
: How to handle faces at the poles.

Pinch:: UVs are pinched at the poles.
Fan:: UVs are fanned at the poles.

Preserve Seams
: Separate projections by islands isolated by seams.

Correct Aspect
: Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to a Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Clip to Bounds
: Any UVs that lie outside the (0 to 1) range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds
: If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Project from View

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *UV ▸ Project from View*
Shortcut:: U

Project from View takes the current view in the 3D Viewport and flattens the mesh as it appears. Use this option if you are using a picture of a real object as a UV Texture for an object that you have modeled. You will get stretching in areas where the model recedes away from you.

Options

Orthographic

Apply an orthographic projection.

Camera Bounds
: Map UVs to the camera region taking resolution and aspect into account

Correct Aspect
: Map UVs will take the image's aspect ratio into consideration. If an image has already been mapped to a Texture Space that is non-square, the projection will take this into account and distort the mapping to appear correctly.

Clip to Bounds
: Any UVs that lie outside the (0 to 1) range will be clipped to that range by being moved to the UV space border it is closest to.

Scale to Bounds
: If the UV map is larger than the (0 to 1) range, the entire map will be scaled to fit inside.

Project from View (Bounds)

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*UV ▸ Project from View (Bounds)*
Shortcut::	U

The same as Project from View, but with *Scale to Bounds* activated by default.

Reset

Reference

Editor::	3D Viewport and UV Editor

Mode::	Edit Mode
Menu::	*UV ▸ Reset*
Shortcut::	U

Reset UVs maps each face to fill the UV grid, giving each face the same mapping.

If you want to use an image that is tileable, the surface will be covered in a smooth repetition of that image, with the image skewed to fit the shape of each individual face. Use this unwrapping option to reset the map and undo any unwrapping (go back to the start).

Properties

- Object Data
  - Vertex Groups
  - Shape Keys
  - UV Maps
  - Color Attributes
  - Attributes
  - Texture Space
  - Remesh
  - Geometry Data
- Vertex Groups
  - Introduction
  - Vertex Groups Panel
  - Assigning a Vertex Group
  - Vertex Weights
- Geometry Data

Object Data

Meshes
: The mesh Data-Block Menu can be used to link the data between objects.

Vertex Groups

Vertex groups can be used to assign a group or weighted group to some operator. An object can have several weight groups and can be assigned in Weight Paint mode, or in Edit Mode via this panel.

See Vertex Groups for more information.

Shape Keys

Shape Keys can be used to transform one shape into another. See Shape Keys Panel for more information.

UV Maps

UV Maps are used to map a 3D object onto a 2D plane that determines where a texture appears on the 3D object. Different UV Maps can be used for different textures. For more information see UV Maps.

Color Attributes

Color data can be applied directly to an object's vertices rather than using a texture or a material. There are two modes to paint color attributes in. Use Vertex Paint mode to paint per face corner by enabling the paint mask in the header. This is useful to achieve sharp edges in the color attribute on low-poly assets. Alternatively use Sculpt mode to paint on a much higher vertex count.

Creating a New Color Attribute

To create a new Color Attribute select the plus icon next to the list of attributes. This action will open a pop-up with the following information.

Name
: The name of the Color Attribute which can be referenced elsewhere in Blender.

Domain
: The associated part of the geometry that stores the attribute. See Attribute Domains for more information.

Vertex::	Color Attributes are stored per each vertex.
Face Corner::	Color Attributes are stored per each corner of a face.

Data Type
: The data type to represent colors internally.

Color::	RGBA color with floating-point precision.
Byte Color::	RGBA color with 8-bit precision.

Color
: The default color to fill for every element in the domain.

Color Attribute Specials

These are operators that are available in the menu to the right of the attribute list.

Duplicate Color Attribute
: Creates a copy of the active color attribute in the list.

Convert Color Attribute
: Changes how the color attribute is stored.

  Domain

The associated part of the geometry that stores the attribute. See Attribute Domains for more information.

Vertex:: Color Attributes are stored per each vertex.
Face Corner:: Color Attributes are stored per each corner of a face.

Data Type
The data type to represent colors internally.

Color:: RGBA color with floating-point precision.
Byte Color:: RGBA color with 8-bit precision.

Attributes

An attribute is data stored per mesh element. Every attribute has a data type, domain and name. This panel only lists custom attributes which excludes all the built-in attributes like `position` and other attributes like vertex groups.

See Attributes Reference for more information.

Texture Space

Each object can have an automatically generated UV map, these maps can be adjusted here.

See Generated UV Properties for more information.

Remesh

Mesh objects, in particular meshes that have been modeled to represent organic objects, often have geometry that is not particularly uniform. This can cause problems if the object needs to be rigged or just needs simpler geometry for workflows such as 3D printing. Remeshing is a technique that rebuilds the geometry with a more uniform topology. Remeshing can either add or remove the amount of topology depending on the defined resolution.

Remeshing is especially useful for sculpting, to generate better topology after blocking out the initial shape.

See Mesh Retopology for more information.

Geometry Data

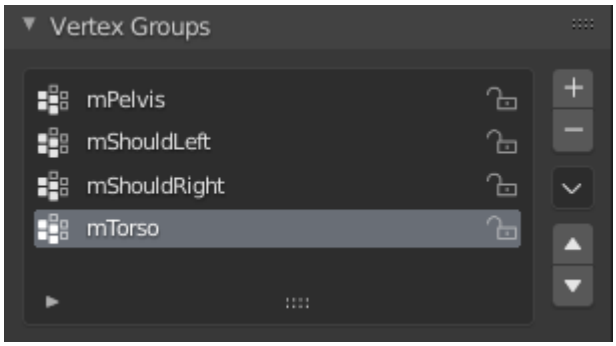
Mesh objects can have different types of custom data attached to them. This data is mostly used internally and can be exported by some exporters. See Geometry Data for more information.

Vertex Groups

- Introduction
  - Usage
- Vertex Groups Panel
  - Editing Vertex Groups
- Assigning a Vertex Group
  - Creating Vertex Groups
  - Deleting Vertex Groups
  - Locking Vertex Groups
  - Working with Content of Vertex Groups
- Vertex Weights
  - Vertex Group Categories
  - Weight Table
  - Operators
  - Locking

Introduction

Vertex groups are mainly used to tag the vertices belonging to parts of a mesh object or Lattice. Think of the legs of a chair or the hinges of a door, or hands, arms, limbs, head, feet, etc. of a character. In addition you can assign different weight values (in the range 0 to 1) to the vertices within a vertex group. Hence vertex groups are sometimes also named 'weight groups'.



The Vertex Groups panel.

Usage

Vertex groups are most commonly used for armatures. But they are also used in many other areas of Blender, like for example:

- Armature deformation *(also called skinning)*
- Shape keys
- Modifiers
- Particle generators
- Physics simulations

See also

Skinning Mesh Objects.

Many more usage scenarios are possible. Actually you can use vertex groups for whatever makes sense to you. In some contexts vertex groups

can also be automatically generated (e.g. for rigged objects). However, in this section we will focus on manually created (user-defined) vertex groups.

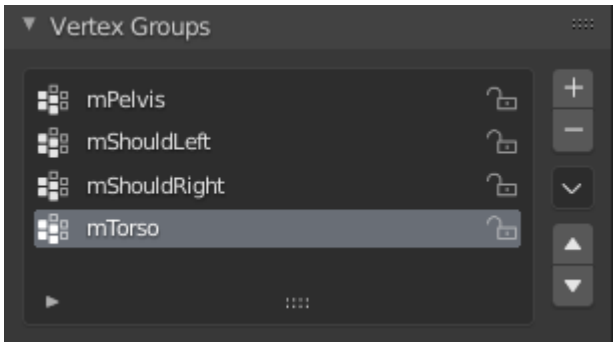
Note

Vertex groups only apply to mesh and lattice objects.

Vertex Groups Panel

Reference

Mode:: All Modes
Panel:: *Object Data tab* ▸ *Vertex Groups*



The Vertex Group panel.

Vertex groups are maintained within the *Object Data* Properties, in the *Vertex Groups* panel.

Active Vertex Group
: A List view.

Lock
: Locks the group from being editable. You can only rename or delete the group.

Add +
: Create an empty vertex group.

Remove –
: Deletes the active vertex group.

Specials
: Sort by Name
: Sorts vertex groups alphabetically.

Sort by Bone Hierarchy
: (Todo)

Duplicate Vertex Group

Add a copy of the active vertex group as a new group. The new group will be named like the original group with "_copy" appended at the end of its name. And it will contain associations to exactly the same vertices with the exact same weights as in the source vertex group.

Copy Vertex Group to Selected
: Copy all vertex groups to other selected objects provided they have matching indices (typically this is true for copies of the mesh which are only deformed and not otherwise edited).

Mirror Vertex Group
: Mirrors weights and/or flips group names. See Mirror Vertex Group for more information.

Mirror Vertex Group (Topology)
: Performs the *Mirror Vertex Group* with the *Topology Mirror* option enabled.

Remove from All Groups
: Unassigns the selected vertices from all (even locked) groups. After this operation has been performed, the vertices will no longer be contained in any vertex group. (Not available for locked groups.)

Clear Active Group
: Remove all assigned vertices from the active group. The group is made empty. Note that the vertices may still be assigned to other vertex groups of the object. (Not available for locked groups.)

Delete All Unlocked Groups
: Remove all vertex groups from the object that are *not* locked.

Delete All Groups
: Remove all vertex groups from the object.

Lock All

Lock all groups.

Unlock All
: Unlock all groups.

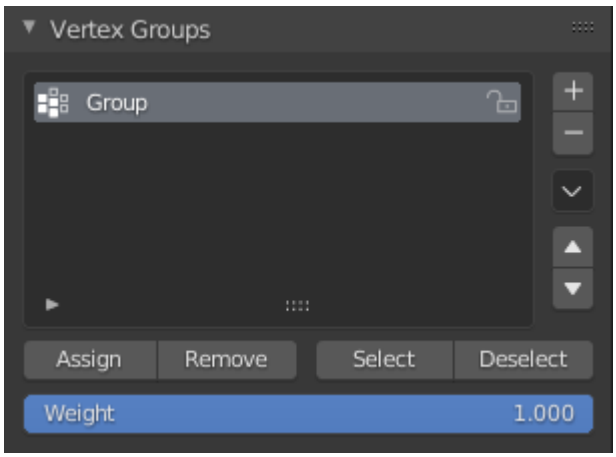
Lock Invert All
: Invert group locks.

Editing Vertex Groups

Reference

Mode::	Edit Mode
Panel::	*Object Data tab* ▸ *Vertex Groups*
Menu::	*Vertex* ▸ *Vertex Groups*
Shortcut::	`Ctrl-G`

When you switch either to Edit Mode or to Weight Paint Mode, vertex weights can be edited. The same operations are available in the 3D Viewport's *Vertex* ▸ *Vertex Groups* menu or `Ctrl-G`.



Vertex Group panel in Edit or Weight Paint Mode.

Assign
: To assign the selected vertices to the active group with the weight as defined in the *Weight* (see below).

Remove
: To remove the selected vertices from the active group (and thus also delete their weight values).

Select
: To select all vertices contained in the group.

Deselect
: To deselect all vertices contained in the group.

Weight
: The weight value that gets assigned to the selected vertices.

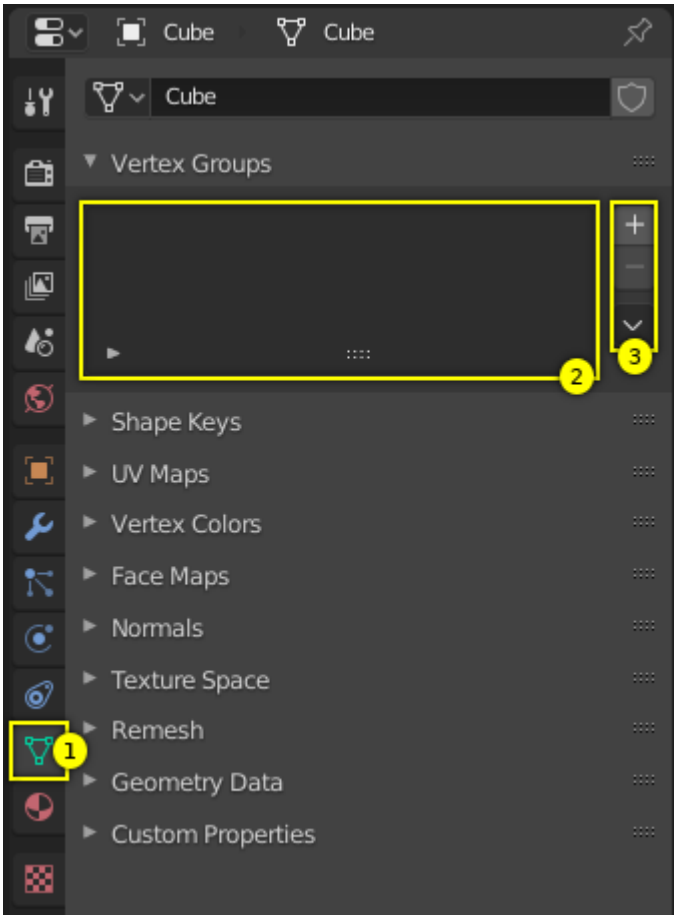
Set Active Group
: Lets you select the group that will become the active one (menu only).

Assigning a Vertex Group

Creating Vertex Groups

Vertex groups are maintained within the *Object Data* tab (1) in the Properties. As long as no vertex groups are defined (the default for new mesh objects), the panel is empty (2).

You create a vertex group by `LMB` on the *Add* button + on the right panel border (3). Initially the group is named “Group” (or “Group.nnn” when the name already exists) and gets displayed in the panel (2) (see next image).



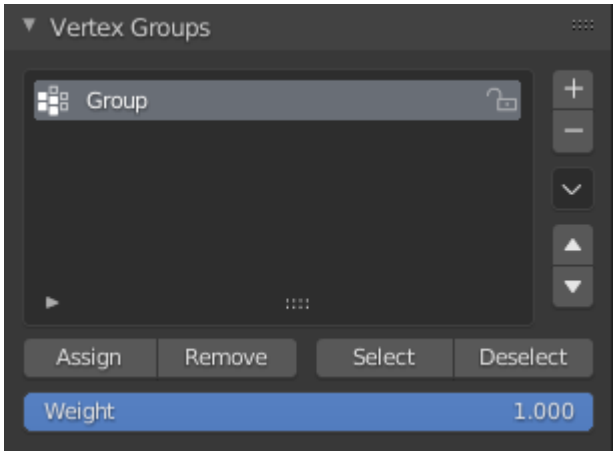
Empty Vertex Groups panel.

Vertex Groups Panel Controls

Once a new vertex group has been added, the new group appears in the Vertex Groups panel. There you find three clickable elements:

Group Name

The group name can be changed by double-clicking LMB on the name itself. Then you can edit the name as you like.



One vertex group.

Filter (arrow icon)

When the little arrow icon in the left lower corner is clicked, a new row opens up where you can enter a search term. This becomes handy when the number of vertex groups gets big.

Drag Handle

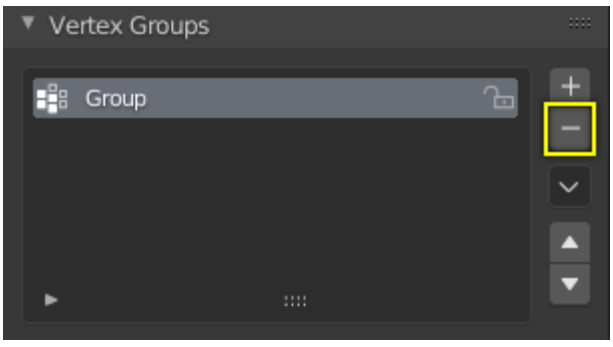
If you have a large number of vertex groups and you want to see more than a few groups, you can LMB on the small drag handle to make the vertex groups list larger or smaller.

Active Group

When a vertex group is created, then it is also automatically marked as the *Active Group*. This is indicated by setting the background of the panel entry to a light gray color. If you have two or more groups in the list, then you can change the active group by LMB on the corresponding entry in the Vertex Groups panel.

Deleting Vertex Groups

You delete a vertex group by first making it the active group (select it in the panel) and then LMB the *Remove* button – at the right panel border.



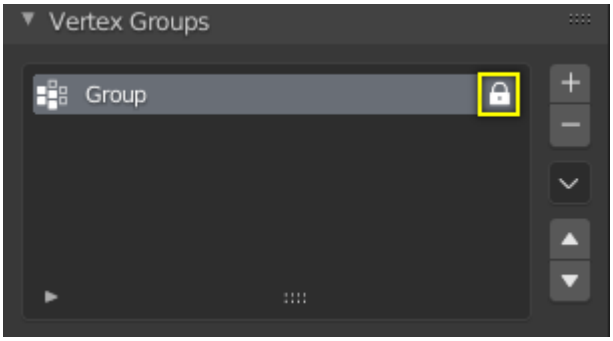
Deleting a vertex group only deletes the vertex assignments to

the group. The vertices themselves are not deleted.

Delete a vertex group.

Locking Vertex Groups

Right after creation of a vertex group, an open padlock icon shows up on the right side of the list entry. This icon indicates that the vertex group can be edited. You can add vertex assignments to the group or remove assignments from the group. And you can change it with the weight paint brushes, etc.



Lock a vertex group.

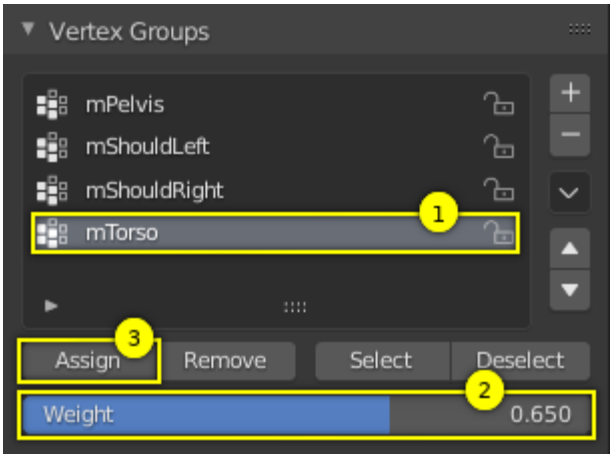
When you click on the icon, it changes to a closed padlock icon and all vertex group modifications get disabled. You can only rename or delete the group, and unlock it again. No other operations are allowed on locked vertex groups, thus all corresponding buttons become disabled for locked vertex groups.

Working with Content of Vertex Groups

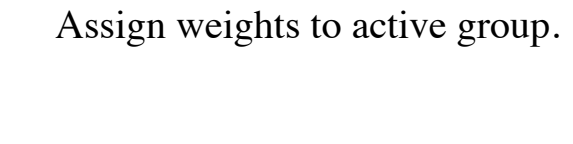
Assigning Vertices to a Group

You add vertices to a group as follows:

- Select the group from the group list, thus making it the active group (1).
- From the 3D Viewport select `Shift-LMB` all vertices that you want to add to the group.



- Set the weight value that shall be assigned to all selected vertices (2).
- LMB the *Assign* button to assign the selected vertices to the active group using the given weight (3).



Note that weight assignment is not available for locked vertex groups. The *Assign* button is grayed out in that case.

Note

Assign is additive

The *Assign* button only adds the currently selected vertices to the active group. Vertices already assigned to the group are not removed from the group.

Also keep in mind that a vertex can be assigned to multiple groups.

Checking Assignments

To be sure the selected vertices are in the desired vertex group, you can try press the deselect button. If the vertices remain selected then they are not yet in the current vertex group.

At this point you may assign them, but take care since all selected vertices will have their weight set to the value in the *Weight* field.

Removing Assignments from a Group

You remove vertices from a group as follows:

- Select the group from the group list (make it the active group).
- Select all vertices that you want to remove from the group.
- LMB click the *Remove* button.

Note that removing weight assignments is not available for locked vertex groups. The *Remove* button is grayed out in that case.

Using Groups for Selecting/Deselecting

You can quickly select all assigned vertices of a group:

- (Optionally) press `Alt-A` to deselect all vertices.
- Select the group from the group list (make it the active group).
- When you now `LMB` click the *Select* button, then the vertices assigned to the active group will be selected and highlighted in the 3D Viewport.
- When you `LMB` click the *Deselect* button instead, then the vertices assigned to the active group will be deselected in the 3D Viewport.

Note

Selecting/Deselecting is additive

If you already have vertices selected in the 3D Viewport, then selecting the vertices of a group will add the vertices but also keep the already selected vertices selected. Vice versa, deselecting the vertices of a vertex group will only deselect the vertices assigned to the group and keep all other vertices selected.

Finding Ungrouped Vertices

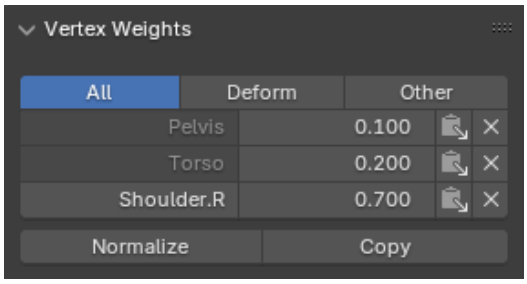
You can find ungrouped vertices as follows:

- Press `Alt-A` to deselect all vertices.
- In the header of the 3D Viewport, navigate to *Select ▸ Select All by Trait ▸ Ungrouped Vertices*.

Vertex Weights

Reference

Mode:: Edit and Weight Paint Modes
Panel:: *Sidebar region ▸ Vertex Weights*



Vertex Weights panel.

A vertex group assigns a weight (a number between 0 and 1) to each vertex it contains. A group can have multiple vertices, and each vertex can be part of multiple groups.

The *Vertex Weights* panel in the 3D Viewport's Sidebar shows the vertex groups for the active vertex, and lets you see and edit the associated weights. It's available in Edit Mode, as well as in Weight Paint Mode when Vertex Selection is enabled in the header.

Vertex Group Categories

While all vertex groups are technically the same, we can still divide them into two types depending on how they're used:

Deform Groups
: Also sometimes called "weight group" or "weight map," this type of vertex group determines which vertices are affected by a certain bone in

the Armature. In other words, it defines which part of the mesh deforms when the bone moves around.

Other Groups
: The remaining vertex groups are used with shape keys, modifiers, and other areas.

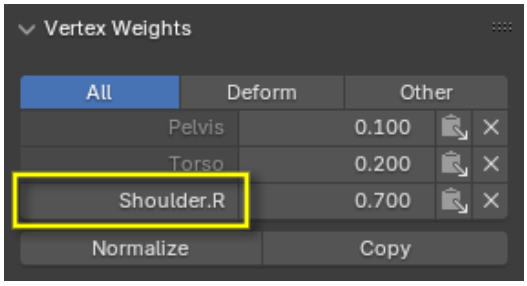
The deform vertex groups are related to each other: the deformation weights of every vertex typically need to add up to 1. For this reason, you can use the filter buttons at the top of the panel to show only these vertex groups (or to exclude them).

Weight Table

The Weight Table shows all the weights associated with the *active vertex*, which is the vertex that was selected last (and is highlighted in white). If there is no active vertex, or it isn't part of any vertex group, the panel is not displayed.

Set the Active Group

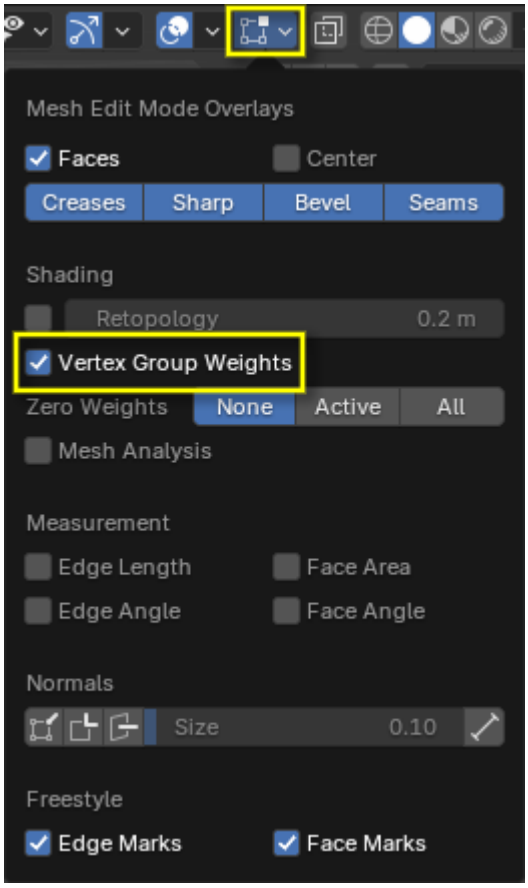
You can click the name of a vertex group to make it the active one.



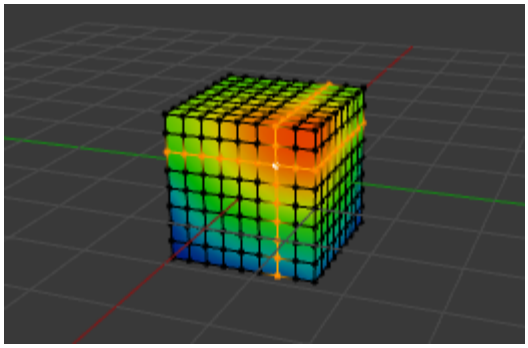
Changing the active vertex group.

Display Weights in Edit Mode

When you are in Edit Mode, you can make the weights of the active group visible on the mesh: open the *Mesh Edit Mode Overlays* popover and enable the *Vertex Group Weights* option.



Enable display of weights in Edit Mode.



Weights in Edit Mode.

Change a Weight

You can change the weight for a vertex group by either clicking the number and typing a new one or by dragging left and right with `LMB`. You can also click the arrows (only shown when hovering) to change the weight in steps of 0.01.



Changing a weight value.

Copy a Weight

The *Paste Weight to Selected* button copies the weight from the active vertex to the other selected vertices. Note that, even though it uses the word "paste," it doesn't interact with the *Copy* button and in fact doesn't use the clipboard at all.



Copying a weight.

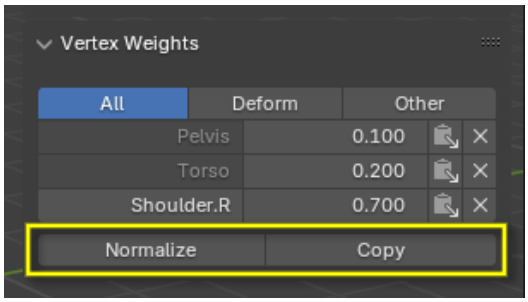
Delete a Weight

The *Delete Weight* button removes the active vertex from the vertex group, making the row disappear from the list.



Deleting a weight.

Operators



Vertex weight operators.

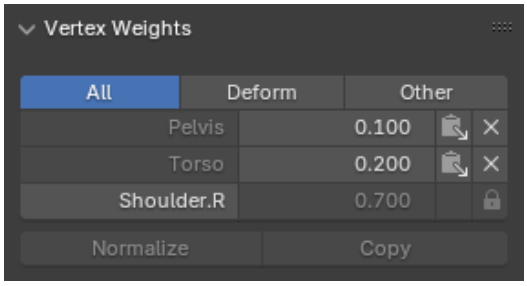
Normalize
: Recalculates the weights of the active vertex so that they add up to 1.0 while retaining their relative magnitude.

Copy
: Copies all the weights from the active vertex to the other selected vertices.

Tip

Both tools only work on the vertex groups that match the current filter setting.

Locking



Locked vertex group.

If a vertex group is locked, its weights become uneditable, and the buttons for copying and normalizing weights become disabled.

Tip

The *Normalize* and *Copy* buttons only become disabled if there's a locked vertex group in the current list. If (for example) only non-deforming vertex groups are locked, you can switch to the *Deform* filter and normalize the groups that way.

Geometry Data

This panel is used to manage any generic data attributes that a mesh could have.

Warning

Clearing any data will result in the data loss of these values.

Clear Sculpt Mask Data
: Deletes the internal `sculpt_mask` attribute. This attribute is used by the Sculpt Masking Feature.

Add/Clear Skin Data
: Used to manage the skin data which is used by the Skin Modifier. This operator can be needed in case a Skin modifier is created but no skin data exist.

Add/Clear Custom Split Normals Data
: Adds Custom Split Normals data, if none exists yet.

UVs

- UVs & Texture Space
  - UV Maps
  - Texture Space
- Unwrapping
  - Introduction
  - Seams
- Tools
  - Toolbar
  - Types
- Editing
  - Transform
  - Randomize
  - Mirror
  - Snap
  - Merge
  - Split
  - Unwrap
  - Pin & Unpin
  - Invert Pins
  - Mark/Clear Seams
  - Seams from Islands
  - Pack Islands
  - Average Island Scale
  - Minimize Stretch
  - Stitch
  - Align
  - Align Rotation
  - Copy UVs
  - Paste UVs
  - Show/Hide Faces
  - Export UV Layout
  - Proportional Editing
  - UV Options

    - 3D Viewport
- Workflows
    - Layout Workflow
    - UDIMs
- Using UV Maps
    - Using the Test Grid

UVs & Texture Space

UV Maps

Reference

Mode:: All Modes
Panel:: *Properties* ▸ *Data* ▸ *UV Maps*



The UV Maps panel in the Data tab.

If you have a mesh object selected, you'll find its UV maps in the Data tab of the Properties editor. After selecting a map, you can view and edit it in the UV editor.

One mesh can have multiple UV maps (e.g. one map per texture), although it's also possible to reuse a UV map for multiple textures.

Active Render
: Click the camera icon to make a certain UV map the default one for rendering. This will be the map that's used by the *UV* output of the Texture Coordinate Node. You can use the UV Map Node to access any other maps.

Add +
: Duplicates the selected UV map, or creates a new one if the list is empty.

Remove –
: Removes the selected UV map.

Texture Space

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Data* ▸ *Texture Space*

This panel lets you configure the object's Texture Space, which is a 3D box used for generating texture coordinates without the use of a UV map. You can visualize the texture space using the option in the Viewport Display panel.

Auto Texture Space
: Calculates the texture space automatically.

  Location, Size
  : Lets you define the texture space manually, relative to the object. Note that you can also edit it in the 3D Viewport – see Editing below.

Texture Mesh *Mesh objects*
: Use another mesh for texture indices. The vertices of the two objects must be perfectly aligned or the UV map will be distorted.

Match Texture Space *Curve objects*
: Modifies the *Location* and *Size* to match the object's bounding box. This disables Auto Texture Space.

Editing

Reference

Mode::	Object Mode and Edit Mode

Menu:: *Object ▸ Transform ▸ Move/Scale Texture Space*

Click one of these menu items, then move the mouse to adjust the texture space and press `LMB` to confirm. While transforming, you can use keyboard shortcuts to lock certain axes; see the status bar.

Accessing

When setting up a material shader, you can use the *Generated* output of the Texture Coordinate Node to read the 3D coordinate inside the object's texture space. You can then pass this coordinate to a texture node.

Tip

Texture spaces do not have rotation support. You can use a Mapping Node to manually rotate the coordinate in the material shader instead.

Unwrapping

- Introduction
  - About UVs
  - Getting Started
- Seams
  - Mark Seam

Introduction

The first step is to unwrap your mesh. Generally, it is recommended to start unwrapping when only minor adjustments to the geometry of your model are required. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you, but you may need to do additional mapping or editing. In this fashion, you can use the UV texture image to guide additional geometry changes.

About UVs

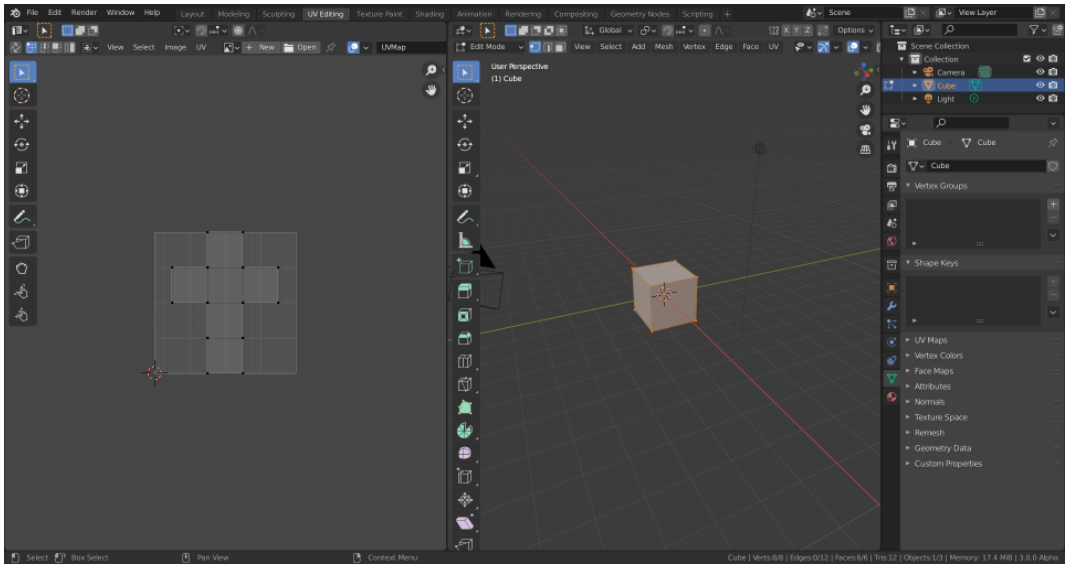
Every point in the UV map corresponds to a vertex in the mesh. The lines joining the UVs correspond to edges in the mesh. Each face in the UV map corresponds to a mesh face. Think of a UV map as projecting the surface of your 3D model onto a 2D image.

Each face of a mesh can have many UV textures. Each UV texture can have an individual image assigned to it. When you unwrap a face to a UV texture in the UV Editor, each face of the mesh is automatically assigned four UV coordinates: These coordinates define the way an image or a texture is mapped onto the face. To distinguish from XYZ coordinates, the U and V axes are used to mark the coordinates of each point. Hence the name, UV unwrapping. These coordinates can be used for rendering or for real-time viewport display as well.

Every face in Blender can have a link to a different image. The UV coordinates define how this image is mapped onto the face. This image then can be rendered or displayed in real-time. A 3D Viewport has to be in “Face Select” mode to be able to assign Images or change UV coordinates of the active mesh object. This allows a face to participate in many UV textures. A face at the hairline of a character might participate in the facial UV texture, *and* in the scalp/hair UV texture.

These are described more fully in the next sections.

Getting Started



Default UV editing workspace.

By default, meshes are not created with UVs. First you must map the faces, then you can edit them. The process of unwrapping your model is done within Edit Mode in the 3D Viewport. This process creates one or more UV Islands in the UV Editor.

To begin, choose the *UV Editing* workspace from the selection list at the top of your screen in the Preferences header. This sets one of the areas to show you the UV Editor, and the other area to the 3D Viewport.

Enter *Edit Mode*, as all unwrapping is done in Edit Mode. You can be in vertex, face, or edge selection mode.

Workflow

The general workflow is as follows, but know that different models may require different approaches to unwrapping:

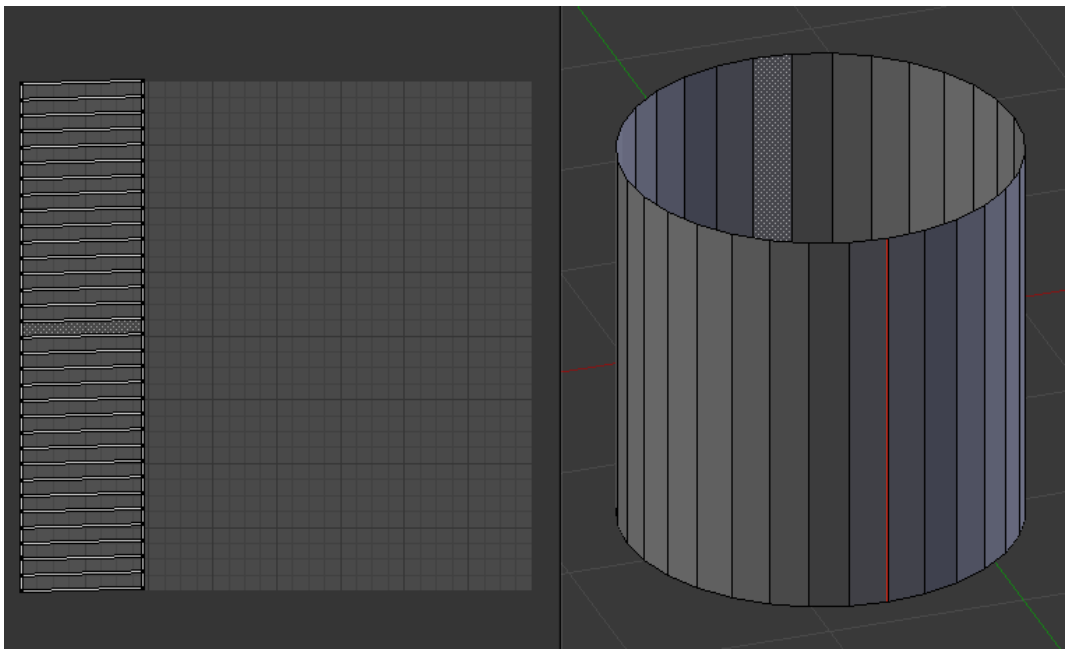
1. Mark Seams if necessary. See more about marking seams.
2. Select mesh faces in the 3D Viewport.

3. Select a UV mapping method from the *UV* ▸ *Unwrap* menu or the UV menu in the 3D Viewport.
4. Adjust the unwrap settings in the Adjust Last Operation panel.
5. Add a test image to see if there will be any distortion. See Applying Images to UVs.
6. Adjust UVs in the UV editor. See Editing UVs.

Seams

For many cases, using the Unwrap calculations of Cube, Cylinder, Sphere, or the regular “Unwrap” operators will produce a good UV layout. But for more complex meshes, especially those with lots of indentations, you may want to define a *seam* to limit and guide the Unwrap operator.

Just like in sewing, a seam is where the ends of the image/cloth are sewn together. In unwrapping, the UV map is discontinuous at the seams. Think of this method as peeling an orange or skinning an animal. You make a series of cuts in the skin, then peel it off. You could then flatten it out, applying some amount of stretching. These cuts are the same as seams.



Simple seam on a cylinder.

When using this method, you need to be aware of how much stretching there is. The more seams there are, the less stretching there is, but this is often an issue for the texturing process. It is a good idea to have as few seams as possible while having the least amount of stretching. Try to hide seams where they will not be seen. In productions where 3D paint is used,

this becomes less of an issue, as projection painting can easily deal with seams, as opposed to 2D texturing, where it is difficult to match the edges of different UV islands.

The workflow is the following:

1. Mark seams.
2. Unwrap.
3. Adjust seams and repeat.
4. Manually adjust UVs.

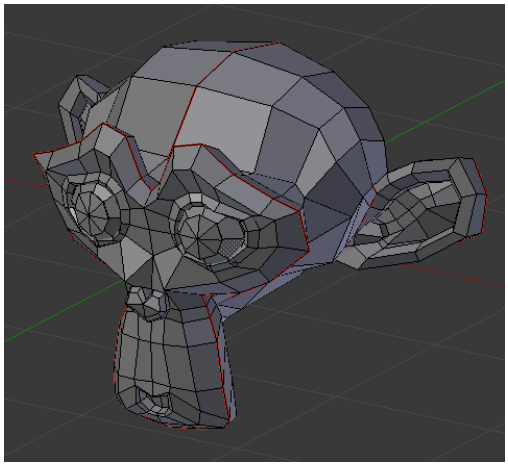
Mark Seam

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Tool:: *UV ▸ Mark/Clear Seam*
Menu:: *Edge ▸ Mark/Clear Seam*

Reference

Editor:: UV Editor
Mode:: View mode
Menu:: *UV ▸ Mark/Clear Seam*



Seamed Suzanne.

To add an edge to a seam, simply select the edge and press `Ctrl-E` to *Mark Seam*, or to remove it, use `Ctrl-E` to *Clear Seam*.

In the example to the right, the back-most edge of the cylinder was selected as the seam (to hide the seam), and the default unwrap calculation was used. In the UV Editor, you can see that all the faces are nicely unwrapped, just as if you cut the seam with a scissors and spread out the fabric.

When marking seams, you can use Select Linked in Face Select Mode to check your work. This menu option selects all faces connected to the selected one, up to a seam. If faces outside your intended seam are selected, you know that your seam is not continuous. You do not need continuous seams, however, as long as they resolve regions that may stretch.

Just as there are many ways to skin a cat, there are many ways to go about deciding where seams should go. In general though, you should think as if you were holding the object in one hand, and a pair of sharp scissors in the other, and you want to cut it apart and spread it on the table with as little tearing as possible. Note that we seamed the outside edges of her ears, to separate the front from the back. Her eyes are disconnected sub-meshes, so they are automatically unwrapped by themselves. A seam runs along the back of her head vertically, so that each side of her head is flattened out.

Another use for seams is to limit the faces unwrapped. For example, when texturing a head, you do not really need to texture the scalp on the top and back of the head since it will be covered in hair. So define a seam at the hairline. Then, when you select a frontal face, and then select linked faces before unwrapping, the select will only go up to the hairline seam, and the scalp will not be unwrapped.

When unwrapping anything that is bilateral, like a head or a body, seam it along the mirror axis. For example, cleave a head or a whole body right down the middle in front view. When you unwrap, you will be able to overlay both halves onto the same Texture Space, so that the image pixels for the right hand will be shared with the left; the right side of the face will match the left, etc.

Note

You **do not** have to come up with “one unwrapping that works perfectly for everything everywhere”. As we will discuss later, you can easily have multiple UV unwrappings, using different approaches in different areas of your mesh.

Seams from Islands

Reference

Mode:: View mode
Menu:: *UV ▸ Seams from Islands*

Adds seams at the boundaries of existing UV islands. This is useful when modifying the UVs of already unwrapped meshes.

Tools

- Toolbar

Types

- Rip
- Grab
- Relax
- Pinch

Toolbar

Select
: Select or moved.

  Select Box
  : Select UVs by dragging a box.

  Select Circle
  : Select UVs by painting on it.

  Select Lasso
  : Select UVs by drawing a lasso.

Cursor
: Change the location of the 2D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

Transform
: Tool to adjust the UVs translation, rotation and scale.

Annotate
: Draw free-hand annotation.

  Annotate Line
  : Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Rip
: The Rip tool separates UV faces from each other.

Grab
: The Grab tool moves UVs around using a brush.

Relax
: The Relax tool makes UVs more evenly distributed using a brush.

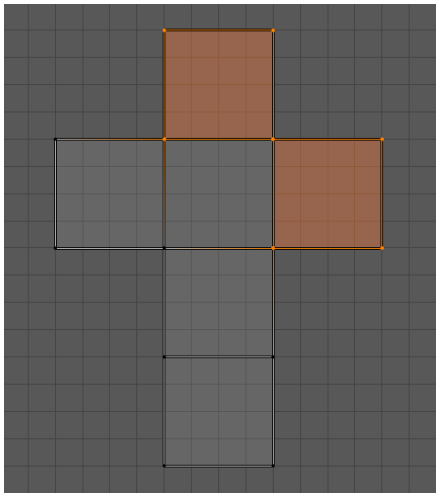
Pinch
: The Pinch tool moves UVs toward the brush's center.

Rip

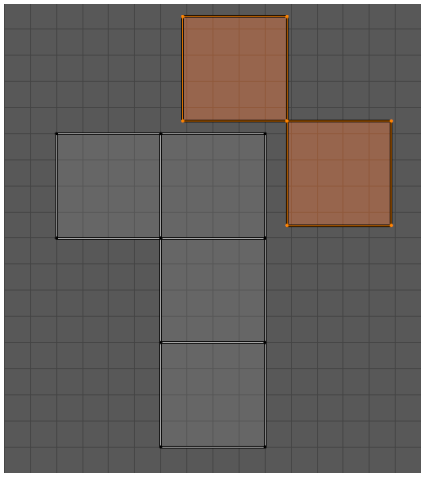
Reference

Mode::	Edit Mode
Tool::	*Toolbar* ‣ *Rip*
Shortcut::	V

The Rip tool separates UV components (vertices, edges, faces) from connected components. The components are ripped in the direction of the mouse pointer position.



Before.



After.

See also

Mesh editing Rip operation.

Grab

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ‣ *Grab*

The Grab tool moves UVs around.

Tool Settings

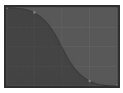
Size
: This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB`. Typing a number then enter while using `F` allows you to enter the size numerically.

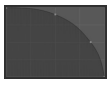
Strength
: Controls how much each application of the brush affects the UVs. You can change the brush strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the brush and then `LMB`. You can enter the size numerically also while in `Shift-F` sizing.

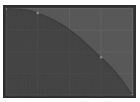
Falloff
: The Falloff allows you to control the *Strength* falloff of the brush. The falloff is mapped from the center of the brush (left part of the curve) towards its borders (right part of the curve). Changing the shape of the curve will make the brush softer or harder. Read more about using the Curve Widget.

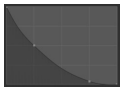
  Curve Preset
  : **Custom::** You can choose how the strength of the falloff is determined from the center of the brush to the borders by manually manipulating the control points within the curve widget. There are also a couple of preset custom curves displayed at the bottom of the curve widget that can be used on their own or as a starting point for tweaking.

    Custom Preset types.


Smooth.


Sphere.


Root.



Sharp.



Linear.



Constant.

Smooth:: The center strength, the border strength, and the falloff transition between them are evenly distributed.

Smoother:: Similar to *Smooth* but produces a wider center point of the brush before tapering off.

Sphere:: The strength of the brush is predominately at its strongest point with a steep falloff near the border of the brush.

Root:: Similar to a Sphere but the center is a more concentrated point.

Sharp:: The center of the brush is the strongest point then exponentially tapers off to a lower strength, creating a fine point.

Linear:: With the center being the strongest, the strength will consistently weaken as it reaches the border of the brush.

Sharper:: Similar to *Sharp* but the center point is more condensed.

Inverse Square:: A hybrid between *Smooth* and *Sphere*.

Constant:: The strength of the brush remains unified across the entire brush. This will create a sharp edge at the border of the brush.

Options

Lock Borders

Locks the boundary of UV islands from being affected by the brush. This is useful to preserve the shape of UV islands.

Sculpt All Islands

To edit all islands and not only the island nearest to the brush center when the sculpt stroke was started.

Relax

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Relax*

The Relax tool can be used to distribute UVs more evenly. It works by pulling vertices along UV edges to bring the UV unwrap into balance.

The Relax tool can be compared with the Minimize Stretch tool which works directly on faces to reduce texture stretching and shearing. You may find that sometimes minimize stretch works better, sometimes the unwrap tool and other times the Relax tool.

First using Unwrap, then Minimize Stretch and touching up with the Relax tool often gives the best results. Remember, you can use "Undo" at any time to return to an earlier state.

Tool Settings

Size
: This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB`. Typing a number then enter while using `F` allows you to enter the size numerically.

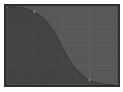
Strength
: Controls how much each application of the brush affects the UVs. You can change the strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the brush and then `LMB`. You can enter the size numerically also while in `Shift-F` sizing.

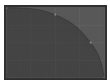
Falloff
: The Falloff allows you to control the *Strength* falloff of the brush. The falloff is mapped from the center of the brush (left part of the curve) towards its borders (right part of the curve). Changing the shape of the curve will make the brush softer or harder. Read more about using the Curve Widget.

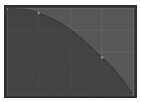
  Curve Preset

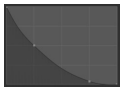
Custom:: You can choose how the strength of the falloff is determined from the center of the brush to the borders by manually manipulating the control points within the curve widget. There are also a couple of preset custom curves displayed at the bottom of the curve widget that can be used on their own or as a starting point for tweaking.

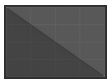
Custom Preset types.


Smooth.


Sphere.


Root.


Sharp.


Linear.


Constant.

Smooth:: The center strength, the border strength, and the falloff transition between them are evenly distributed.

Smoother:: Similar to *Smooth* but produces a wider center point of the brush before tapering off.

Sphere:: The strength of the brush is predominately at its strongest point with a steep falloff near the border of the brush.

Root:: Similar to a Sphere but the center is a more concentrated point.

Sharp:: The center of the brush is the strongest point then exponentially tapers off to a lower strength, creating a fine point.

Linear:: With the center being the strongest, the strength will consistently weaken as it reaches the border of the brush.

Sharper:: Similar to *Sharp* but the center point is more condensed.

Inverse Square:: A hybrid between *Smooth* and *Sphere*.

Constant:: The strength of the brush remains unified across the entire brush. This will create a sharp edge at the border of the brush.

Options

Lock Borders

Locks the boundary of UV islands from being affected by the brush. This is useful to preserve the shape of UV islands.

Sculpt All Islands
To edit all islands and not only the island nearest to the brush center when the sculpt stroke was started.

Method
How to determine the edge weighting:

Laplacian:: The classic discrete laplace operator applied to the UV graph. Each edge has equal weighting, resulting in triangles which resemble a honeycomb shape, or quads aligned into square grid.

HC:: Similar to Laplacian, the HC method uses equal weighting while trying to preserve a gradient between dense regions of the mesh and regions with fewer edges.
Note, this method uses the "Humphrey's Classes" operator as described in the paper: "Improved Laplacian Smoothing of Noisy Surface Meshes".

Geometry:: Edges are weighted according to the discrete Laplace operator (cotangent formula) applied to the 3D geometry. This tries to bring the relative lengths of edges in UV closer to the relative lengths of edges in 3D, resulting in a UV unwrap with less distortion across edge boundaries.

Pinch

Reference

Mode:: Edit Mode
Tool:: *Toolbar ▸ Pinch*

The Pinch tool moves UVs toward the brush's center. The pinch tool can be inverted by pressing `Ctrl-LMB`.

Tool Settings

Size
: This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB`. Typing a number then enter while using `F` allows you to enter the size numerically.

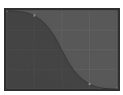
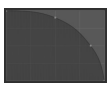
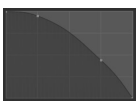
Strength
: Controls how much each application of the brush affects the UVs. You can change the brush strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the brush and then `LMB`. You can enter the size numerically also while in `Shift-F` sizing.

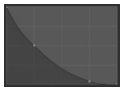
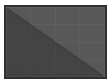
Falloff
: The Falloff allows you to control the *Strength* falloff of the brush. The falloff is mapped from the center of the brush (left part of the curve) towards its borders (right part of the curve). Changing the shape of the curve will make the brush softer or harder. Read more about using the Curve Widget.

  Curve Preset

  Custom:: You can choose how the strength of the falloff is determined from the center of the brush to the borders by manually manipulating the control points within the curve widget. There are also a couple of preset custom curves displayed at the bottom of the curve widget that can be used on their own or as a starting point for tweaking.

  Custom Preset types.

Smooth.	Sphere.	Root.
 Sharp.	 Linear.	 Constant.

Smooth:: The center strength, the border strength, and the falloff transition between them are evenly distributed.

Smoother:: Similar to *Smooth* but produces a wider center point of the brush before tapering off.

Sphere:: The strength of the brush is predominately at its strongest point with a steep falloff near the border of the brush.

Root:: Similar to a Sphere but the center is a more concentrated point.

Sharp:: The center of the brush is the strongest point then exponentially tapers off to a lower strength, creating a fine point.

Linear:: With the center being the strongest, the strength will consistently weaken as it reaches the border of the brush.

Sharper:: Similar to *Sharp* but the center point is more condensed.

Inverse Square:: A hybrid between *Smooth* and *Sphere*.

Constant:: The strength of the brush remains unified across the entire brush. This will create a sharp edge at the border of the brush.

Options

Lock Borders

Locks the boundary of UV islands from being affected by the brush. This is useful to preserve the shape of UV islands.

Sculpt All Islands

To edit all islands and not only the island nearest to the brush center when the sculpt stroke was started.

Editing

After unwrap, you will likely need to arrange the UV maps, so that they can be used in texturing or painting. Your goals for editing are:

- Stitch pieces (of UV maps) back together.
- Minimize wasted space in the image.
- Enlarge the faces where you want more detail.
- Re-size/enlarge the faces that are stretched.
- Shrink the faces that are too grainy and have too much detail.

With a minimum of dead space, the most pixels can be dedicated to giving the maximum detail and fineness to the UV texture. A UV face can be as small as a pixel (the little dots that make up an image) or as large as an entire image. You probably want to make major adjustments first, and then tweak the layout.

Transform

Reference

Editor::	UV Editor
Mode::	Edit Mode
Tool::	*Toolbar ▸ Move, Rotate, Scale, Transform*
Menu::	*UV ▸ Transform*

- Move `G`
- Rotate `R`
- Scale `S`
- Shear `Shift-Ctrl-Alt-S`

Axis Locking

Transformations can be locked to an axis by pressing `X` or `Y` after one of the transform tools. Also, holding the `MMB` will constrain movement to the X or Y axis.

Vertex Slide

Reference

Mode:: Edit Mode
Menu:: *UV ▸ Transform ▸ Vertex Slide*

Vertex Slide will transform a vertex along one of its adjacent edges. Use `Shift-V` to activate tool. The nearest selected vertex to the mouse cursor will be the control one. Move the mouse along the direction of the desired edge to specify the vertex position. Then press `LMB` to confirm the transformation.

Factor
: Determines the amount of slide performed. Negative values correspond to slides toward one vertex, while positive ones, refer to the other one.

Even `E`
: By default, the offset value of the vertices is a percentage of the edges length along which they move. When Even mode is active, the vertices are shifted by an absolute value.

Flipped `F`
: When Flipped is active, vertices move the same distance from adjacent vertices, instead of moving from their original position.

Clamp `Alt` or `C`
: Toggle clamping the slide within the edge extents.

Edge Slide

Reference

Mode:: Edit Mode
Menu:: *UV ▸ Transform ▸ Edge Slide*

Slides one or more edges across adjacent faces with a few restrictions involving the selection of edges (i.e. the selection *must* define a valid loop, see below).

Factor
: Determines the amount of slide performed. Negative values correspond to slides toward one face, while positive ones, refer to the other one.

Even `E`
: Forces the edge loop to match the shape of the adjacent edge loop. You can flip to the opposite vertex using `F`.

Flipped `F`
: When Even mode is active, this flips between the two adjacent edge loops the active edge loop will match.

Clamp `Alt` or `C`
: Toggle clamping the slide within the edge extents.

Mirror Editing
: Lets you propagate the operation to the symmetrical elements of the mesh (if present, in local X direction).

Randomize

Reference

Editor:: UV Editor
Mode:: Edit Mode
Menu:: *UV ▸ Transform ▸ Randomize*

Randomize the scale, rotation and offset of selected UV islands. The *Randomize Transform* tool in the UV editor works similar to *Randomize Transform* tool in the 3d view.

Random Seed
: Changes the random seed used by the pseudo-random number generator, producing a different transform result for each seed value.

Location
: Amount to randomize location.

Rotation
: Amount to randomize rotation.

Scale Even
: Apply the same scale to the U coordinate and V coordinate.

Scale
: Amount to randomize scale in U and V coordinates.

Mirror

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Mirror*
Shortcut::	`Ctrl-M`

UVs can be mirrored on the Y axis or the X axis:

- Mirror X
- Mirror Y

You can also use the hotkeys `X` or `Y`, or hold the `MMB` and drag in the mirror direction.

Copy Mirrored UV Coordinates

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Copy Mirrored UV Coordinates*

Copies UVs from one side of the mirrored mesh to the other. Affects only selected vertices (on both sides).

Axis Direction
: Positive/Negative

Precision
: Tolerance for finding vertex duplicates.

Snap

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Snap*
Shortcut::	`Shift-S`

Snapping in the UV Editor is similar to Snapping in 3D. For the snap to pixel options to work an image has to be loaded.

Selected to Pixels
: Moves selection to nearest pixel. See also *Round to Pixels* below.

Selected to Cursor
: Moves selection to 2D cursor location.

Selected to Cursor (Offset)
: Moves selection center to 2D cursor location, while preserving the offset of the vertices from the center.

Selected to Adjacent Unselected
: Moves selection to adjacent unselected element.

Cursor to Pixels
: Snaps the cursor to the nearest pixels.

Cursor to Selected
: Moves the Cursor to the center of the selection.

Cursor to Origin
: Places the cursor to the location (0, 0, 0).

Merge

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Merge*
Shortcut::	`M`

At Center
: Moves selected UVs to their average position.

At Cursor
: Moves selection UVs to 2D cursor location.

By Distance

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Merge ▸ By Distance*

Merges selected UVs within the specified *Merge Distance*.

Merge Distance
: Maximum distance between merged vertices.

Unselected
: Merge selected to other unselected vertices.

Shared Vertex
: Merge UVs that correspond to the same mesh vertex, even if they have different UV coordinates.

Split

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Split*
Shortcut::	`Alt-M`

Selection `Y`
: Splits (disconnects) the selection from the rest of the UV. The border edge to any non-selected elements are duplicated.

  Note that the “copy” is left exactly at the same position as the original, so you must move it to see it clearly.

Unwrap

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Unwrap*
Shortcut::	`U`

Blender offers several ways of mapping UVs. The simpler projection methods use formulas that map 3D space onto 2D space, by interpolating the position of points toward a point, axis or plane through a surface. The

more advanced methods can be used with more complex models, and have more specific uses.

- Unwrap
- Smart UV Project
- Lightmap Pack
- Follow Active Quads
- Cube Projection
- Cylinder Projection
- Sphere Projection

Pin & Unpin

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Pin/Unpin*
Shortcut::	`P`, `Alt-P`

You can pin UVs so they do not move between multiple unwrap operations. When Unwrapping a model it is sometimes useful to "Lock" certain UVs, so that parts of a UV layout stay the same shape, and/or in the same place. Pinning is done by selecting a UV, then selecting *Pin* from the *UVs* menu, or the shortcut `P`. You can *Unpin a UV* with the shortcut `Alt-P`.

Pinning is most effective when using the Unwrap method of UV mapping, for organic objects. An example is when you are modeling a symmetrical object using the Mirror Modifier. Some of the UVs on the mirror axis may be shared across the mirrored counterparts. You could pin the UVs that correspond to the midline, then align them on the X axis, and they will stay in that location.

The sculpting tools, *Pinch* and *Relax*, will not move any pinned UVs. This allows you to pin the borders, or around interior holes, and gives even more control to the sculpt tools.

Pinning also works great with the *Live Unwrap* tool. If you pin two or more UVs, with *Live Unwrap* on, moving or scaling the pinned UVs will interactively unwrap the model. You can even use the *Grab* sculpting tool to move the pinned UVs. This helps with fitting a UV island to a certain shape or region.

Invert Pins

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ‣ Invert Pins*

Pin all un-pinned selected UVs and un-pin all currently selected pinned UVs.

Mark/Clear Seams

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ‣ Mark/Clear Seam*

See Seams.

Seams from Islands

Reference

Mode::	View mode
Menu::	*UV ‣ Seams from Islands*

Adds seams at the boundaries of existing UV islands. This is useful when modifying the UVs of already unwrapped meshes.

Pack Islands

Reference

Editor:: UV Editor
Mode:: Edit Mode
Menu:: *UV ▸ Pack Islands*

The *Pack Islands* tool can be used to optimize the UV layout by adjusting existing islands to efficiently fill the Texture Space. Based on the options selected, the tool will scale, translate and rotate the islands, ensuring a specified margin exists between them to maximize the usage of the UV space. Pinned islands can have additional restrictions applied to customize the packing process even further.

Shape Method
: The method to use when considering the shape of each island.

  Exact Shape (Concave)::
  : Use the complete shape of the island, including filling any holes or concave regions around the island.

  Boundary Shape (Convex)::
  : Takes into account the boundary (Convex Hull) of the island. This method will not place islands inside holes.

  Bounding Box:: Uses the simple bounding box of the island.

Scale
: Scale the islands to fill the unit square, or pack islands towards the lower left corner.

Rotate

Allows the rotation of islands, as well as translation and scaling, to optimize texture usage.

Rotation Method

The allowable rotations to use for each each island.

Any::	Any rotation which improves the packing is allowed.
Axis-aligned::	The island will first be rotated into a smallest rectangle. Additional rotation will only be in 90-degree turns.
Cardinal::	Like the four cardinal directions on a compass, North, South, East and West, only 90-degree turns will be allowed.

Margin Method

The method to use when calculating the empty space between islands.

Scaled::	Use scale of existing UVs to multiply margin.
Add::	Simple method, just add the margin.
Fraction::	Precisely specify the fraction of the UV unit square for margin. (Slower than other two methods.)

Margin

The scale for the empty space between islands.

Lock Pinned Islands

An island which has any of its UVs pinned is considered a *Pinned Island*. With this option, *Pinned Islands* will be unable to move. The other islands will pack around them.

Lock Method

Change the way *Pinned Islands* are packed

Scale::	The scale of the *Pinned Islands* will not change.
Rotation::	*Pinned Islands* will not rotate.
Rotation and Scale::	

Pinned Islands can translate, but not scale nor rotate.

Merge Overlapping

Before the main packing operation, overlapping islands are detected and temporarily combined. During packing, the relative rotation and position of the merged islands are preserved.

Pack To

Determines the final placement of UV islands after completing the packing operation.

Closest UDIM:: Pack islands to the UDIM grid nearest to the center of the selection.

Active UDIM:: Pack islands to the active UDIM image tile or, if no image is available, the UDIM grid tile where the 2D cursor is located.

Original bounding box:: Find the original bounding box of the selection, packs the islands, and then moves them back inside the original box.

Note

The performance of the *Pack Islands* operator is heavily affected by the options selected, and sometimes the options can combine in different ways to produce unexpectedly slower results.

The fastest results can be obtained by using the “Bounding Box” shape method and the “Add” margin method.

Although enabling the “Rotate” option slightly impacts performance, it will often enhance efficiency, making it a good choice to always keep enabled.

However the “Fraction” margin method requires significantly more computation to find the exact scale. For certain layouts, it may even take up to 10 times longer to complete then using the simpler “Add” or “Scaled” methods.

Similarly, the “Exact shape” and “Boundary shape” methods are much slower than the simple “Bounding Box” method.

Average Island Scale

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Average Island Scale*

Using the *Average Island Scale* tool, will scale each UV island so that they are all approximately the same scale.

Non-Uniform
: Reduces average texture stretching within islands by scaling the U and V axes independently.

Shear
: Reduces average texture shearing within islands by shearing the U axis.

Minimize Stretch

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Minimize Stretch*

The *Minimize Stretch* tool, reduces UV stretch by minimizing the difference between the angles in 3D and the angles in UV space. This operation is similar to the Relax tool with the *Geometry* Relaxation Method, but uses a different algorithm.

Fill Holes
: Just during *minimize stretch*, internal holes will be filled with temporary polygons to prevent stretching and overlaps of the surrounding UVs.

Blend
: The fraction between 0 and 1 of the original UVs to blend in once the stretch is minimized. A blend of 0 is the fully minimized stretch. Blend of 0.5 is halfway between the original UVs and the minimize stretch UVs.

Iterations
: More iterations result in smoother UVs, but take longer to process.

Stitch

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Stitch*
Shortcut::	`Alt-V`

The *Stitch* tool, will join selected UVs that share vertices. You set the tool to limit stitching by distance in the Adjust Last Operation panel, by activating *Use Limit* and adjusting the *Limit Distance*.

Align

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Align*
Shortcut::	`Shift-W`

Moves the selected UV vertices to a line, where that line is specified in different ways by *Axis*.

Axis

Straighten:: Positions UV vertices along the line defined by the endpoints.

Straighten X:: Positions UV vertices horizontally along the line defined by the endpoints.

Straighten Y:: Positions UV vertices vertically along the line defined by the endpoints.

Align Auto:: Positions UV vertices automatically chooses the direction based on which is most alignment already.

Align Vertically:: Positions UV vertices vertically along the line defined by the midpoint of the selection.

Align Horizontally:: Positions UV vertices horizontally along the line defined by the midpoint of the selection.

Align Rotation

Reference

Editor:: UV Editor
Mode:: Edit Mode
Menu:: *UV ▸ Align Rotation*

The *Align Rotation* tool aligns entire islands to either the U or V axis.

The tool has three different methods of operation. The different methods specify the source for the alignment, and also whether to align with both the U and V axes, or just the V axis alone.

When using the *Auto* method, islands are aligned so that UV edges are aligned to either the U axis or the V axis. This method works best with quads and meshes representing organic subjects.

When using the *Edge* method, only the selected edges are considered, and the islands will be aligned such that the selected edges are aligned with the V axis. This method works with the selection, so it works best when a particular edge, or edge loop, needs to be aligned in UV coordinates.

When using the *Geometry* method, the geometry is taken into consideration. Either the *X* axis, the *Y* axis, or the *Z* axis can be used. Suppose the *X* axis is chosen. Using this method, edges which have a positive extent in the *X* axis will be rotated in the UV map so that the edge extends upwards in the *V* axis. This method works best to align multiple islands which share some common geometric property, either in the X, Y or Z axis.

Note that in the *Auto* method, edges can end up aligned either up or down or left or right depending on the orientation of the island prior to activating the tool. In the *Edge* method, the alignment of selected edges can be either up or down in the V axis, whatever is closest to the current orientation of the UV island. By comparison, in the *Geometry* method, the alignment will always be pointing up in the V axis, ignoring any previous orientation.

Copy UVs

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Copy UVs*

For each selected UV island, the *Copy UVs* tool will copy it's topology and UV coordinates into a temporary clipboard for later use with the *Paste UVs* tool.

Note

The *Copy UVs* tool currently uses an internal clipboard which is not shared between instances of blender.

Paste UVs

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Paste UVs*

For each selected UV island, the *Paste UVs* tool will attempt to match the topology of an island stored in the internal clipboard. If a match is found, the UVs stored in the clipboard for the original island will be pasted onto the currently selected island.

For example, if a triangle attached to a quad attached to a quad is in the clipboard, then a different triangle <=> quad <=> quad is selected, then the topologies match, and the UVs will be pasted over the current selection.

For best results, you may want to use the Rip tool, or *UV > Split > Selection*, prior to using *Paste UVs*.

Show/Hide Faces

Reference

Editor::	UV Editor
Mode::	Edit Mode
Menu::	*UV ▸ Show/Hide Faces*

- Reveal Hidden `Alt-H`
- Hide Selected `H`
- Hide Unselected `Shift-H`

Export UV Layout

Reference

Editor:: UV Editor
Mode:: Edit Mode
Menu:: *UV ▸ Export UV Layout*

If you are using an external application, you need to know where on the mesh you are painting.

Note

This is an add-on activated by default.

Proportional Editing

Reference

Editor:: UV Editor
Mode:: Edit Mode
Header:: *Proportional Editing*
Menu:: *UV ▸ Proportional Editing*
Shortcut:: 0

Proportional Editing is available in UV editing. The controls are the same as in the 3D Viewport. See Proportional Editing in 3D for a full reference.

UV Options

Reference

Editor:: UV Editor
Mode:: Edit Mode
Menu:: *UVs*

Live Unwrap

Continuously unwraps the selected UV islands while transforming pinned vertices. Note, this is different than the Live Unwrap option in the 3D Viewport.

Round to Pixels

During UV transforms, you can use Round to Pixels to help with matching features in the image or ensure your UVs have precise horizontal, vertical or diagonal alignment.

Note that Round to Pixels is applied after any snapping modes.

Disabled:: UVs will not be rounded.
Corner:: Will force the UVs to round to the corner of the nearest pixel of an image if loaded.
Center:: Will force the UVs to round to the center of the nearest pixel of an image if loaded.

Constraining to Image Bounds

For standard textures, this option prevents UVs from being moved outside the 0 to 1 UV range. For UDIMs textures, this option prevents UVs from being moved outside the nearest UDIM tile.

3D Viewport

Rotate UVs

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *Face ▸ Face Data ▸ Rotate UVs*

The orientation of the UV texture is defined by each face. If the image is, for example, upside down or laying on its side, use the *Face ▸ Rotate UVs* (in the 3D Viewport in Face Select mode) menu to rotate the UVs per face in 90-degree turns.

Reverse UVs

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Face ▸ Face Data ▸ Reverse UVs*

The *Face ▸ Reverse UVs* tool mirrors the UVs per face, which flips the image over, showing you the image reversed.

Workflows

- Layout Workflow
  - Transferring UV Maps
  - Multiple UV Maps
  - Optimizing the UV Layout
  - Refining the Layout
- UDIMs
  - Workflow
  - File Substitution Tokens
  - UDIM Tiles

Layout Workflow

Transferring UV Maps

You can copy a UV map from one mesh to another mesh provided both meshes have the same geometry/vertex order. This is useful for example when you want to recreate a UV map from an earlier version of your model with intact UVs. This works by:

1. Select the target mesh (to which you want to copy the UV map).
2. `Shift` select the source mesh (that contains the intact UV map).
3. *Object menu* ▸ *Make Links…* ▸ *Transfer UV Layouts* (Shortcut: `Ctrl-L` …).

The target mesh will now have a UV map that matches the original mesh.

Multiple UV Maps

You are not limited to one UV map per mesh. You can have multiple UV maps for parts of the mesh by creating new UV maps. This can be done by clicking the *Add* button next to UV maps list and unwrapping a different part of the mesh. UV maps always include the whole mesh.

Optimizing the UV Layout

When you have unwrapped, possibly using seams, your UV layout may be quite disorganized and chaotic. You may need to proceed with the following tasks: Orientation of the UV mapping, arranging the UV maps, stitching several maps together.

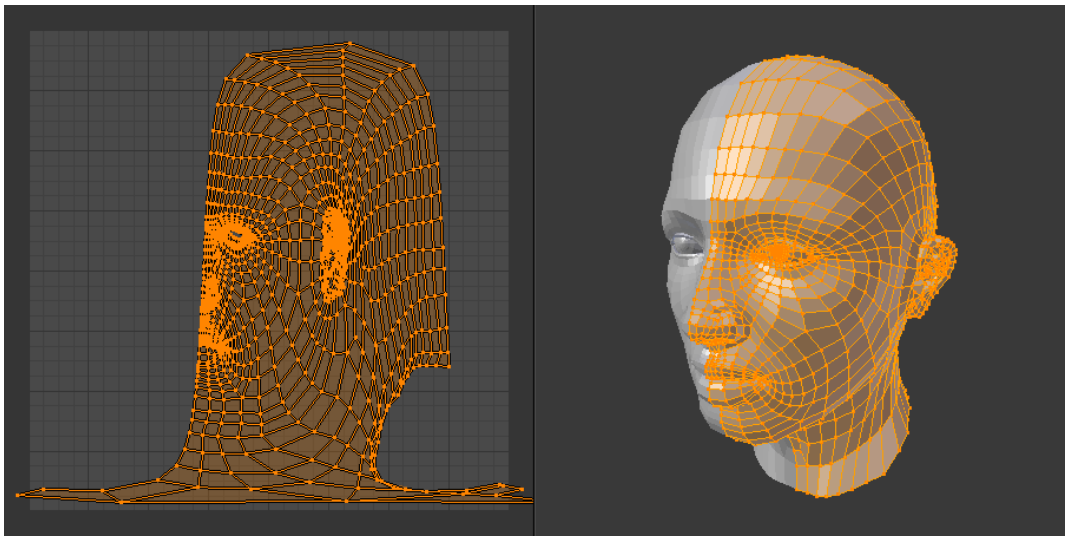
The next step is to work with the UV layouts that you have created through the unwrap process. If you do add faces or subdivide existing faces when a model is already unwrapped, Blender will add those new faces for you. In

this fashion, you can use the UV texture image to guide additional geometry changes.

When arranging, keep in mind that the entire view is your workspace, but only the UV coordinates within the grid are mapped to the image. So, you can put pieces off to the side while you arrange them. Also, each UV unwrap is its own linked set of coordinates.

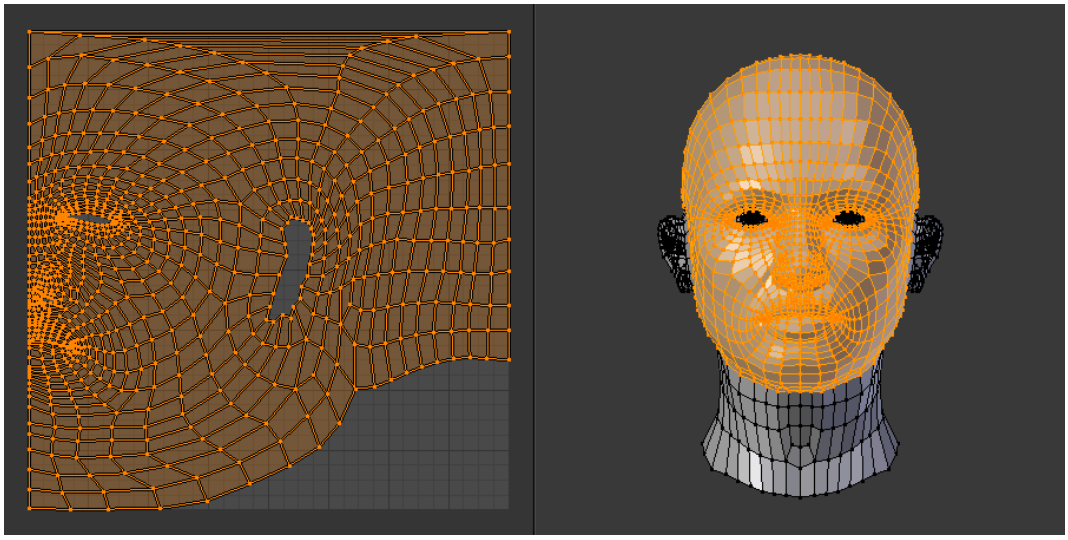
You can lay them on top of one another, and they will onion skin (the bottom one will show through the top one). To move only one though, `RMB` select one of the UV coordinates, and use *Select ▸ Linked UVs*, `Ctrl-L` to select connected UVs, not box select because UVs from both will be selected.

Combining UV Maps



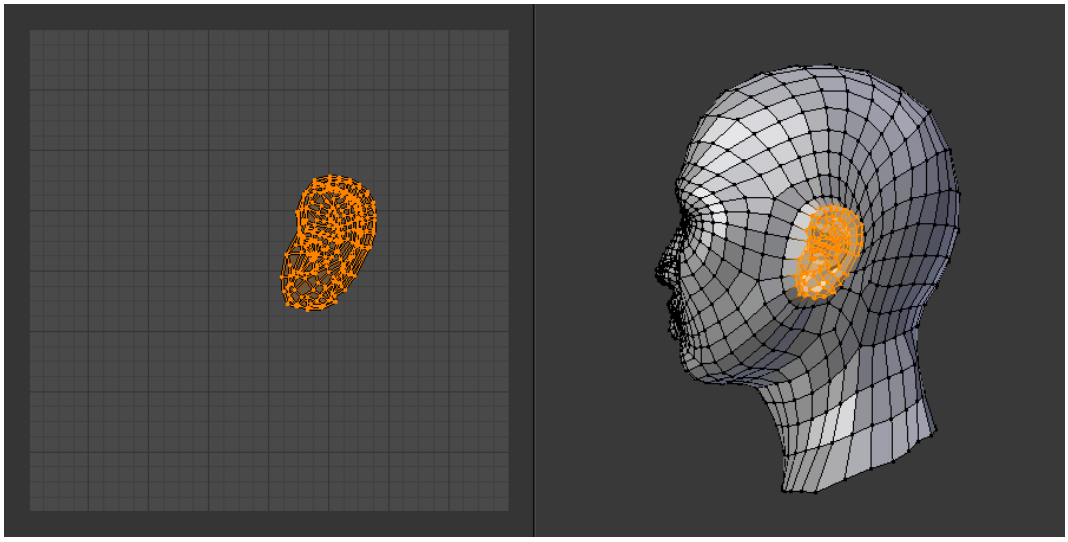
Bad unwrap, note ear and neck.

Very often you will unwrap an object, such as the face example we have been using, and get it “mostly right” but with parts of the mesh that did not unwrap properly, or are horribly confusing. The picture to the right shows an initial unwrap of the face using the Unwrap from sphere option. The issues are with the ear; it is just a mush of UVs, and the neck, it is stretched and folded under. Too much work to clean up.



Unwrap face only, without ear or neck.

We can tell that the ear would unwrap nicely with just a straightforward projection from the side view, and the neck with a tubular unwrap. So, our general approach will be to unwrap different parts of the object (face, ears, and so on) using different unwrap tool, selecting each tool according to which one works best for that part. So let us begin: We select only the "face" faces, unwrap them using the *Sphere* projection, and scale and rotate them a bit to fit logically within the image area of the UV Editor.



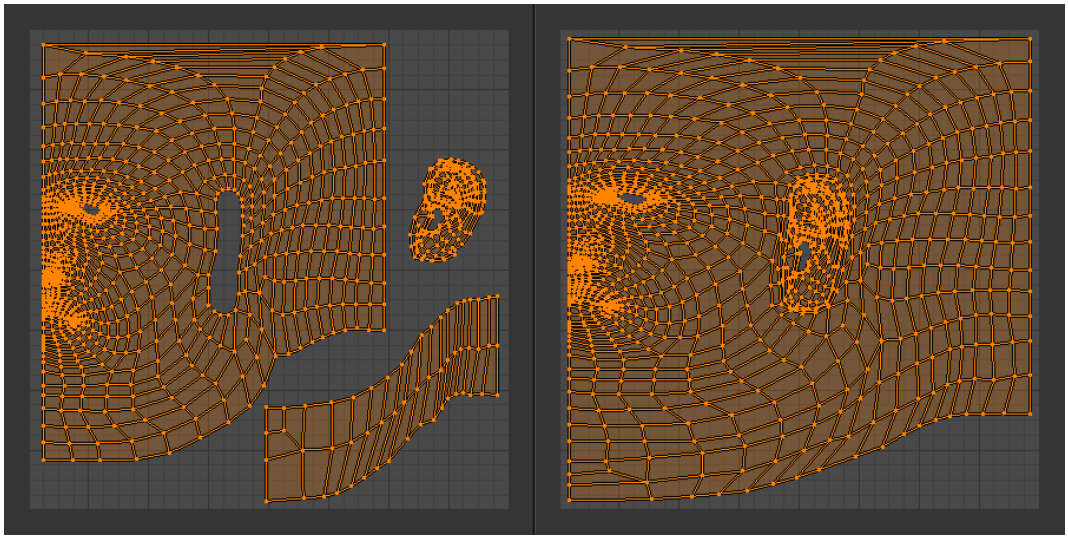
Unwrap ear only, using *Project from View*.

Once we are satisfied with the face, it is time to turn our attention to the ear. First, deselect the faces you were working with. Their UVs will disappear from the UV Editor, but they are still there, just not shown. (To verify this, you can select a few faces in 3D Viewport and it will show up in the UV Editor.)

To work on the ear, in the 3D Viewport, we now select only the "ear" faces. You can use vertex groups to select the ear faces. Selecting sub-meshes is easy too, since they are not connected to the rest of the mesh. Simply selecting linked vertices will select that entire submesh. Basically, since you are in Edit Mode, all the selecting/deselecting features are available to you.

Now unwrap the ear again using the *Project* tool from side view, and scale and rotate them a bit (discussed in the next section), and place them off to the side. You can do this repetitively, using different unwrapping algorithms; each re-apply just puts those UVs for the selected faces somewhere else. Choose the tool for each part that gives you the best fit and most logical layout for subsequent painting of that part.

When all parts of the mesh have been unwrapped using the various methods, you should end up with something that looks like the example to the right. All sections of the mesh have been mapped, and all those maps are laid out in the same UV texture map. Congratulations! From here, it is a simple matter of "stitching" (discussed in the next section) to construct the entire UV map as a single map.



UV maps arranged together and stitched.

When you have completed arranging and stitching, you will end up with a consolidated UV map, like that shown to the right, arranged such that a single image will cover, or paint, all of the mesh that needs detailed painting. All the detailed instructions on how to do this are contained in the next section. The point of this paragraph is to show you the ultimate goal. Note that the mesh shown is Mirrored along the Z axis, so the right side of the face is virtual; it is an exact copy of the right, so only one set of UVs actually exist. (If more realism is desired, the Mirror Modifier would be applied, resulting in a physical mirror and a complete head. You could then make both side physically different by editing one side and not the other. Unwrapping would produce a full set of UVs (for each side) and painting could thus be different for each side of the face, which is more realistic).

Iteration & Refinement

At least for common people, we just do not "get it right the first time." It takes building on an idea and iterating our creative process until we reach that magical milestone called "Done". In software development, this is called the 'spiral methodology'.

Applied to computer graphics, we cycle between modeling, texturing, animating, and then back to making modifications to mesh, UV mapping, tweaking the animation, adding a bone or two, finding out we need a few more faces, so back to modeling, etc. We continue going round and round like this until we either run out of time, money, or patience, or, in rare cases, are actually happy with our results.

Refining the Layout

Refinement comes into play when we finally look at our character, and realize that we need more detail in a particular spot. For example, areas around the eyes might need crow's feet, or we need to add a logo to the vest. As you start to edit the image, you realize that there just are not enough pixels available to paint the detail that you want.

Your only choice is to expand the size (scale out) that UV face. Using the *Minimize Stretch* or *Scale* tools, you expand the UV faces around the eyes or chest, allocating more pixels to those areas, but at the same time taking away pixels (detail) from somewhere else, like the back of the head. After refining the UV map, you then edit the image so that it looks right and contains the details you want.

Reusing Textures

Another consideration is the need to conserve resources. Each image file is loaded in memory. If you can reuse the same image on different meshes, it saves memory. So, for example, you might want to have a generic face painting, and use that on different characters, but alter the UV map and shape and props (sunglasses) to differentiate.

You might want to have a "faded blue jeans" texture, and unwrap just the legs of characters to use that image. It would be good to have a generic skin image, and use that for character's hands, feet, arms, legs, and neck. When modeling a fantasy sword, a small image for a piece of the sword blade would suffice, and you would *Reset Unwrap* the sword faces to reuse that image down the length of the blade.

UDIMs

Using UV maps can have one disadvantage, they consist of one texture for the entire mesh. Most of the time this is sufficient but the disadvantage is that the texture is one resolution for the entire mesh. This causes issues if you have a very large mesh with geometry of different importance. When using a singular texture, the resolution might be too low to cover larger UV islands while being inefficient for smaller, less important islands.

UDIM offers a solution to this by being able to spread UV islands across several different textures. UDIM which stands for U DIMension is based on a tile system where each tile is a different texture in the overall UDIM texture array. Basically each tile consists of its own UV space (0-1, 1-2, 2-3) and have its own image assigned to that tile. Tiles are managed in the UDIM Tiles panel where they can have a generated image assigned to them. Generally, you create several textures of different resolutions; for example, you may have a 4k resolution texture for the major details, and 2k and/or 1k textures for less important details.

The UDIM array consists of one main tile, this tile is given the index number of `1001`. The next tile that gets added will be `1002` and will be placed to the right of the main tile. The overall UDIM array is ten tiles wide, so tiles `1001` through `1010` are created on the first row. After ten tiles a new row of tiles is started above the main tile; so `1011` will be place directly above `1001`.

Workflow

To start using a UDIM workflow, you should unwrap a mesh as you would for any other UV map. After that you should decide how many textures you want to split your UV map into. This will be different for every mesh and workflow but a good minimum is 3: one 4k, one 2k, and one 1k image. Then create the desired textures to match how many textures you want.

After this it is the same process of moving UVs to the appropriate tile and scaling and managing them like any other UV map. See Layout Workflow for information on laying out UVs.

When the UVs are correctly set up across the multiple UV islands it is time to add proper textures the UDIM array. Currently, existing textures cannot be added to a tile, to fill a tile with an existing texture you first must:

1. Create the desired tiles.
2. Save the image.
3. Replace the saved image file with the desire texture by deleting the file and replacing it with a new image file, keeping the old file name. Or by opening the image in another application and modifying the contents of the image.

Other than using a third-party application to edit the UDIM texture it is possible to paint on UDIM textures. This works for either 2D Painting or 3D Painting.

File Substitution Tokens

Substitution tokens are special sequences of characters in a filename that can be replaced with more meaningful and context aware information. In this case, tokens are identified by being text wrapped in angle bracket characters.

This substitution is used while load loading or saving an image to automatically identify the tile associated with a particular texture in the UDIM array.

The following tokens are supported:

- `<UDIM>`: A 4-digit notation calculated as `1001 + u-tile + v-tile * 10`.
- `<UVTILE>`: A notation defined as `u(u-tile + 1)_v(v-tile + 1)`.

Examples:

- `monster-basecolor.<UDIM>.png` will load/save files like `monster-basecolor.1021.png` etc.
- `monster-basecolor.<UVTILE>.png` will load/save files like `monster-basecolor.u1_v3.png` etc.

UDIM Tiles

Reference

Editor::	Image Editor, UV Editor
Mode::	All Modes
Panel::	*Sidebar ▸ Image ▸ UDIM Tiles*

In this panel UDIM tiles are managed; new tiles can be added, tiles can be removed, or tiles can filled with a generated texture.

UDIM Tile List
: List all UDIM tiles associated with the main index (`1000` tile). Double clicking on the tile name allows you to alter the tiles *Label*.

Add Tile +
: Adds new UDIM tiles to the group.

  Number
  : The starting tile index number. UDIMs must start with the `1001` tile and typically increase in incremental order.

  Count
  : The number of tiles to add.

  Label
  : An optional label can be used instead of the index number. These labels are shown in the 2D Viewport.

  Fill
  : Occupy the UDIM tile with a generated image; see *Fill Tile* below.

Remove Tile –

Deletes the selected UDIM tile from the group. If this tile is not saved and contains data, that data will be lost.

Fill Tile

Occupy the UDIM tile with a Generated Image.

Warning

If a tile is not filled, it will not be saved with the image.

Using UV Maps

Sooner or later, you may want to use an image texture on your model. The UV Editor allows you to map textures directly to the mesh faces. The 3D Viewport shows you the object being textured. If you set the 3D Viewport into Textured viewport shading, you will immediately see any changes made in the UV Editor, and vice versa. This is because no real rendering is taking place; it is all just viewport shading. If you were to apply an image to UVs then render, the texture would not show up by default. So to render an image, you must:

- Create a Material for the object.
- Tell Blender to use the UV textures on faces when rendering.

To create a Material, you have to click *Add New* Material in the Shading context.

There are two ways to tell Blender to use the UV texture when rendering; the quick way and the proper way.

The quick way is to use generated UV coordinates. Generated coordinates is used by default by all Texture nodes, with the exception of Image textures that use UV coordinates by default. To use generated coordinates for images as well use the *Generated* output of the Texture Coordinate node.

The proper way is to use UV Unwrapping to manually generate UV coordinates. To use UV mapping, use the Texture Coordinate node (UV output) or the UV Map node and select the UV map to use, “UVMap” by default.

Full details of using Image textures are on the Image Textures page.

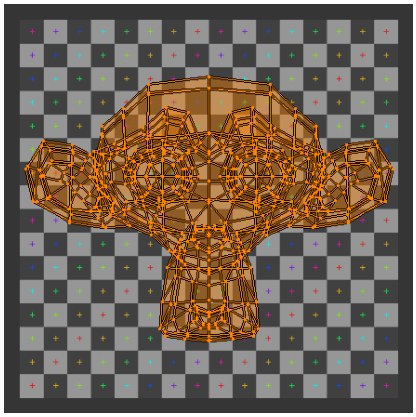
Note

Material is Required for Rendering

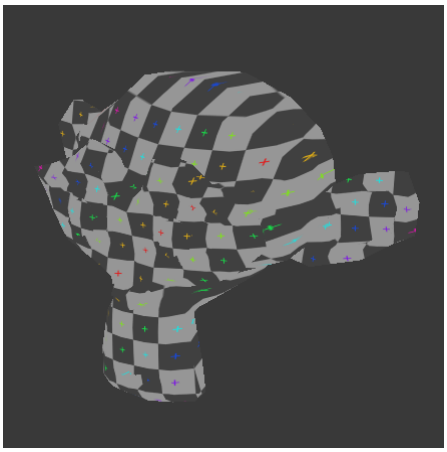
You can perform UV texturing on a mesh within Blender without assigning a material, and you will even see it in your 3D Viewport in textured viewport mode. However, when you render, you will just get a default gray if the object does not have a Material assigned. You will get a black if you do not load an image. If you do not create a texture that uses the image, your object will render according to the procedural material settings.

Using the Test Grid

If your image is a base uniform pattern and you want the application of that image to your model to look like cloth, you do **not** want any stretching (unless you want the cloth to look like spandex). You may also need to test your UV mapping with a test image:



The test grid applied to the UVs.



A preview of the texture on the geometry.

When you render, the mesh will have the test grid as its colors, and the UV texture will be the size image you specified. Note that Blender has a built-in test image. To use it, press the New button of the data-block menu in the Image editor header and change the *Generated Type* to UV Grid.

Modifying Your Image Texture

See also

- Render Bake
- Texture Paint

The advantage to saving as a separate file is that you can easily switch textures just by copying other image files over it, and you can use external editing programs to work on it. The advantage of packing is that your whole project is kept in the blend-file, and that you only have to manage one file.

Mesh Analysis

Reference

Mode::	Edit Mode
Panel::	*Header ▸ Overlays ▸ Mesh Analysis*

Mesh analysis is useful for displaying attributes of the mesh, that may impact certain use cases.

The mesh analysis works in *Edit Mode* and *Solid* Viewport shading. It shows areas with a high value in red, and areas with a low value in blue. Geometry outside the range is displayed gray.

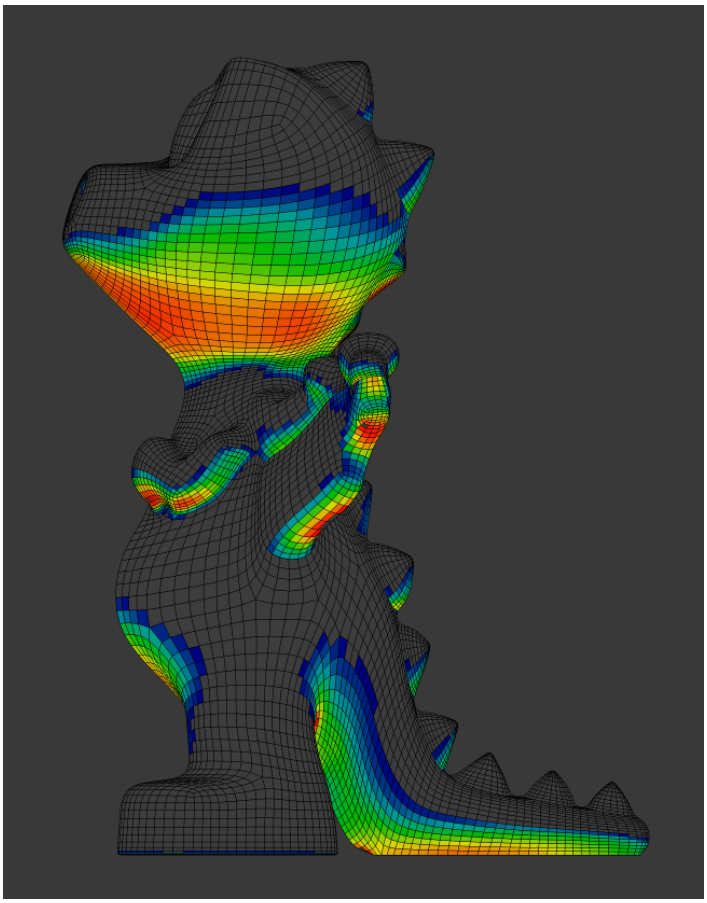
Currently the different modes target 3D printing as their primary use.

Overhang

Extrusion 3D printers have a physical limit to the overhang that can be printed, this display mode shows the overhang with angle range and axis selection.

Minimum/Maximum
: Minimum/Maximum angle to display.

Axis
: Axis and direction to use as the bases to calculate the angle to visualize.



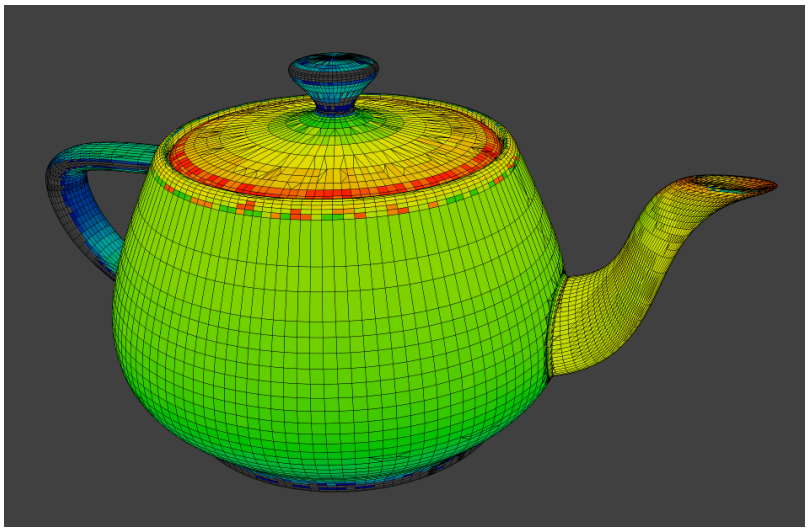
Overhang.

Thickness

Printers have a limited *wall-thickness* where very thin areas cannot be printed, this test uses ray casting and a distance range to the thickness of the geometry.

Minimum/Maximum
: Minimum/Maximum thickness to display.

Samples
: Number of samples to use to calculate the thickness.

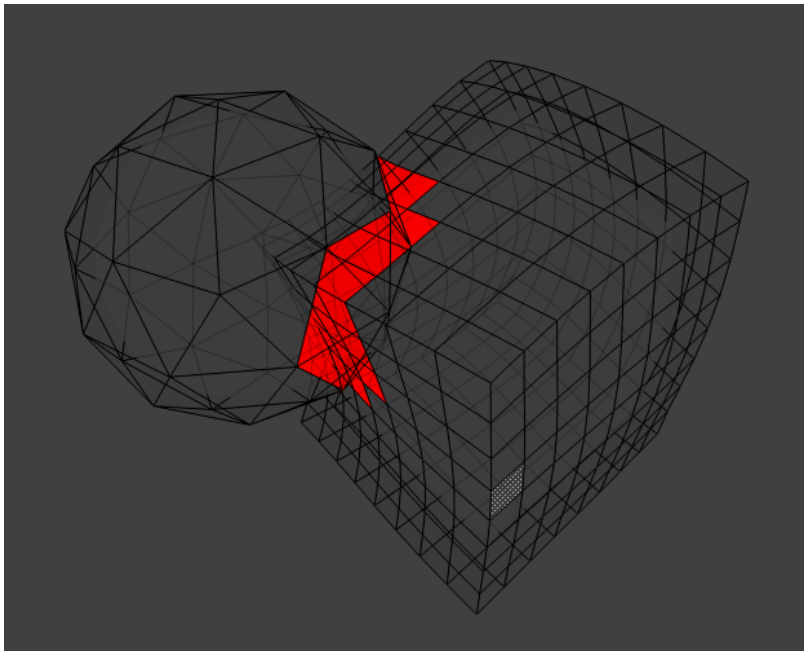


Thickness.

Intersections

Another common cause of problems for printing are intersections between surfaces, where the inside/outside of a model cannot be reliably detected.

Unlike other display modes, intersections have no variance and are either on or off.



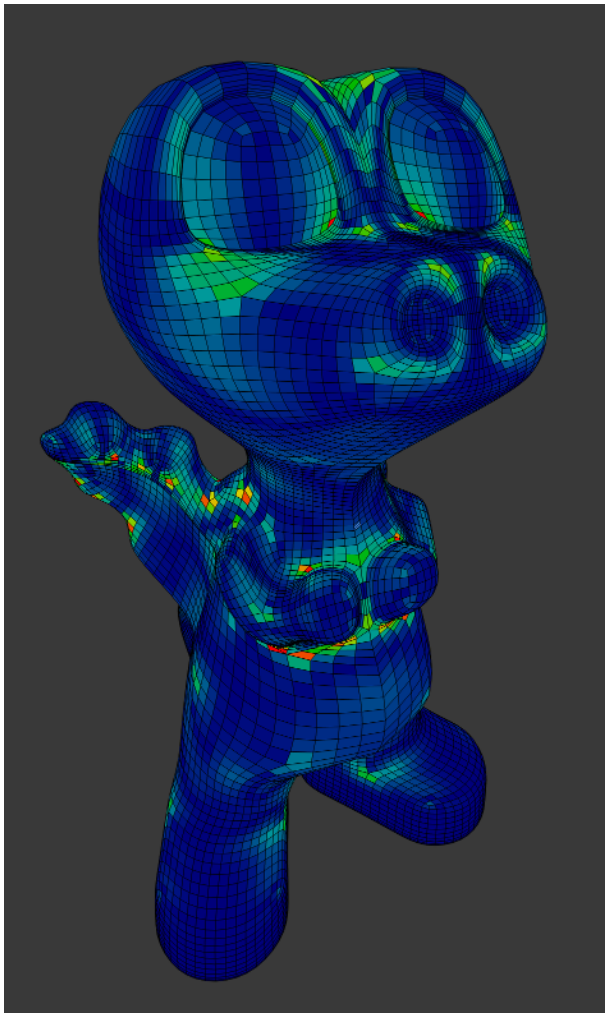
Intersecting faces.

Distortion

Distorted geometry can cause problems since the triangulation of a distorted n-gon is undefined.

Distortion is measured by faces which are not flat, with parts of the face pointing in different directions.

Minimum/Maximum
: Minimum/Maximum distortion to display.

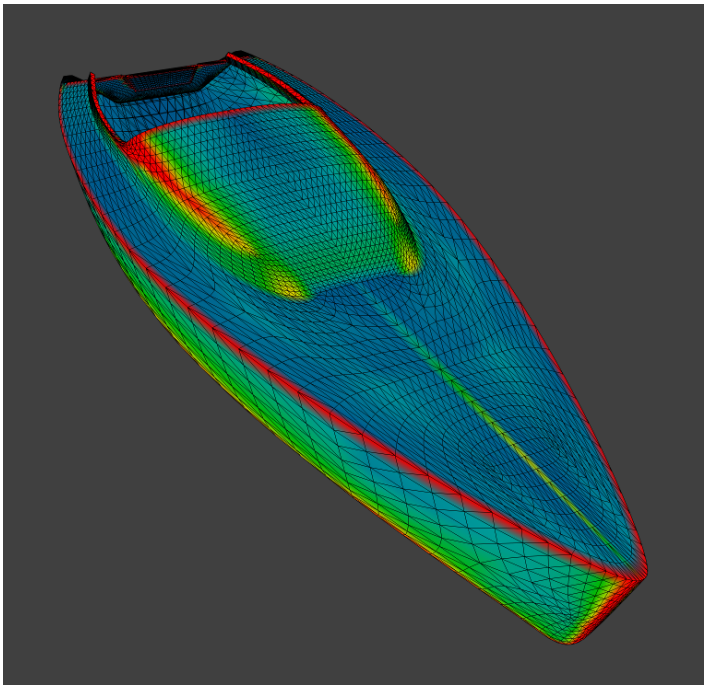


Distorted Faces.

Sharp Edges

Similar to wall-thickness, sharp edges can form shapes that are too thin to be able to print.

Minimum/Maximum
: Minimum/Maximum angle to display.



Sharp edges.

Known Limitations

There are some known limitations with mesh analysis:

- Currently only displayed with Deform Modifiers.
- For high-poly meshes the performance is low while editing.

Retopology

Retopology is the process of simplifying the topology of a mesh to make it cleaner and easier to work with. Retopology is needed for mangled topology resulting from sculpting or generated topology, for example from a 3D scan. Meshes often need to be retopologized if the mesh is going to be deformed in some way. Deformations can include rigging or physics simulations such as cloth or soft body. Retopology can be done by hand by manipulating geometry in Edit Mode or through automated methods.

Using the Poly Build Tool

Todo 2.81.

Remeshing

Reference

Mode::	Object Mode, Sculpt Mode
Panel::	*Properties* ▸ *Object Data* ▸ *Remesh*

Remeshing is a technique that automatically rebuilds the geometry with a more uniform topology. Remeshing can either add or remove the amount of topology depending on a defined resolution. This technique is especially useful for sculpting, to generate better topology after blocking out the initial shape.

Note

Limitations:

- Remeshing only works on the original mesh data and ignores generated geometry from modifiers, shape keys, rigging, etc.

- Remeshing will not work with the Multiresolution Modifier.

See also

Remesh modifier

Voxel

The Voxel Remesher uses OpenVDB to generate a new manifold mesh from the current geometry. It produces a mesh with perfectly even distributed topology and it does not have any performance penalty once the new mesh is calculated. This makes the voxel remesher great for sculpting as it is possible to sculpt at a much higher level of detail than using other features like dyntopo which often adds more performance overhead.

Voxel Size
: The resolution or the amount of detail the remeshed mesh will have. The value is used to define the size, in object space, of the Voxel. These voxels are assembled around the mesh and are used to determine the new geometry. For example a value of 0.5 m will create topological patches that are about 0.5 m (assuming *Preserve Volume* is enabled). Lower values preserve finer details but will result in a mesh with a much more dense topology.

Adaptivity
: Reduces the final face count by simplifying geometry where detail is not needed. This introduce triangulation to faces that do not need as much detail. Note, an *Adaptivity* value greater than zero disables *Fix Poles*.

Fix Poles
: Tries to produce less Poles at the cost of some performance to produce a better topological flow.

Preserve
: Volume

Tells the algorithm to try to preserve the original volume of the mesh. Enabling this could make the operator slower depending on the complexity of the mesh.

Paint Mask
: Reprojects the paint mask onto the new mesh.

Face Sets
: Reprojects Face Sets onto the new mesh.

Color Attributes
: Reprojects the Color Attributes onto the new mesh.

Voxel Remesh
: Performs the remeshing operation to create a new manifold mesh based on the volume of the current mesh. Performing this will lose all mesh object data layers associated with the original mesh.

Quad

The Quad remesh uses the Quadriflow algorithm to create a Quad based mesh with few poles and edge loops following the curvature of the surface. This method is relatively slow but generates a higher quality output for final topology.

Warning

Performing *Quadriflow Remesh* will lose all mesh object data layers associated with the original mesh.

Quadriflow Remesh
: Opens a pop-up used to set parameters for the remesh operation.

Use Paint Symmetry
: Generates a symmetrical mesh using the Mesh Symmetry options.

Preserve Sharp

Tells the algorithm to try to preserve sharp features of the mesh. Enabling this could make the operator slower depending on the complexity of the mesh.

Preserve Mesh Boundary

Tells the algorithm to try to preserve the original volume of the mesh. Enabling this could make the operator slower depending on the complexity of the mesh.

Preserve Paint Mask

Reprojects the Paint Mask onto the new mesh.

Smooth Normals

Applies the Smooth Normals operator to the resulting mesh.

Mode

How to specify the amount of detail for the new mesh.

Ratio::	Specify target number of faces relative to the current mesh.
Edge Length::	Input target edge length in the new mesh.
Faces::	Input target number of faces in the new mesh.

Seed

Random Seed to use with the solver; different seeds will cause the remesher to generate different quad layouts on the mesh.

Curves

- Introduction
- Tools
  - Toolbar
  - Types
- Structure
  - Splines
  - Spline Types
- Primitives
  - Bézier Curve
  - Bézier Circle
  - NURBS Curve
  - NURBS Circle
  - Path
- Selecting
  - Select Menu
  - All
  - None
  - Invert
  - Box Select
  - Circle Select
  - Lasso Select
  - Select Random
  - Checker Deselect
  - Select More/Less
  - Select Linked
  - Select Similar
  - (De)select First/Last
  - Select Next/Previous
  - Pick Shortest Path
- Editing
  - Transform Panel
  - Curve
  - Control Points

    - Segments
    - Other
- Properties
    - Shape
    - Geometry
    - Path Animation
    - Active Spline
- Curve Display

Introduction

Curves and Surfaces are particular types of Blender objects. They are expressed by mathematical functions (interpolation) rather than linear interpolation between a series of points.

Blender offers both Bézier and NURBS. Both Bézier curves and NURBS curves and surfaces are defined in terms of a set of “control points” (or “control vertices”) which define a “control polygon”.



Blender logo made from Bézier curves.

Both Bézier and NURBS curves are named after their mathematical definitions, and choosing between them is often more a matter of how they are computed behind the scenes than how they appear from a modeler’s perspective. Bézier curves are generally more intuitive because they start and end at the control points that you set, but NURBS curves are more efficient for the computer to calculate when there are many twists and turns in a curve.

The main advantage to using curves instead of polygonal meshes is that curves are defined by less data and so can produce results using less memory and storage space at modeling time. However, this procedural approach to surfaces can increase demands at render time.

Certain modeling techniques, such as extruding a profile along a path, are possible only using curves. On the other hand, when using curves, vertex-level control is more difficult and if fine control is necessary, mesh editing may be a better modeling option.

Bézier curves are the most commonly used curves for designing letters or logos.

They are also widely used in animation, both as for objects to move along (see constraints below) and as F-Curves to change the properties of objects as a function of time.

See also

Modifiers & Constraints

- Curve Modifier
- Follow Path Constraint
- Clamp To Constraint

Tools

- Toolbar

Types

- Draw
- Curve Pen

Toolbar

Curve Edit Mode tools:

Select
: Select or move.

  Select Box
  : Select objects by dragging a box.

    All objects that intersect the box will be selected.

  Select Circle
  : Select objects by dragging a circle. All objects that intersect the path of the circle will be selected.

  Select Lasso
  : Select objects by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform

Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line

Draw straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previous drawn annotations.

Measure

Measure distances in the scene.

Draw

Free-hand drawing of new curves.

Curve Pen

Construct and edit splines.

Extrude

Extrude the curve by adding new control points.

Radius

Control the radius value of the control points.

Tilt

Control the rotation value of the control points around the curve's axis.

Randomize

Move selected control points in pseudo-random directions.

Draw

Reference

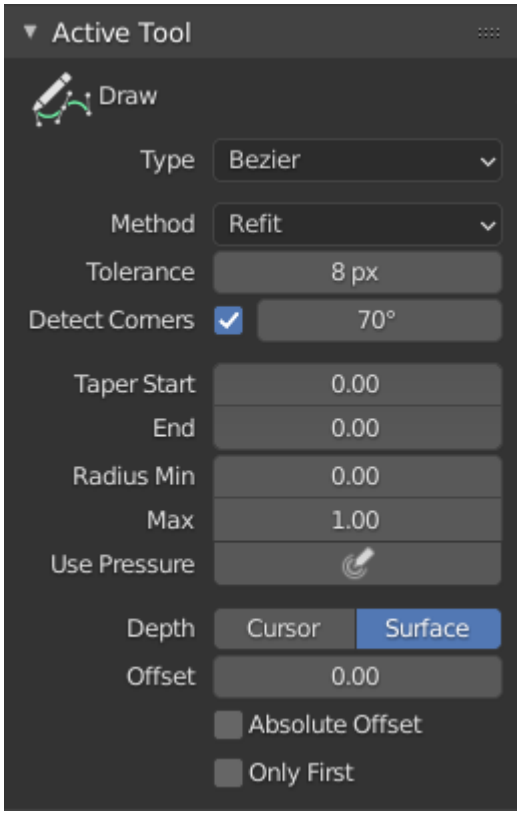
Mode:: Edit Mode
Tool:: *Toolbar ▸ Draw*

The Curve draw tool allows you to free-hand draw curves.

Tool Settings

Type
: Type of curve to use for drawing.

Poly::



Curve Stroke panel.

Bézier Curve with straight line segments (auto handles).

Bézier:: Tolerance

Lower values give a result that is closer to the drawing stroke, while higher values give more smoothed results.

Method

Refit:: Incrementally refits the curve (gives best results).

Split:: Splits the curve until the tolerance is met (gives a better drawing performance).

Detect Corners

Detects corners while drawing based on a specified angle; Any angles above the specified value are considered corners. If a corner is detected, the curve uses non-aligned handles for the corner resulting in a more crisp corner.

Taper Start, End

Taper factor for the radius of the start and end points along the curve.

Radius Min

Minimum radius when the minimum pressure is applied (also the minimum when tapering).

Max

Radius to use when the maximum pressure is applied (or when a tablet is not used).

Use Pressure

Uses stylus pressure to control the radius of the curve.

Depth

Controls where and how the curves are drawn.

Cursor:: Uses the depth under the cursor to draw curves.

Surface:: Used to draw on top of other objects.

Project Onto Selected
: Only project the strokes onto selected objects.

Offset
: Distance to offset the curve from the surface.

Absolute Offset
: Applies a fixed offset (does not scale by the curve radius).

Only First
: Only uses the start of the stroke for the depth.

Plane
: The orientation plane to draw on, available when *Only First* is enabled.

  Normal to Surface:: Draws aligned to the surface.

  Tangent to Surface:: Draws perpendicular to the surface.

  View:: Draws aligned to the viewport.

Options

After the tool is run, these options are available in the Adjust Last Operation panel.

Error
: Error distance in object units. This can be seen similar to a subdivision rate for the curve. Lower values give a result that is closer to the drawing stroke while higher values give more smoothed results.

Fit Method

Refit:: Incrementally refits the curve (gives best results).
Split:: Splits the curve until the tolerance is met (gives a better drawing performance).

Corner Angle
Any angles above this are considered corners.

Cyclic
Toggles whether or not the curve is Cyclic.

Curve Pen

Reference

Mode::	Edit Mode
Tool::	*Toolbar ‣ Curve Pen*

The Curve Pen tool allows you to construct and edit curves rapidly.

Usage

The following preferences can be configured from: *Preferences ‣ Keymap ‣ 3D View ‣ Curve ‣ 3D View Tool: Edit Curve, Curve Pen*.

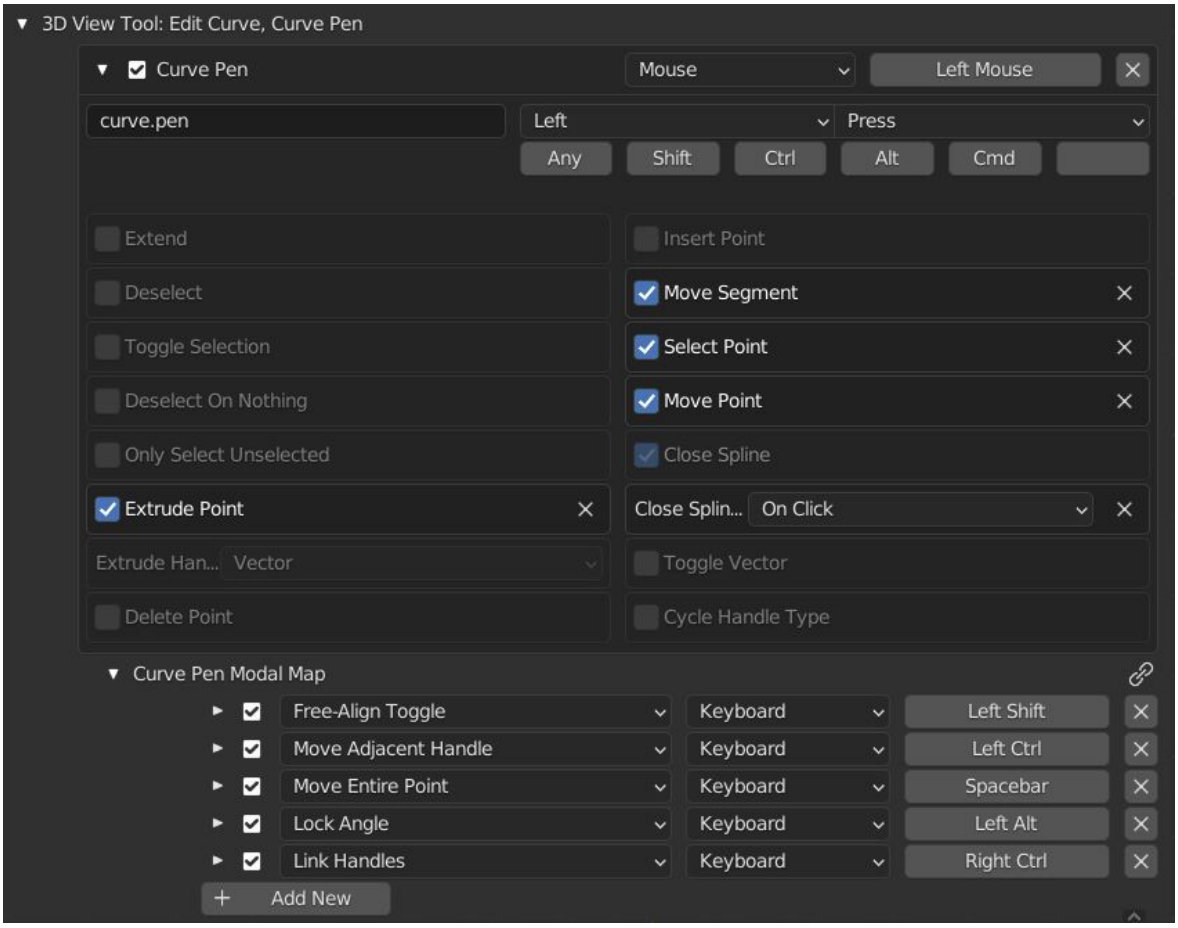
Extrude Point
: `LMB` click to add a new point connected to an existing point.

Extrude Handle Type
: The handle type of the extruded points. Can be either *Vector* or *Auto*. However, the handle type switches to *Align* when handles are moved (See *Move Point*).

Delete Point
: `Ctrl-LMB` click on an existing point to delete it.

Insert Point
: `Ctrl-LMB` click on a Curve Segment to insert a new control point between the two adjacent control points. `Ctrl-LMB` click and drag to control the handles of the inserted points.

Move Segment
: `LMB` drag on a segment in between two control points to adjust the handles, changing the shape of the curve without affecting the location of any control points.



Curve Pen Preferences

Select Point
: `LMB` click to select a single point or handle at a time.

Move point
: `LMB` drag to move existing points or handles. With an endpoint of a spline selected, click and drag on empty space to *Extrude Point* and move the handle at the same time.

Close Spline
: Make the spline Cyclic by clicking the endpoints consecutively.

Close Spline Method
: The condition for *Close Spline* to activate.

None:: Turn off the Close Spline functionality.

On Press:: Close the spline on mouse down. With this option, you may click and drag to adjust the handles of the endpoint.

On Click:: Activate on mouse release. With this option, the *Close Spline* functionality will not be triggered on click and drag.

Toggle Vector

Double `LMB` click on a handle to switch handle between *Vector* and *Auto* handle types. Can be used to easily switch between sharp corners and smooth curves.

Cycle Handle Type

Double `LMB` click on the control point to cycle through all handle types.

Hotkeys

Free-Align Toggle

Hold `LeftShift` while dragging a handle to switch between `Free` and `Align` handle types. Can be used to create sharp corners along the curve.

Move Adjacent Handle

Hold `LeftCtrl` while dragging a handle to move the closer handle of the adjacent control point. Can be helpful to make adjustments to newly created curve segments.

Move Entire

Hold `Spacebar` while dragging a handle to move the entire point.

Link Handles

Press `RightCtrl` while dragging a handle to mirror its movement on the opposite handle of the same point.

Lock Handle Angle

Hold `LeftAlt` while dragging a handle to limit the movement of the handle to its current direction, so only its length can be adjusted.

Structure

Splines

Splines are a substructure of curves and are the individual elements that make curve objects. A curve object can be composed of several different splines, just like mesh objects can have different discontinuous meshes under the same object. A spline defines the shape of the curve and can be transformed by altering its Control Points. Splines come in three distinct types, which are described in Spline Types. Each spline type has a slightly different algorithm for computing how bends in the spline are represented.

Splines have their own separate properties from curves and can be altered by selecting the spline in Edit Mode and using the Active Spline panel.

Control Points

Similar to meshes splines have control points or vertices. Control points connect to other control points to form splines. Control points can be selected and transformed to alter the resulting shape of the spline.

See also

Curve Editing

Spline Types

Poly

Poly splines are the most simple spline type as they do not interpolate the shape of the curve between control points.

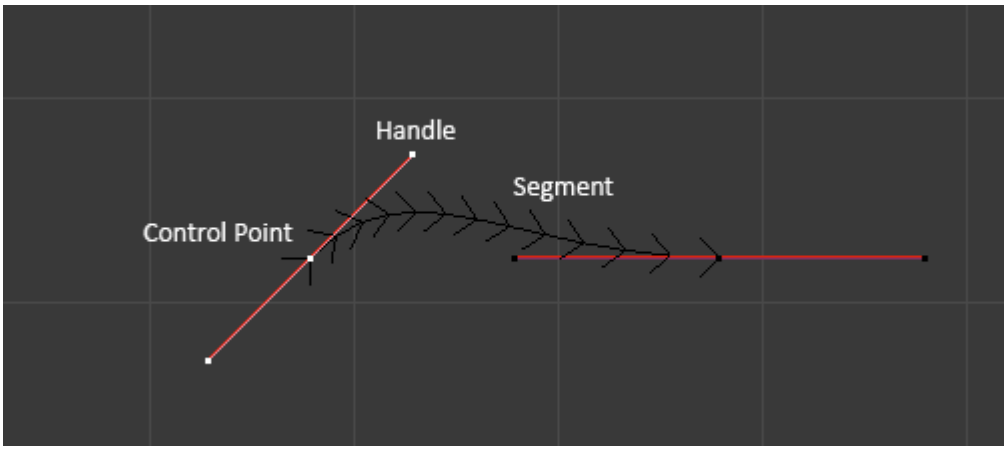
Poly Curves are used when converting meshes to curves. Because they do not interpolate the shape, Poly splines are able to give an accurate representation of the original mesh object.

This is the primary use case of splines, most of the time Bézier or NURBS splines are used instead; both of which interpolate the shape and give smooth results.

Bézier

The main elements used in editing Bézier curves are the control points and handles. A segment (the actual curve) is found between two control points. The handles define the curvature of the segment.

In the image below, the control points can be found in the middle of the pink line, while the handles comprise the extensions from the control point. The arrows visualize the normals of the curve, which indicate i.e. the direction and the tilt.



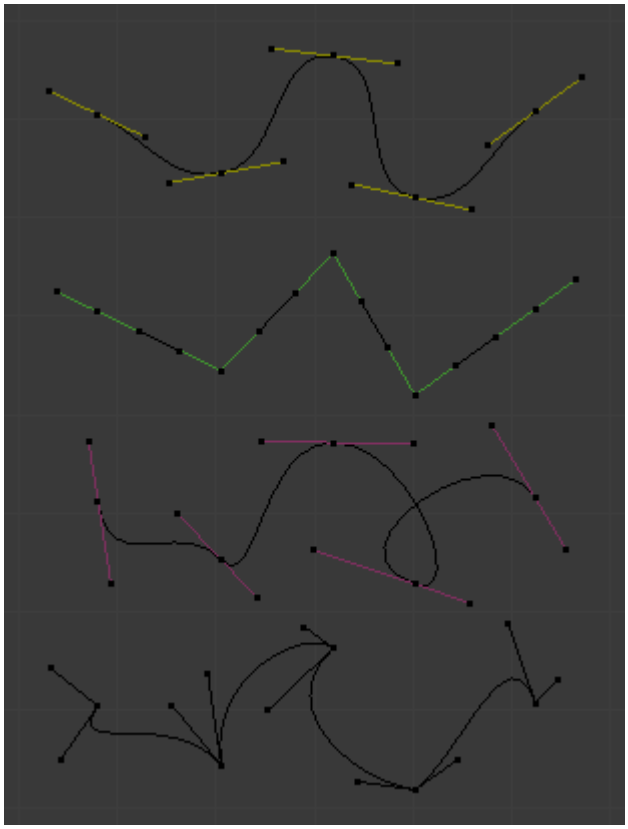
Bézier Curve in Edit Mode.

Handle Types

There are four Bézier curve handle types. They can be accessed by pressing `V` and selecting from the list that appears.

Automatic (yellow handles)
: This handle has a completely automatic length and direction which is set by Blender to ensure the smoothest result. These handles convert to *Aligned* handles when moved.

Vector (green handles)
: Both parts of a handle always point to the previous handle or the next handle which allows you to create curves or sections thereof made of straight lines or with sharp corners. Vector handles convert to *Free* handles when moved.



Bézier Curve Handle Types.

Aligned (purple handles)
: These handles always lie in a straight line, and give a continuous curve without sharp angles.

Free (black handles)
: The handles are independent of each other.

Note

When a control point is selected, its handles are overlaid with a red hilight that changes their color. For example, Vector handles that are normally green will look yellow when selected, which can make it difficult to distinguish them from unselected Automatic handles.

If you prefer not to have this color change, you can disable it by opening the Theme preferences, finding the entry *3D Viewport* ▸ *Active Spline*, and

changing it to black. From then on, selected curve handles will simply look a bit brighter than unselected ones.

NURBS

N.U.R.B.S. is the abbreviation for Non-Uniform Rational B-Splines. One of the major differences between Bézier objects and NURBS objects is that Bézier curves are approximations. For example, a Bézier circle is an *approximation* of a circle, whereas a NURBS circle is an *exact* circle. NURBS theory can be a *very* complicated topic. For an introduction, please consult the Wikipedia page.

NURBS spline control points are different than other spline types because they have a special weight property. This weight property controls how much influence the control point has on the surface. This weight should not be confused with the Goal Weight, which is used only for soft body simulations. The NURBS control point weight can be adjusted in the *W* number field of the Transform panel.

Note

If all the control points have the same *Weight* then each effectively cancels each other out. It is the difference in the weights that cause the curve to move towards or away from a control point.

Primitives

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Add* ▸ *Curve*
Shortcut::	`Shift-A`

See also

When adding curves there are some common options like other Objects.

Note

Eventually all the primitive curves will be replaced to use the same curve system used for hair curves. Until this is done, their features will diverge.

They can be converted interchangeably to access the full range of edit and sculpting functionalities.

In Object/Edit Mode, the *Add Curve* menu, provides a few different curve primitives:

Bézier Curve

Adds an open 2D Bézier curve with two control points.

Bézier Circle

Adds a closed, circle-shaped 2D Bézier curve (made of four control points).

NURBS Curve

Adds an open 2D NURBS curve, with four control points, with *Uniform* knots.

NURBS Circle

Adds a closed, circle-shaped 2D NURBS curve (made of eight control points).

Path

Adds a NURBS open 3D curve made of five aligned control points, with *Endpoint* knots and the *Curve Path* setting enabled.

Selecting

This page discusses specific selecting tools for curve objects in Edit Mode. The Curve Edit more also uses the general select tools used which are described in the interface section.

Curve selection in *Edit Mode* has fewer options than with meshes. Mainly this is, because there is only one selectable element type, the control points (no select mode needed here…). These points are a bit more complex than simple vertices, however, especially for Bézier curves, as there is the central vertex, and its two handles…

The basic tools are the same as with meshes, so you can select a simple control point with the `LMB`, add to current selection with `Shift-LMB`, Box Select `B`, and so on.

One word about the Bézier control points: when you select the main central vertex, the two handles are automatically selected too, so you can move it as a whole, without creating an angle in the curve. However, when you select a handle, only this vertex is selected, allowing you to modify this control vector…

Note that, unlike mesh edges, you cannot directly select a segment. Instead, select all of the control points that make up the segment you want to edit.

Select Menu

With curves, all "advanced" selection options are grouped in the *Select* menu of the 3D Viewport header.

All

Reference

Mode::	Edit Mode
Menu::	*Select ▸ All*
Shortcut::	`A`

Select all selectable elements.

None

Reference

Mode::	Edit Mode
Menu::	*Select ▸ None*
Shortcut::	`Alt-A`

Deselect all elements, but the active element stays the same.

Invert

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Invert*
Shortcut::	`Ctrl-I`

Selects all the geometry that are not selected, and deselect currently selected components.

Box Select

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Box Select*
Shortcut::	`B`

Interactive box selection.

Circle Select

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Circle Select*
Shortcut::	`C`

Interactive circle selection.

Lasso Select

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Lasso Select*
Shortcut::	`Ctrl-Alt-LMB`

See Select Lasso.

Select Random

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Random*

Select Random control points.

Percent
: Selects the defined percentage of control points.

Random Seed

Seed used by the pseudo-random number generator.

Action
: Controls whether the operator *Selects* or *Deselects* control points.

Checker Deselect

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Checker Deselect*

This tool applies an alternating selected/deselected checker pattern. This only works if you already have more than one control point selected.

It works by changing the current selection so that only every Nth control points will remain selected, starting from the active one.

Deselected
: The number of deselected elements in each pattern repetition.

Selected
: The number of selected elements in each pattern repetition.

Offset
: Offset from the starting point.

Select More/Less

Reference

Mode::	Edit Mode
Menu::	*Select ▸ More/Less*
Shortcut::	`Ctrl-NumpadPlus`, `Ctrl-NumpadMinus`

Their purpose, based on the currently selected control points, is to reduce or enlarge this selection.

More
: For each selected control point, select *all* its linked points (i.e. one or two…).

Less
: For each selected control point, if *all* points linked to this point are selected, keep this one selected. Otherwise, deselect it.

This implies two points:

1. When *all* control points of a curve are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no control points are selected.
2. Second, these tools will never "go outside" of a curve (they will never "jump" to another curve in the same object).

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Linked*
Shortcut::	`L`, `Ctrl-L`, `Shift-L`

`L` (or `Ctrl-L` for all) will add to the selection the cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same curve. Note that for Bézier, using `L` with a handle selected will select the whole control point and all the linked ones.

Select Similar

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Similar*
Shortcut::	`Shift-G`

Selects control points that have certain similar properties to the active one. The Adjust Last Operation panel provides several selection options:

Type
: Type
  : Selects splines that have the same spline Type i.e. Bézier, NURBS or Poly.

: Radius
  : Selects control points that have a similar Radius value.

: Weight
  : Selects all points that have a similar Weight value.

: Direction
  : Selects control points that have a similar handles direction.

Compare
: For quantitative properties, this property selects the type of comparison to between the two numerical values.

Equal::	Select items with the same value as the active item's chosen property.
Greater::	Select items with a larger value as the active item's chosen property.
Less::	Select items with a smaller value as the active item's chosen property.

Threshold
: For quantitative properties, this property controls how close the property's values have to be in the comparison.

(De)select First/Last

Reference

Mode::	Edit Mode
Menu::	*Select ▸ (De)select First*, *Select ▸ (De)select Last*

These operators will toggle the selection of the first or last control point(s) of the curve(s) in the object. This is useful to quickly find the start of a curve (e.g. when using it as path…).

Select Next/Previous

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Next*, *Select ▸ Select Previous*

These operators will select the next or previous control point(s), based on the current selection (i.e. the control points following or preceding the selected ones along the curve). In case of a cyclic curve, the first and last points are not considered as neighbors.

Pick Shortest Path

Reference

Mode::	Edit Mode
Menu::	*Menu Search ▸ Pick Shortest Path*
Shortcut::	`Ctrl-LMB`

Selects the curve segments between two control points: the active and the one under the cursor. In the case of a closed curve, the shortest path will be selected.

Editing

- Transform Panel
- Curve
  - Transform
  - Mirror
  - Snap
  - Spin
  - Add Duplicate
  - Split
  - Separate
  - Toggle Cyclic
  - Set Spline Type
  - Show/Hide
  - Cleanup
  - Delete
- Control Points
  - Extrude Curve and Move
  - Make Segment
  - Tilt
  - Clear Tilt
  - Set Handle Type
  - Recalculate Handles
  - Smooth
  - Smooth Curve Tilt
  - Smooth Curve Radius
  - Smooth Curve Weight
  - Hooks
  - Make Vertex Parent
- Segments
  - Subdivide
  - Switch Direction
- Other
  - Set Goal Weight
  - Add Vertex

Transform Panel

Reference

Mode::	Edit Mode
Panel::	*Sidebar ▸ Transform*

When nothing is selected, the panel is empty. When more than one vertex is selected, the median values are edited and "Median" is added in front of the labels.

Control Point, Vertex
: The first controls (X, Y, Z) show the coordinates of the selected point or handle (vertex). In case of a NURBS curve, there is a fourth component available (W), which defines the weight of the selected control point or the median weight.

Space
: The Space radio buttons let you choose if those coordinates are relative to the object origin (local) or the global origin (global).

  Global, Local

Weight
: Controls the "goal weight" of selected control points, which is used when a curve has Soft Body physics, forcing the curve to "stick" to their original positions, based on the weight.

Radius
: Controls the width of the extrusion/bevel along the "spinal" curve. The radius will be interpolated from point to point (you can check it with the normals).

Tilt

Controls how the normals (visualized as arrows) twist around each control point – so it is only relevant with 3D curves! The tilt will be interpolated from point to point (you can check it with the normals).

Curve

This page covers the basics of curve editing.

Transform

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Transform*

A Bézier curve can be edited by transforming the locations of both control points and handles. NURBS curve on the other hand have only control points.

Move, Rotate, Scale
: Like other elements in Blender, curve control points and handles can be moved, rotated, or scaled as described in Basic Transformations.

To Sphere, Shear, Bend, Push/Pull, Warp, Randomize
: The transform tools are described in the Transformations sections.

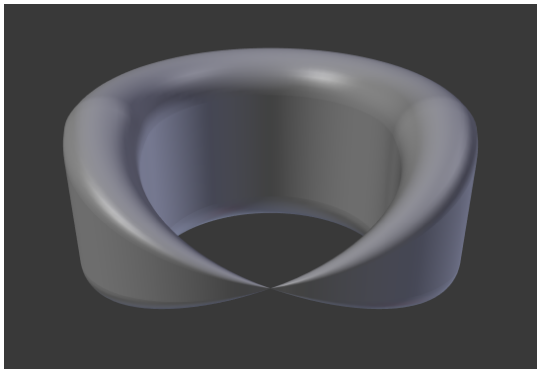
Move/Scale Texture Space
: Like other objects, curves have textures spaces which can be edited.

Radius

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Radius*
Menu::	*Curve* ▸ *Transform* ▸ *Radius*
Shortcut::	`Alt-S`

Controls the width of the extrusion along the “spinal” curve. The radius will be interpolated from point to point (you can check it with the normals). The *Radius* of the points is set using the *Radius* transform tool. Or in the Sidebar *Transform* panel.



One control point radius set to zero.

Mirror

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Mirror*
Shortcut::	`Ctrl-M`

The *Mirror* tool is also available, behaving exactly as with mesh vertices.

Snap

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Snap*
Shortcut::	`Shift-S`

Mesh snapping also works with curve components. Both control points and their handles will be affected by snapping, except for within itself (other components of the active curve). Snapping works with 2D curves but points will be constrained to the local XY axes.

Spin

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Spin*

The *Spin* operator only works for one dimensional surface objects. Its use for curves is currently not possible, the full feature is documented in Surface editing.

Add Duplicate

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Add Duplicate*
Shortcut::	`Shift-D`

Duplicates the selected control points, along with the curve segments implicitly selected (if any). If only a handle is selected, the full point will be duplicated too. The copy is selected so you can move it to another place.

Split

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Split*

Shortcut:: `Y`

Splits a selected segment of a curve from the rest of the curve. This curve can then be moved or altered without affecting the other curve. If a single control point is selected the *Split* operator will create a new singular loose control point; leaving the previously selected control point attached to the rest of the curve.

Separate

Reference

Mode:: Edit Mode
Menu:: *Curve ▸ Separate*
Shortcut:: `P`

Separates curve objects that are made of multiple distinct curves into their own objects.

Note, if there is only one curve in a Curve object, This operation will create a new Curve object with no control points.

Toggle Cyclic

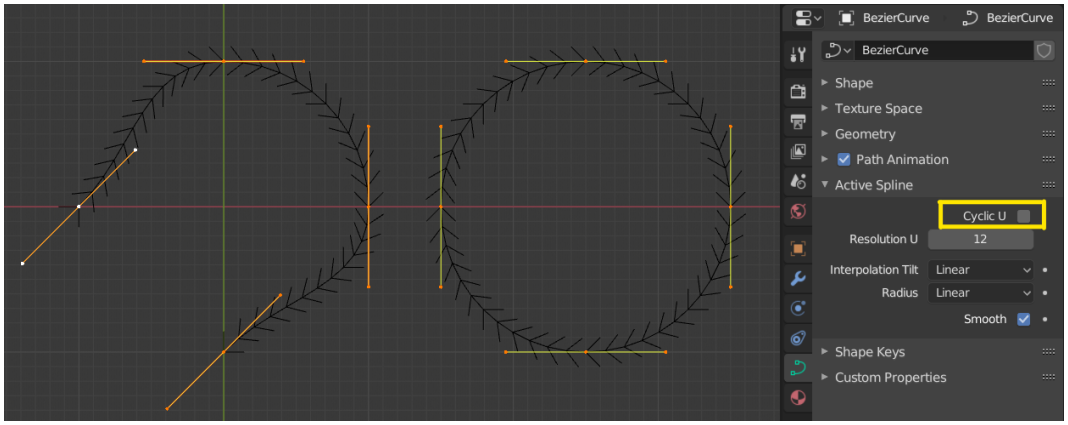
Reference

Mode:: Edit Mode
Menu:: *Curve ▸ Toggle Cyclic*
Shortcut:: `Alt-C`

Toggles between an open curve and closed curve (Cyclic). Only curves with at least one selected control point will be closed/open. The shape of the closing segment is based on the start and end handles for Bézier curves, and as usual on adjacent control points for NURBS. The only time a handle is

adjusted after closing is if the handle is an *Auto* one. Fig. Open and Closed curves. is the same Bézier curve open and closed.

This action only works on the original starting control point or the last control point added. Deleting a segment(s) does not change how the action applies; it still operates only on the starting and last control points. This means that `Alt-C` may actually join two curves instead of closing a single curve! Remember that when a 2D curve is closed, it creates a renderable flat face.



Open and Closed curves.

Set Spline Type

Reference

Mode:: Edit Mode
Menu:: *Curve ▸ Set Spline Type*

Converts splines in a curve object between Bézier, NURBS, and Poly curves. Note, this is not a "smart" conversion, i.e. Blender does not try to keep the same shape, nor the same number of control points. For example, when converting a NURBS to a Bézier, each group of three NURBS control points become a unique Bézier one (center point and two handles).

See also

Convert/from Mesh.

Show/Hide

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Show/Hide*
Shortcut::	`Alt-H`, `H`, `Shift-H`

When in *Edit Mode*, you can hide and reveal elements from the display. You can only show or hide control points, as segments are always shown, unless all control points of the connected curve are hidden, in which case the curve is fully hidden.

See Show/Hide in *Object Mode*. See also the Curve Display panel.

Cleanup

Decimate Curve

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Clean Up ▸ Decimate Curve*

The *Decimate Curve* operator reduces the number of control points while trying to maintain the curves original shape. This operator works similar to its mesh counterpart.

Ratio
: The percentage of control points to remove.

Note

This tool can only decimate Bézier curves.

Delete

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Delete*
Shortcut::	`X`, `Delete`

Options for the *Delete* pop-up menu:

Vertices
: This will delete the selected control points, *without* breaking the curve (i.e. the adjacent points will be directly linked, joined, once the intermediary ones are deleted). Remember that NURBS order cannot be higher than its number of control points, so it might decrease when you delete some control point. Of course, when only one point remains, there is no more visible curve, and when all points are deleted, the curve itself is deleted.

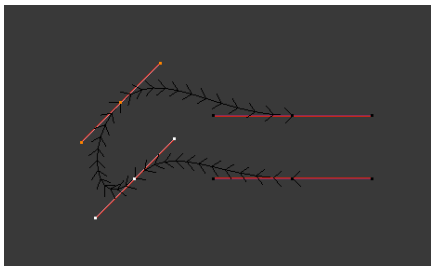
Segment
: Deletes the segment that connects the selected control points and disconnecting them.

Dissolve Vertices

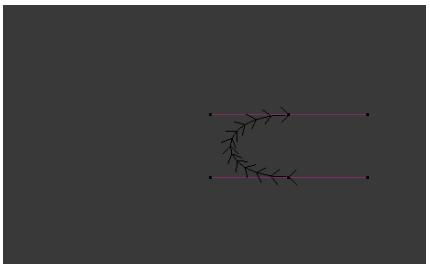
Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Delete ▸ Dissolve Vertices*
Shortcut::	`Ctrl-X`

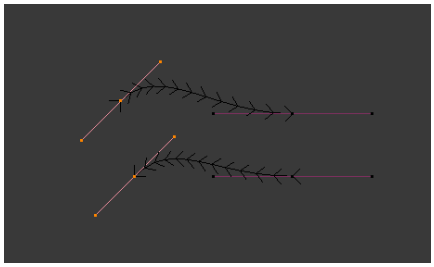
Deletes the selected control points, while the remaining segment is fitted to the deleted curve by adjusting its handles.



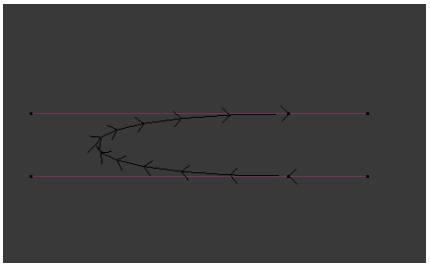
Before deleting.



Deleting vertices.



Deleting segment.



Dissolve vertices.

Control Points

Extrude Curve and Move

Reference

Mode::	Edit Mode
Menu::	*Control Points ▸ Extrude Curve and Move*
Shortcut::	E

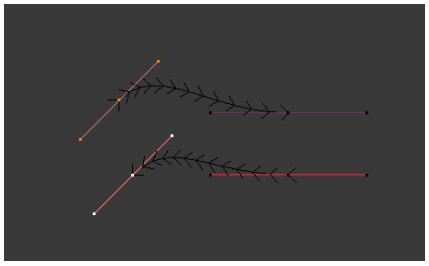
Extrudes points by duplicating the selected points, which then can be moved, and connecting those points back to the original curve creating a continuous curve.

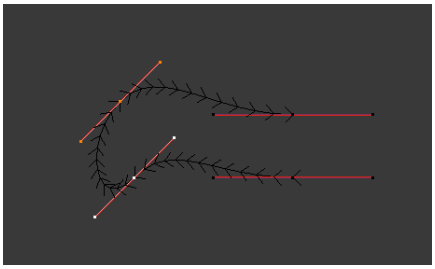
Make Segment

Reference

Mode::	Edit Mode
Menu::	*Control Points ▸ Make Segment*
Shortcut::	F

Connects two disconnected control points. The selection must be loose points, or the first/last point of a curve, then press F. If the points belong to different curves, these are joined by a segment to become a single curve.





Two curves before. Curve after joining.

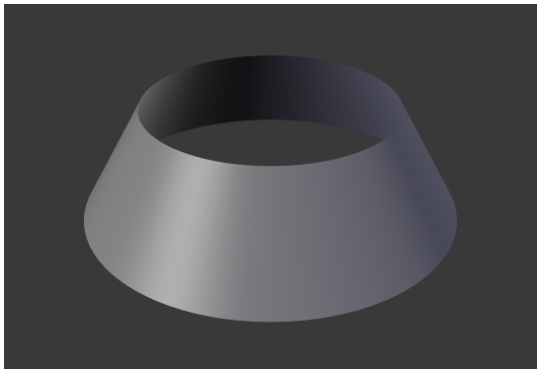
Note that you can only join curves of the same type (i.e. Bézier with Bézier, NURBS with NURBS). Additionally, you can close a curve by toggling cyclic.

Tilt

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Tilt*
Menu::	*Control Points* ▸ *Tilt*
Shortcut::	`Ctrl-T`

This setting controls how the normals (visualized as arrows) twist around each control point – so it is only relevant with 3D curves! The tilt will be interpolated from point to point (you can check it with the normals).



30 degree Mean Tilt of all control points.

Clear Tilt

Reference

Mode::	Edit Mode

Menu::	*Control Points* ▸ *Clear Tilt*
Shortcut::	`Alt-T`

You can also reset the tilt to its default value (i.e. perpendicular to the original curve plane). With NURBS, the tilt is always smoothly interpolated. However, with Bézier, you can choose the interpolation algorithm.

Set Handle Type

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Set Handle Type*
Shortcut::	`V`

Handle types are a property of Bézier curves and can be used to alter features of the curve. For example, switching to *Vector handles* can be used to create curves with sharp corners. Read the Bézier curves page for more details.

Toggle Free/Align
: Additionally, this operator can be used to toggle between Free and Aligned handle types.

Recalculate Handles

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Recalculate Handles*
Shortcut::	`Shift-N`

The *Recalculate Handles* operator rotates the selected control point's handle to be tangential to the curve. This can be used to make curves smoother and

more consistent looking.

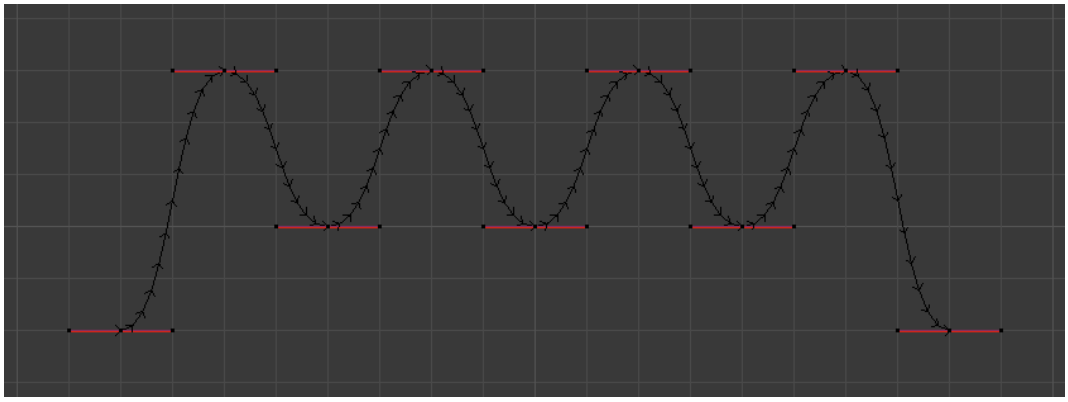
Length
: Recalculates the length of the handles so they are all the same length.

Smooth

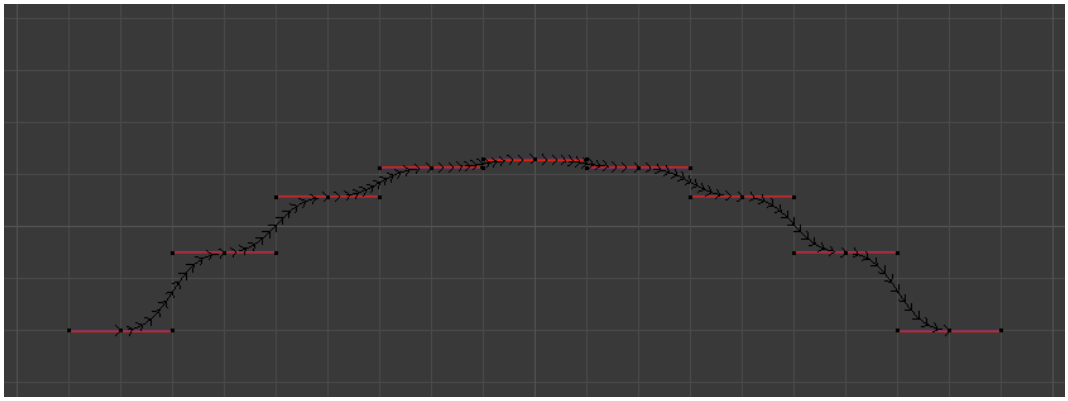
Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Smooth*

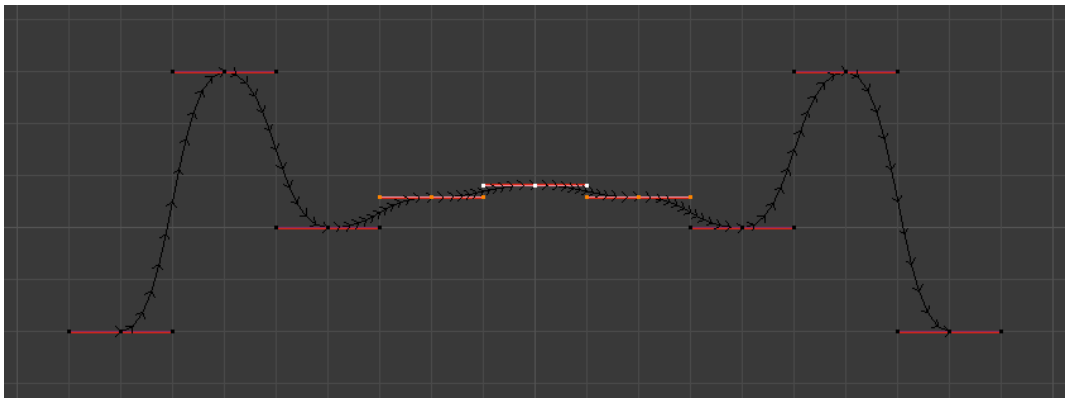
For Bézier curves, this smoothing operation reduces the distance between the selected control point(s) and their neighbors, while keeping the neighbors anchored. Does not effect control point tangents.



Original, unsmoothed Curve.



Entire curve smoothed over 20 times by holding `Shift-R` to repeat last step.



Only three control points in the center smoothed over 20 times.

Smooth Curve Tilt

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Smooth Curve Tilt*

The *Smooth Curve Tilt* operator interpolates the *Tilt* value for the selected control points. This will reduce sharp changes in the curve's *Tilt* and give a smooth transition between points.

Smooth Curve Radius

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Smooth Curve Radius*

The *Smooth Curve Radius* operator interpolates the *Radius* value for the selected control points. This will reduce sharp changes in the curve's *Radius* and give a smooth transition between points.

Smooth Curve Weight

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Smooth Curve Weight*

The *Smooth Curve Weight* operator interpolates the *Weight* value for the selected control points. This will reduce sharp changes in the curve's *Weight* and give a smooth transition between points.

Hooks

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Hooks*
Shortcut::	`Ctrl-H`

Hooks can be added to control one or more points with other objects.

Make Vertex Parent

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Make Vertex Parent*
Shortcut::	`Ctrl-P`

You can make other selected objects children of one or three control points, as with mesh objects.

To select a mesh (that is in view) while editing a curve, `Ctrl-P` click on it. Select either one or three control points, then `Ctrl-LMB` the object and use

`Ctrl-P` to make a vertex parent. Selecting three control points will make the child follow the median point between the three vertices. An alternative would be to use a Child Of constraint. See also the Curve modifier.

Segments

Subdivide

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Segments ▸ Subdivide*

Curve subdivision simply subdivides all selected segments by adding one or more control points between the selected segments.

Number of Cuts
: The number of subdivisions to perform.

Switch Direction

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Segments ▸ Switch Direction*

This tool will “reverse” the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). This is mainly useful when using a curve as path, or using the bevel and taper options.

Other

This page describes other curve editing tools that are not accessible via the edit menus.

Set Goal Weight

Reference

Mode::	Edit Mode
Menu::	*Context Menu* ‣ *Set Goal Weight*

Sets the curve's Weight for the selected control point to the specified value. If more than one control point is selected this will set the *Mean Weight*.

Add Vertex

Reference

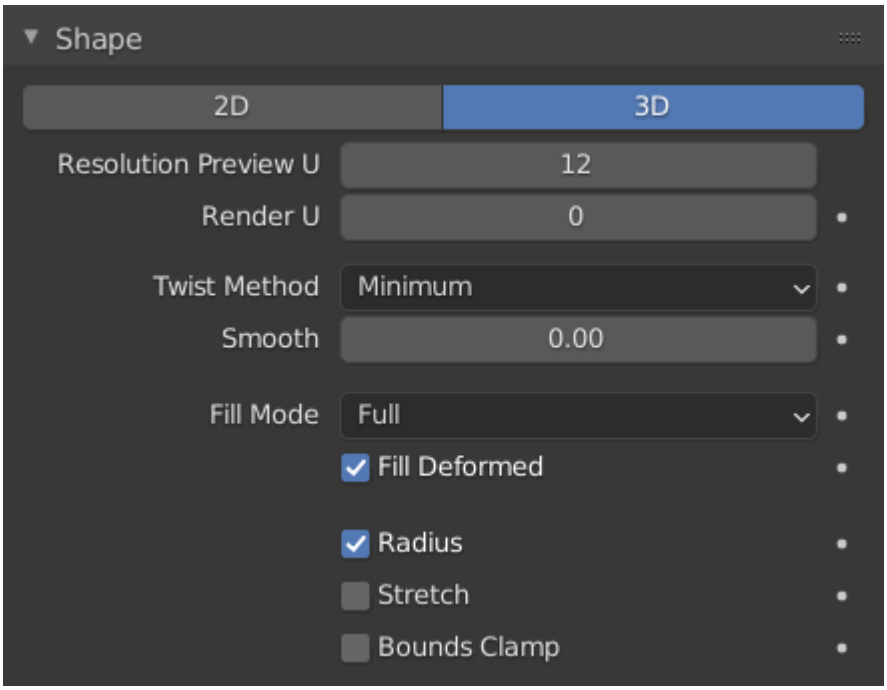
Mode::	Edit Mode
Shortcut::	`Ctrl-RMB`

Interactively places new points with `Ctrl-RMB` at the cursor position. With the selection it deals in same manner as the *Extrude Curve and Move* tool.

Properties

- Shape
- Geometry
  - Bevel
  - Start & End Mapping
  - Examples
- Path Animation
  - Example
- Active Spline
  - Poly
  - Bézier
  - NURBS

Shape



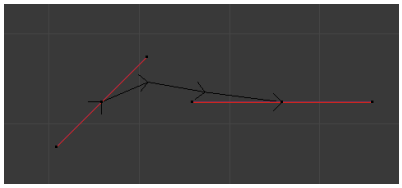
Shape panel.

Dimensions

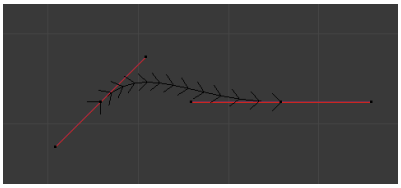
By default, new curves are set to be 3D, which means that control points can be placed anywhere in 3D space. Curves can also be set to 2D which constrain the control points to the curve's local XY axis.

Resolution Preview/Render U

The *resolution* property defines the number of points that are computed between every pair of control points. Curves can be made more or less smooth by increasing and decreasing the resolution respectively. The *Preview U* setting determines the resolution in the 3D Viewport while the *Render U* setting determines the curve's render resolution. If *Render U* is set to zero (0), then the *Preview U* setting is used for both the 3D Viewport and render resolution.



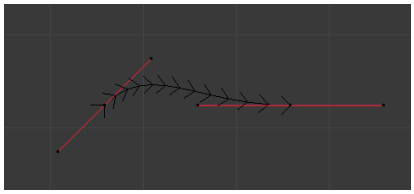
Curves with a resolution of 3.



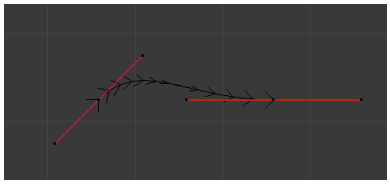
Curves with a resolution of 12.

Twist Method

A 3D curve has control points that are not located on the curve's local XY plane. This gives the curve a twist which can affect the curve normals. You can alter how the twist of the curve is calculated by choosing from *Minimum, Tangent* and *Z-Up* options from the select menu.



Curves with a twist of Minimum.



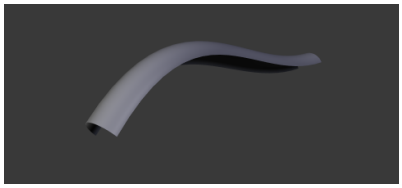
Curves with a twist of Tangent.

Smooth

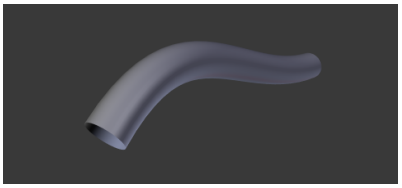
Interactively removes twists from the curve. This is useful if a curve has noticeable "kinks" from over twisting; which can be possible when converting meshes to curves.

Fill Mode

Fill determines the way a curve is displayed when it is beveled (see below for details on Beveling). When set to *Half* the curve is displayed as half a cylinder.



Curves with a fill of Half.



Curves with a fill of Full.

Fill Deformed

Fills the curve after applying all modifications that might deform the curve (i.e. shape keys and modifiers).

Curve Deform

Radius

Causes the deformed object to be scaled by the set curve radius. Utilized when using a curve as a path or when using the Curve Modifier.

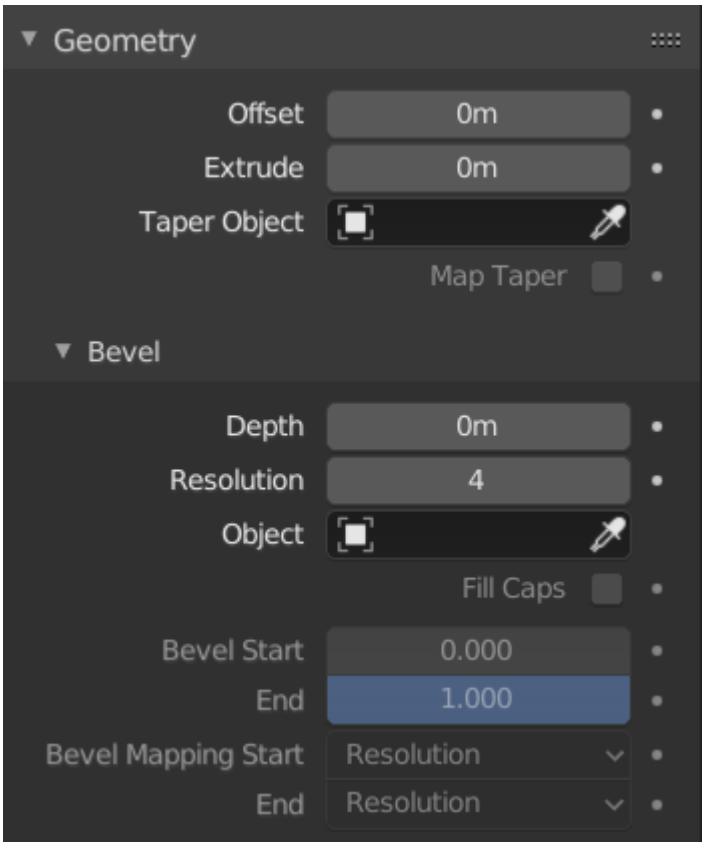
Stretch

The *Stretch* curve option allows you to let the mesh object stretch, or squeeze, over the entire curve. To get the expected result, use this together with the *Bounds Clamp* option. Utilized when using the Curve Modifier.

Bounds Clamp

When this option is enabled, the object and mesh offset along the deformation axis is ignored. This can be useful with the *Stretch* option or when using a negative axis. Utilized when using the Curve Modifier.

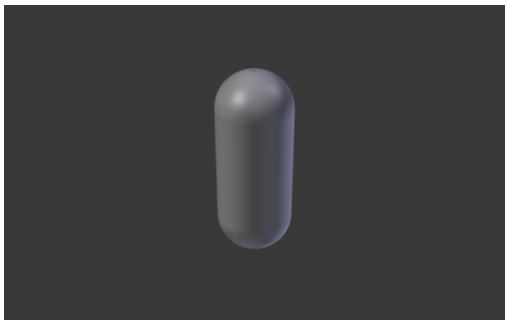
Geometry



Geometry panel.

Offset

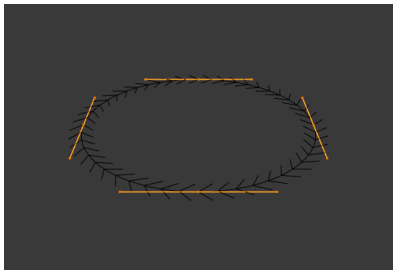
Moves the extrusion parallel to the curve normals.



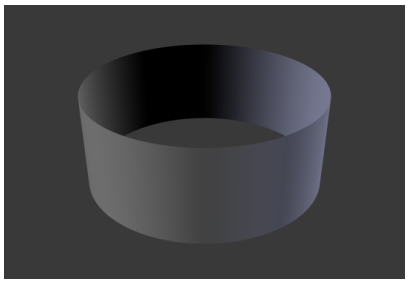
Bézier Circle -1 offset, 0.5 extrusion, 0.25 Bevel Depth, 10 Bevel resolution.

Extrude

Will extrude the curve along both the positive and negative local Z axes. Turns a one-dimensional curve into a two-dimensional curve by giving it height. With a scale is the sum of both directions, perpendicular to the curve's normals.



Bézier Circle 0.0 extrude (Edit Mode).



Extruded by 0.5 (Object Mode).

Taper Object

Tapering a curve causes it to get thinner towards one end. You can also alter the proportions of the Taper throughout the tapered object by moving/scaling/rotating the control points of the Taper Object. The taper curve is evaluated along the local X axis, using the local Y axis for width control. In order for this to work the Taper Object can only be another *open curve*.

The details are:

- The taper is applied independently to all curves of the extruded object.
- Only the first curve in a *Taper Object* is evaluated, even if you have several separated segments.
- The scaling starts at the first control point on the left and moves along the curve to the last control point on the right.

- Negative scaling, (e.g. negative local Y on the taper curve) is possible as well. However, rendering artifacts may appear.
- You may need to increase the curve resolution to see more detail of the taper.
- The Taper Object is distributed by control points. Therefor unevenly spaced control points may likelier to stretch the shape of the taper. Subdividing segments causes those points to use a larger fraction of the overall taper shape.
- With closed curves, the taper curve in *Taper Object* acts along the whole curve (perimeter of the object), not just the length of the object, and varies the extrusion depth. In these cases, you want the relative height of the *Taper Object* Taper curve at both ends to be the same, so that the cyclic point (the place where the endpoint of the curve connects to the beginning) is a smooth transition.

Hint

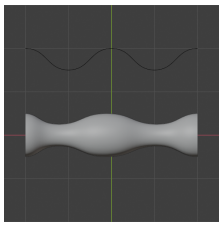
Editing the handles and control points of the Taper Object will instantly change the shape of the original object.

Taper Mode

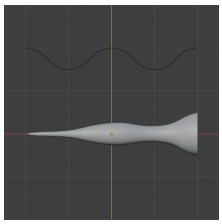
For curves using a *Taper Object*, this option defines how the effective curve radius is computed from the Taper Object.

Override::	The curve radius is ignored and the effective radius is equal to the taper radius.
Multiply::	The effective radius is computed by multiplying the taper radius with the curve radius.
Add::	The effective radius is computed by adding the taper radius to the curve radius.

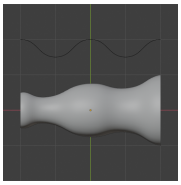
Examples of a curve with a radius of zero on one end and a radius of one on the other end.



Override mode.



Multiply mode.



Add mode.

Map Taper

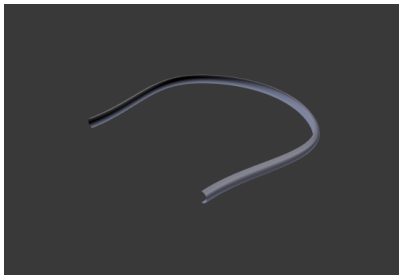
For curves using a Taper Object and with modifications to the *Start*/*End Bevel Factor* the *Map Taper* option will apply the taper to the beveled part of the curve (not the whole curve).

Bevel

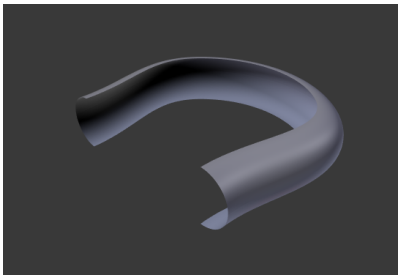
Round

Depth

Changes the size of the bevel.



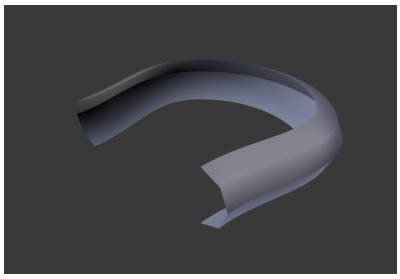
A curve with different Bevel depths applied (Depth of 0.05).



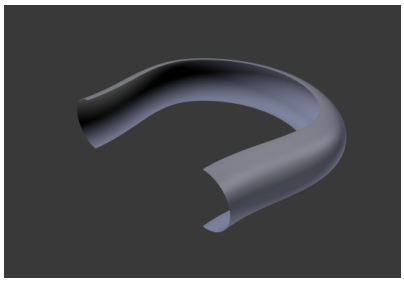
A curve with different Bevel depths applied (Depth of 0.25).

Resolution

Alters the smoothness of the bevel.



A curve with different resolutions applied (Resolution of 1).



A curve with different resolutions applied (Resolution of 12).

Fill Caps
: Seals the ends of a beveled curve.

Object

Object
: Here you can specify a profile curve object (opened or closed) which will be extruded along the path curve. If the profile curve's shape is 3D, it will be projected to its local XY plane before the extrusion. You can check what the projected object looks like by switching its shape to 2D.

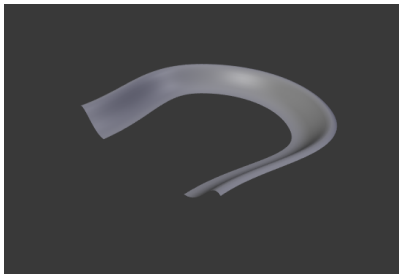
Important

Make sure the shape you want to extrude is in the Object's local XY plane. If it is in the local XZ or YZ plane, it will be reduced to a line when it is projected to the local XY plane. Because of this, the extruded shape will be a flat plane.

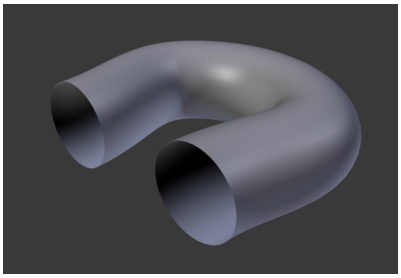
Note

If the selected curve has modifiers, these will not be taken into account. The extrusion will use the original curve shape. The reason for this behavior is that curves turn into meshes internally when they have modifiers on them, at which point they can't be used as bevel objects anymore.

To work around this, you can disable beveling on the path curve, and instead add geometry nodes to it: retrieve the profile curve using an Object Info Node, convert it from a mesh back into a curve using a Mesh to Curve Node, and finally pass both the path curve and the profile curve to a Curve to Mesh Node.



A curve with a Bézier curve as the Bevel Object.



A curve with a Bézier circle as the Bevel Object.

Profile

This Curve Widget allows the creation of a user-defined profile with more complexity than with the single profile parameter. The modal tool allows toggling the custom profile, but the shape of the profile is only editable in the options panel after the operation is confirmed.

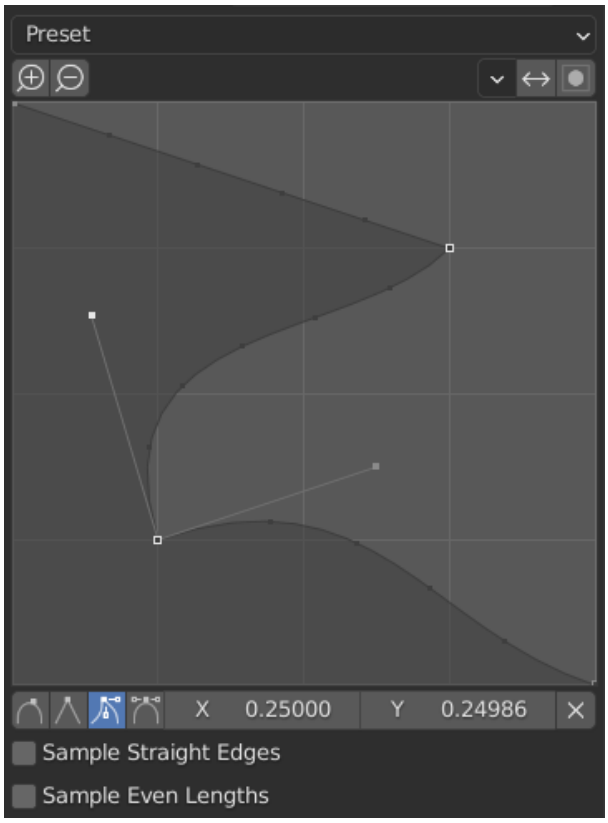
The profile starts at the bottom right of the widget and ends at the top left, as if it were between two edges intersecting at a right angle. Control points are created in the widget and then the path is sampled with the number of segments from the Bevel modifier.

Note

The *Profile* curve widget stays active when miters are enabled because it still controls the shape of the miter profiles.

Presets

The *Support Loops* and *Steps* presets are built dynamically depending on the number of segments in the bevel. If the number of segments is changed, the preset will have to be re-applied.



The custom profile widget.

Sampling

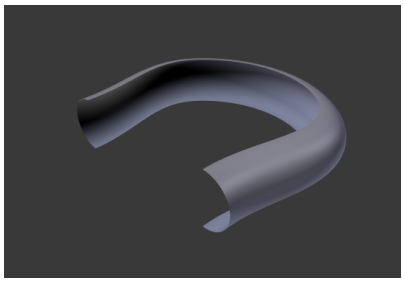
Samples will first be added to each control point, then if there are enough samples, they will be divided evenly between the edges. The *Sample Straight Edges* option toggles whether the samples are added to edges with sharp control points on either side. If there aren't enough samples to give each edge the same number of samples, they will just be added to the most curved edges. So it is recommended to use at least as many segments as there are control points.

Start & End Mapping

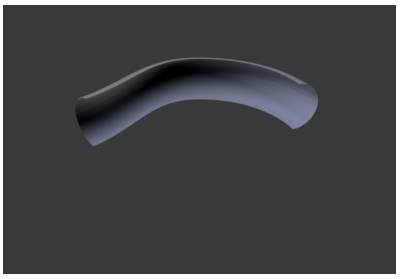
Factor Start, End

These options determine where to start/end the geometry of the curve. This allows to make a curve which is not fully covered with geometry.

A Start value to 0.5 will start the geometry at 50% of the distance from the start of the curve (in effect shortening the curve). An End value of 0.75 will start the geometry at 25% of the distance from the end of the curve (again, shortening the curve).



A curve with no *Factor Start, End*.



A curve with a 0.6 End factor.

Mapping Start, End

Allows to control the relation between the *Factor Start, End* (number between 0 and 1) and the rendered start and end point of the spline's geometry.

Resolution:: Maps the start and end factor to the number of subdivisions of a spline (U resolution).

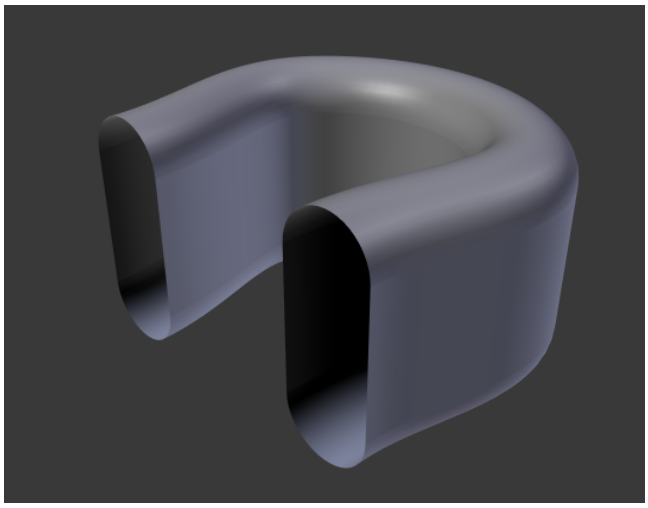
Segments:: Maps the start and end factor to the length of its segments. Mapping to segments treats the subdivisions in each segment of a curve as if they would have all the same length.

Spline:: Maps the start and end factor to the length of a spline.

Examples

Open 2D Curve

The extrusion will create a "wall" or "ribbon" following the curve shape. If using a *Bevel Depth*, the wall becomes a sort of slide or gutter. If your normals are facing the wrong way you can switch their direction as shown here.

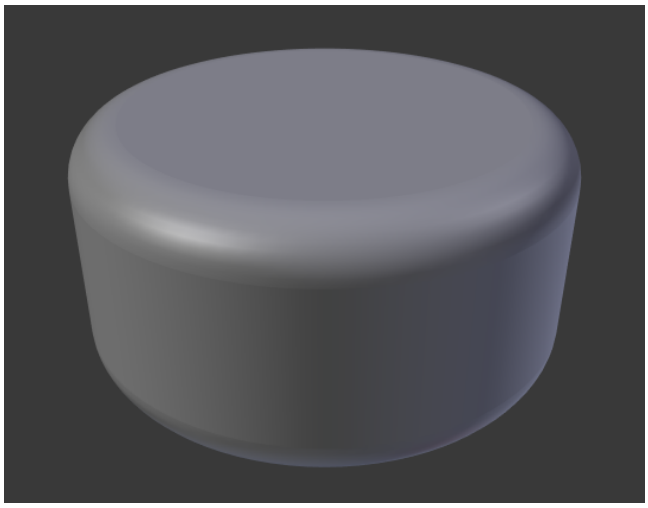


Open 2D Curve with `Alt-C`, fill set to none, zero offset, 0.5 extrusion, 0.25 Bevel Depth, 10 Bevel resolution.

Closed 2D Curve

This is probably the most useful situation, as it will quickly create a volume, with (by default) two flat and parallel surfaces filling the two sides of the extruded “wall”. You can remove one or both of these faces by choosing the fill mode: both, front, back, or none.

The optional bevel depth will always create a 90 degree bevels here.



Closed 2D Curve, 0.5 extrude, 0.25 Bevel Depth, 10 Bevel resolution, Fill: Both.

3D Curve

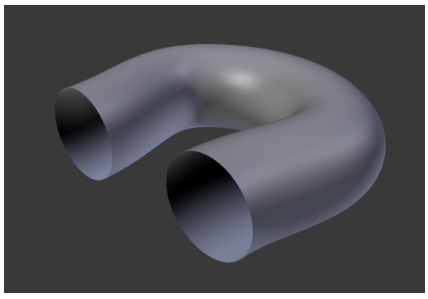
Here the fact that the curve is closed or not has no importance – you will never get a volume with an extruded 3D curve, only a wall or ribbon, like with open 2D curves.

However, there is one more feature with 3D curves: the *Tilt* of the control points (see above). It will make the ribbon twist around the curve to create a Möbius strip, for example.

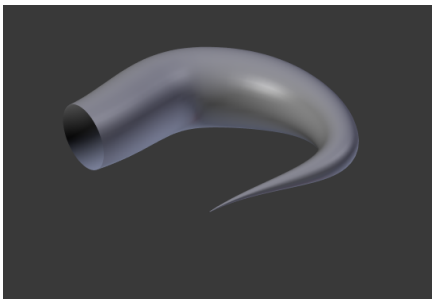
Taper

Let us taper a simple curve circle extruded object using a taper curve. Add a curve, then exit *Edit Mode*. Add another one (a closed one, like a circle); call it "BevelCurve", and enter its name in the *Bevel Object* field of the first curve (*Curve* tab). We now have a pipe. Add a third curve while in *Object Mode* and call it "TaperCurve". Adjust the left control point by raising it up about 5 units.

Now return to the Object tab, and edit the first curve's *Taper Object* field in the Geometry panel to reference the new taper curve which we called "TaperCurve". When you hit enter the taper curve is applied immediately, with the results shown in Fig. Circle curve set as Bevel Object..

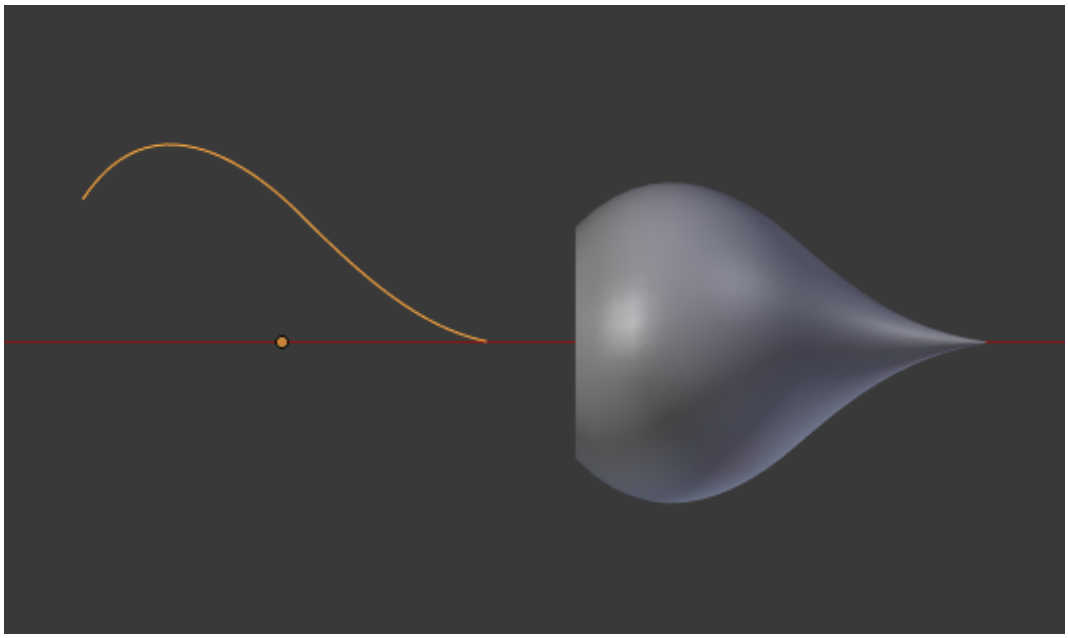


Circle curve set as Bevel Object.



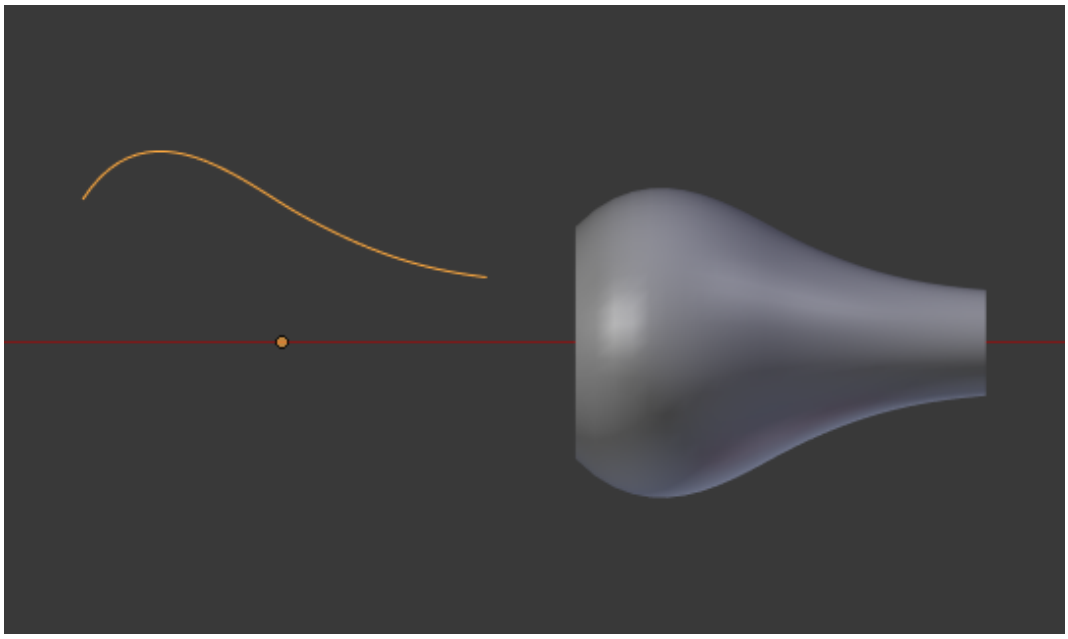
Taper extruded curve.

You can see the *taper curve* being applied to the *extruded object*. Notice how the pipe's volume shrinks to nothing as the taper curve goes from left to right. If the taper curve went below the local Y axis the pipe's inside would become the outside, which would lead to rendering artifacts. Of course as an artist that may be what you are looking for!



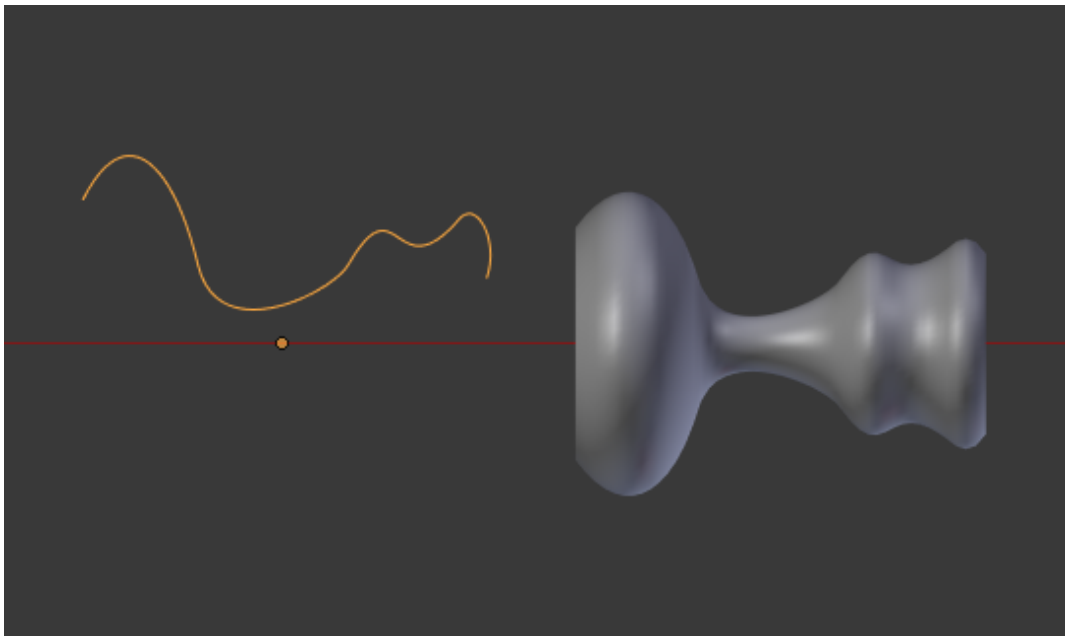
Taper example 1.

In Fig. Taper example 1. you can clearly see the effect the left taper curve has on the right curve object. Here the left taper curve is closer to the object origin and that results in a smaller curve object to the right.



Taper example 2.

In Fig. Taper example 2. a control point in the taper curve to the left is moved away from the origin and that gives a wider result to the curve object on the right.



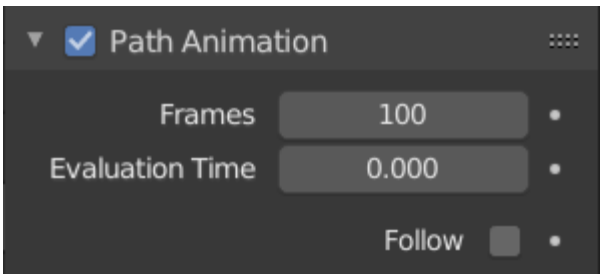
Taper example 3.

Path Animation

The *Path Animation* settings can be used to determine how child objects move along a certain path.

Note

This feature is deprecated, but still available. A more future-proof method is the Follow Path Constraint.



Path Animation panel.

Frames
: The number of frames that are needed to traverse the path, defining the maximum value for the *Evaluation Time* setting.

Evaluation Time
: Parametric position along the length of the curve that object following it should be at (the position is evaluated by dividing by the *Frames* value). By default, it is linked to the global frame number, but could be keyframed to give more control over the path animation.

Clamp
: Clamp the curve path children so they can't travel past the start/end point of the curve.

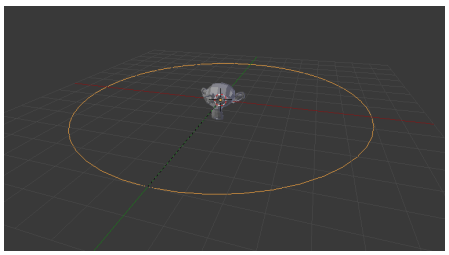
Follow

Make the curve path children rotate along the curvature of the path.

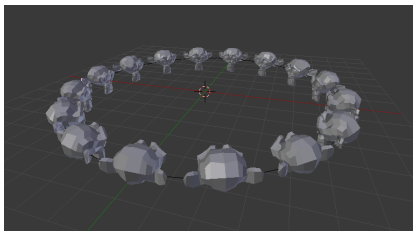
Example

This example shows you how setup a *Path Animation*.

1. Add an object you want to animate and a path along which this object will move. In this example it's the *Monkey* and the *Bézier Circle*.
2. To parent the monkey to the Bézier circle, first select the monkey then the curve (so that the curve is the active object), press `Ctrl-P` and select *Follow Path*. It will automatically animate *Evaluation Time* and activate *Follow* option in the *Path Animation* panel.
3. Select the monkey and Clear Origin to reset its offset.
4. You can change the orientation of the monkey by changing the Tracking Axis.



Monkey parented to the Bézier Circle.



The final result.

Active Spline

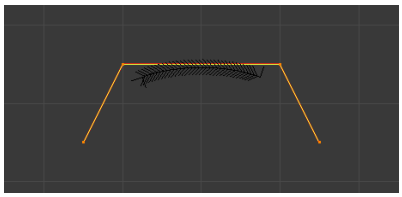
Reference

Mode::	Edit Mode
Menu::	*Properties* ▸ *Curve* ▸ *Active Spline*

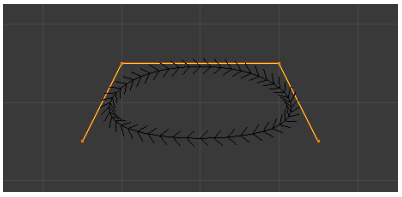
The *Active Spline* panel is used in Edit Mode to control properties of the currently selected spline.

Common Options

Cyclic U
: Closes the active spline.



Default NURBS curve.



A NURBS curve with Cyclic applied.

Resolution U
: Alters the resolution of each segment by changing the number of subdivisions.

Smooth
: Use Smooth Shading for any 3D geometry.

Poly

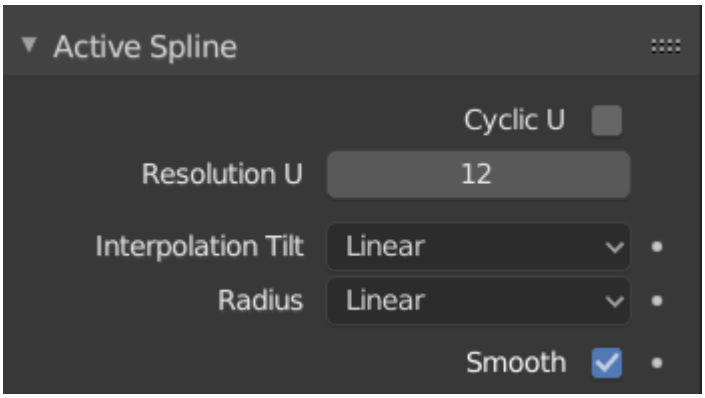


Active Spline panel: Poly Spline.

Cyclic U
See Common Options.

Smooth
See Common Options.

Bézier



Active Spline panel: Bézier Spline.

Cyclic U
See Common Options.

Resolution U
See Common Options.

Interpolation Tilt
Alters how the tilt of a segment is calculated.

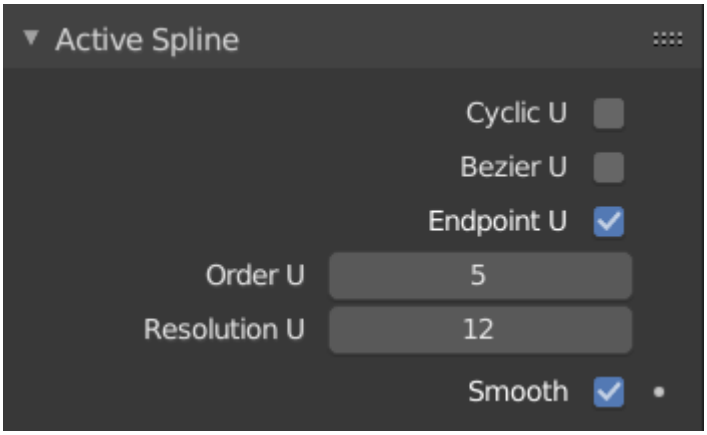
Radius

Alters how the radius of a beveled curve is calculated. The effects are easier to see after increasing the radius.

Smooth
See Common Options.

NURBS

One of the characteristics of a NURBS object is the *knot vector*. This is a sequence of numbers used to determine the influence of the control points on the curve. While you cannot edit the knot vectors directly, you can influence them through the *Endpoint* and *Bézier* options in the Active Spline panel. Note that, the *Endpoint* and *Bézier* settings only apply to open NURBS curves.

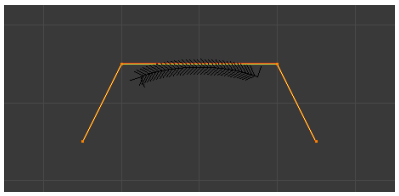


Active Spline: NURBS Spline.

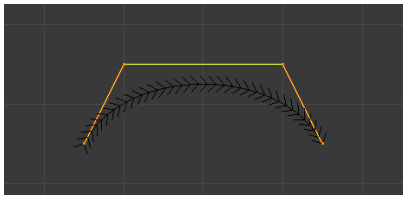
Cyclic U
See Common Options.

Bézier U
Makes the NURBS curve act like a Bézier curve. The NURBS control points act like *Free* handles of Bézier curve.

Endpoint U
Makes the curve contact the end control points.



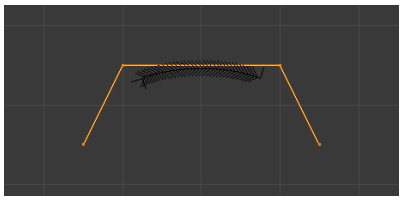
Default NURBS curve.



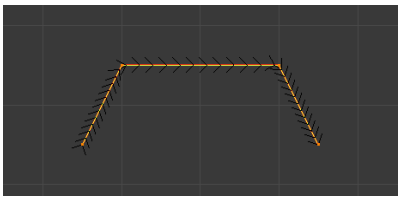
A NURBS curve with Endpoint enabled.

Order U

The order of the NURBS curve determines the area of influence of the control points over the curve. Higher order values means that a single control point has a greater influence over a greater relative proportion of the curve. The valid range of *Order* values is 2-6 depending on the number of control points present in the curve.



NURBS curves with orders of 4.



NURBS curves with orders of 2.

Resolution U

See Common Options.

Smooth

See Common Options.

Curve Display

Reference

Mode::	Edit Mode
Panel::	*3D Viewport* ▸ *Viewport Overlays* ▸ *Curve Edit Mode*

When in Edit Mode, curves have special overlays to control how curves are displayed in the 3D Viewport.

Handles
: Toggles the option to display the Bézier handles.

Normals
: Toggles the display of the curve normals.

  Normal Size
  : Length of the axis that points the direction of the normal.

Curves (New)

- Tools
  - Toolbar
  - Types
- Primitives
  - Empty Hair
  - Fur
- Selecting
  - Selection Modes
  - All
  - None
  - Invert
  - Select Random
  - Select More/Less
  - Select Linked
  - Select Endpoints
- Editing
  - Curves
  - Control Points
  - Segments
- Properties
  - Attributes
  - Surface

Tools

- Toolbar

Types

- Draw

Toolbar

Curves Edit Mode tools:

Select
: Select or move.

  Select Box
  : Select objects by dragging a box.

    All objects that intersect the box will be selected.

  Select Circle
  : Select objects by dragging a circle. All objects that intersect the path of the circle will be selected.

  Select Lasso
  : Select objects by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform

Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line

Draw straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previous drawn annotations.

Measure

Measure distances in the scene.

Draw

Free-hand drawing of new curves.

Radius

Control the radius value of the control points.

Tilt

Control the rotation value of the control points around the curve's axis.

Draw

Reference

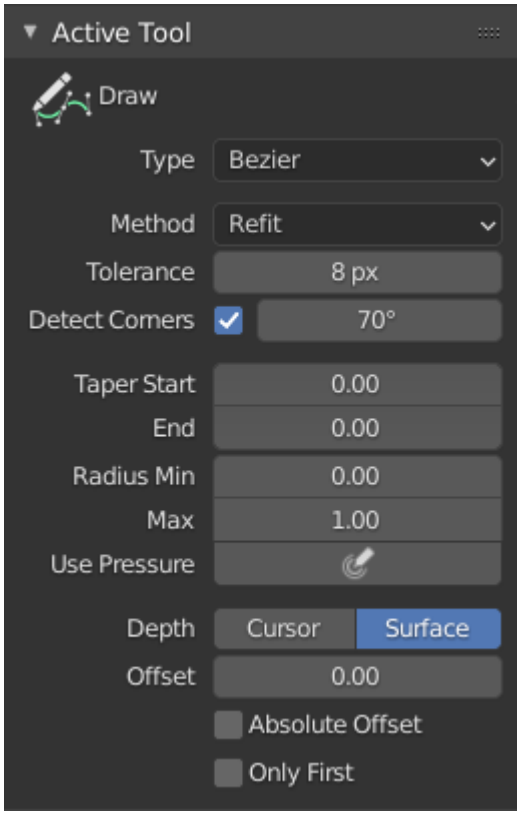
Mode:: Edit Mode
Tool:: *Toolbar ▸ Draw*

The Curve draw tool allows you to free-hand draw curves.

Tool Settings

Type
: Type of curve to use for drawing.

Poly::



Curve Stroke panel.

Bézier Curve with straight line segments (auto handles).

Bézier:: Tolerance

Lower values give a result that is closer to the drawing stroke, while higher values give more smoothed results.

Method

Refit:: Incrementally refits the curve (gives best results).

Split:: Splits the curve until the tolerance is met (gives a better drawing performance).

Detect Corners

Detects corners while drawing based on a specified angle; Any angles above the specified value are considered corners. If a corner is detected, the curve uses non-aligned handles for the corner resulting in a more crisp corner.

Taper Start, End

Taper factor for the radius of the start and end points along the curve.

Radius Min

Minimum radius when the minimum pressure is applied (also the minimum when tapering).

Max

Radius to use when the maximum pressure is applied (or when a tablet is not used).

Use Pressure

Uses stylus pressure to control the radius of the curve.

Depth

Controls where and how the curves are drawn.

Cursor:: Uses the depth under the cursor to draw curves.

Surface:: Used to draw on top of other objects.

Project Onto Selected
: Only project the strokes onto selected objects.

Offset
: Distance to offset the curve from the surface.

Absolute Offset
: Applies a fixed offset (does not scale by the curve radius).

Only First
: Only uses the start of the stroke for the depth.

Plane
: The orientation plane to draw on, available when *Only First* is enabled.

  Normal to Surface:: Draws aligned to the surface.

  Tangent to Surface:: Draws perpendicular to the surface.

  View:: Draws aligned to the viewport.

Curve 2D
: Project the curve on the Z axis.

As NURBS
: Draw curves as a NURBS curve with Bézier knot mode, instead of a Bézier curve.

Options

After the tool is run, these options are available in the Adjust Last Operation panel.

Error

Error distance in object units. This can be seen similar to a subdivision rate for the curve. Lower values give a result that is closer to the drawing stroke while higher values give more smoothed results.

Fit Method

Refit:: Incrementally refits the curve (gives best results).

Split:: Splits the curve until the tolerance is met (gives a better drawing performance).

Corner Angle

Any angles above this are considered corners.

Cyclic

Toggles whether or not the curve is Cyclic.

Primitives

Empty Hair

Adds an empty high-performance curves object and automatically:

- Assigns the active object as the Surface.
- Set the surface object as the parent of the new object.
- Adds a Geometry Nodes modifier to deform the curves on the surface.

Fur

Adds a fur setup to the selected objects. The fur setup is based on Geometry Nodes and built with Hair Node Groups that come with Blender as bundled assets.

See Quick Fur for more information.

Selecting

Hair curves, while similar to regular curves are a bit different and have their own selection tools. Many of these match their regular curve tools but are implemented differently All hair curve selection operators are documented below for completeness.

These selection operators work in both Sculpt and Edit modes.

Selection Modes

Reference

Mode:: Edit Mode
Menu:: *3D Viewport Header ▸ Select Mode*
Shortcut:: 1, 2

Note, this is only supported for “Hair Curves”.

Selection modes limits selection operators to certain curve domains. This feature is makes it easy to select whole segments at once, or to give more granular control over editing.

Control Points:: Allows selection of individual control points.
Curve:: Limits selection to whole curve segments.

All

Reference

Mode:: Edit Mode
Menu:: *Select ▸ All*
Shortcut::

`A`

Select all selectable elements.

None

Reference

Mode::	Edit Mode
Menu::	*Select ▸ None*
Shortcut::	`Alt-A`

Deselect all elements, but the active element stays the same.

Invert

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Invert*
Shortcut::	`Ctrl-I`

Selects all the geometry that are not selected, and deselect currently selected components.

Select Random

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Random*

Select Random control points.

Seed
: Seed used by the pseudo-random number generator.

Probability
: Selects the defined percentage of control points.

Select More/Less

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *More/Less*
Shortcut::	`Ctrl-NumpadPlus`, `Ctrl-NumpadMinus`

Their purpose, based on the currently selected control points, is to reduce or enlarge this selection.

More
: For each selected control point, select *all* its linked points (i.e. one or two…).

Less
: For each selected control point, if *all* points linked to this point are selected, keep this one selected. Otherwise, deselect it.

This implies two points:

1. When *all* control points of a curve are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no control points are selected.
2. Second, these tools will never "go outside" of a curve (they will never "jump" to another curve in the same object).

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Linked*
Shortcut::	`L`, `Ctrl-L`, `Shift-L`

`L` (or `Ctrl-L` for all) will add to the selection the cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same curve. Note that for Bézier, using `L` with a handle selected will select the whole control point and all the linked ones.

Select Endpoints

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Endpoints*

Select endpoints of curves. Only supported in the Control Point selection mode.

Editing

The curves can be edited via sculpting.

Curves objects also have basic editing support in “Edit Mode”.

- Curves
  - Transform
  - Duplicate
  - Delete
  - Toggle Cyclic
  - Set Curve Type
- Control Points
  - Extrude Curve and Move
  - Tilt
  - Clear Tilt
  - Set Handle Type
- Segments
  - Subdivide
  - Switch Direction

Curves

Transform

Reference

Mode::	Edit Mode
Menu::	*Curves* ▸ *Transform*

A curves objects can be edited by transforming the locations of control points.

Move, Rotate, Scale
: Like other elements in Blender, control points can be moved, rotated, or scaled as described in Basic Transformations.

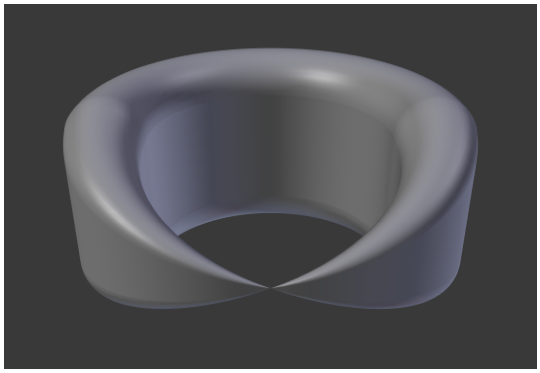
To Sphere, Shear, Bend, Push/Pull
: The transform tools are described in the Transformations sections.

Radius

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Radius*
Menu::	*Curves* ▸ *Transform* ▸ *Radius*
Shortcut::	`Alt-S`

The Radius allows you to directly control the width of the extrusion along the “spinal” curve. The radius will be interpolated from point to point (you can check it with the normals). The *Radius* of the points is set using the *Radius* transform tool. Or in the Sidebar *Transform* panel.



One control point radius set to zero.

Duplicate

Reference

Mode::	Edit Mode
Menu::	*Curves ▸ Duplicate*
Shortcut::	`Shift-D`

This operator duplicates the selected control points, along with the curve segments implicitly selected (if any). .. If only a handle is selected, the full point will be duplicated too. The copy is selected so you can move it to another place.

Delete

Reference

Mode::	Edit Mode
Menu::	*Curves ▸ Delete*
Shortcut::	`X`

The Delete operator can remove Control Points or Segments. Deleting can be used to make curves shorter or simplify segments by deleting control points in the mid section of a segment.

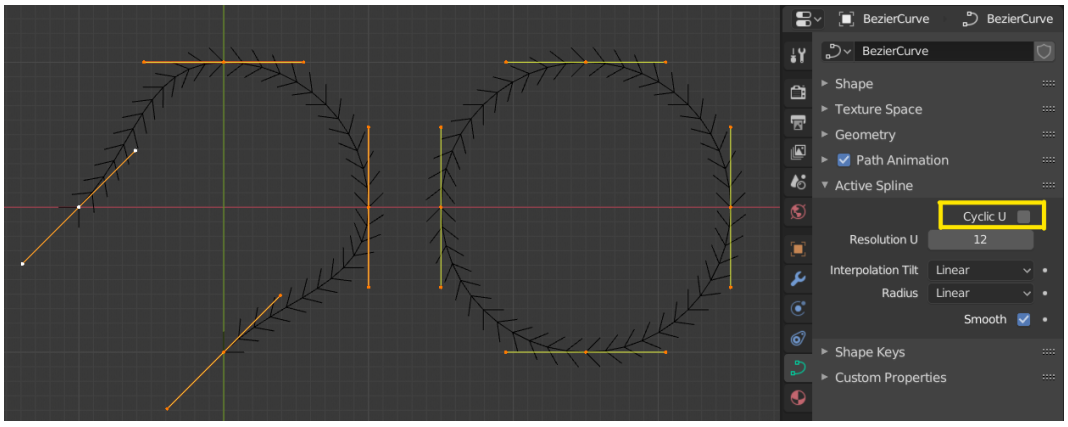
Toggle Cyclic

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Toggle Cyclic*
Shortcut::	`Alt-C`

Toggles between an open curve and closed curve (Cyclic). Only curves with at least one selected control point will be closed/open. The shape of the closing segment is based on the start and end handles for Bézier curves, and as usual on adjacent control points for NURBS. The only time a handle is adjusted after closing is if the handle is an *Auto* one. Fig. Open and Closed curves. is the same Bézier curve open and closed.

This action only works on the original starting control point or the last control point added. Deleting a segment(s) does not change how the action applies; it still operates only on the starting and last control points. This means that `Alt-C` may actually join two curves instead of closing a single curve! Remember that when a 2D curve is closed, it creates a renderable flat face.



Open and Closed curves.

Set Curve Type

Reference

Mode:: Edit Mode
Menu:: *Curves ▸ Set Curve Type*

Converts splines in a curve object between Bézier, NURBS, and Poly curves. Note, this is not a "smart" conversion, i.e. Blender does not try to keep the same shape, nor the same number of control points. For example, when converting a NURBS to a Bézier, each group of three NURBS control points become a unique Bézier one (center point and two handles).

Handles
    Take handle information into account in the conversion

Control Points

Extrude Curve and Move

Reference

Mode::	Edit Mode
Menu::	*Control Points ▸ Extrude Curve and Move*
Shortcut::	`E`

Extrudes points by duplicating the selected points, which then can be moved, and connecting those points back to the original curve creating a continuous curve.

Tilt

Reference

Mode::	Edit Mode
Tool::	*Toolbar ▸ Tilt*
Shortcut::	`Ctrl-T`

This setting controls how the normals twist around each control point. The tilt will be interpolated from point to point (you can check it with the normals).

Clear Tilt

Reference

Mode::	Edit Mode
Shortcut::	`Alt-T`

You can also reset the tilt to its default value (i.e. perpendicular to the original curve plane).

Set Handle Type

Reference

Mode::	Edit Mode
Menu::	*Control Points* ▸ *Set Handle Type*
Shortcut::	V

Handle types are a property of Bézier curves and can be used to alter features of the curve. For example, switching to *Vector handles* can be used to create curves with sharp corners. Read the Bézier curves page for more details.

Segments

Subdivide

Reference

Mode:: Edit Mode
Menu:: *Curves ▸ Segments ▸ Subdivide*

Curve subdivision simply subdivides all selected segments by adding one or more control points between the selected segments.

Number of Cuts
The number of subdivisions to perform.

Switch Direction

Reference

Mode:: Edit Mode
Menu:: *Curves ▸ Segments ▸ Switch Direction*

This tool will "reverse" the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). This is mainly useful when using a curve as path, or using the bevel and taper options.

Properties

Hair Curves have different properties than regular Curve objects; these properties are documented below.

Attributes

The *Attributes* panel contains different hair characteristics such as the position and color of hair strands.

Use the List View to manage attributes.

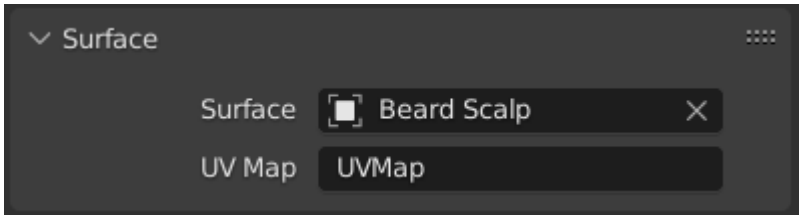
See also

See the Attribute Reference for details on attributes.

Surface

Surface
: The curve surface is an optional mesh that is used to anchor the curves, and behave as a scalp for hair grooming. When adding a new Curves object via the **Add Menu** the active object is automatically set as the surface.

  To set a new surface press `Ctrl-P` and select *Object (Attach Curves to Surface)* in the *Set Parent To* pop-up menu. This option can be seen as part of the Curves settings in the Properties Editor.



Surface UV Map

The name of the attribute on the surface mesh used to define the attachment of each curve.

Note

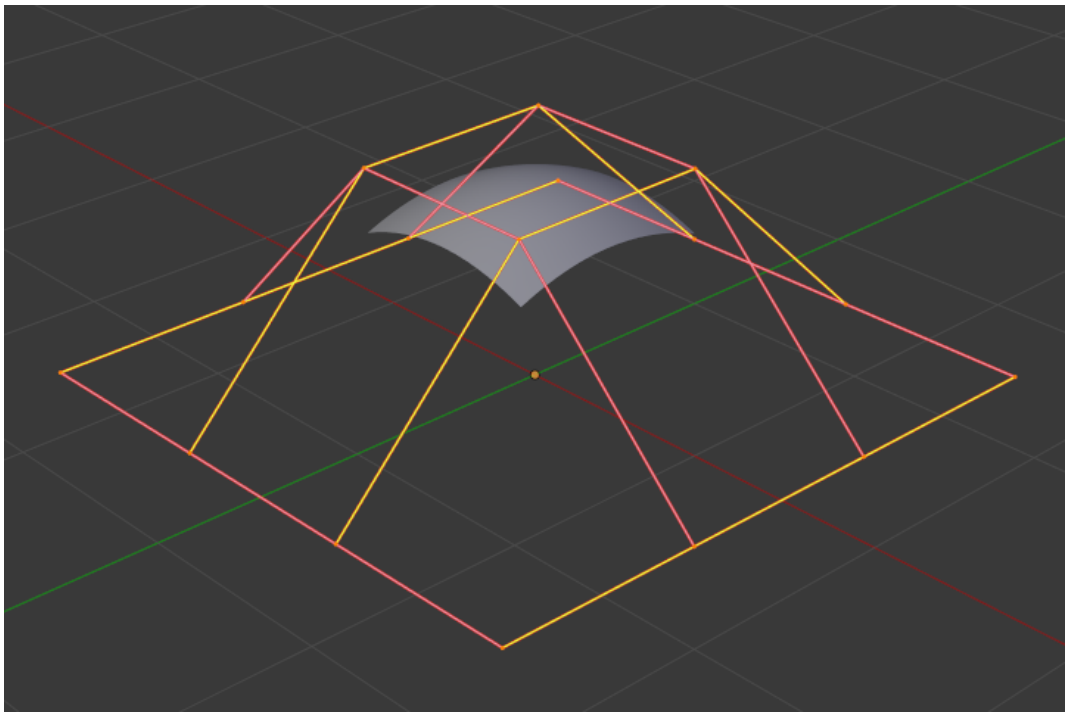
If the UV from the surface changed, run Snap to Nearest Surfaces to re-attach the curves.

Surfaces

- Introduction
  - Visualization
  - Conversion
- Toolbar
- Structure
  - Control Points, Rows and Grid
  - Weight
- Primitives
  - NURBS Curve
  - NURBS Circle
  - NURBS Surface
  - NURBS Cylinder
  - NURBS Sphere
  - NURBS Torus
- Selecting
  - Select Menu
  - Select Random
  - Checker Deselect
  - Select Linked
  - Select Similar
  - Select Control Point Row
  - Select More/Less
- Editing
  - Transform Panel
  - Surface
  - Control Points
  - Segments
- Properties
  - Shape
  - Active Spline

Introduction

Curves are 2D objects, and surfaces are their 3D extension. Note however, that in Blender, you only have NURBS surfaces, no Bézier (you have the *Bézier* knot type, though; see below), nor polygonal (but for these, you have meshes!). Even though curves and surfaces share the same object type (with texts also…), they are not the same thing; for example, you cannot have in the same object both curves and surfaces.



NURBS surface in Edit Mode.

As surfaces are 2D, they have two interpolation axes, U (as for curves) and V. It is important to understand that you can control the interpolation rules (knot, order, resolution) *independently* for each of these two dimensions (the U and V fields for all these settings, of course).

You may ask yourself “but the surface appears to be 3D, why is it only 2D?”. In order to be 3D, the object needs to have “Volume”, and a surface, even when it is closed, does not have volume; it is infinitely thin. If it had a

volume the surface would have a thickness (its third dimension). Hence, it is only a 2D object, and has only two interpolation dimensions or axes or coordinates (if you know a bit of math, think of non-Euclidean geometry – well, surfaces are just non-Euclidean 2D planes…). To take a more "real-world" example, you can roll a sheet of paper to create a cylinder; well, even if it becomes a "volume", the sheet itself will remain a (nearly…) 2D object!

In fact, surfaces are very similar to the results you get when extruding a curve.

Visualization

There is nearly no difference from NURBS curves, except that the U direction is indicated by yellow grid lines, and the V one is materialized by pink grid lines, as you can see in Fig. NURBS surface in Edit Mode..

You can hide and reveal control points just as with curves.

Conversion

As there are only NURBS surfaces, there is no "internal" conversion here.

However, there is an "external" conversion available, from surface to mesh, that only works in Object Mode. It transforms a surface object into a mesh one, using the surface resolutions in both directions to create faces, edges and vertices.

Toolbar

Surface Edit Mode tools:

Select
: Select or move.

  Select Box
  : Select objects by dragging a box.

    All objects that intersect the box will be selected.

  Select Circle
  : Select objects by dragging a circle. All objects that intersect the path of the circle will be selected.

  Select Lasso
  : Select objects by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform

Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line

Draw straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previous drawn annotations.

Measure

Measure distances in the scene.

Structure

Many of the concepts from curves, especially NURBS ones, carry directly over to NURBS surfaces, such as control points, *Order*, *Weight*, *Resolution*, etc. Here we will just talk about the differences.

It is very important to understand the difference between NURBS curves and NURBS surfaces: the first one has one dimension, the latter has two. Blender internally treats NURBS surfaces and NURBS curves completely differently. There are several attributes that separate them but the most important is that a NURBS curve has a single interpolation axis (U) and a NURBS surface has two interpolation axes (U and V).

However, you can have "2D" surfaces made of curves (using the extrusion tools, or, to a lesser extent, the filling of closed 2D curves). And you can have "1D" curves made of surfaces, like a NURBS surface with only one row (either in U or V direction) of control points produces only a curve…

Visually you can tell which is which by entering Edit Mode and looking at the 3D Viewport header: either the header shows *Surface* or *Curve* as one of the menu choices. Also, you can extrude a whole NURBS surface curve to create a surface, but you cannot with a simple NURBS curve.

Control Points, Rows and Grid

Control points for NURBS surfaces are the same as for NURBS curves. However, their layout is quite constraining. The concept of "segment" disappears, replaced by "rows" and the overall "grid".

A "row" is a set of control points forming one "line" in one interpolation direction (a bit similar to edge loops for meshes). So you have "U rows" and "V rows" in a NURBS surface. The key point is that *all* rows of a given type (U or V) have the *same* number of control points. Each control point belongs to exactly one U row and one V row.

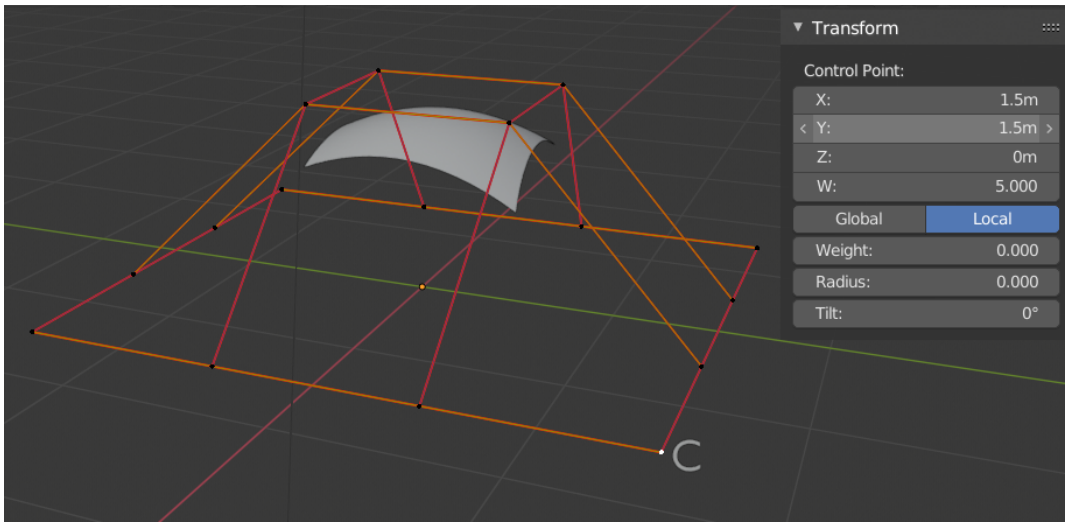
All this forms a “grid”, or “cage”, the shape of which controls the shape of the NURBS surface. A bit like a lattice…

This is very important to grasp: you cannot add a single control point to a NURBS surface; you have to add a whole U or V row at once (in practice, you will usually use the *Extrude* tool, or perhaps the *Duplicate* one, to add those…), containing exactly the same number of points as the others. This also means that you will only be able to “merge” different pieces of surfaces if at least one of their rows matches together.

Weight

Similar to NURBS Splines NURBS Surface control points have a weight property. This weight property controls how much influence the control point has on the surface. This weight should not be confused with the Goal Weight, which is used only for soft body simulations. The NURBS control point weight can be adjusted in the *W* number field of the Transform panel.

In Fig. One control point with a weight of 5. a single control point, labeled “C”, has had its *Weight* set to 5.0 while all others are at their default of 1.0. As you can see, that control point *pulls* the surface towards it.



One control point with a weight of 5.

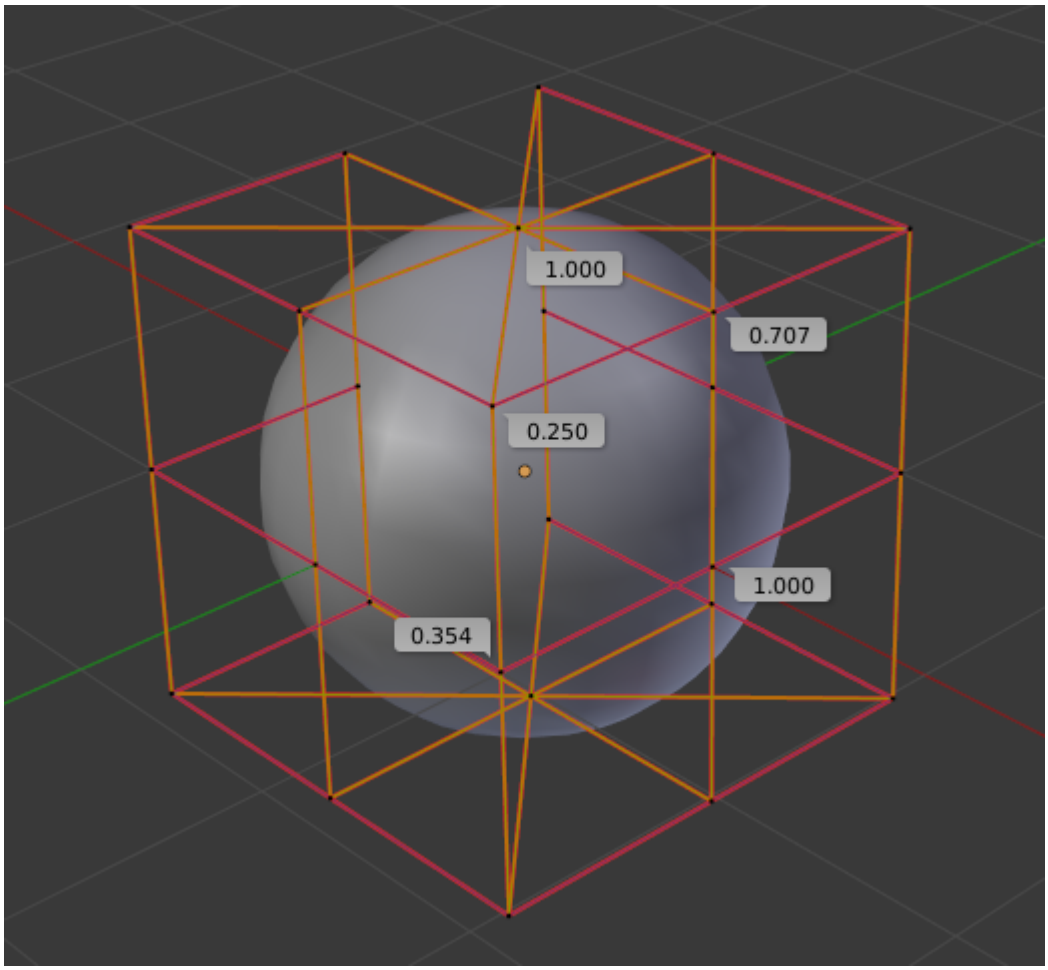
Note

If all the control points have the same *Weight* then each effectively cancels each other out. It is the difference in the weights that cause the surface to move towards or away from a control point.

Preset Weights

NURBS can create pure shapes such as circles, cylinders, and spheres (note that a Bézier circle is not a pure circle). To create pure circles, spheres, or cylinders, you must set to specific values the weights of the control points. This is not intuitive, and you should read more on NURBS before trying this.

To create a sphere with 2D surfaces, its the same principle as with a 2D circle. You will note that the four different weights needed for creating a sphere (1.0, 0.707 = sqrt(0.5), 0.354 = sqrt(2)/4, and 0.25).



A sphere surface.

Primitives

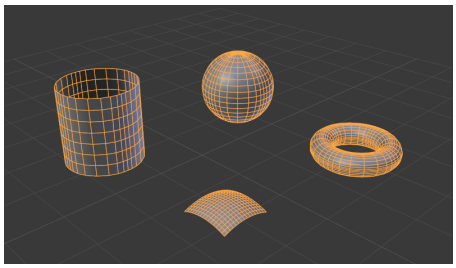
Reference

Mode::	Object Mode and Edit Mode
Menu::	*Add* ▸ *Curve*
Shortcut::	`Shift-A`

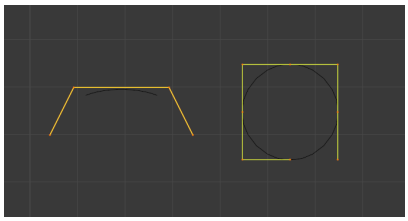
See also

When adding curves there are some common options like other Objects.

In Object/Edit Mode, the *Add Surface* menu, provides six different surface primitives:



NURBS surface primitives.



NURBS curve primitives.

NURBS Curve

Adds a generic curve of four control points forming an arc.

NURBS Circle

Adds an a closed loop of control point forming a circle. Note, a circle NURBS surface is never filled, unlike its “real” curve counterpart…

NURBS Surface

Adds a generic surface patch consisting of a 4×4 grid plane with the center grid slightly raised.

NURBS Cylinder

Adds an open end cylinder, consisting of an extruded *NURBS Circle*.

NURBS Sphere

Adds a generic sphere constructed by revolving a grid of control points about an axis.

NURBS Torus

Adds a doughnut-shaped primitive created by rotating a circle around an axis.

Selecting

This page discusses specific selecting tools for surface objects in Edit Mode. The Surface Edit also uses the general select tools used which are described in the interface section.

Surface selection in *Edit Mode* is very similar to NURBS curve selection. The basic tools are the same as with meshes, so you can select a simple control point with an `LMB`-click, add to current selection with `Shift–LMB` clicks, Border Select, and so on.

Select Menu

The *Select* menu (in the 3D Viewport header) is even simpler than for curves…

All these options have the same meaning and behavior as in Object Mode and mesh Edit Mode.

All `A`
: Select all.

None `Alt–A`
: Select none.

Inverse `Ctrl–I`
: Selects all the geometry that is not selected, and deselect currently selected components.

Box Select `B`
: Interactive box selection.

Circle Select `C`
: Interactive circle selection.

Lasso Select
: Interactive free-form selection.

Select Random
: Select random control points.

Checker Deselect
: Select every Nth control point.

Select Linked `Ctrl-L`
: Select control points that are connected to the current selection.

Select Similar `Shift-G`
: Select control points that have similar properties to the current selection.

Select Control Point Row
: Select a whole row of control points.

Select More/Less
: Select objects based on their parent child relationships.

Select Random

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Random*

Select random control points.

Percent
: Selects the defined percentage of control points.

Random Seed

Seed used by the pseudo-random number generator.

Action
: Controls whether the operator *Selects* or *Deselects* control points.

Checker Deselect

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Checker Deselect*

This tool applies an alternating selected/deselected checker pattern. This only works if you already have more than one control point selected.

It works by changing the current selection so that only every Nth control points will remain selected, starting from the active one.

Deselected
: The number of deselected elements in each pattern repetition.

Selected
: The number of selected elements in each pattern repetition.

Offset
: Offset from the starting point.

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Linked*
Shortcut::	`L`, `Ctrl-L`

Select Linked will add to the selection the mouse cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same surface.

Select Similar

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Similar*
Shortcut::	`Shift-G`

Selects control points that have certain similar properties to the active one. The Adjust Last Operation panel provides several selection options:

Type
: Type
  : Selects splines that have the same spline Type i.e. Bézier, NURBS or Poly.

: Radius
  : Selects control points that have a similar Radius value.

: Weight
  : Selects all points that have a similar Weight value.

: Direction
  : Selects control points that have a similar handles direction.

Compare
: For quantitative properties, this property selects the type of comparison to between the two numerical values.

Equal::	Select items with the same value as the active item's chosen property.
Greater::	Select items with a larger value as the active item's chosen property.
Less::	

Select items with a smaller value as the active item's chosen property.

Threshold
: For quantitative properties, this property controls how close the property's values have to be in the comparison.

Select Control Point Row

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Control Point Row*
Shortcut::	`Shift-R`

This option works a bit like edge loop selection for meshes, inasmuch it selects a whole row of control points, based on the active (the last selected) one. The first time you press `Shift-R`, the V row passing through (containing) the active point will be added to the *current* selection. If you use again this shortcut, you will toggle between the U and V row of this point, removing *everything else* from the selection.

Select More/Less

Reference

Mode::	Edit Mode
Menu::	*Select ▸ More/Less*
Shortcut::	`Ctrl-NumpadPlus` / `Ctrl-NumpadMinus`

Expand or contract the selection based on current selected control points.

More
: For each selected control point, select **all** its linked points (i.e. two, three or four).

Less

For each selected control point, if **all** points linked to this point are selected, keep it selected. For all other selected control points, deselect them.

This implies two points:

1. First, when **all** control points of a surface are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no control point is selected.
2. Second, these tools will never "go outside" of a surface (they will never "jump" to another surface in the same object).

Editing

- Transform Panel
- Surface
  - Transform
  - Mirror
  - Snap
  - Spin
  - Add Duplicate
  - Split
  - Separate
  - Toggle Cyclic
  - Set Spline Type
  - Show/Hide
  - Cleanup
  - Delete
- Control Points
  - Extrude Curve and Move
  - Make Segment
  - Smooth
  - Hooks
  - Make Vertex Parent
- Segments
  - Subdivide
  - Switch Direction

Transform Panel

Reference

Mode::	Edit Mode
Panel::	*Sidebar ▸ Transform*

When nothing is selected, the panel is empty. When more than one control point is selected, the median values are edited and “Median” is added in front of the labels.

Control Point, Median
: The first controls (X, Y, Z) show the coordinates of the selected point or handle (vertex). The last control (W), defines the weight of the selected control point or the median weight.

Space
: The Space radio buttons let you choose if those coordinates are relative to the object origin (local) or the global origin (global).

  Global, Local

Weight
: Controls the “goal weight” of selected control points, which is used when a surface has Soft Body physics, forcing the surface to “stick” to their original positions, based on the weight.

Radius
: Surface objects do not have a *Radius* property, this value has no effect.

Tilt
: Surface objects do not have a *Radius* property, this value has no effect.

Surface

Surface editing has even fewer tools and options than its curve counterpart, but has many common points with it… So this page covers (or tries to cover) all the subjects, from the basics of surface editing to more advanced topics, like retopology.

Transform

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Transform*

A surface can be edited by transforming the locations of the control points.

Move, Rotate, Scale
: Like other elements in Blender, control points can be moved, rotated, or scaled as described in Basic Transformations.

To Sphere, Shear, Warp, Bend, Push/Pull, Warp, Randomize
: These transform tools are described in the Transformations sections.

Move/Scale Texture Space
: Like other objects, surfaces have textures spaces which can be edited.

Mirror

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Mirror*
Shortcut::	`Ctrl-M`

The *Mirror* tool is also available, behaving exactly as with mesh vertices.

Snap

Reference

Mode::	Edit Mode
Menu::	*Curve* ▸ *Snap*
Shortcut::	`Shift-S`

Mesh snapping also works with control points, except for within itself (other components of the active spline). Snapping works with 2D surfaces but points will be constrained to the local XY axes.

Spin

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Spin*

This tool is a bit similar to its mesh counterpart but with less control and options (in fact, there is none!).

It only works on selected "surfaces" made of *one U row* (and not with one V row), so-called "surface curves", by "extruding" this "cross section" in a square pattern. While automatically adjusting the weights of control points to get a perfect circular extrusion (this also implies closing the surface along the V axis), following exactly the same principle as for the *NURBS Tube* or *NURBS Torus* primitives.

Add Duplicate

Reference

Mode::	Edit Mode
Menu::	*Surface ▸ Add Duplicate*
Shortcut::	`Shift-D`

Similar as with meshes and curves, this tool duplicates the selection. The copy is selected and placed in move mode, so you can move it to another place.

However, with surfaces there are some selections that cannot be duplicated, in which case they will just be placed in move mode… In fact, only selections forming a *single* valid sub-grid are copyable; let us see this in practice:

- You can copy a single control point. From it, you will be able to "extrude" a "surface curve" along the U axis, and then extrude this unique U row along the V axis to create a real new surface.
- You can copy a single continuous part of a row (or a whole row, of course). This will give you a new *U row*, even if you selected (part of) a V row!
- You can copy a single whole sub-grid.

Note

Trying to duplicate several valid "sub-grids" (even being single points) at once will not work; you will have to do it one after the other…

Split

Reference

Mode::	Edit Mode
Menu::	*Surface ▸ Split*
Shortcut::	`Y`

The *Split* operator splits a selected segment of a surface from the rest of the surface. This segment can then be moved or altered without affecting the other surface. If a single control point is selected the *Split* operator will create a new singular loose control point; leaving the previously selected control point attached to the rest of the surface.

Separate

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Separate*
Shortcut::	`P`

Surface objects that are made of multiple distinct parts can be separated into their own objects by selecting the desired segments and using *Separate*. Note, if there is only one surface in a surface object, *Separate* will create a new surface object with no control points.

Toggle Cyclic

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Toggle Cyclic*
Shortcut::	`Alt-C`

As in curves, surfaces can be closed (cyclic) or open. However, as surfaces are 2D, you can control this property independently along the U and V axes.

To toggle the cyclic property of a surface along one axis, use *Toggle Cyclic* and choose either *Cyclic U* or *Cyclic V* from the pop-up menu. The corresponding surface's outer edges will join together to form a "closed" surface.

Note

Inner and Outer

Surfaces have an “inner” and “outer” face, the first being black whereas the latter is correctly shaded. When you close a surface in one or two directions, you might get an entirely black object! In this case, just Switch Direction of the surface.

Set Spline Type

Reference

Mode::	Edit Mode
Menu::	*Surface ▸ Set Spline Type*

This feature only works for Curves.

Show/Hide

Reference

Mode::	Edit Mode
Menu::	*Curve ▸ Show/Hide*
Shortcut::	`Alt-H`, `H`, `Shift-H`

When in *Edit Mode*, you can hide and reveal elements from the display. You can only show or hide control points, as segments are always shown, unless all control points of the connected surface are hidden, in which case the surface is fully hidden.

See also

See Show/Hide in *Object Mode*.

Cleanup

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Cleanup*

This feature only works for Curves.

Delete

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Delete*
Shortcut::	`X`, `Delete`

The selection must abide by the following rules:

- Whole rows, and only whole rows must be selected.
- Only rows along the same axis must be selected (i.e. you cannot delete both U and V rows at the same time).

Also remember that NURBS order cannot be higher than its number of control points in a given axis, so it might decrease when you delete some control points… Of course, when only one row remains, the surface becomes a "surface curve"; when only one point remains, there is no more visible surface; and when all points are deleted, the surface itself is deleted.

Vertices
: This will delete the selected rows, *without* breaking the surface (i.e. the adjacent rows will be directly linked, joined, once the intermediary ones

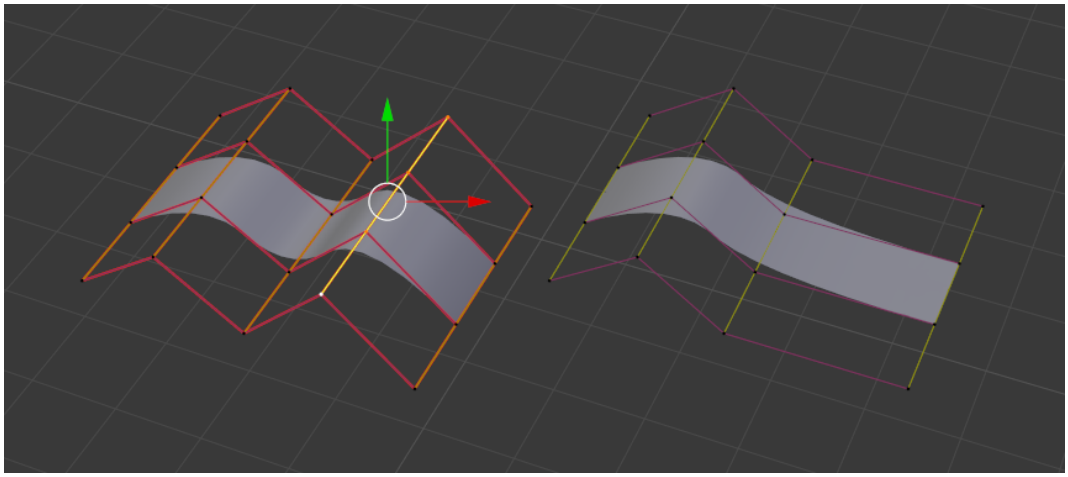
are deleted). Remember that NURBS order cannot be higher than its number of control points, so it might decrease when you delete some control point. Of course, when only one point remains, there is no more visible curve, and when all points are deleted, the curve itself is deleted.

Segment
: Deletes the segment that connects the selected control points and disconnects them.

Dissolve Vertices `Ctrl-X`
: This feature only works for Curves.

Example

In the image below (left), a row of control points has been selected by initially selecting the one control point and using Select Control Point Row to select the remaining control points. Then, using Delete *Vertices*, the selected row of control points is erased, resulting in the image below (right).



Before and after.

Control Points

Extrude Curve and Move

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Extrude Curve and Move*
Shortcut::	E

Unlike meshes or curves, you cannot generally directly add new control points to a surface, as you can only extend a surface by adding a whole U or V row at once. The only exception is when working on a NURBS surface curve, i.e. a surface with only one control point on each U or V row. In this special case, all works exactly as with curves.

Most of the time, only extrusion is available. As usual, once the tool is activated the extrusion happens immediately and you are placed into *select mode*, ready to drag the new extruded surface to its destination.

There are two things very important to understand:

1. Surfaces are *2D* objects. So you cannot extrude anything *inside* a surface (e.g. "inner" row); it would not make any sense!
2. The control "grid" *must* remain "squarish", which means that you can only extrude a whole row, not parts of rows here and there…

To summarize, the *Extrude* tool will only work, when one and only one whole border row is selected, otherwise nothing happens.

Note

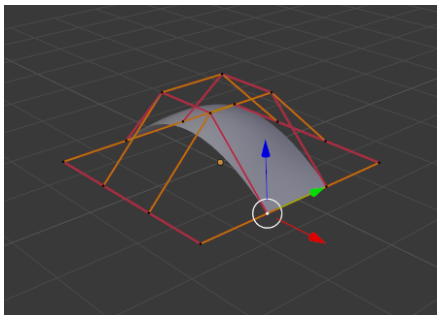
As for curves, you cannot create a new surface in your object out of nowhere. However, unlike for curves, there is no "cut" option allowing

you to separate a surface into several parts, so you only can create a new surface by Duplicating an existing one, or adding a new one with the *Add* menu.

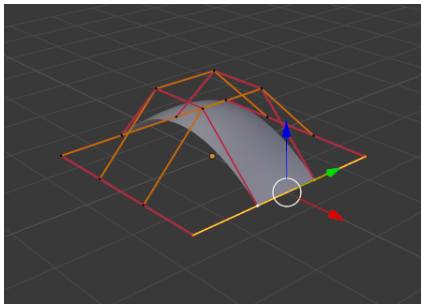
Examples

Images Fig. Selecting control point. to Fig. Extruding. show a typical extrusion along the side of a surface.

In Fig. Selecting control point. and Select Control Point Row., a border row of control points were highlighted by selecting a single control point, and then using Select Control Point Row to select the rest of the control points.

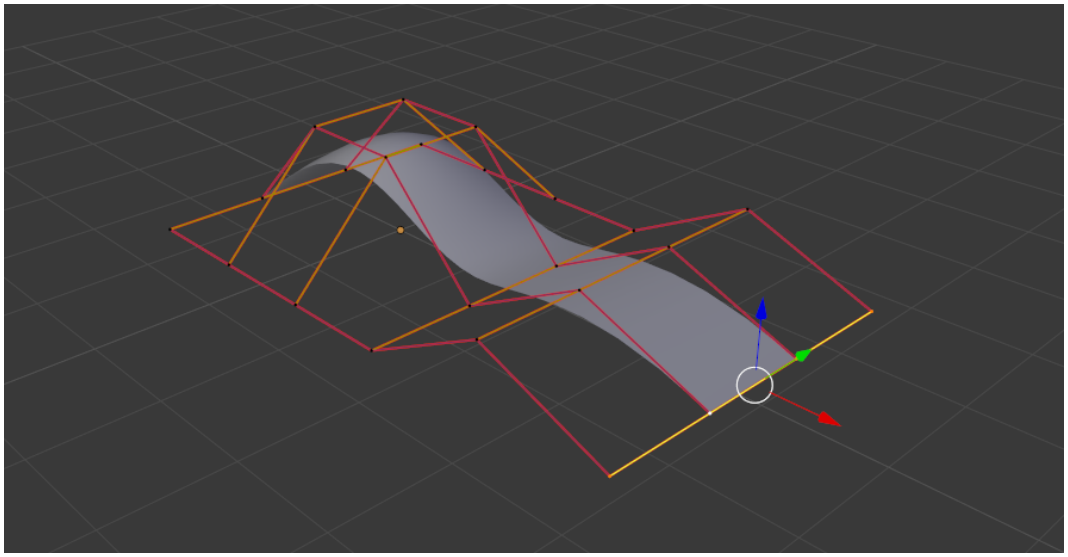


Selecting control point.



Select Control Point Row.

The edge is then extruded as shown in Fig. Extruding.. Notice how the mesh has bunched up next to the highlighted edge. That is because the *new* extruded surface section is bunched up there as well.



Extruding.

By moving the new section away from the area, the surface begins to "unbunch".

You can continue this process of extruding or adding new surface sections until you have reached the final shape for your model.

Make Segment

Reference

Mode::	Edit Mode
Menu::	*Surface* ‣ *Make Segment*
Shortcut::	F

Just like curves, merging two surfaces requires that a single edge, a border row of control points, from two separate surfaces is selected. This means that the surfaces must be part of the same object. For example, you cannot join two surfaces while in *Object Mode* – but you can of course, as with any objects of the same type, join two or more *Surface* objects – they just will not be "linked" or merged in a single one.

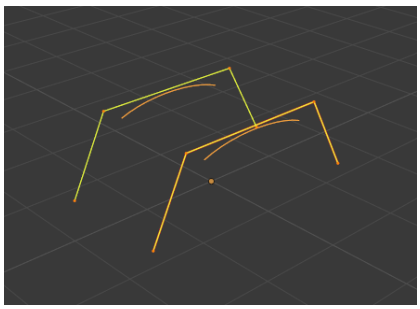
This tool is equivalent to creating edges or faces for meshes (hence its shortcut). The selection must contain only border rows of the same resolution (with the same number of control points), else Blender will try to do its best to guess what to merge with what, or the merge will fail (either silently, or stating that `Resolution does not match` if rows with different number of points are selected, or that there is `Too few selections to merge` if you only selected points in one surface…). To select control points of different surfaces, in the same object, you must use either box select or circle select; `Ctrl-LMB` will not work.

So to avoid problems, you should always only select border rows with the same number of points… Note that you can join a border U row of one surface with a border V row of another one, Blender will automatically "invert" the axis of one surface for them to match correctly.

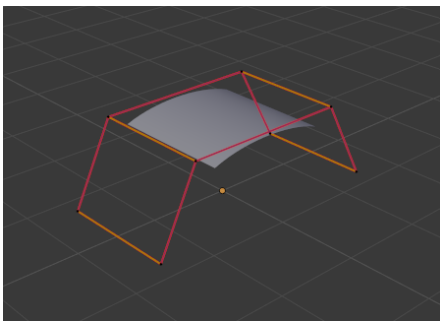
NURBS surface curves are often used to create objects like hulls, as they define cross sections all along the object, and you just have to "skin" them as described above to get a nice, smooth and harmonious shape.

Examples

Fig. Joining ready. is an example of two NURBS surface curves, **not** NURBS curves, in *Edit Mode*, ready to be joined. Fig. Joining complete. is the result of joining the two curves.



Joining ready.



Joining complete.

Smooth

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Control Points* ▸ *Smooth*

Iteratively smooths the selected control points by reducing the distance between neighboring control points.

Hooks

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Control Points* ▸ *Hooks*
Shortcut::	`Ctrl-H`

Hooks can be added to control one or more points with other objects.

Make Vertex Parent

Reference

Mode::	Edit Mode
Menu::	*Surface* ▸ *Control Points* ▸ *Make Vertex Parent*
Shortcut::	`Ctrl-P`

You can make other selected objects children of one or three control points, as with mesh objects.

To select a mesh (that is in view) while editing a surface, `Ctrl-P` click on it. Select either one or three control points, then `Ctrl-LMB` the object and use `Ctrl-P` to make a vertex parent. Selecting three control points will make the

child follow the median point between the three vertices. An alternative would be to use a Child Of constraint. See also the Curve modifier.

Segments

Subdivide

Reference

Mode::	Edit Mode
Menu::	*Surface ▸ Segments ▸ Subdivide*

The *Subdivision* operator simply subdivides all selected segments by adding one or more control points between the selected segments. Selected grids will be split into four smaller ones.

If used on a 1D surface (a "surface curve"), this tool works exactly as with curves.

Number of Cuts
: The number of subdivisions to perform.

Switch Direction

Reference

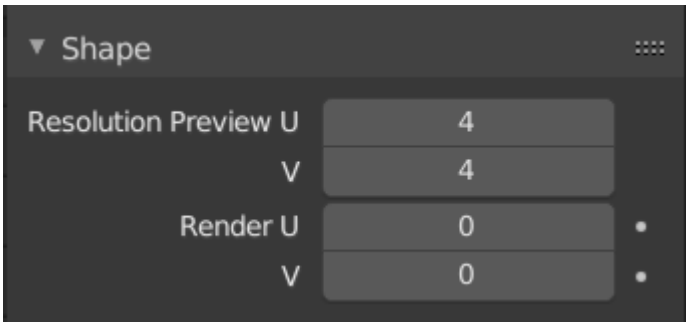
Mode::	Edit Mode
Menu::	*Surface ▸ Segments ▸ Switch Direction*

This tool will "reverse" the direction of any curve with at least one selected element (i.e. the start point will become the end one, and *vice versa*). Mainly useful when using a curve as path, or the bevel and taper options…

Properties

- Shape
- Active Spline

Shape

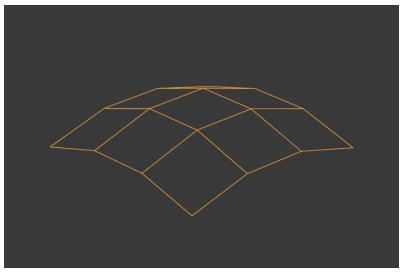


Shape panel.

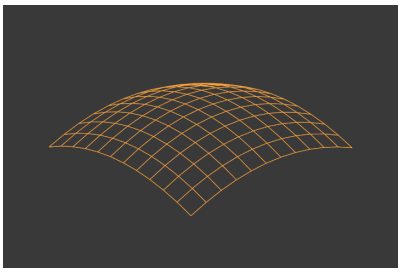
Resolution Preview U/V
: Resolution to use in the 3D Viewport.

Render U/V
: Just like NURBS curves, *Resolution* controls the detail of the surface. The higher the *Resolution* the more detailed and smoother the surface is. The lower the *Resolution* the rougher the surface. However, here you have two resolution settings, one for each interpolation axis (U and V).

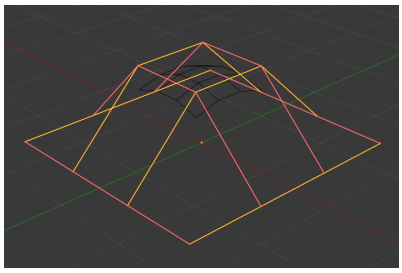
  You can adjust the resolution separately for both preview and render, to not slow things down in the viewport, but still get good render results.



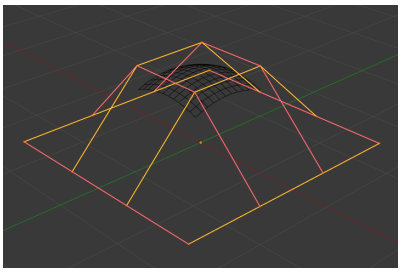
Resolution 1×1.



Resolution 3×3.



Resolution of 1 for both U and V.



Resolution of 3 for both U and V.

See also

The panels of the *Curve and Surface* tab are the same as for curves, just with fewer options…

Active Spline

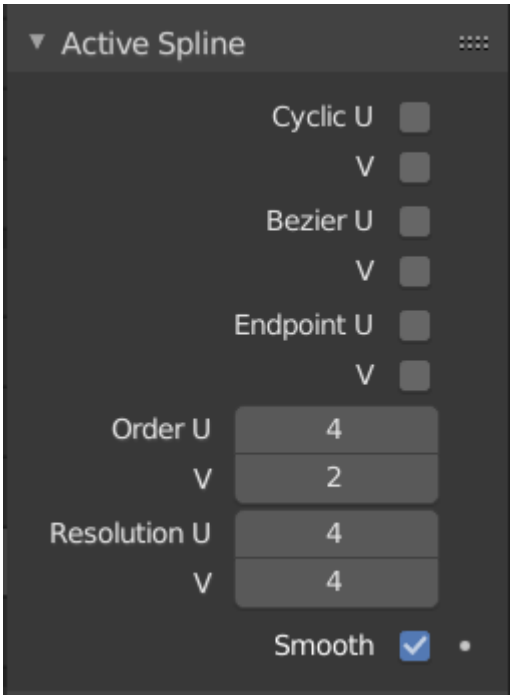
Reference

Mode:: Edit Mode
Menu:: *Properties* ▸ *Curve* ▸ *Active Spline*

See also

Active Spline for curves.

The *Active Spline* panel is used in Edit Mode to control properties of the currently selected spline.



Active Spline panel.

Cyclic U/V

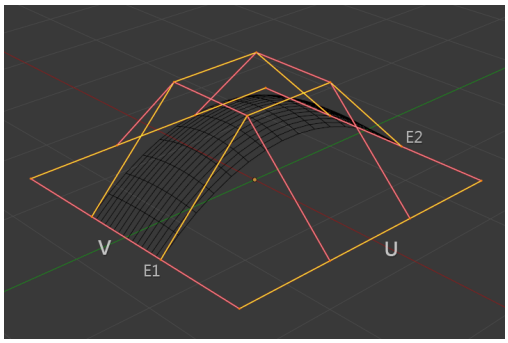
Like curves, surfaces can be closed (cyclical) or open, independently in both directions, allowing you to easily create a tube, torus or sphere shape, and they can be viewed as "solids" in *Edit Mode*. This can be set per interpolation axis.

Bézier U

Makes the surface act like a Bézier curve. The control points act like *Free* handles of a Bézier curve. Depending on the *Order*, 3 or 4 control points form one spline segment. This can be set per interpolation axis.

Endpoint U/V

Makes the surface contact the end control points. This can be set per interpolation axis.



Endpoint U.

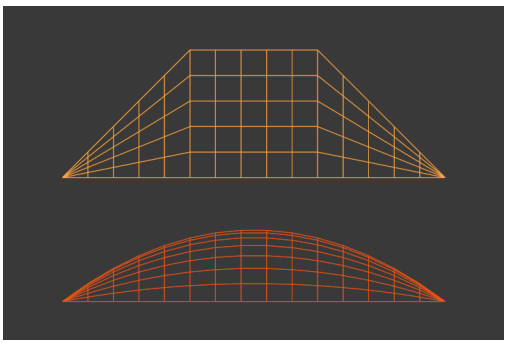
In the image below, the U interpolation axis is labeled as "U" and the V interpolation axis is labeled as "V". The U's interpolation axis has been set to *Endpoint* and as such the surface now extends to the outer edges from E1 to E2 along the U interpolation axis.

To cause the surface to extend to all edges, *Endpoint* would be set for the V's axis as well.

Order U/V

This property is the same as with NURBS Curves; it specifies how much the control points are taken into account for calculating the curve of the surface shape. For high Orders 1 the surface pulls away from the control points, creating a smoother surface by assuming that the

Resolution U/V is high enough. For lowest Orders 2 the surface follows the control points, creating a surface that tends to follow the grid cage.



Order 2 and Order 4 surface.

For illustration purposes, in both Fig. Order 2 and Order 4 surface., the knot vectors were set to *Endpoint*, causing the surface to extend to all edges.

You can set independently the order for each interpolation axis, and like curves, it **cannot** be lower than 2, and higher than 6 or the number of control points on the relevant axis.

Resolution U/V

Alters the resolution of each segment by changing the number of subdivisions. This can be set per interpolation axis.

Smooth

Use Smooth Shading for any 3D geometry.

Metaball

- Introduction
  - Visualization
- Toolbar
- Structure
  - Technical Details
  - Type
- Primitives
  - Options
- Editing
  - Deleting Elements
  - Conversion
  - Object Families
  - Examples
- Properties
  - Metaball
  - Active Element

Introduction

Metaball objects (short meta) are *implicit surfaces*, meaning that they are *not explicitly* defined by vertices (as meshes are) or control points (as surfaces are): they exist *procedurally*. Meta objects are literally mathematical formulas that are calculated on-the-fly by Blender.

A very distinct visual characteristic of metas is that they are fluid *mercurial*, or *clay-like* forms that have a "rounded" shape. Furthermore, when two meta objects get close to one another, they begin to interact with one another. They "blend" or "merge", as water droplets do, especially in zero-g (which, by the way, makes them very handy for modeling streams of water when you do not want to do a fluid simulation). If they subsequently move away from one another, they restore their original shape.

Each of these is defined by its own underlying mathematical structure, and you can at any time switch between them using the *Active Element* panel.

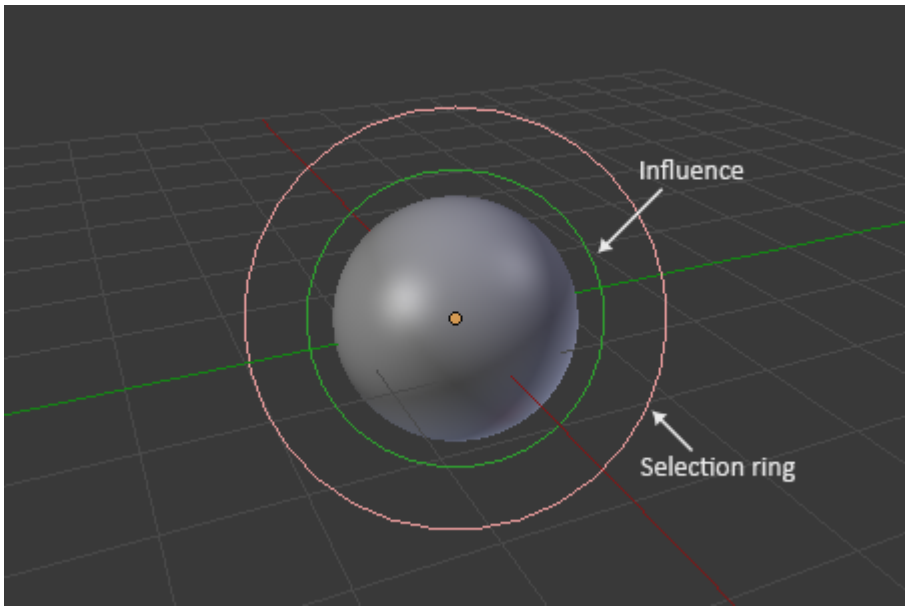
Typically *Meta* objects are used for special effects or as a basis for modeling. For example, you could use a collection of metas to form the initial shape of your model and then convert it to a mesh for further modeling or sculpting. Meta objects are also very efficient for ray tracing.

Warning

Names of *Meta* objects are very important, as they define families, and only objects within a same family interact with each other. Unlike other object types, even edition (transformations) in Object Mode will affect the generated geometry within the edited families.

Visualization

In Object Mode, the calculated mesh is shown, along with a black "selection ring".



Meta Ball in Edit Mode.

In Edit Mode (Fig. Meta Ball in Edit Mode.), a meta is displayed as a mesh (either shaded or as black wireframe, but without any vertex of course), with two colored circles: a red one for selection (pink when selected), and a green one for a direct control of the meta's stiffness (light green when active). Note that except for the scale transformation, having the green circle highlighted is equivalent to having the red one.

Toolbar

Metaball Edit Mode tools:

Tweak
: Select or move.

  Select Box
  : Select objects by dragging a box. All objects that intersect the box will be selected.

  Select Circle
  : Select objects by dragging a circle. All objects that intersect the path of the circle will be selected.

  Select Lasso
  : Select objects by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform
: Tool to adjust the objects translation, rotations and scale.

Annotate
: Draw free-hand annotation.

Annotate Line
: Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Measure
: Measure distances in the scene.

Shear
: Shear selected items along a defined axis.

Structure

Technical Details

A more formal definition of a meta object can be given as a *directing structure* which can be seen as the source of a static field. The field can be either positive or negative and hence the field generated by neighboring directing structures can attract or repel.

The implicit surface is defined as the surface where the 3D field generated by all the directing structures assume a given value. For example a meta ball, whose directing structure is a point, generates an isotropic (i.e. identical in all directions) field around it and the surfaces at constant field value are spheres centered at the directing point.

Meta objects are nothing more than mathematical formula that perform logical operations on one another (AND, OR), and that can be added and subtracted from each other. This method is also called *Constructive Solid Geometry* (CSG). Because of its mathematical nature, CSG uses little memory, but requires lots of processing power to compute.

See also

The Wikipedia page about metaballs.

Type

Reference

Mode::	Edit Mode
Panel::	*Sidebar region* ▸ *Transform panel* ▸ *Type*, *Metaball tab* ▸ *Active Element panel* ▸ *Type*

Blender has five types of metas, each determined by its underlying (or directing) structure.

In Edit Mode, you can change this structure, either using the relevant buttons in the *Active Element* panel, or the selector in the *Transform* panel in the Sidebar region. Depending on the structure, you might have additional parameters, located in both *Transform* panel and *Active Element* panel.

Ball (point, zero-dimensional structure)
: This is the simplest meta, without any additional setting. As it is just a point, it generates an isotropic field, yielding a spherical surface (this is why it is called *Meta Ball* or *Ball* in Blender).

Capsule (straight line, one-dimensional structure)
: This is a meta which surface is generated by the field produced by a straight line of a given length. This gives a cylindrical surface, with rounded closed ends.

  Size X
  : The length of the line (and hence, of the capsule).

Plane (rectangular plane, two-dimensional structure)
: This is a meta which surface is generated by the field produced by a rectangular plane. This gives a parallelepipedal surface, with a fixed thickness, and rounded borders.

  Size X/Y
  : The length and width of the rectangle.

Ellipsoid (ellipsoidal volume, three-dimensional structure)
: This is a meta which surface is generated by the field produced by an ellipsoidal volume. This gives an ellipsoidal surface.

  Size X/Y/Z
  : The length, width and height of the ellipsoid.

Cube (parallelepipedal volume, three-dimensional structure)

This is a meta which surface is generated by the field produced by a parallelepipedal volume. This gives a parallelepipedal surface, with rounded edges.

Size X/Y/Z

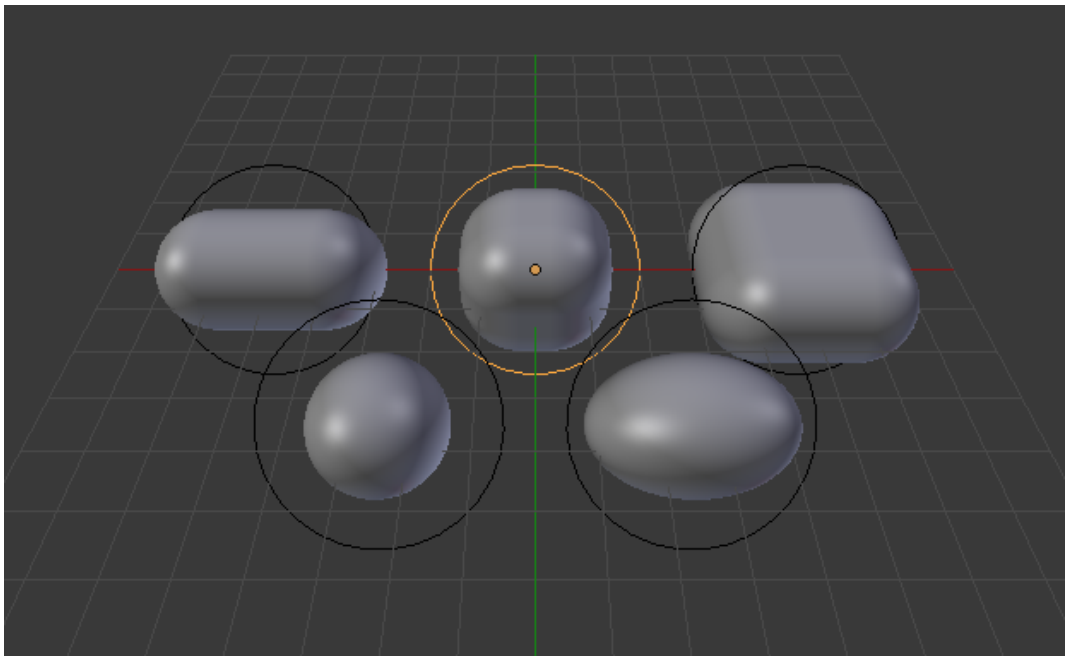
The length, width and height of the parallelepiped.

Primitives

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Add ▸ Metaball*
Shortcut::	`Shift-A`

There are five predefined metaball “primitives” (or configurations) available in the *Add ▸ Metaball* submenu:



The five Metaball primitives.

Options

Primitive
: Ball
: Adds a meta with a point underlying structure.

Capsule
: Adds a meta with a line segment underlying structure.

Plane
: Adds a meta with a planar underlying structure.

Ellipsoid
: Adds a meta with an ellipsoidal underlying structure.

Cube
: Adds a meta with a volumetric cubic underlying structure.

See also

A more detailed explanation of each primitive in the Structure page.

Radius, Align to View, Location, Rotation
: See Common Object Options.

Editing

In addition to having several meta objects in a same family, you can also have several meta primitives in a single object (just add some more while in Edit Mode). Each will be an element, with its own shape, editing rings (in the viewport), and settings.

Deleting Elements

Reference

Shortcut:: `X`, `Delete`

You can only delete the active element, no fancy options here.

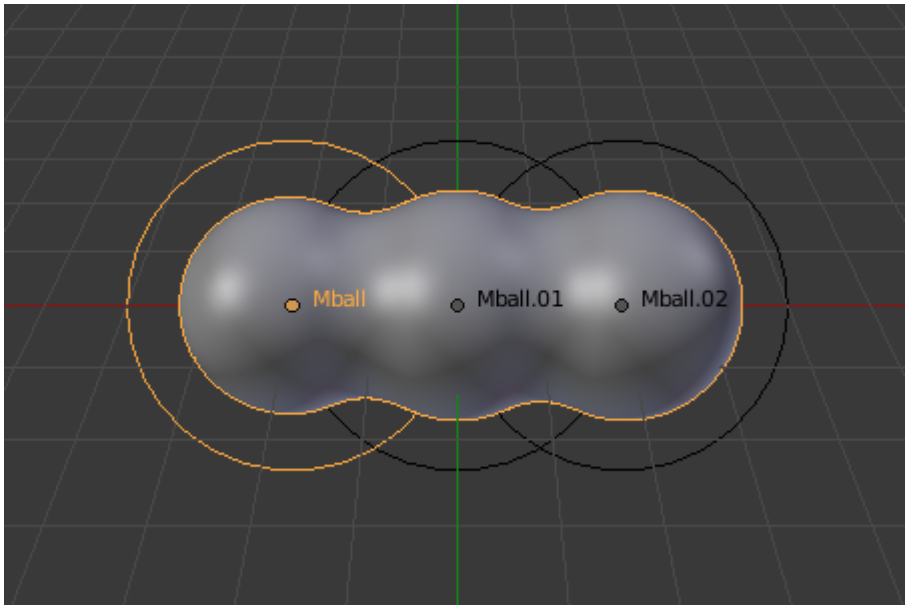
Conversion

To convert the meta to a real mesh, use Convert in Object Mode.

Object Families

A "family" is a way to regroup several meta objects, producing something very similar to having several metas inside the same object.

It is defined by the left part of an object's name (the one before the first dot). Remember, an object's name is the one in the *Object Name* field, in most panels, **not** the *Metaball Name* field, which is the meta data-block's name… For example, the *family* part of "MetaPlane.001" is `MetaPlane`. Each meta object in the same "family" is associated with one another as discussed below.



Metaball family.

Families of metas are controlled by a *base* meta object which is identified by an object name **without** a dot in it. For example, if we have three metas called `MetaThing`, `MetaThing.001`, `MetaThing.round`, the *base* meta object would be `MetaThing`.

The *base* meta object determines the basis, the resolution, the threshold, *and* the transformations. It also has the material and texture area. In a way, the *base* meta is the "owner" of the other metas in the family (i.e. it is as if the other metas were "included" or joined into the base one).

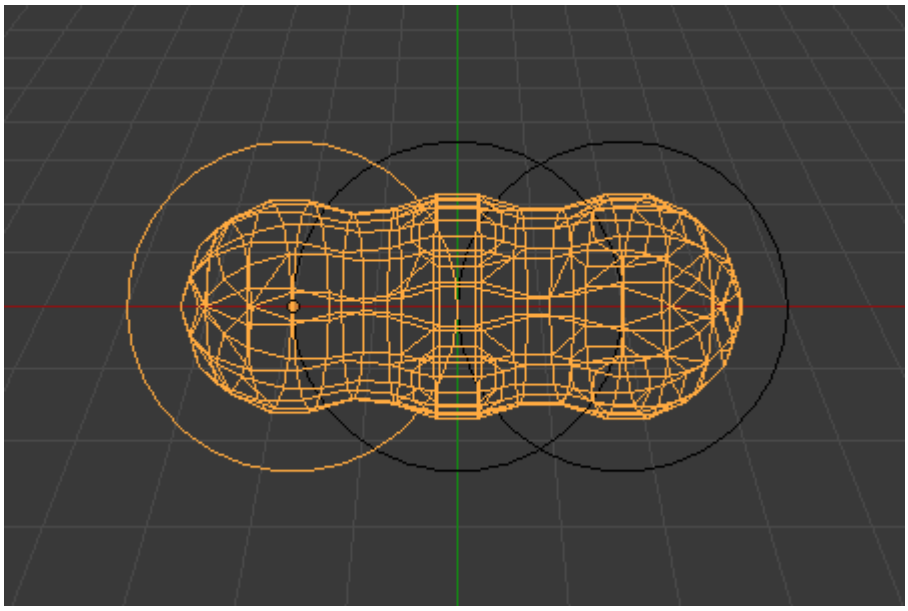
Hint

When working with multiple scenes, take care naming your meta objects so the *base* is always in the same scene as other metas.

Failing to do so will give confusing behaviors (like invisible meta objects).

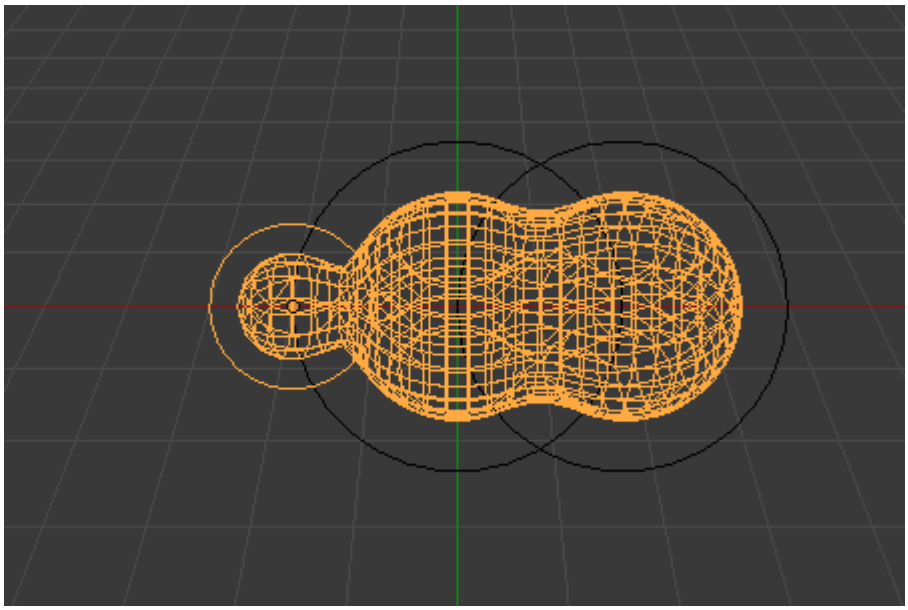
Examples

Fig. Meta ball base. shows the *base* meta labeled “B”. The other two *Meta* objects are *children*. Children’s selection rings are always black, while the group’s mesh is orange. Because the metas are grouped, they form a unified mesh which can always be selected by selecting the mesh of any meta in the group.



Meta ball base.

For example, in Fig. Meta ball base., only the lower sphere (the parent) has been selected, and you see that both the parent’s mesh *and* all of the children’s meshes are now highlighted.



Scaling the “base”.

The *base* meta object controls the *polygonalization* (mesh structure) for the group, and as such, also controls the polygonalization for the children (non-base) metas. If we transform the *base* meta, the children’s polygonalization changes. However, if we transform the children, the polygonalization remains unchanged.

Hint

This discussion of “polygonalization” does *not* mean that the various meshes do not deform towards or away from each other (meta objects always influence one another in the usual way, within a same family).

Rather, it means that the underlying mesh structure changes only when the *base* object transforms. For example, if you scale the *base*, the children’s mesh structure changes.

In Fig. Scaling the “base”., the *base* has been scaled down, which has the effect of scaling the mesh structure of each of the children. As you can see, the children’s mesh resolution has increased, while the *base* decreased. The children did *not* change size!

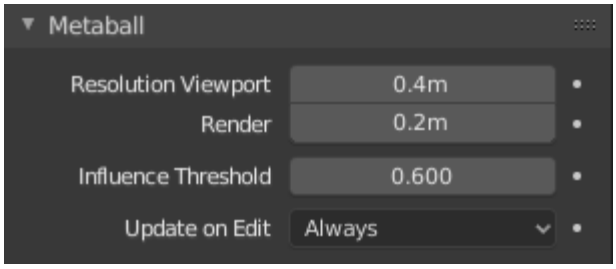
Properties

All meta objects of a same family in a scene interact with each other. The settings in the *Metaball* section apply to all meta objects of the active family. In Edit Mode, the *Active Element* panel is shown for editing individual metaball elements.

Metaball

Reference

Mode:: Object and Edit Mode
Panel:: *Properties* ▸ *Metaball* ▸ *Metaball*



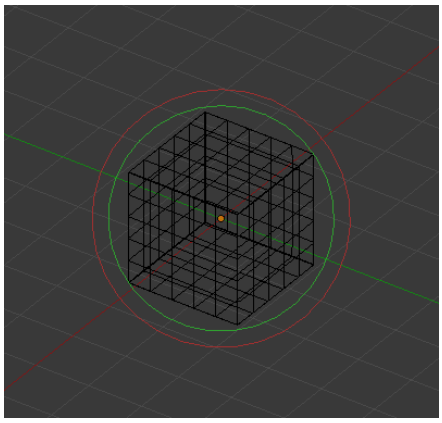
Family meta properties.

Resolution Viewport
: Controls the resolution of the resultant mesh as generated by the *Meta* objects. The 3D Viewport resolution of the generated mesh; finest to coarsest.

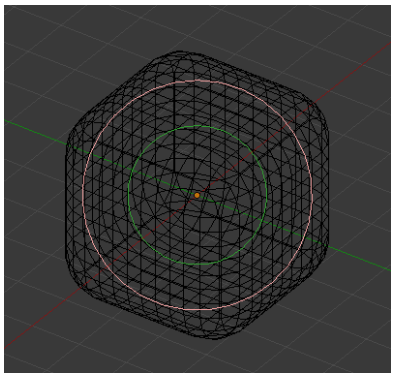
Render
: The rendered resolution of the generated mesh; finest to coarsest.

Tip

One way to see the underlying mathematical structure is to lower the *Resolution*, increase the *Threshold* and set the *Stiffness* (see below) a fraction above the *Threshold*. Fig. Underlying structure. is a meta cube with the above mentioned configuration applied as follows: *Resolution* of 0.410, *Threshold* of 5.0 and *Stiffness* a fraction above at 5.01.



Underlying structure.



Meta cube shape.

You can clearly see the underlying cubic structure that gives the meta cube its shape.

Influence Threshold

Defines how much a meta's surface "influences" other metas. It controls the *field level* at which the surface is computed. The setting is global to a group of *Meta* objects. As the threshold increases, so does the influence that each meta has on each other.

There are two types of influence: *positive* or *negative*.

Update on Edit

While transforming metas (move, scale, etc.), you have four "modes" of visualization.

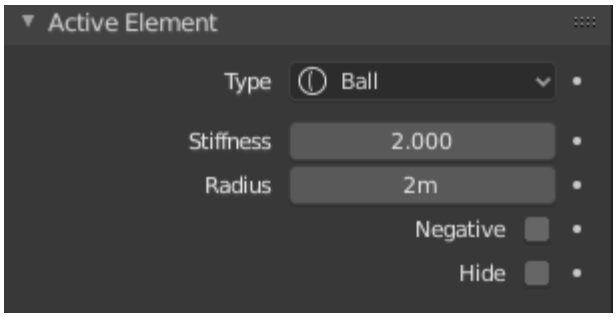
This should help you if you experience difficulties (metas are quite computationally intensive…), but with modern computers, this should not happen, unless you use many metas, or very high resolutions…

Always:: Fully display the meta during transformations.
Half:: During transformations, display the meta at half its *Viewport* resolution.
Fast:: Do not display meta during transformations.
Never:: Never show meta mesh (not a very recommended option, as the meta is only visible at render time!).

Active Element

Reference

Mode:: Edit Mode
Panel:: *Properties* ▸ *Metaball* ▸ *Active Element*



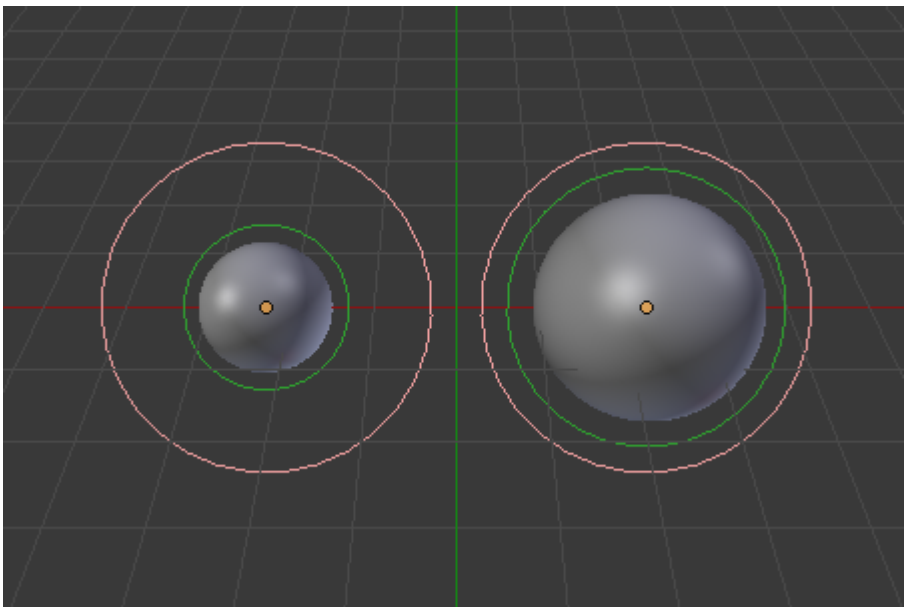
Active Element panel.

These settings apply only to the selected metaball element.

Type
Changes the primitive shape of the meta object.

Stiffness
Controls the influence range for individual metaball elements, unlike *Influence Threshold* which controls the influence for the entire meta family. This essentially defines how sensitive a meta is to being affected by other metas. With a low stiffness, the meta will begin to deform from further away. A higher value means the meta needs to be close to another one to begin merging. The *Stiffness* is visualized by the *green ring* and can be selected and scaled to also change the *Stiffness* value.

To be visible, the *Stiffness* must be slightly larger than the *Threshold* value.



The left meta ball, has a smaller *Stiffness* value than the right one.

Radius

Controls the physical size of the metaball. This works the same as scaling the metaball in Object Mode. The *Radius* is visualized by the *white ring* and can be selected and scaled to also change the *Radius* value.

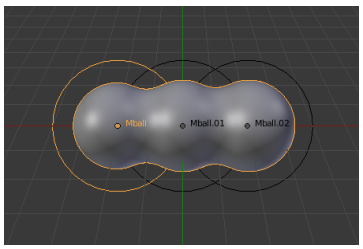
Negative

Controls whether the influence is *positive* or *negative*.

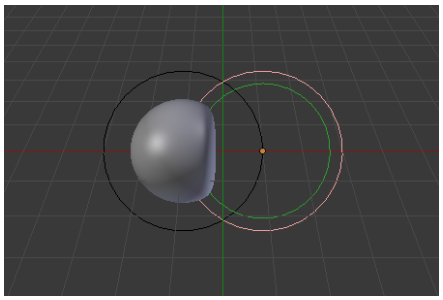
A *positive* influence is defined as an attraction, meaning that the meshes will stretch towards each other as the *rings of influence* intersect. The opposite effect would be a *negative* influence where the objects repel each other.

Note

If a metaball has *Negative* influence the meta is not visible in the 3D Viewport, only the surrounding circles are shown.



Positive influence of three metaballs.



Negative influence of a meta ball; the first is negative and the second positive.

Hide

As in Show/Hide in Object Mode, you can hide the selected meta(s), and then reveal what was hidden. This is very handy for cleaning your views up a bit.

Note

- Hiding a meta does not *only* hide it, but also disables it from the meta computation, which will affect the final geometry.
- The two red and green rings always remain visible in Edit Mode, as well as the select circle in Object Mode.

Text

- Introduction
- Selecting
- Editing
  - Cut
  - Copy
  - Paste
  - Paste File
  - To Uppercase
  - To Lowercase
  - Insert Unicode
  - Special Characters
  - Toggle Bold, Italics, Underline, Small Caps
  - Kerning
  - Delete
  - Inserting Text
  - Converting to a Mesh or Curve
  - Assigning Materials
- Properties
  - Shape
  - Texture Space
  - Geometry
  - Font
  - Paragraph
  - Text Boxes

Introduction

Text objects contain some text, and are in the same object type family as curves and surfaces ones, as fonts are vector data (they are made of curves).

Blender uses a “Font System” to manage mapping letter codes to geometry representing them in the 3D Viewport. This font system has its own *built-in* font, but it can use external fonts too, including *PostScript Type 1*, *OpenType* and *TrueType* fonts. And moreover, it can use any objects existing in the current blend-file as letters.



An example of an extruded text.

Text objects allow you to create and render 2D or 3D text, with various advanced layout options, like justifying and frames. By default, letters are just flat filled surfaces, exactly like any closed 2D curve. But, just like curves, you can extrude them, and apply modifiers to them (e.g. to make them follow a curve).

Text in Blender can be laid out in some relatively advanced ways, defining columns or blocks of text, using different alignments, and so on.

Those features are similar in concept to what you can find in DTP software (like Scribus), although at a very basic level currently.

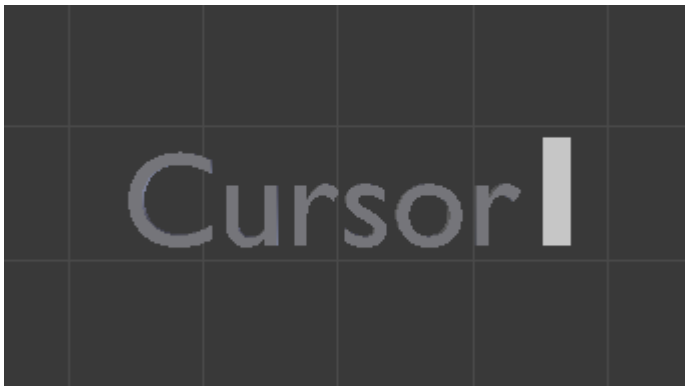
Tip

You can convert a text object, either to a curve, or directly to a mesh, using Convert in Object Mode.

Note

A maximum of 50,000 characters is allowed per text object. However, be forewarned that the more characters a single text object has, the slower the object will respond interactively.

Selecting



Text in Edit Mode.

In Edit Mode, your text has a white cursor, and as in any text editor, it determines where new chars will be inserted.

Select All `Ctrl-A`
: Selects the full text.

Top/Bottom `Shift-Ctrl-Home`/ `Shift-Ctrl-End`
: Moves the cursor to the start or end of the text object.

Next/Previous Character `Left`/ `Right`
: You can move the cursor with the arrow keys.

Next/Previous Word `Ctrl-Left`/ `Ctrl-Right`
: To move the cursor on a word's boundary.

Line Begin/End `Home`/ `End`
: Move the cursor to the beginning and end of a line respectively.

Next/Previous Line `Up`/ `Down`
: To jump between lines.

Next/Previous Block `PageUp`/ `PageDown`

To jump back/forward ten lines at a time.

Hold `Shift` while using the arrow keys to select a part of the text. You can use it to specify different materials, the normal/bold/italic style…

Editing

Editing text is quite different from other object types in Blender, and happens mainly in two areas. First, the 3D Viewport, where you type your text, and have a few shortcuts, e.g. for applying styles (see Font) – note however, that most Blender shortcuts you know in Edit Mode do not exist for texts. The second place is the Properties, especially the *Font* tab.

Editing text objects is similar to using a standard text editor but is not as full-featured and has some differences. The menu of the 3D Viewport header offers few options. You have no transform nor mirror tools, and so on. To leave Edit Mode use `Tab` as it does not insert a tab character in the text, but rather enters and exits Edit Mode, as with other object types.

Cut

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header* ▸ *Text* ▸ *Cut*
Shortcut::	`Ctrl-X`

To cut and copy text to the buffer, use the shortcut or the matching entry in the *Edit* menu.

Copy

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header* ▸ *Text* ▸ *Copy*

Shortcut:: `Ctrl-C`

To copy text to the buffer, use the shortcut or the matching entry in the *Edit* menu.

Paste

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *Header ▸ Text ▸ Paste*
Shortcut:: `Ctrl-V`

To paste text from the buffer, use the shortcut or the matching entry in the *Edit* menu.

Paste File

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *Header ▸ Text ▸ Paste File*

Inserts text from and external text file. This will bring up a File Browser for navigating to a valid UTF-8 file. As usual, be careful that the file does not have too many characters, as interactive response will slow down.

To Uppercase

Reference

Editor:: 3D Viewport

Mode:: Edit Mode
Menu:: *Header ▸ Text ▸ To Uppercase*

Changes the case of the selected text to uppercase.

To Lowercase

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *Header ▸ Text ▸ To Lowercase*

Changes the case of the selected text to lowercase.

Insert Unicode

Reference

Editor:: 3D Viewport
Mode:: Edit Mode
Menu:: *Header ▸ Text ▸ Insert Unicode*

Opens a dialog box to input the entry of any Unicode character by entering its hexadecimal codepoint value.

See Wikipedia for a List of Unicode characters which list there respective hexadecimal codepoint values.

Special Characters

Reference

Editor:: 3D Viewport

Mode::	Edit Mode
Menu::	*Header ▸ Text ▸ Special Characters*

This is a limited character map to insert characters which aren't available from the keyboard. Many other special characters can be "composed", see Accent Characters. If you need others, you will have to copy-paste them from an external editor or character map program.

Note

The text buffer is in sync with the desktop clipboard. But if it is used within Blender the text formatting will be copied as well. For other ways of inserting a text, see Inserting Text.

Toggle Bold, Italics, Underline, Small Caps

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header ▸ Text ▸ Toggle Bold/Italics/Underline/Small Caps*

To apply the *Bold*, *Italics*, *Underline* or *Small Caps* attribute to a set of characters, you either turn on the related setting prior to typing characters, or select existing text, and then toggle desired style from the menu.

Warning

Blender's *Bold* and *Italic* buttons do not work the same way as in other applications, as they also serve as placeholders for you to load up other fonts manually.

Kerning

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header ▸ Text ▸ Kerning*

Font kerning is the space between individual characters.

Decrease Kerning `Alt-Left`
: Decreases the spacing between the characters on either side of the cursor.

Increase Kerning `Alt-Right`
: Increase the spacing between the characters on either side of the cursor.

Reset Kerning
: Sets the spacing between the characters on either side of the cursor to their initial value.

Delete

Reference

Editor::	3D Viewport
Mode::	Edit Mode
Menu::	*Header ▸ Text ▸ Delete*

Previous/Next Character
: Deletes the character before or after the cursor.

Previous/Next Word
: Deletes the word before or after the cursor.

Inserting Text

You can insert text in two ways: from the internal text buffer or from a text file.

Using an existing text data-block, you can convert it to an object from the Text editor's header, select *Edit* ▸ *Text to 3D Object*, *One Object* or *One Object per Line* depending on your needs.

It is also possible to paste from the clipboard or a file, while editing 3D text.

Accent Characters

Many special characters (such as accented chars, which are not directly available on your keyboard) can be "composed" using a combination of two other characters. To do so, type the main char, press `Alt-Backspace`, and then press the desired "modifier" to produce the special character. Some examples are given below:

- ã: `A`, `Alt-Backspace`, `~`
- á: `A`, `Alt-Backspace`, `'`
- à: `A`, `Alt-Backspace`, `\`
- â: `A`, `Alt-Backspace`, `^`
- å: `A`, `Alt-Backspace`, `O`
- æ: `A`, `Alt-Backspace`, `E`
- ª: `A`, `Alt-Backspace`, `-`
- ë: `E`, `Alt-Backspace`, `"`
- ç: `C`, `Alt-Backspace`, `,`
- ¢: `C`, `Alt-Backspace`, `|`
- ø: `O`, `Alt-Backspace`, `/`
- §: `S`, `Alt-Backspace`, `S`
- †: `|`, `Alt-Backspace`, `-`
- ‡: `|`, `Alt-Backspace`, `=`
- ©: `O`, `Alt-Backspace`, `C`
- ®: `O`, `Alt-Backspace`, `R`
- ™ : `T`, `Alt-Backspace`, `M`
- ½: `1`, `Alt-Backspace`, `2`
- ÷: `-`, `Alt-Backspace`, `:`
- ±: `-`, `Alt-Backspace`, `+`

Converting to a Mesh or Curve

In Object Mode, it is possible to convert a text object to a mesh or curve one, see Convert.

Tip

The topology of the result is usually a bit messy, so it may be useful to use a *Limited Dissolve* deletion, or apply a Remesh modifier at a low threshold, to clean up your mesh.

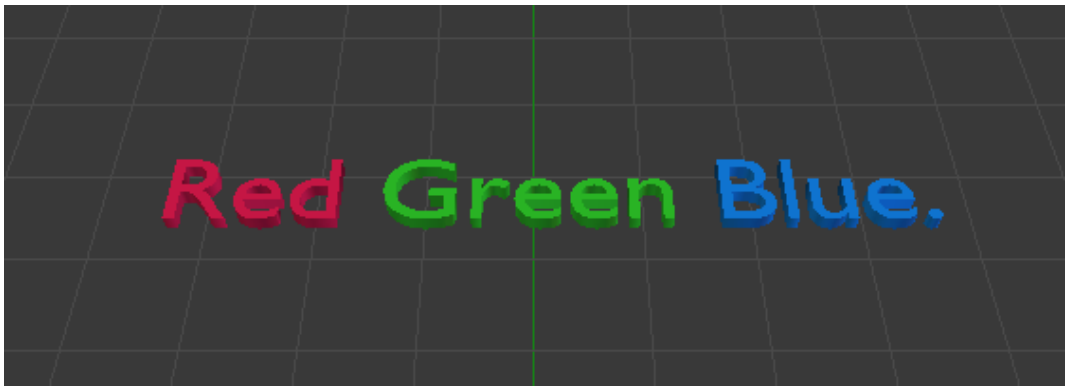
Assigning Materials

Reference

Mode::	Edit
Panel::	*Properties* ▸ *Materials*

Each character can have a different *Material index* in order to have different materials on different characters.

You can assign indices either as you type, or after by selecting blocks of text and clicking on the *Assign* button in the *Materials* panel.



Red Green Blue text example.

Properties

Shape

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Text* ▸ *Shape*

Most of the settings in the *Shape* panel are shared with those of Curves data-blocks, please refer to those for details.

Fast Editing
: Does not fill the letters in Edit Mode, only show their outline.

Texture Space

Each Object can have an automatically generated UV map, these maps can be adjusted here.

See Generated UV Properties for more information.

Geometry

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Font* ▸ *Geometry*

Offset
: Offset the control points of the curves defining the letters, which will make them thinner or thicker. Use with care, it can quickly lead to artifacts…

Extrusion, Taper & Bevel

The remaining settings of that panel, which are used to give volume to the letters, are also shared with the Curves data-blocks, please refer to those for details.

Note

How the Taper object effect works depends on how the curves defining the letters are built. The results can often look quite random…

Note

Bevel is applied to the curves defining the letters, which means that usually it will follow their outlines (there will be two parallel beveled curves, and not a single one, as one might expect).

Font

Reference

Mode:: All Modes
Panel:: *Properties* ▸ *Font* ▸ *Font*

Blender comes with a *built-in* font by default that is displayed in each of the four font style data-block menus. The *built-in* font is always present and shows in this list as "Bfont". The data-block menu contains a list displaying the currently loaded fonts. Select one for each font style.

To load a different *Font*, click one of the *Load* buttons in the *Font* panel and navigate to a font file. The File Browser will give all valid fonts a capital "F" icon.

If you select a font that is unsupported by Blender, you will get the error `Not a valid font.`

Note

Location of Fonts on Unix

Fonts are typically located under `/usr/lib/fonts`, or some variant like `/usr/lib/X11/fonts`, but not always. They may be in other locations as well, such as `/usr/share/local` or `/usr/local/share`, and possibly related sub-trees.

Remember that the same font will be applied to all chars with same style in a text, but that a separate font is required for each style. For example, you will need to load an *Italics* font in order to make characters or words italic. Once the font is loaded you can apply that font "Style" to the selected characters or the whole object. In all, you would need to load a minimum of four different types of fonts to represent each style (Normal, Italics, Bold, Bold & Italics).

It is important to understand, that Blender does not care what font you load for "normal", "bold", etc., styles. This is how you can have up to four different fonts in use in the same text, but you have to choose between different styles of a same font, or different fonts. Blender has a number of typographic controls for changing the style and layout of text, found in the *Font* panel.

Bold
: With no text selected, toggles new text to be **bold**. With text selected, toggles the selected text to be **bold**. Text can also be set to bold by selecting it then using the *Bold* entry in the *Text* menu of the 3D Viewport.

Italic
: With no text selected, toggles new text to be *italic*. With text selected, toggles the selected text to be *italic*. Text can also be set to italic by selecting it then using the *Bold* entry in the *Text* menu of the 3D Viewport.

Underline

With no text selected, toggles new text to be underline. With text selected, toggles the selected text to be underline. Text can also be set to underline by selecting it then using the *Underline* entry in the *Text* menu of the 3D Viewport.

See also underline settings below.

Small Caps

With no text selected, toggles new text to be small capitals. With text selected, toggles the selected text to be small capitals. Text can also be set to small caps by selecting it then using the *Small Caps* entry in the *Text* menu of the 3D Viewport.

The size of the *Small Caps* can be changed with the Small Caps Scale setting.

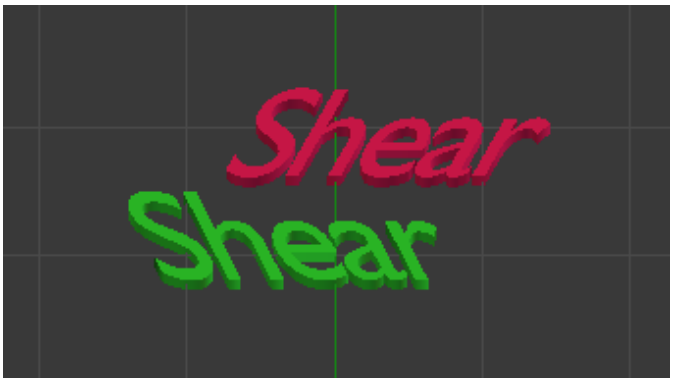
Transform

Size

Controls the size of the whole text (no way to control each char size independently). Note however, that chars with different fonts (different styles, see below) might have different visible sizes.

Shear

Controls the inclination of the whole text. As similar as it may seem, this is not the same thing as italics style.



Shear example.

Object Font

Allows individual objects to be used to render fonts, you can create/model your own complex font inside Blender! This field is used to select the objects prefix name (object "family") to be used to locate the individual characters used for typing. This is quite a complex process, so here are detailed steps to follow:

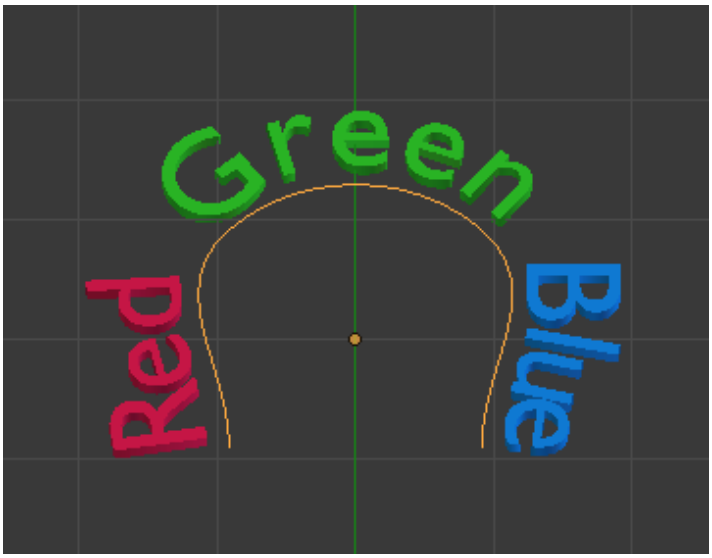
1. Create the font characters, each character can be any object type (mesh, curve, etc.). They must all have a name following the naming schema: "common prefix" followed by the "character name" (e.g. "ft.a", "ft.b", etc.).
2. For the text object, enable Instancing Vertices.
3. In the *Font* tab, fill the *Object Font* field with the "common prefix" of your "font" objects. Now, each time a character in your text matches the *suffix part* of a "font" object's name, this object is duplicated on this character.

Note

The objects are duplicated so that their center is positioned at the *lower right corner* of the corresponding characters.

Text on Curve

Select a curve object for the text object to follow.



Text on curve.

Tip

You should rather use the Curve modifier, which offers more control, and is the standard way to achieve such effects in modern Blender.

Underline Position
: This allows you to shift vertically the position of the underline.

Underline Thickness
: This controls the thickness of the underline.

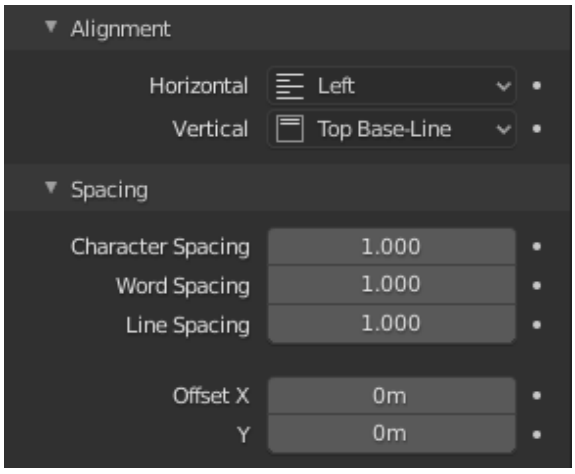
Small Caps Scale
: The scaling applied to capital letters to turn them into small caps.

Paragraph

Reference

Mode:: All
Panel:: *Properties* ▸ *Font* ▸ *Paragraph*

The *Paragraph* Panel has settings for the alignment and spacing of text.



The Paragraph panel.

Alignment

Horizontal

Left:: Aligns text to the left of the frames when using them, else uses the origin of the text object as the starting point of the text (which grows to the right).

Center:: Centers text in the frames when using them, else uses the origin of the text object as the mid-point of the text (which grows equally to the left and right).

Right:: Aligns text to the right of the frames when using them, else uses the origin of the text object as the ending point of the text (which grows to the left).

Justify:: Only flushes a line when it is terminated by a word wrap (**not** by a newline), and uses *whitespace* instead of *character spacing* (kerning) to fill lines.

Flush:: Always flushes the line, even when it is still being typed-in. It uses character spacing (kerning) to fill lines.

Note

Both *Justify* and *Flush* only work within frames.

Vertical

Top::

- With text boxes, aligns the top of the text to the to of the frames.
- With no text box, aligns the top of the text to the origin of the object, and grows to the bottom.

Top Baseline::

- With text boxes, aligns the 'top' baseline of the te to the top of the frames.
- With no text box, aligns the actual baseline of the text to the origin of the object, and grows to the bottom.

Note

That difference of reference point in the first line depending on usage of boxes or not is indeed

confusing.

Middle::
- With text boxes, centers the text in the frames.
- With no text box, centers the text on the origin of the object, and grows in both top and bottom directions equally.

Bottom Baseline::
- With text boxes, aligns the baseline of the text to the bottom of the frames.
- With no text box, aligns the baseline of the text to the origin of the object, and grows to the top.

Bottom::
- With text boxes, align the bottom of the text to the bottom of the frames.
- With no text box, align the bottom of the text to th origin of the object, and grows to the top.

Spacing

Character Spacing
: A factor by which space between each character (kerning) is scaled in width.

  In Edit Mode in the 3D Viewport, you can also control individual kerning at text cursor position by pressing `Alt-Left` / `Alt-Right` to decrease/increase it.

Word Spacing
: A factor by which whitespace between words is scaled in width.

Line Spacing
: A factor by which the vertical space between lines is scaled.

Offset X, Y
: These settings control the X and Y offset of the text position within the object. This applies relatively to the object's origin, either to the whole text or, when using text boxes, to each frame.

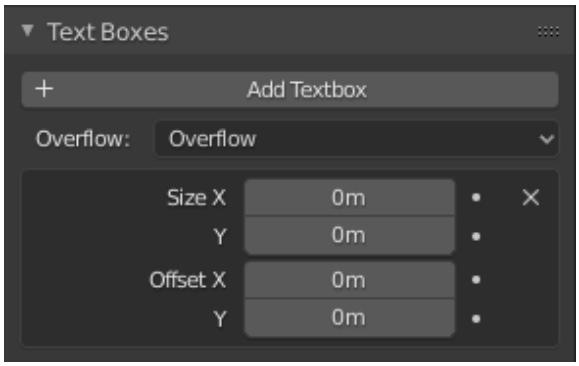
Text Boxes

Reference

Mode:: All
Panel:: *Properties ▸ Font ▸ Text Boxes*

Text boxes (or frames) allow you to distribute the text among rectangular areas within a single text object. An arbitrary number of freely movable and re-sizable text frames are allowed per text object.

The text flows continuously from the lowest-numbered frame to the highest-numbered frame with text inside each frame word-wrapped. It flows between frames when a lower-numbered frame cannot fit any more text. If the last frame is reached, text overflows out of it (by default, see options below).



Text Boxes panel.

Add Textbox
: Inserts a new frame, just after the current one (in text flow order). The new frame will have the same size and position as the selected one.

.._bpy.ops.font.textbox_remove:

(Remove Text Block)
: Delete the current frame.

Overflow
: How to handle text overflowing available space in the defined boxes.

Overflow:: Just keep text running out of the last box.

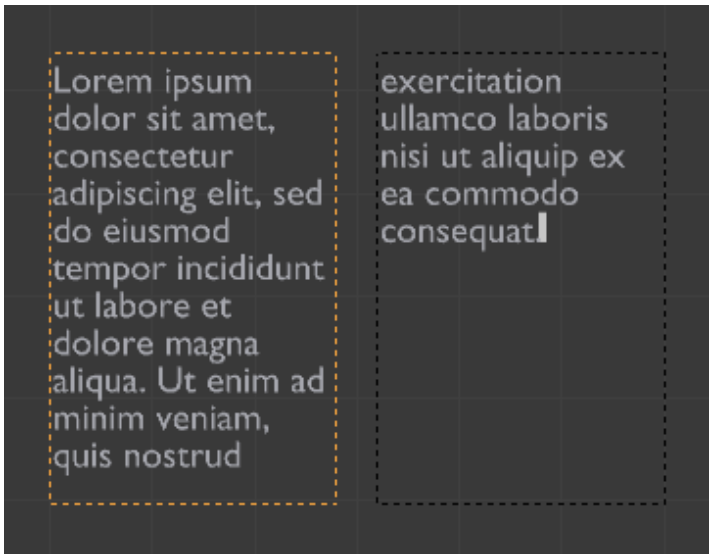
Scale to Fit:: Scale text to fit into the available space.

Truncate:: Hide the end of the text that does not fit into the available space.

> **Note**
> It will only truncate in *Object Mode*, in *Edit Mode* the whole text remains visible (and overflows as needed).

Size X, Y
: Specifies the width and height of the text box, if set to **zero** no word wrapping happens (it is ignored, and the whole text box system is disabled if all are set to a null size).

Offset X, Y
: Controls the *X* and *Y* offset of the frame, i.e. its position.



Multiple columns, text flowing between boxes.

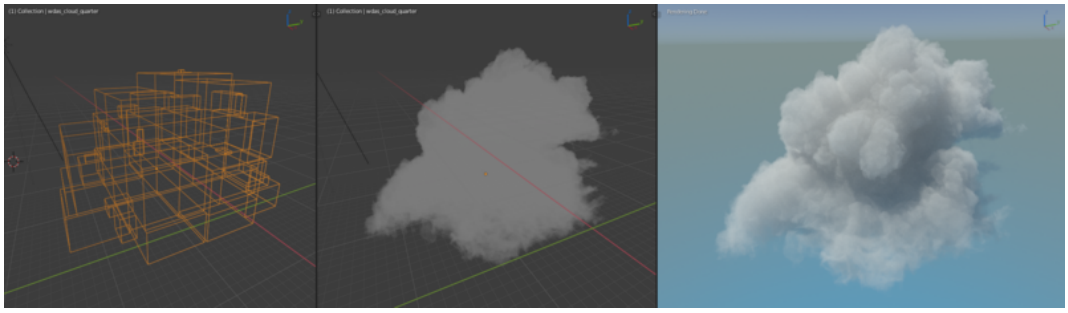
Volumes

- Introduction
  - Rendering
  - Limitations
- Properties
  - Grids
  - OpenVDB File
  - Viewport Display
  - Render

Introduction

Volume objects are containers used to represent OpenVDB files in Blender. OpenVDB is a library and file format for the interoperability and storage of volumetric data. OpenVDB files may be generated by other software such as Houdini, or from Blender's fluid simulation cache.

Volume objects can be created from the Add menu in the 3D Viewport, or by dragging and dropping vdb-files into Blender. For animations, a frame sequence of OpenVDB files can be imported.



WDAS cloud data set rendered in wireframe, Workbench, and Cycles.

Rendering

Rendering volumes works the same as rendering smoke simulations. By default, the Principled Volume shader is used for rendering volume objects. It will use grids named `density`, `color` and `temperature` by default. If these are not available, another grid name must be chosen in the shader nodes.

Limitations

- OpenVDB excels at representing sparse volumes, that aren't necessarily concentrated within a tight bounding box but may be

spread out through space. However, in Blender, these are still rendered as dense volumes which is not ideal for performance and memory usage. This will be improved in future releases.
- OpenVDB files can also store level sets and points. While level set grids can be read, there is no current support for rendering them as surfaces. Importing OpenVDB points is not supported.

Properties

Grids

The List View shows the grids in the OpenVDB-file, listing their name and data type. A “grid” is a set of volumetric data, which typically stores the density of each voxel but can also contain temperatures, velocities and so on.

Click a grid to make the volume object display it.

OpenVDB File

File Path
: The VDB file to use.

Sequence
: Loads further VDB files, one for each frame in an animation. Much like with image sequences, all the files should have a numerical suffix in their name; so if you selected smoke-000.vdb in the *File Path*, there should be a smoke-001.vdb, a smoke-002.vdb and so on.

  Frames
  : How many frames of the sequence to use.

  Start
  : Scene frame at which the sequence should start.

  Offset
  : How many frames of the sequence to skip at the beginning.

  Mode
  : How the volume should behave before the sequence’s first frame (*Start*) and after its last (*Start* + *Frames*).

Clip::	Show nothing.
Extend::	Keep showing the first/last frame of the sequence.
Repeat::	Play the sequence again (and again, and again…).
Ping-Pong::	Play the sequence forwards, then backwards, then forwards again and so on.

Viewport Display

Wireframe

Method used to represent volumes in wireframe shading mode. For heavy volume data sets, it can be useful to set the object to always display as wireframe. This way, the 3D Viewport remains responsive but the volume still appears in the final render.

None::	The volume is not displayed in wireframe mode.
Bounds::	Displays the volume as a Bounding Box for the entire grid.
Boxes::	Displays bounding boxes for nodes in the volume tree.
Points::	Displays points for nodes in the volume tree.

Detail

The amount of detail to display for *Boxes* or *Points* wireframe mode.

Coarse::	Display one box or point for each intermediate tree node.
Fine::	Display a box or point for each leaf node containing 8×8 voxels.

Density

Thickness of the volume in the 3D Viewport. The density of the volume in the render is adjusted via Volume Shading.

Interpolation

Interpolation method to use for the visualization of the fluid grid.

Linear:: Linear interpolation between voxels. Gives good smoothness and speed.

Cubic:: Cubic interpolation between voxels. Gives smoothed high quality interpolation, but is slower.

Closest:: No interpolation between voxels. Gives raw voxels.

Slice

Renders only a single 2D section of the domain object.

Axis

Auto:: Adjust slice direction according to the view direction.

X/Y/Z:: Slice along the X, Y, or Z axis.

Position

Position of the slice relative to the length of the respective domain side.

Render

Space

Specifies how volume density and step size are computed.

Object:: Keeps volume *Density* and *Detail* the same regardless of object scale.

World:: Specify *Step Size* and *Density* in world space.

Step Size *Cycles Only*

Distance between volume samples. Lower values render more detail at the cost of performance. If set to zero, the step size is automatically determined based on voxel size.

Clipping *Cycles Only*

Value under which voxels are considered empty space to optimize rendering.

Precision *Cycles Only*

Specifies volume data precision. Lower values reduce memory consumption at the cost of detail.

Full::	Full float (Use 32 bit for all data).
Half::	Half float (Use 16 bit for all data).
Variable::	Automatically use less precision for less noticeable areas.

Velocity Grid *Cycles Only*

The name of the grid that contains voxel velocities, for calculating motion blur. This can be the name of a single grid containing 3D vectors, or a prefix of three separate grids containing scalar values. In the latter case, the X grid should have a name suffix of `x`, `.x` or `_x`, with similar conventions for the Y and Z grids.

Velocity Unit *Cycles Only*

Whether the velocity grid(s) specify distances per frame or per second.

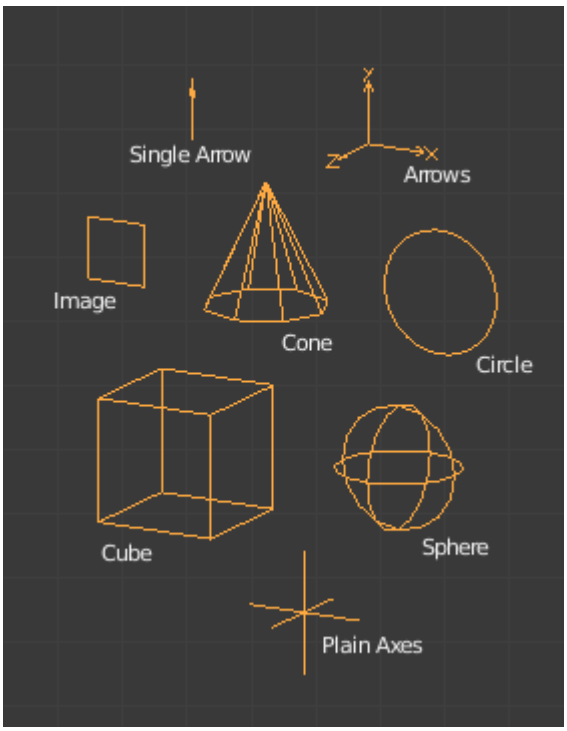
Velocity Scale *Cycles Only*

A custom multiplier to apply to the velocities in the VDB.

Empties

The “empty” is a single coordinate point with no additional geometry. Because an empty has no volume and surface, it cannot be rendered. Still it can be used as a handle for many purposes.

Primitives



Empty Display Types.

Plain Axes

Displays as six lines, initially with one pointing in each of the +X, -X, +Y, -Y, +Z, and -Z axis directions.

Arrows

Displays as arrows, initially pointing in the positive X, Y, and Z axis directions, each with a label.

Single Arrow

Displays as a single arrow, initially pointing in the +Z axis direction.

Circle

Displays as a circle initially in the XZ plane.

Cube

Displays as a cube, initially aligned to the XYZ axes.

Sphere

Displays as an implied sphere defined by three circles. Initially, the circles are aligned, one each, to the X, Y, and Z axes.

Cone

Displays as a cone, initially pointing in the +Y axis direction.

Image

Empties can display images. This can be used to create reference images, including blueprints or character sheets to model from. The image is displayed regardless of the 3D display mode.

Empty Displays settings can be accessed from *Properties* ▸ *Object Data* ▸ *Empty* panel.

Offset X, Y
: Offset the image origin (where 1.0 represents the width/height of the image).

X=0.5, Y=0.5:: Object origin at image center.
X=0.0, Y=0.0:: Object origin at image bottom, left.
X=1.0, Y=1.0:: Object origin at image top, right.

Depth

Default:: Use normal depth behavior.
Front:: Always display on top of other objects.
Back:: Always display behind of other objects.

Tip

When using the image as a reference for modeling, it can be useful to set the depth to *Front*, with a low *Opacity*.

Side

Both:: Display both the front and back of the empty.
Front:: Only display the front of the image.
Back:: Only display the back of the image.

Tip

This is useful if you are using an image as a reference where you have photos from both the front and back, so two empty images can be set only to show when viewed from the correct side.

Show in

Orthographic

Show in orthographic view.

Perspective

Show in perspective view.

Hint

It's often useful to disable this so reference images don't *get in the way* when viewing a model.

Only Axis Aligned

Only displays the image contents when the view is aligned with the object's local axis.

Opacity
: Blends the image with the background. The value slider adjusts the opacity of the image, changing how much of the image is blended with the background.

Editing

An empty can only be edited in Object Mode, which includes its transformation and parenting properties. For other tools see the Object section.

Apply Scale `Ctrl-A`
: While empties don't exactly have any object data attached to them which can be used for supporting "true" apply scale (i.e. with non-uniform scaling), they do have *Display Size* which controls how large the empties are displayed (before scaling). This works by taking the scale factor on the most-scaled axis, and combines this with the existing empty *Display Size* to maintain the correct dimensions on that axis.

Properties

Display As
: The Primitives empty type to display in the 3D Viewport.

Size
: Controls the size of the empties visualization. This does not change its scale, but functions as an offset.

Usage

Empties can serve as transform handles. Some examples of ways to use them include:

Parent object for a group of objects

An empty can be parented to any number of other objects. This gives the user the ability to control a group of objects easily, and without affecting a render.

Target for constraints

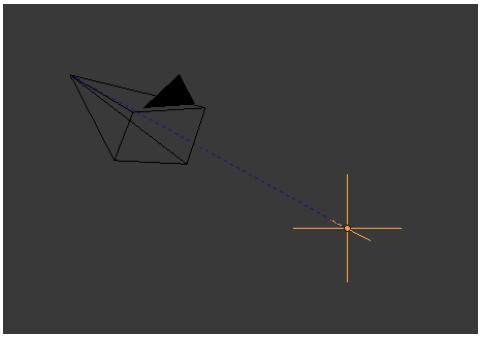
An empty can also be used as a target for normal, or bone constraints. This gives the user far more control; for instance, a rig can easily be set up to enable a camera to point towards an empty using the *Track to* constraint.

Array offset

An empty can be used to offset an Array Modifier, meaning complex deformations can be achieved by only moving a single object.



An example of an empty being used to control an array.



An example of an empty being used to control the Track To constraint.

Other common uses:

- Placeholders
- Rigging controls
- DOF distances

- Reference Images

Modifiers

- Introduction
  - Categories
  - Interface
  - Example
- Common Modifier Options
  - Vertex Group
  - Texture

Built-In Modifiers

- Edit
  - Data Transfer Modifier
  - Mesh Cache Modifier
  - Mesh Sequence Cache Modifier
  - UV Project Modifier
  - UV Warp Modifier
  - Vertex Weight Edit Modifier
  - Vertex Weight Mix Modifier
  - Vertex Weight Proximity Modifier
- Generate
  - Array Modifier
  - Bevel Modifier
  - Boolean Modifier
  - Build Modifier
  - Decimate Modifier
  - Edge Split Modifier
  - Geometry Nodes Modifier
  - Mask Modifier
  - Mesh to Volume Modifier
  - Mirror Modifier
  - Multiresolution Modifier
  - Remesh Modifier

    - Screw Modifier
    - Skin Modifier
    - Solidify Modifier
    - Subdivision Surface Modifier
    - Triangulate Modifier
    - Volume to Mesh Modifier
    - Weld Modifier
    - Wireframe Modifier
- Deform
    - Armature Modifier
    - Cast Modifier
    - Curve Modifier
    - Displace Modifier
    - Hook Modifier
    - Laplacian Deform Modifier
    - Lattice Modifier
    - Mesh Deform Modifier
    - Shrinkwrap Modifier
    - Simple Deform Modifier
    - Smooth Modifier
    - Smooth Corrective Modifier
    - Smooth Laplacian Modifier
    - Surface Deform Modifier
    - Volume Displace Modifier
    - Warp Modifier
    - Wave Modifier
- Normals
    - Normal Edit Modifier
    - Weighted Normal Modifier
    - Smooth By Angle Modifier
- Physics
    - Cloth Modifier
    - Collision Modifier
    - Dynamic Paint Modifier
    - Explode Modifier
    - Fluid Modifier
    - Ocean Modifier

- Particle Instance Modifier
- Particle System Modifier
- Soft Body Modifier

Introduction

Modifiers are automatic operations that affect an object's geometry in a non-destructive way. With modifiers, you can perform many effects automatically that would otherwise be too tedious to do manually (such as subdivision surfaces) and without affecting the base geometry of your object.

They work by changing how an object is displayed and rendered, but not the geometry which you can edit directly. You can add several modifiers to a single object to form The Modifier Stack and *Apply* a modifier if you wish to make its changes permanent.

They can be added to the active object using the Add Modifier operator, the "Add Modifier" button at the top of Modifiers tab in the Properties, or using `Shift-A` in the same tab. New modifiers are always added at the bottom of the stack (i.e. will be applied last).

There are many built-in modifiers but Blender also allows users to make their own modifiers through Geometry Nodes.

Categories

There are four categories of built-in modifiers:

Edit
: Similar to the *Deform* modifiers (see below), however, they usually do not directly affect the geometry of the object, but some other data, such as vertex groups.

Generate
: Constructive/destructive modifiers that will affect the whole Topology of the mesh. They can change the general appearance of the object, or add new geometry to it…

Deform
: Unlike *Generate* ones above, these modifiers only change the shape of an object, without altering its topology.

Simulate
: Represent physics simulations. In most cases, they are automatically added to the modifiers stack whenever a *Particle System* or *Physics* simulation is enabled. Their only role is to define the position in the modifier stack from which is taken the base data for the simulation they represent. As such, they typically have no properties, and are controlled by settings exposed in separate sections of the Properties.

You will also notice a category called “Hair”, this category comes from a bundled Asset Library that is distributed with Blender. See Hair Nodes for more information.

Users can make their own categories by making geometry node groups assets and assigning them to a Asset Catalog. This catalog name will be the category name. If a user creates a catalog with the same name as one of the built-in categories the node group will be added to the bottom of the corresponding menu.

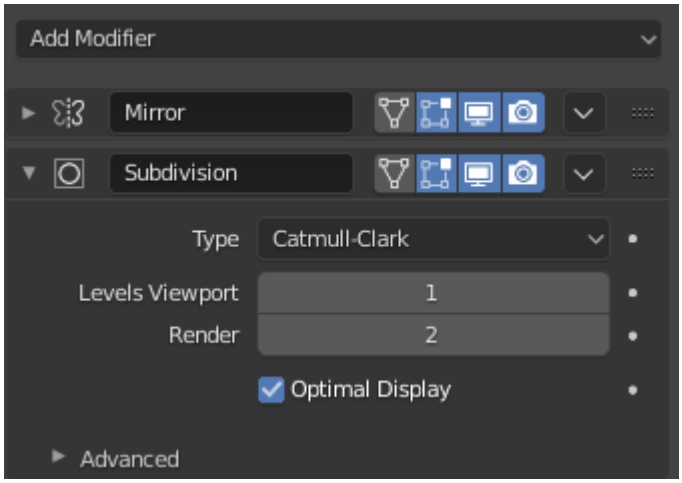
Node Groups that are non-assets or that do not belong to a category will be available in the “Unassigned” sub-menu.

Note

Geometry Node Groups must have the Modifier property enabled for the node group to show up in the Add Modifier menu.

Interface

Each modifier’s interface shares the same basic components, see Fig. Panel layout (Subdivision Surface as an example)..



Panel layout (Subdivision Surface as an example).

At the top is the panel header. The icons each represent different settings for the modifier (left to right):

Expand (down/right arrow icon)
: Collapse modifier to show only the header and not its options.

Type
: An icon as a quick visual reference of the modifier's type.

Name
: Every modifier has a unique name per object. Two modifiers on one object must have unique names, but two modifiers on different objects can have the same name. The default name is based on the modifier type.

Show on Cage (vertices triangle icon) – Meshes only
: Depends on the previous setting, if enabled, the modified geometry can also be edited directly, instead of the original one.

Warning

While it shows edited items in their final, modified positions, you are still actually editing original data. This can lead to strong and unpredictable effects with some tools, and should be disabled

whenever you need to perform complex or precise editing on the mesh.

Show in Edit Mode (vertices square icon)
: Display the modified geometry in Edit Mode, as well as the original geometry which you can edit.

Show in Viewport (screen icon)
: Toggle visibility of the modifier's effect in the 3D Viewport.

Render (camera icon)
: Toggle visibility of the modifier's effect in the render.

Note

The *Square*, *Triangle* and *Surface* icons may not be available, depending on the type of object and modifier.

Apply On Spline Points (point surface icon) – Curves, surfaces and texts only
: Apply the whole modifier stack up to and including that one on the curve or surface control points, instead of their tessellated geometry.

Note

By default, curves, texts and surfaces are always converted to mesh-like geometry before that the modifier stack is evaluated on them.

Extras
: Apply `Ctrl-A`
  : Makes the modifier "real": converts the object's geometry to match the applied modifier's results, and deletes the modifier.

    When applying a modifier to an object that shares Object Data between multiple objects, the object must first be made a Single

User which can be performed by confirming the pop-up message.

Warning

Applying a modifier that is not first in the stack will ignore the stack order (it will be applied as if it was the first one), and may produce undesired results.

Apply as Shape Key
: Stores the result of that modifier in a new relative shape key and then deletes the modifier from the modifier stack. This is only available with modifiers that do not affect the topology (typically, *Deform* modifiers only).

Note

Even though it should work with any geometry type that supports shape keys, currently it will only work with meshes.

Save as Shape Key
: Stores the result of that modifier in a new relative shape key and keeps the modifier in the modifier stack. This is only available with modifiers that do not affect the topology (typically, *Deform* modifiers only).

Duplicate `Shift-D`
: Creates a duplicate of the modifier just below current one in the stack.

Copy to Selected
: Copies the modifier from the Active object to all selected objects.

Move to First/Last
: Moves the modifier to the first or last position in the modifier stack.

Pin to Last

Keeps the modifier at the end of the modifier stack. When a modifier is pinned, a pin icon will be displayed on the right side of the panel's header.

Move to Nodes

Converts the existing Geometry Nodes Modifier node tree to a group node to be reused in other node trees. See Move to Nodes Operator for more information.

This operator is only available for the Geometry Nodes Modifier.

Delete `X`, `Delete`

Delete the modifier.

Move `::::`

Move the modifier up/down in the stack, changing the evaluation order of the modifiers.

A modifier is not movable if Pin to Last is enabled.

Below this header, all of the options unique to each modifier will be displayed.

Tip

Use `Alt` to affect all selected objects at once when performing operators such as add, apply, remove, and move to index.

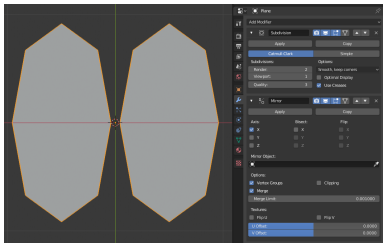
See Multi-Object Editing for more information.

The Modifier Stack

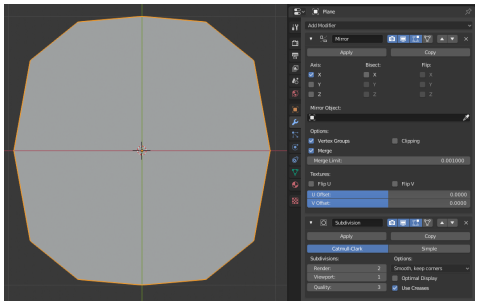
Modifiers are a series of non-destructive operations which can be applied on top of an object's geometry. You can be apply them in almost any order. This kind of functionality is often referred to as a "modifier stack" and is also found in several other 3D applications.

In a modifier stack, the order in which modifiers are applied has an effect on the result. Therefore the modifiers can be re-arranged by clicking the grab widget (`::::`) in the top right, and moving the selected modifier up or down. For example, the image below shows Subdivision Surface and Mirror modifiers that have switched places.

Modifier Stack example.



The Mirror modifier is the last item in the stack and the result looks like two surfaces.



The Subdivision Surface modifier is the last item in the stack and the result is a single merged surface.

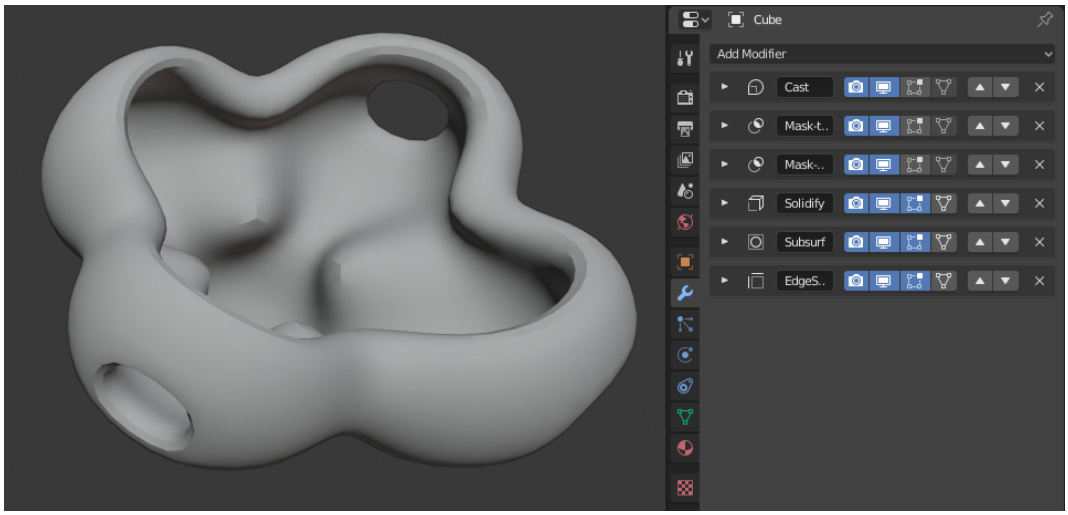
Modifiers are calculated from top to bottom in the stack. In this example, the desired result (on right) is achieved by first mirroring the object, and then calculating the subdivision surface.

Active Modifier

A modifier in the stack can be selected to mark in as Active, the active modifier displays an outline around the modifier's panel. To set an active modifier, select an area of the modifier's panel background, the modifier's icon, or, select a modifier in the Outliner.

The active modifier is used by the Geometry Node Editor to determine which node group is being modified.

Example



In this example a simple subdivided cube has been transformed into a rather complex object using a stack of modifiers.

Download example file.

Common Modifier Options

Some options are commonly used by many modifiers, and share the same behavior across all of those. In particular, many offer ways to precisely mask and weight their effect on a vertex basis (using either vertex groups and/or textures).

Vertex Group



Typical modifier Vertex Group options.

Vertex Groups are an easy way to control which vertices are affected by a modifier, and to which extent (using their weights). They are available when modifying meshes or lattices.

Tip

Vertex groups can also be edited and even animated using the Vertex Weight modifiers.

Vertex Group
: The vertex group name.

Warning

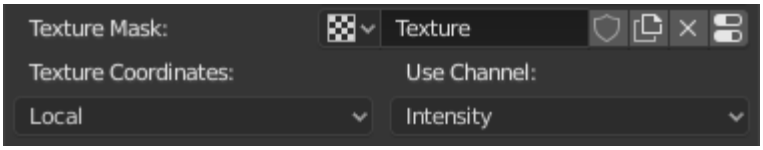
The group is referenced by its name. That means that if you rename it, the link to the renamed vertex group will be lost by all modifiers using

it (their field will turn red), and you'll have to select the proper group again in all of them.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group. Only available in some modifiers.

Texture



Typical modifier Texture options.

Those options allow to use any kind of image (including parametric ones) to control the modifier's effect. Most of the time, only the value (grayscale) of the texture is used, but in some cases (like with some modes of the Displace modifier), the whole RGB color components might be exploited.

Tip

Textures can be animated (either using videos, or by animating the mapping coordinates…).

Texture
: The texture data-block to use.

Tip

By clicking on the right-most button of this field (with the settings icon), you can go directly to the selected texture's settings in the *Texture Properties* tab.

Texture Coordinates

The texture's coordinates to get each vertex' value:

UV

Take texture coordinates from face UV coordinates.

UV Map

The UV Map from which to take texture coordinates. If the object has no UV coordinates, it falls back to the *Local* coordinate system. If this field is blank, but there is a UV map available (e.g. just after adding the first UV map to the mesh), the currently active UV map will be used.

Note

Since UV coordinates are specified per face, the UV texture coordinate system currently determines the UV coordinate for each vertex from the first face encountered which uses that vertex. Any other faces using that vertex are ignored.

This may lead to artifacts if the mesh has non-contiguous UV coordinates.

Object

Take the texture coordinates from another object's coordinate system.

Object

The object from which to take texture coordinates. Moving the object will therefore alter the coordinates of the texture mapping.

If this field is blank, it falls back to the *Local* coordinate system.

Note

Moving the original object will **also** result in a texture coordinate update. As such, if you need to maintain a displacement coordinate system while moving the modified object, consider parenting the coordinate object to the modified object.

Global
: Take the texture coordinates from the global coordinate system.

Local
: Take the texture coordinates from the object's local coordinate system.

Use Channel
: Which channel to use as value source (only available with a few modifiers currently, others follow the *Intensity* behavior, unless otherwise specified).

Intensity
: The average of the RGB channels (if RGB(1.0, 0.0, 0.0) value is 0.33).

Red/Green/Blue/Alpha
: One of the color channels' values.

Hue
: The hue from the HSV color model (i.e; the color in the standard wheel, e.g. blue has a higher hue value than yellow).

Saturation
: The saturation from the HSV color model (e.g. the value for pure red is 1.0, for gray is 0.0).

Value
: The value from the HSV color model.

Note

All of the channels above are gamma corrected, except for *Intensity*.

Edit

- Data Transfer Modifier
- Mesh Cache Modifier
- Mesh Sequence Cache Modifier
- UV Project Modifier
- UV Warp Modifier
- Vertex Weight Edit Modifier
- Vertex Weight Mix Modifier
- Vertex Weight Proximity Modifier

Data Transfer Modifier

The Data Transfer modifier transfers several types of data from one mesh to another. Data types include vertex groups, UV maps, Color Attributes, custom normals…

Transfer works by generating a mapping between source mesh's elements (vertices, edges, etc.) and destination ones, either on a one-to-one basis, or mapping several source elements to a single destination one, using interpolation.



Transferring normals between objects, see example blend-file.

See also

Transfer Mesh Data Operator

Options

Source
: Mesh object to copy data from.

  If the button to the right of the field is unset, both the source and the destination geometry is considered in global space when generating the mapping, otherwise they are evaluated in local space (i.e. as if both object's origins were at the same place).

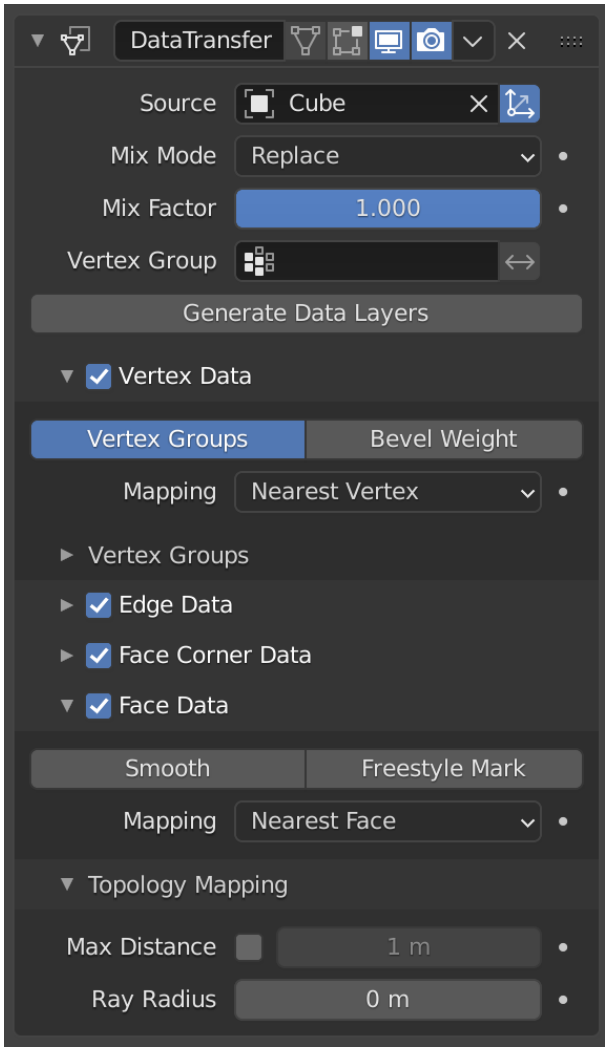
Mix Mode
: Controls how destination data are affected:

  All
  : Replaces everything in destination (note that *Mix Factor* is still used).

  Above Threshold
  : Only replaces destination value if it's above given threshold *Mix Factor*. How that threshold is interpreted depends on the data type, note that for Boolean values this option fakes a logical AND.

  Below Threshold
  : Only replaces destination value if it's below given threshold *Mix Factor*. How that threshold is interpreted depends on the data type, note that for Boolean values this option fakes a logical OR.

  Mix, Add, Subtract, Multiply
  : Apply that operation, using mix factor to control how much of source or destination value to use. Only available for a few types



Data Transfer Modifier.

(vertex groups, Color Attributes).

Mix Factor
: How much of the transferred data gets mixed into existing one (not supported by all data types).

Vertex Group
: Allows per-element fine control of the mix factor. Vertex group influence can be reverted using the small “arrow” button to the right.

Generate Data Layers
: This modifier cannot generate needed data layers itself. Once the set of source data to transfer is selected, this button shall be used to generate matching destination layers, if needed.

Selection of Data to Transfer

To keep the size of the modifier reasonable, the kind of elements to be affected must be selected first (vertices, edges, face corners and/or faces).

Mapping Type
: How is generated the mapping between those source and destination elements. Each type has its own options, see Geometry Mapping below for details.

Data Types
: The left column of toggle buttons, to select which data types to transfer.

Multi-layers Data Types Options
: In those cases (vertex groups, Color Attributes, UVs), one can select which source layers to transfer (usually, either all of them, or a single specified one), and how to affect destination (either by matching names, matching order/position, or, if a single source is selected, by specifying manually the destination layer).

Islands Handling Refinement
: This setting only affects UV transfer currently. It allows to avoid a given destination face to get UV coordinates from different source UV islands.

Keeping it at 0.0 means no island handling at all. Typically, small values like 0.02 are enough to get good results, but if you are mapping from a very high-poly source towards a very low-poly destination, you may have to raise it quite significantly.

Usage

First key thing to keep in mind when using this modifier is that it will **not** create destination data layers. *Generate Data Layers* button shall always be used for this purpose, once the set of source data to transfer has been selected. It should also be well understood that creating those data layers on destination mesh is **not** part of the modifier stack, which means e.g. that they will remain even once the modifier is deleted, or if the source data selection is modified.

Geometry Mapping

Geometry mapping is how a given destination mesh relates to a source mesh. In this process a destination vertex/edge/… gets a part of the source mesh assigned with functions as its data source. It is crucial to understand this topic well to get good results with this modifier.

Topology
: The simplest option, expects both meshes to have identical number of elements, and match them by order (indices). Useful e.g. between meshes that were identical copies, and got deformed differently.

One-To-One Mappings
: Those always select only one source element for each destination one, often based on shortest distance.

  Vertices
  : Nearest Vertex
    : Uses source's nearest vertex.

    Nearest Edge Vertex
    : Uses source's nearest vertex of source's nearest edge.

Nearest Face Vertex
: Uses source's nearest vertex of source's nearest face.

Edges

Nearest Vertices
: Uses source's edge which vertices are nearest from destination edge's vertices.

Nearest Edge
: Uses source's nearest edge (using edge's midpoints).

Nearest Face Edge
: Uses source's nearest edge of source's nearest face (using edge's midpoints).

Face Corners

A face corner is not a real element by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, …) aspects are used to match them together.

Nearest Corner and Best Matching Normal
: Uses source's corner having the most similar *split* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner and Best Matching Face Normal
: Uses source's corner having the most similar *face* normal with destination one, from those sharing the nearest source's vertex.

Nearest Corner of Nearest Face
: Uses source's nearest corner of source's nearest face.

Faces

Nearest Face
: Uses source's nearest face.

Best Normal-Matching
: Uses source's face which normal is most similar with destination one.

Interpolated Mappings

Those use several source elements for each destination one, interpolating their data during the transfer.

Vertices

Nearest Edge Interpolated

Uses nearest point on nearest source's edge, interpolates data from both source edge's vertices.

Nearest Face Interpolated

Uses nearest point on nearest source's face, interpolates data from all that source face's vertices.

Projected Face Interpolated

Uses point of face on source hit by projection of destination vertex along its own normal, interpolates data from all that source face's vertices.

Edges

Projected Edge Interpolated

This is a sampling process. Several rays are cast from along the destination's edge (interpolating both edge's vertex normals), and if enough of them hit a source's edge, all hit source edges' data are interpolated into destination one.

Face Corners

A face corner is not a real element by itself, it's some kind of split vertex attached to a specific face. Hence both vertex (location) and face (normal, …) aspects are used to match them together.

Nearest Face Interpolated

Uses nearest point of nearest source's face, interpolates data from all that source face's corners.

Projected Face Interpolated

Uses point of face on source hit by projection of destination corner along its own normal, interpolates data from all that

source face's corners.

Faces

Projected Face Interpolated

This is a sampling process. Several rays are cast from the whole destination's face (along its own normal), and if enough of them hit a source's face, all hit source faces' data are interpolated into destination one.

Topology Mapping

Max Distance

When the "pressure stylus" icon button to the right is enabled, this is the maximum distance between source and destination to get a successful mapping. If a destination element cannot find a source one within that range, then it will get no transferred data.

This allows to transfer a small sub-detailed mesh onto a more complete one (e.g. from a "hand" mesh towards a "full body" one).

Ray Radius

The starting ray radius to use when Ray Casting against vertices or edges. When transferring data between meshes Blender performs a series of ray casts to generate mappings. Blender starts with a ray with the radius defined here, if that does not return a hit then the radius is progressively increased until a positive hit or a limit is reached.

This property acts as an accuracy/performance control; using a lower ray radius will be more accurate however, might take longer if Blender has to progressively increase the limit. Lower values will work better for dense meshes with lots of detail while larger values are probably better suited for simple meshes.

Mesh Cache Modifier

The Mesh Cache modifier main use is for animated mesh data to be applied to a mesh and played back, deforming the mesh.

This works in a similar way to shape keys, but is aimed at playing back external files and is often used for interchange between applications.

Tip

Both MDD and PC2 depend on the vertex order on the mesh remaining unchanged. This is a limitation of this method, so take care not to add, remove or reorder vertices once this modifier is used.

Options

Format
: The input file format (currently `.mdd` and `.pc2` are supported).

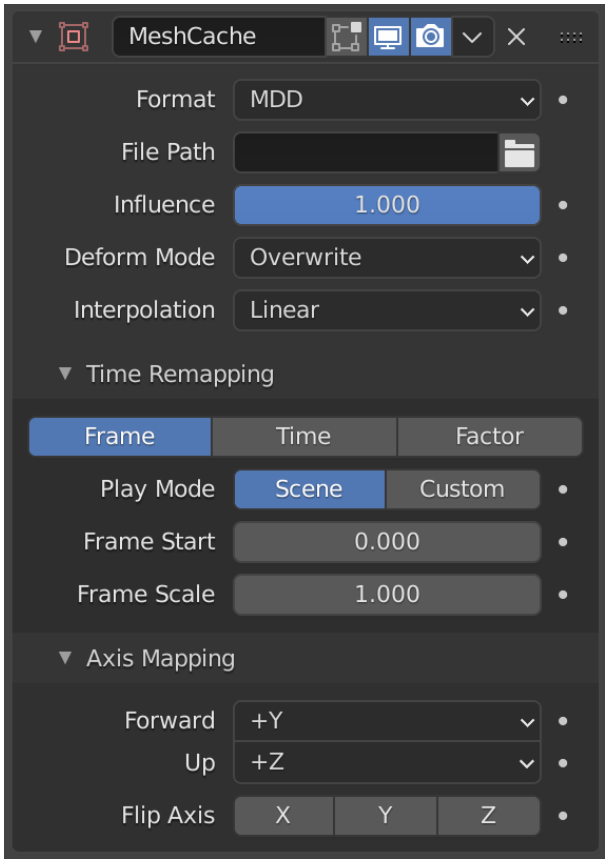
File Path
: Path to the cache file.

Influence
: Factor to adjust the influence of the modifier's deformation, useful for blending in/out from the cache data.

Deform Mode
: This setting defaults to *Overwrite* which will replace the vertex locations with those in the cache file. However, you may want to use shape keys, for example, and mix them with the mesh cache. In this case you can select the *Deform* option which integrates deformations with the mesh cache result.

Note

This feature is limited to making smaller, isolated edits and will not work for larger changes such as re-posing limbs.



Mesh Cache Modifier.

Interpolation
: *None*, or *Linear*, which will blend between frames. Use linear when the frames in the cache file do not match up exactly with the frames in the blend-file.

Vertex Group
: If set, restrict the effect to the only vertices in that vertex group.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group.

Time Remapping

Time Mode
: Select how time is calculated.

Frame:: Allows you to control the frames, which will ignore timing data in the file but is often useful since it gives simple control.

Time::

Evaluates time in seconds, taking into account timing information from the file (offset and frame-times).

Factor:: Evaluates the entire animation as a value in the [0, 1] range.

Play Mode
: Select how playback operates.

Scene:: Use the current frame from the scene to control playback.

Frame Start
: Play the cache starting from this frame.

Frame Scale
: Scale time by this factor (applied after the start value).

Custom:: Control animation timing manually.

Evaluation Value
: Property used for animation time, this gives more control of timing (typically this value will be animated).

Axis Mapping

Forward/Up Axis
: The axis for forward and up used in the source file.

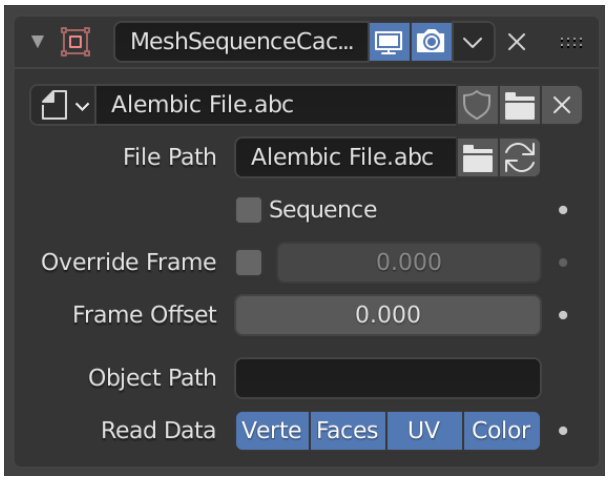
Flip Axis
: In rare cases you may also need to flip the coordinates on an axis.

Mesh Sequence Cache Modifier

The Mesh Sequence Cache modifier loads data from Alembic and USD files. It supports static meshes, but is mostly used to load animated meshes. Despite its name, this modifier also supports curves. It also handles file sequences, as well as meshes and curves with varying topology (like the result of fluid simulations).

When importing an Alembic or USD file, Mesh Sequence Cache modifiers are automatically added to time-varying meshes. For time-varying object transforms (so animation of rotation, location, or scale), the Transform Cache Constraint is used. Files other than Alembic or USD, like MDD and PC2 files, can be loaded using the Mesh Cache modifier.

Options



Cache File
: Data-block menu to select the Alembic or USD file.

File Path
: Path to Alembic or USD file.

Object Path
: The path to the Alembic or USD object inside the archive or stage.

Read Data
: Type of data to read for a mesh object, respectively: vertices, polygons, UV maps, and Color Attributes.

  Vertices, Faces, UV, Color

Time

Sequence
: Whether or not the cache is separated in a series of files.

Override Frame
: Whether to use a custom frame for looking up data in the cache file, instead of using the current scene frame.

  The *Frame* value is the time to use for looking up the data in the cache file, or to determine which to use in a file sequence.

Frame Offset
: Subtracted from the current frame to use for looking up the data in the cache file, or to determine which file to use in a file sequence.

Velocity

Velocity Attribute
: The name of the Alembic attribute used for generating motion blur data; by default, this is `.velocities` which is standard for most Alembic files.

  Note

  The *Velocity Attribute* option is currently for Alembic files only.

Velocity Unit
: Defines how the velocity vectors are interpreted with regard to time.

  Frame:: The velocity unit was encoded in frames and does not need to be scale by scene FPS.

  Second:: The velocity unit was encoded in seconds and needs to be scaled by the scene FPS (1 / FPS).

  Note

The *Velocity Unit* option is currently for Alembic files only.

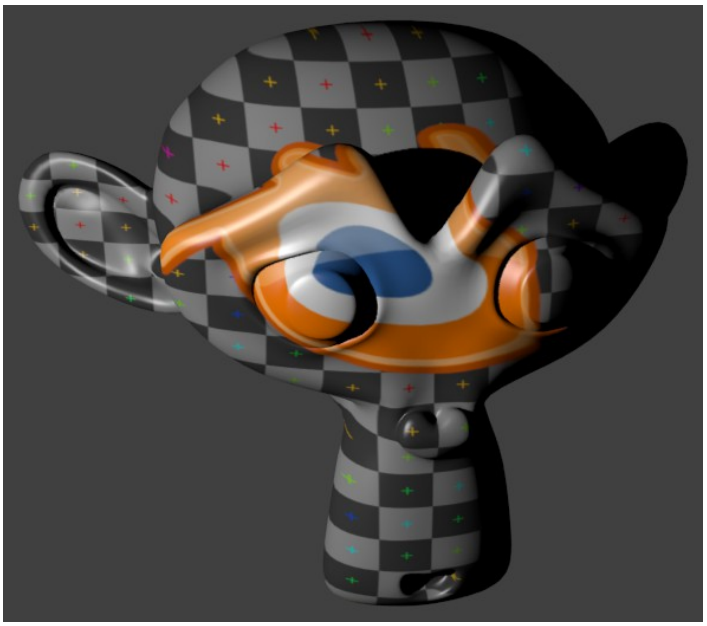
Velocity Scale

Multiplier used to control the magnitude of the velocity vector for time effects such as motion blur.

Note

The *Velocity Scale* option is currently for Alembic files only.

UV Project Modifier



Projecting the Blender logo onto Suzanne.

The *UV Project* modifier acts like a slide projector. It emits a UV map from the negative Z axis of a controller object (such as an empty object), and applies it to the object as the "light" hits it.

Download an example.

Options

UV Map
: Which UV map to modify. Defaults to the active rendering layer.

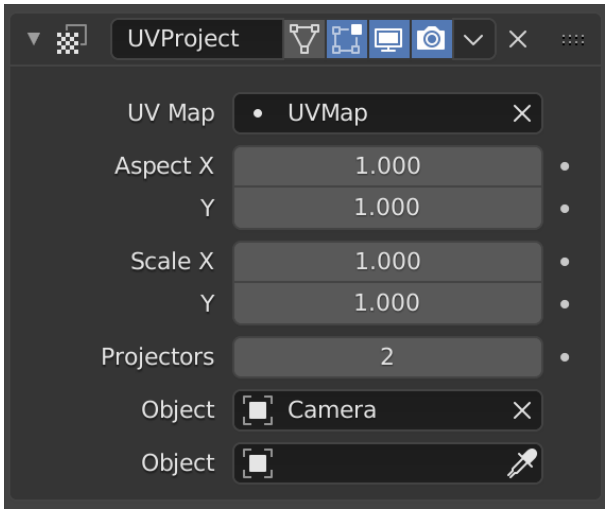
Aspect X/Y
: Changes the image's aspect ratio. Only apply when a camera is used as projector object.

Scale X/Y

Scales the image. Only apply when a camera is used as projector object.

Projectors
: Up to ten projector objects are supported. Each face will choose the closest and aligned projector with its surface normal. Projections emit from the negative Z axis (i.e. straight down a camera or light). If the projector is a camera, the projection will adhere to its perspective/orthographic setting.

Object
: Specify the projector object(s).



The UV Project modifier.

Usage

General

UV Project is great for making spotlights more diverse, and also for creating decals to break up repetition.

Usually, an Image Texture node mapped to the UV map that the modifier targets is added to the object's material.

Known Limitations

Vertices Behind the Camera

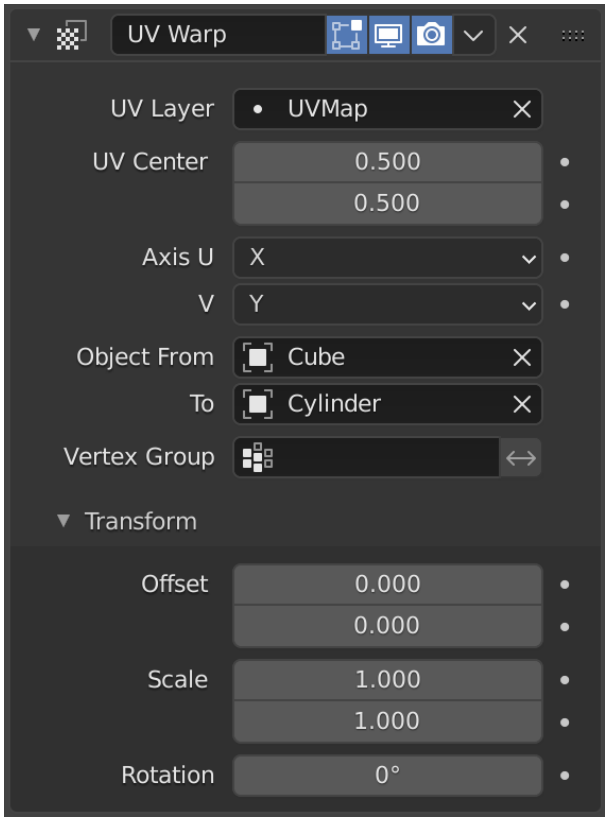
When projecting geometry in a perspective view, vertices behind the camera are not properly mapped. You can workaround this by subdividing

geometry so that faces in front of the camera have correctly mapped UVs.

UV Warp Modifier

The *UV Warp* modifier transforms an object's UV map based on values or two objects. Its purpose is to give you direct control over the object's UVs in the 3D Viewport, allowing you to directly move, rotate, and scale existing UV coordinates using defined values or a controller object or bone.

Options



UV Layer
: Which UV map to modify; if not set it defaults to the active rendering layer.

UV Center
: The center point of the UV map to use when applying scale or rotation. With (0, 0) at the bottom left and (1, 1) at the top right.

Axis U/V
: The axes to use when mapping the 3D coordinates into 2D.

Object From, To
: The two objects used to define the transformation. See Usage below.

Vertex Group
: The vertex group can be used to scale the influence of the transformation per vertex.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group.

Transform

Offset
: Amount to move the UV map.

Scale
: Amount to scale the UV map.

Rotate
: Amount to rotate the UV map.

Usage

How the UVs are warped is determined by the difference between the transforms (location, rotation and scale) of the *from* and *to* objects.

If the *to* object has the same transforms as the *from* object, the UVs will not be changed.

Assuming the *UV Axis* of the modifier is X/Y and the scale of the objects is (1, 1, 1), if the *to* object is one unit away from the *from* object on the X axis, the UVs will be transformed on the U axis (horizontally) by one full UV space (the entire width of the image).

Vertex Weight Edit Modifier

This modifier is intended to edit the weights of a vertex group.

The general process is the following, for each vertex:

- (Optional) It does the mapping, either through one of the predefined functions, or a custom mapping curve.
- It applies the influence factor, and optionally the vertex group or texture mask (0.0 means original weight, 1.0 means fully mapped weight).
- It applies back the weight to the vertex, and/or it might optionally remove the vertex from the group if its weight is below a given threshold, or add it if it is above a given threshold.

Important

This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

Note

You can view the modified weights in Weight Paint Mode. This also implies that you will have to disable the *Vertex Weight Edit* modifier if you want to see the original weights of the vertex group you are editing.

Options

Vertex Group
: The vertex group to affect.

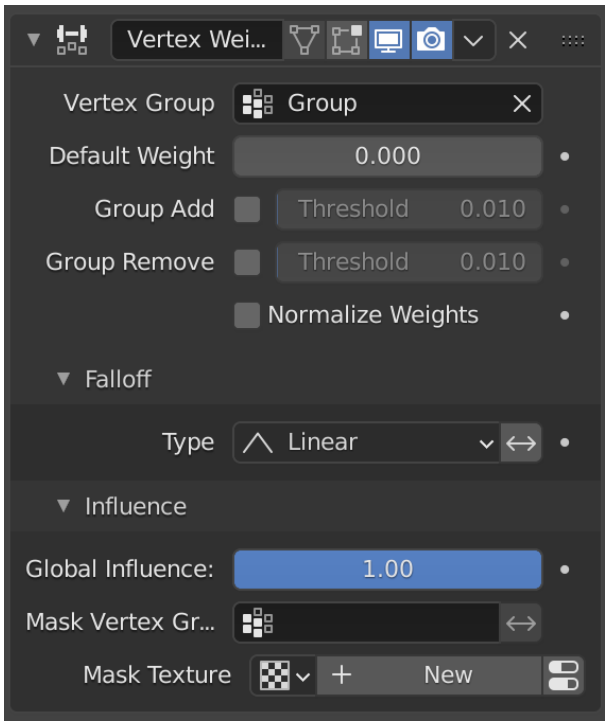
Default Weight

The default weight to assign to all vertices not in the given vertex group.

Group Add
: Adds vertices with a final weight over *Add Threshold* to the vertex group.

Group Remove
: Removes vertices with a final weight below *Remove Threshold* from the vertex group.

Normalize Weights
: Scale the weights in the vertex group to keep the relative weight but the lowest and highest values follow the full 0 - 1 range.



The Vertex Weight Edit modifier panel.

Falloff

Falloff Type
: Type of mapping.

  Linear
  : No mapping.

  Custom Curve
  : Allows you to manually define the mapping using a curve.

  Sharp, Smooth, Root and Sphere
  : These are classical mapping functions, from spikiest to roundest.

  Random
  : Uses a random value for each vertex.

Median Step
: Creates binary weights (0.0 or 1.0), with 0.5 as cutting value.

Invert <-->
: Inverts the falloff.

Influence

Those settings are the same for the three *Vertex Weight* modifiers.

Global Influence
: The overall influence of the modifier (0.0 will leave the vertex group's weights untouched, 1.0 is standard influence).

> **Important**
>
> Influence only affects weights, adding/removing of vertices to/from vertex group is not prevented by setting this value to 0.0.

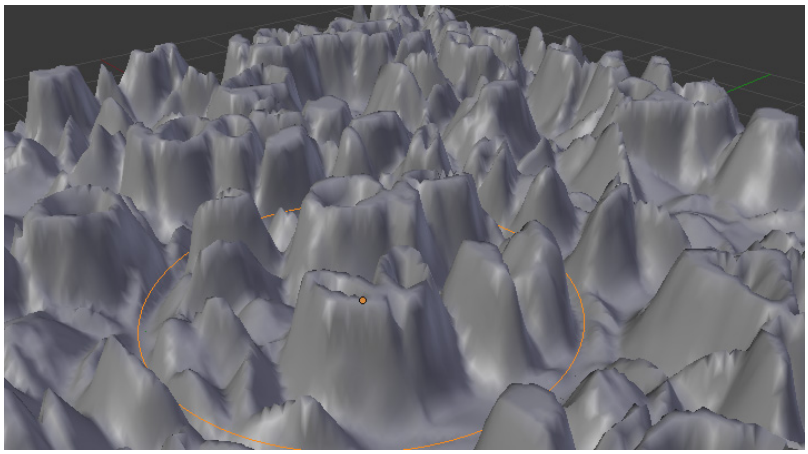
In addition, a per-vertex fine control of the effect is possible using either a vertex group or a texture (both are mutually exclusive). The per-vertex values from those will be multiplied with the *Global Influence*.

See common masking options for a complete reference.

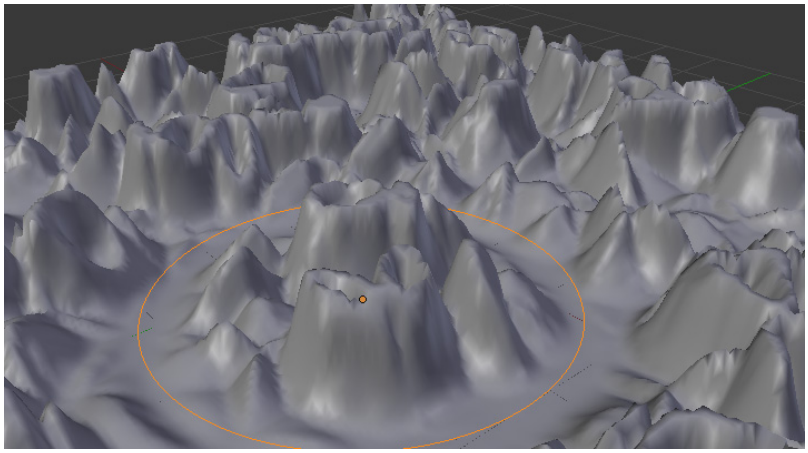
Example

Here is an example of various effects achieved using *Vertex Weight Edit* modifier (together with the Vertex Weight Proximity modifier) to generate weights used by the Displace modifier.

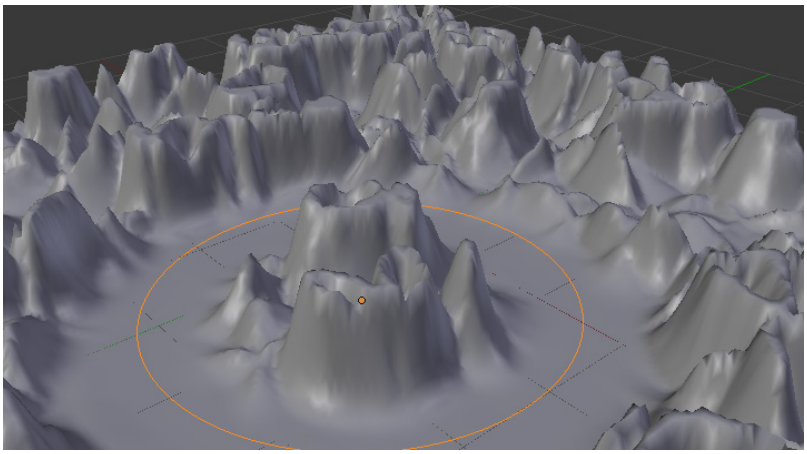
Curve Map variations.



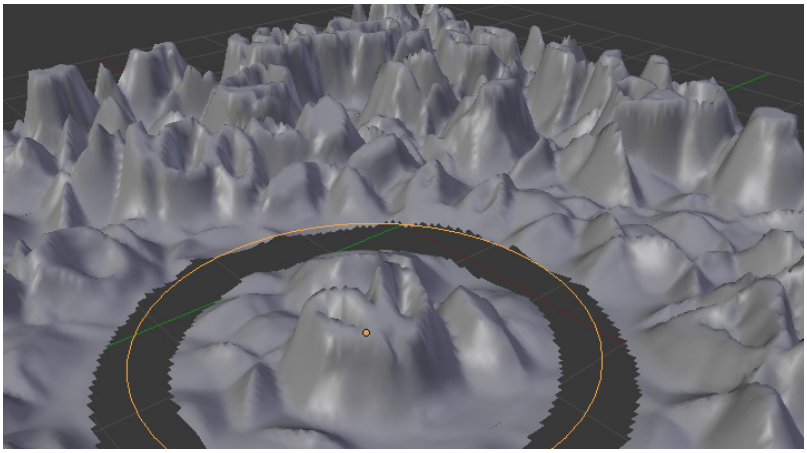
Concave-type mapping curve.



No mapping curve (linear).



Convex-type mapping curve.



Vertices with a computed weight below 0.1 removed from the vertex group.

Vertex Weight Mix Modifier

This modifier mixes a second vertex group (or a simple value) into the affected vertex group, using different operations.

Important

This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

Note

You can view the modified weights in Weight Paint Mode. This also implies that you will have to disable the *Vertex Weight Mix* modifier if you want to see the original weights of the vertex group you are editing.

Options

Vertex Group A, B

- **A**: The vertex group to affect.
- **B**: The second vertex group to mix into the affected one. Leave it empty if you only want to mix in a simple value.

Invert Weights A/B

Invert the influence of the vertex group.

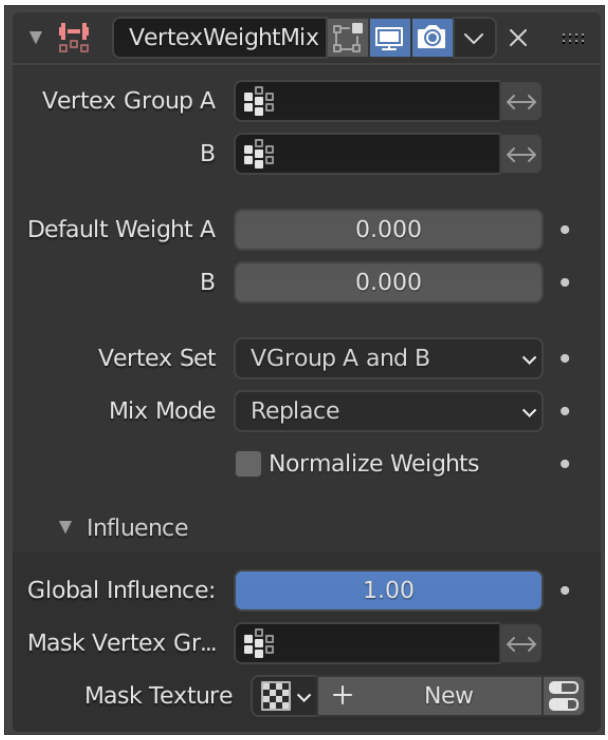
Default Weight A, B

- **A**: The default weight to assign to all vertices not in the given vertex group.
- **B**: The default weight to assign to all vertices not in the given second vertex group.

Vertex Set

Choose which vertices will be affected.

All::



The Vertex Weight Mix modifier panel.

Affects all vertices, disregarding the vertex groups content.

Vertex Group A:: Affects only vertices belonging to the affected vertex group.

Vertex Group B:: Affects only vertices belonging to the second vertex group.

Vertex Group A or B::

Affects only vertices belonging to at least one of the vertex groups.

Vertex Group A and B::

Affects only vertices belonging to both vertex groups.

Important

When using *All vertices*, *Vertices from group B* or *Vertices from one group*, vertices might be added to the affected vertex group.

Mix Mode

How the vertex group weights are affected by the other vertex group's weights.

Replace::	Replaces affected weights with the second group's weights.
Add::	Adds the values of *Group B* to *Group A*.
Subtract::	Subtracts the values of *Group B* from *Group A*.
Multiply::	Multiplies the values of *Group B* with *Group A*.
Divide::	Divides the values of *Group A* by *Group B*.
Difference::	Subtracts the smaller of the two values from the larger.
Average::	Adds the values together, then divides by 2.
Minimum::	Uses the smallest weight value of VGroup A's or VGroup B's weights.
Maximum::	Uses the largest weight value of VGroup A's or VGroup B's weights.

Normalize Weights

Scale the weights in the vertex group to keep the relative weight but the lowest and highest values follow the full 0 - 1 range.

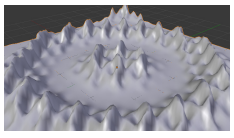
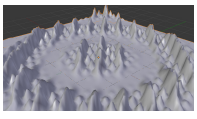
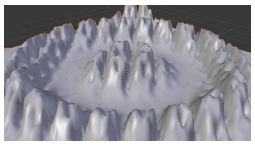
Influence

Those settings are the same for the three *Vertex Weight* modifiers, see the Vertex Weight Edit modifier page.

Example

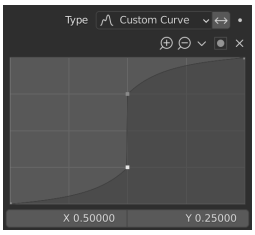
Here is and example of using a texture and the mapping curve to generate weights used by the Wave modifier.

Texture channel variations.

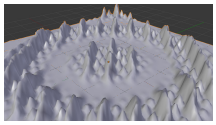




Using intensity.	Using Red.	Using Saturation.

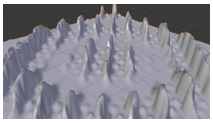
Custom mapping curve with a Vertex Weight Edit modifier.



A customized mapping curve.



Custom Mapping disabled.



Custom Mapping enabled.

The blend-file, TEST_4 scene.

Vertex Weight Proximity Modifier

This modifier sets the weights of the given vertex group, based on the distance between the object (or its vertices), and another target object (or its geometry).

Warning

This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

Note

You can view the modified weights in Weight Paint Mode. This also implies that you will have to disable the *Vertex Weight Proximity* modifier if you want to see the original weights of the vertex group you are editing.

Options

Vertex Group
: The vertex group to affect.

Target Object
: The object from which to compute distances.

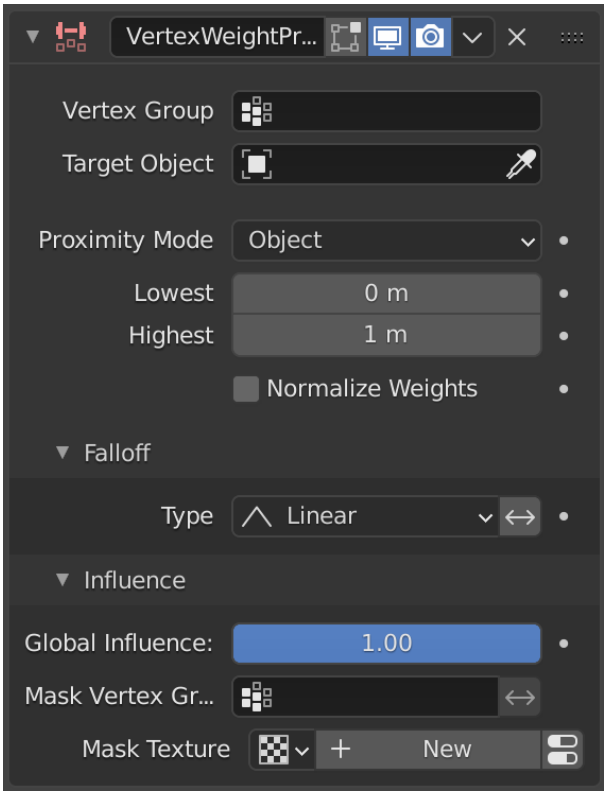
Proximity Mode
: Object Distance
  : Use the distance between the modified mesh object and the target object as weight for all vertices in the affected vertex group.

  Geometry Distance

Use the distance between each vertex and the target object, or its geometry.

Vertex
: This will set each vertex's weight from its distance to the nearest vertex of the target object.

Edge
: This will set each vertex's weight from its distance to the nearest edge of the target object.



The Vertex Weight Proximity modifier panel.

Face
: This will set each vertex's weight from its distance to the nearest face of the target object.

Note

If you enable more than one of them, the shortest distance will be used. If the target object has no geometry (e.g. an empty or camera), it will use the location of the object itself.

Lowest
: Distance mapping to 0.0 weight.

Highest
: Distance mapping to 1.0 weight.

Tip

Lowest can be set above *Highest* to reverse the mapping.

Normalize Weights
: Scale the weights in the vertex group to keep the relative weight but the lowest and highest values follow the full 0 - 1 range.

Falloff

Type
: Type of mapping.

  Linear
  : No mapping.

  Custom Curve
  : Allows you to manually define the mapping using a curve.

  Sharp, Smooth, Root and Sphere
  : These are classical mapping functions, from spikiest to roundest.

  Random
  : Uses a random value for each vertex.

  Median Step
  : Creates binary weights (0.0 or 1.0), with 0.5 as cutting value.

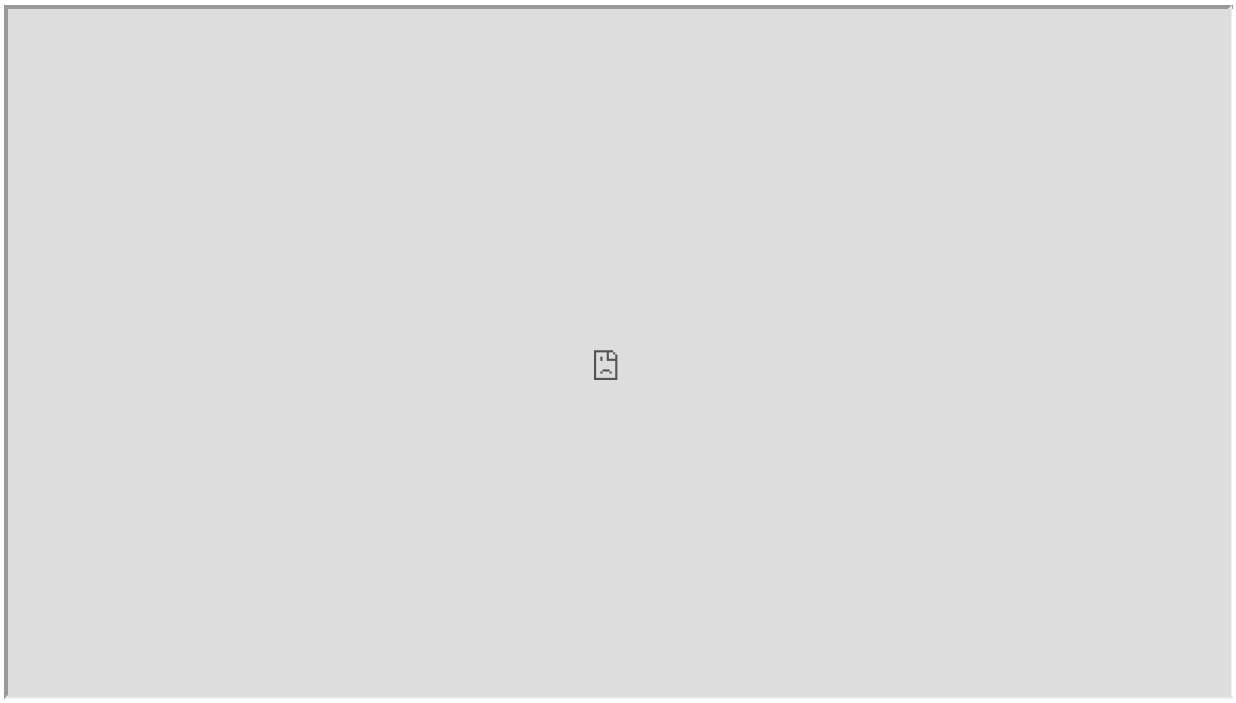
Invert <-->
: Inverts the falloff.

Influence

Those settings are the same for the three *Vertex Weight* modifiers, see the Vertex Weight Edit modifier page.

Example

This example shows the usage of distance from a target object to dynamically control a Wave modifier with a modified vertex group:



The blend-file, TEST_1 scene.

Generate

- Array Modifier
- Bevel Modifier
- Boolean Modifier
- Build Modifier
- Decimate Modifier
- Edge Split Modifier
- Geometry Nodes Modifier
- Mask Modifier
- Mesh to Volume Modifier
- Mirror Modifier
- Multiresolution Modifier
- Remesh Modifier
- Screw Modifier
- Skin Modifier
- Solidify Modifier
- Subdivision Surface Modifier
- Triangulate Modifier
- Volume to Mesh Modifier
- Weld Modifier
- Wireframe Modifier

Array Modifier

The *Array* modifier creates an array of copies of the base object, with each copy being offset from the previous one in any of a number of possible ways. Vertices in adjacent copies can be merged if they are nearby, allowing smooth Subdivision Surface frameworks to be generated.

This modifier can be useful when combined with tileable meshes for quickly developing large scenes. It is also useful for creating complex repetitive shapes.

Multiple Array modifiers may be active for an object at the same time (e.g. to create complex three-dimensional constructs).

Options

Fit Type
: Controls how the length of the array is determined. There are three choices, activating respectively the display of the *Curve*, *Length* or *Count* settings explained below:

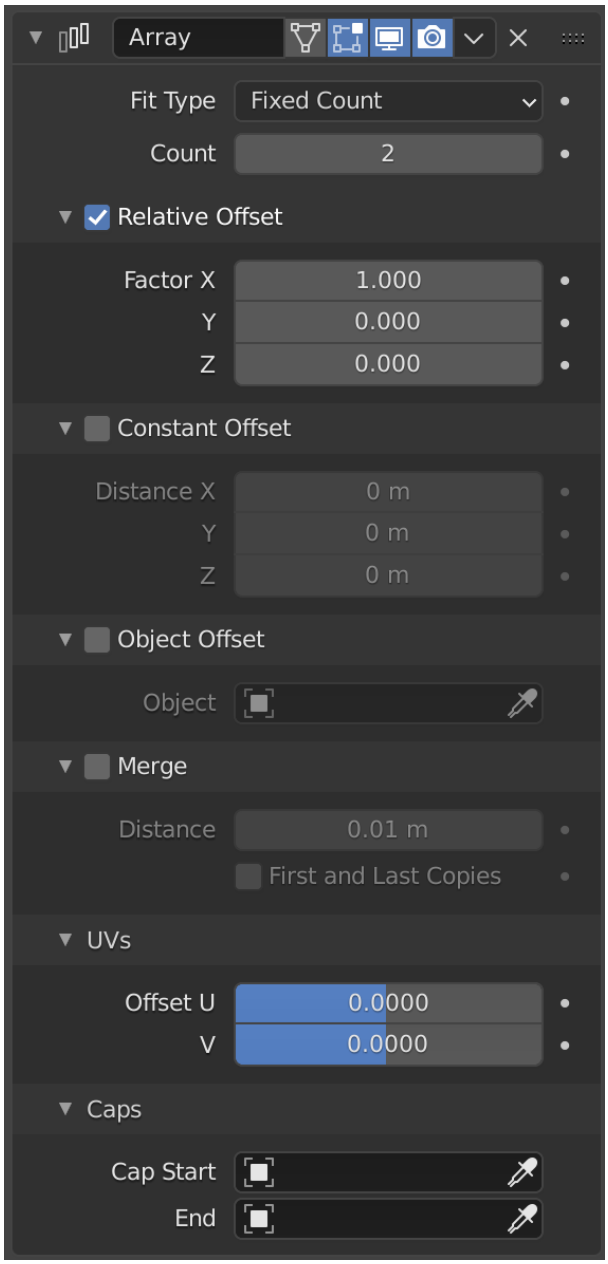
  Fit Curve
  : Generates enough copies to fit within the length of the curve object specified in *Curve*.

  Fit Length
  : Generates enough copies to fit within the fixed length given by *Length*.

  Fixed Count
  : Generates the number of copies specified in *Count*.

Note

- Both *Fit Curve* and *Fit Length* use the local coordinate system size of the base object, which means that scaling the base object in Object Mode will not change the number of copies generated by the modifier.
- *Fit Curve* uses the local coordinate system length of the curve, which means that scaling the curve in Object Mode will not change the number of copies generated by the modifier.
- Applying the scale can be useful for both.



The Array modifier.

Relative Offset

Factor X/Y/Z

Adds a translation equal to the object's bounding box size along each axis, multiplied by a scaling factor, to the offset. X, Y and Z scaling factors can be specified.



Relative offset (0.5, 1.0 and 1.5) examples.

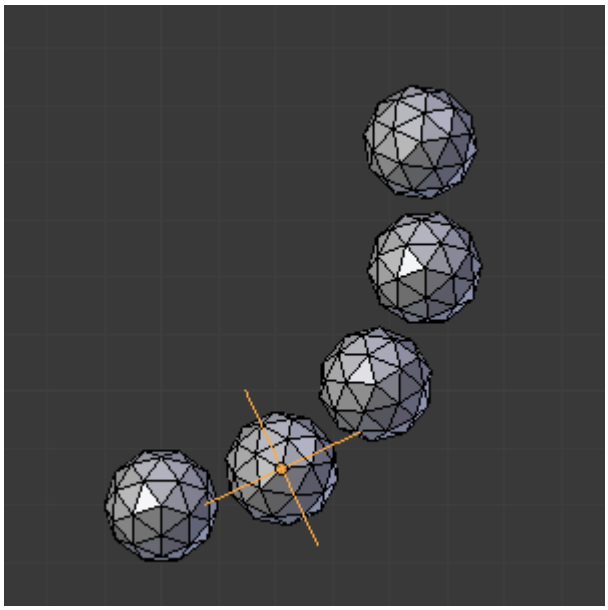
Constant Offset

Distance X/Y/Z

Adds a constant translation component to the duplicate object's offset. X, Y and Z constant components can be specified.

Object Offset

Adds a transformation taken from an object (relative to the current object) to the offset. It is good practice to use an empty object centered or near to the initial object. E.g. by rotating this empty a circle or helix of objects can be created.



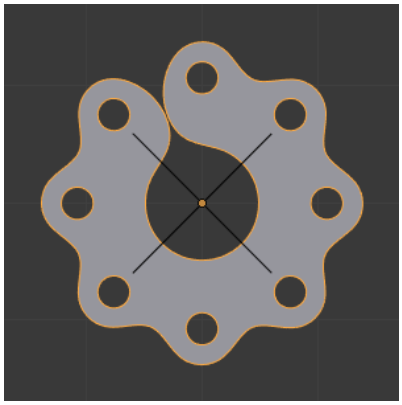
Object offset example.

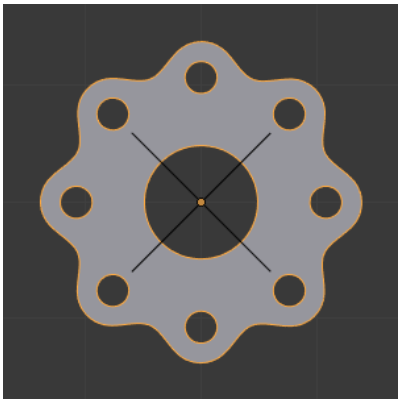
Merge

If enabled, vertices in each copy will be merged with vertices in the next copy that are within the given *Distance*.

First and Last Copies
: If enabled **and** *Merge* is enabled, vertices in the first copy will be merged with vertices in the last copy, again if they are within *Distance* range. This is useful for circular objects.

First and Last Copies merge example.





Subdivision discontinuity caused by not merging vertices between first and last copies (*First and Last Copies* off).	Subdivision discontinuity eliminated by merging vertices between first and last copies (*First and Last Copies* on).

Distance
: Controls the merge distance for *Merge* and *First and Last Copies*.

UVs

Offset U/V
: Shifts UVs of each new duplicate by a settable amount.

Caps

Cap Start, End
: This allows either endpoints of the array to have a different mesh subsisted.

  For the *start*: as if it was in position -1, i.e. one “array step” before the first “regular” array copy. For the *end*: as if it was in position $n + 1$, i.e. one “array step” after the last “regular” array copy.

  When *Merge* is activated, the *cap* vertices within the *Distance* threshold will be merged.

  Note

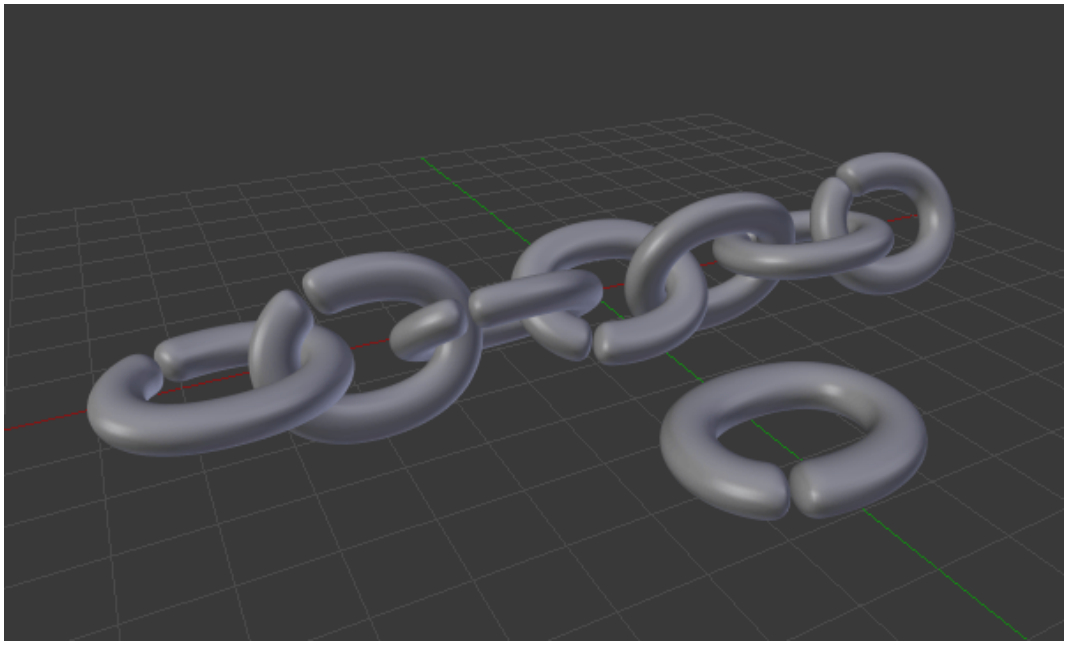
  The start/end cap objects currently do not support the *First and Last Copies* option.

Hints

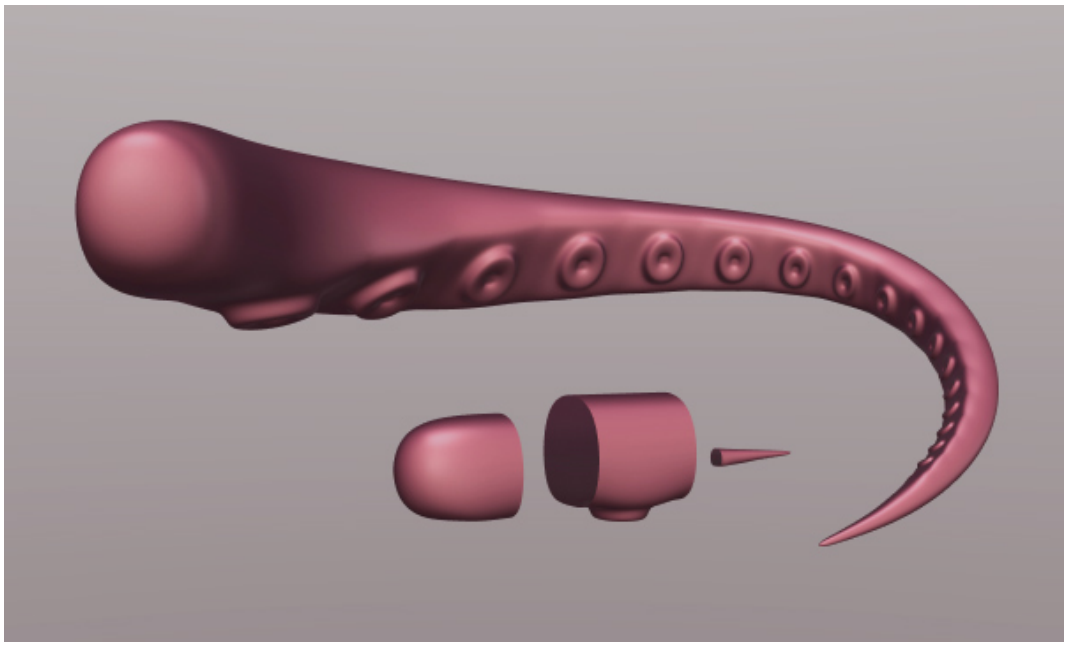
Offset Calculation

The transformation applied from one copy to the next is calculated as the sum of the three different components (*Relative*, *Constant* and *Object*), each of which can be enabled/disabled independently of the others. This allows, for example, a relative offset of (1.0, 0.0, 0.0) and a constant offset of (0.1, 0.0, 0.0), giving an array of objects neatly spaced along the X axis with a constant 0.1 unit between them, whatever the original object's size.

Examples



A chain created from a single link. Sample blend-file.



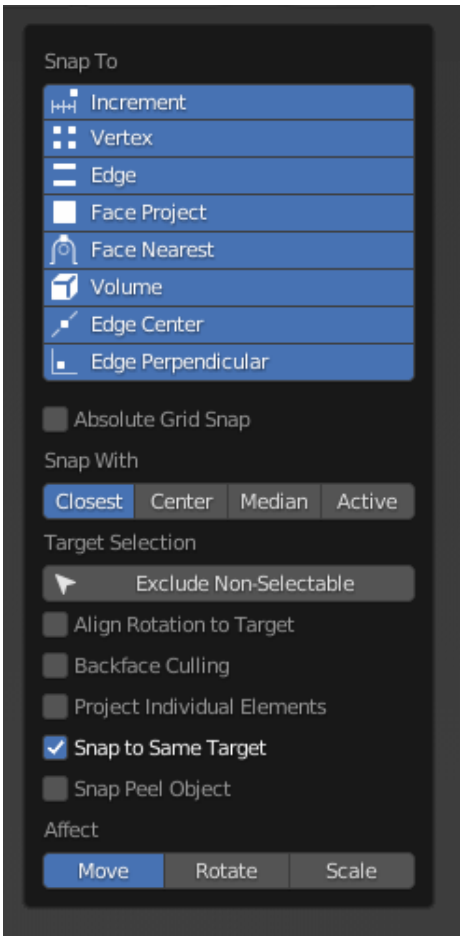
A tentacle created with an Array Modifier followed by a Curve Modifier.

The segment in the foreground is the base mesh for the tentacle; the tentacle is capped by two specially-modeled objects deformed by the same Curve object as the main part of the tentacle. Sample blend-file.

Fractal



Multi-level array animated with motion blur.



Fractal created with multiple arrays. Sample blend-file.

Bevel Modifier

The *Bevel* modifier bevels the edges of the mesh it is applied to, with some control of how and where the bevel is applied to the mesh.

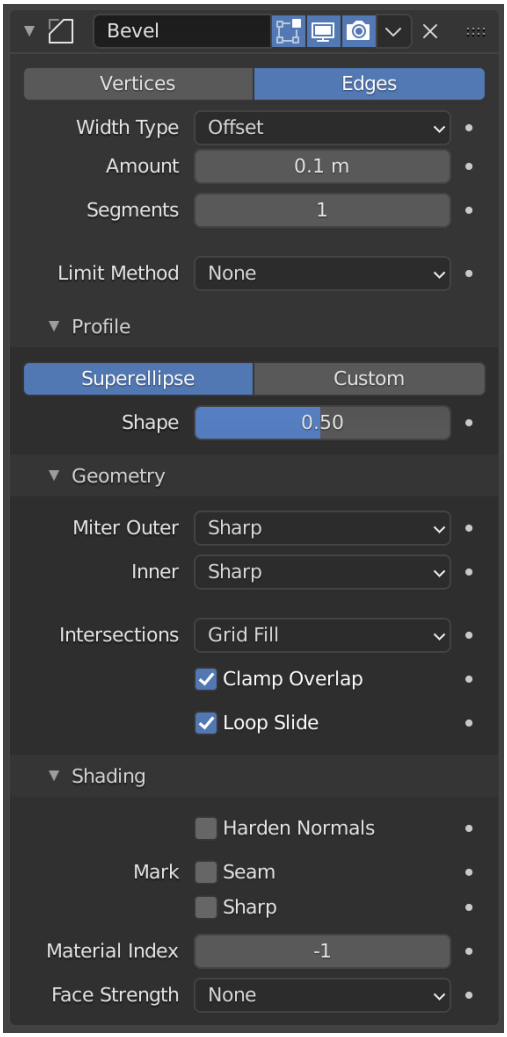
It is a non-destructive alternative to the Bevel Operation in Edit Mode.

Side views of a cube.



Not beveled.

Beveled.



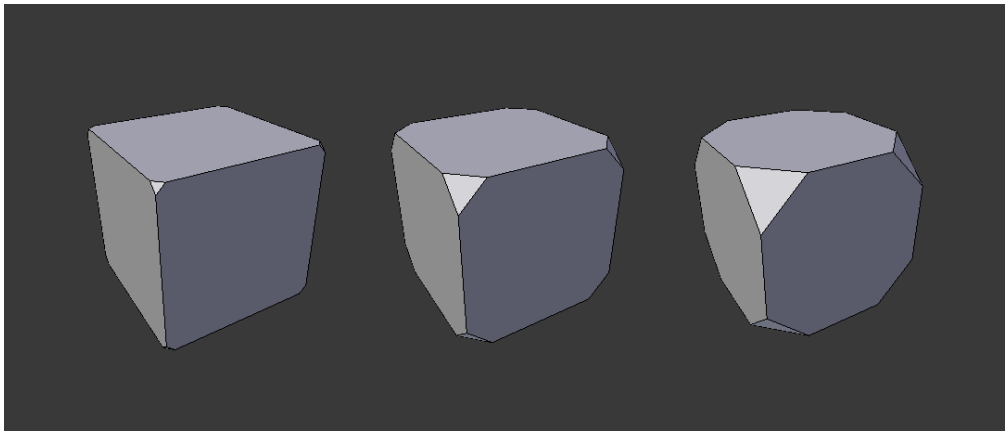
The Bevel modifier.

Options

Affect

Vertices:: Only the areas near vertices are beveled, the edges remain unchanged.

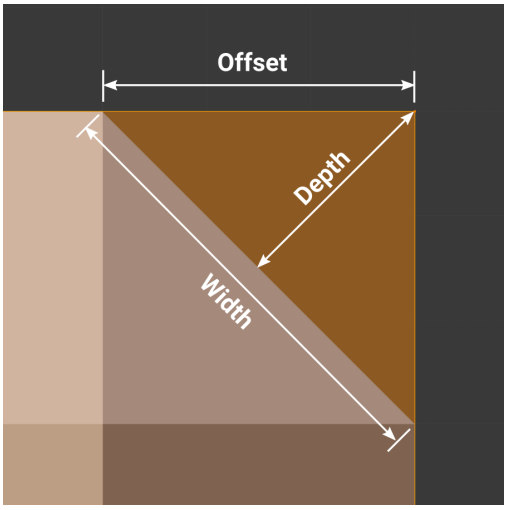
Edges:: Bevel the edges, creating intersections at vertices.



Three cubes with 0.1, 0.3 and 0.5 bevel widths, with *Vertices* option selected.

Width Type

Defines how *Width* will be interpreted to determine the amount of bevel.



Offset:: The distance from the new edge to the original.

Width:: The distance between the two new edges formed by the bevel (or the edges on either side of the bevel if there is more than one segment).

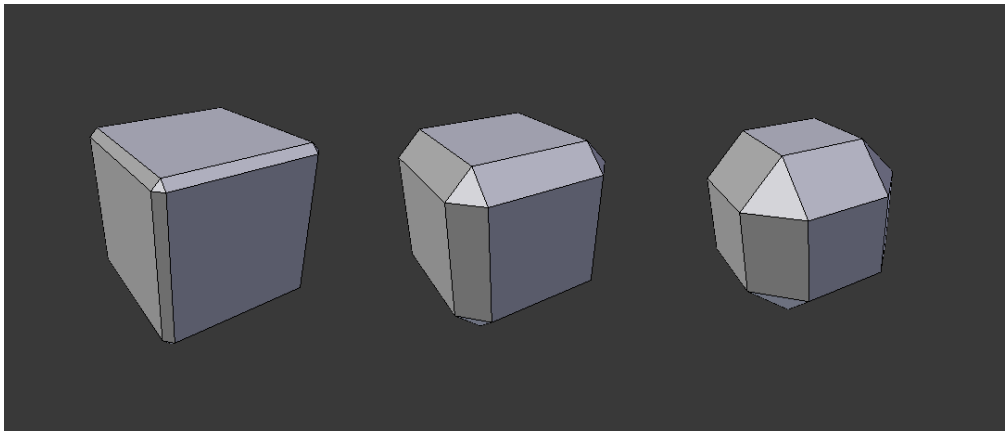
Depth:: Value is the perpendicular distance from the new bevel face to original edge.

Percent:: The percentage of the length of adjacent edge length that the new edges slide along.

Absolute:: The exact distance along edges adjacent to the beveled edge. A difference from **Offset** is visible when the unbeveled edges attached to beveled edges meet at an angle besides a right angle.

Width

The size of the bevel effect. See *Width Method* below.



Three Cubes with 0.1, 0.3 and 0.5 bevel widths.

Segments

The number of edge loops added along the bevel's face.

Limit Method

Used to control where a bevel is applied to the mesh.

None:: No limit, all edges will be beveled.

Angle:: Only bevels edges whose angle of adjacent face normals plus the defined *Angle* is less than 180 degrees. Intended to allow you to bevel only the sharp edges of an object without affecting its smooth surfaces.

Weight:: Use an attribute to determine the width of a bevel. When the bevel weight is 0.0, no bevel is applied.
Any attribute on the input mesh can be chosen. By default the *bevel_weight_edge* and *bevel_weight_vert* attributes adjusted in edit mode are used. For convenience, the modifier lists the attributes from the original mesh in the dropdown, but attributes created by previous modifiers can also be used. Attributes with non-

matching domains or types will be automatically interpolated to the correct type.

Vertex Group:: Use weights from a vertex group to determine the width of the bevel. When the vertex weight is 0.0, no bevel is applied. An edge is only beveled if both of its vertices are in the vertex group. See here about adjusting vertex group weights.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Profile

Superellipse

Creates a bevel with a uniform concave or convex curve.

Shape

The shape of the bevel, from concave to convex. It has no effect if *Segments* is less than 2.

Custom Profile

This widget allows the creation of a user-defined profile with more complexity than with the single profile parameter. The modal tool allows toggling the custom profile, but the shape of the profile is only editable in the options panel after the operation is confirmed.

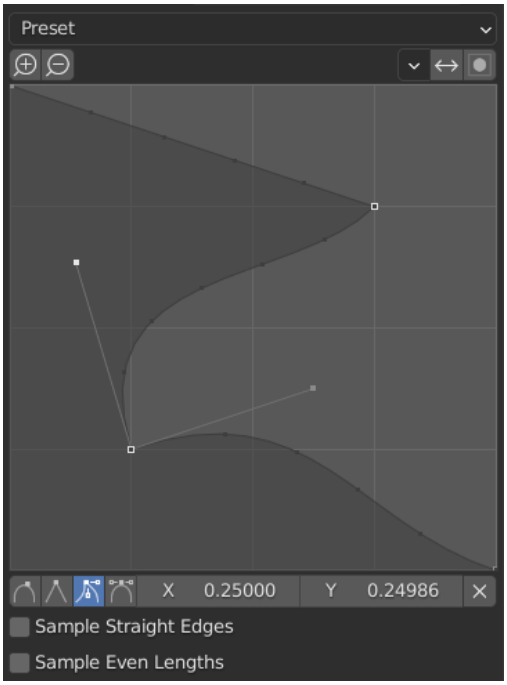
The profile starts at the bottom right of the widget and ends at the top left, as if it were between two edges meeting at a right angle. Control points are created in the widget and then the path is sampled with the number of segments from the Bevel modifier.

Miter Shape

The shape of the miter patterns, from concave to convex. It has no effect if *Segments* is less than 2.

Note

The *Miter Shape* slider stays active when miters are enabled because it still controls the shape of the miter profiles.



The custom profile widget.

Presets

The *Support Loops* and *Steps* presets are built dynamically depending on the number of segments in the bevel. If the number of segments is changed, the preset will have to be re-applied.

Sampling

Samples will first be added to each control point, then if there are enough samples, they will be divided evenly between the edges. The *Sample Straight Edges* option toggles whether the samples are added to edges with sharp control points on either side. If there aren't enough samples to give each edge the same number of samples, they will just be added to the most curved edges, so it is recommended to use at least as many segments as there are control points.

Geometry

Miter Inner/Outer

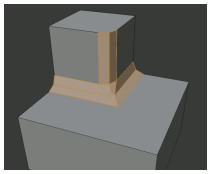
A *miter* is formed when two beveled edges meet at an angle. On the side where the angle is greater than 180 degrees, if any, it is called an *outer miter*. If it is less than 180 degrees, then it is called an *inner miter*. The outer and inner miters can each be set to one of these patterns:

Sharp:: Edges meet at a sharp point, with no extra vertices introduced on the edges.

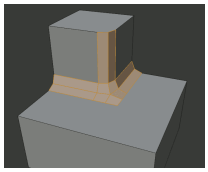
Patch:: Edges meet at a sharp point but in addition, two extra vertices are introduced near the point so that the edges and faces at the vertex may be less pinched together than what occurs in the *Sharp* case. This pattern does makes no sense for inner miters, so it behaves like *Arc* for them.

Arc:: Two vertices are introduced near the meeting point, and a curved arc joins them together.
The *Spread* slider controls how far the new vertices are from the meeting point.
The *Profile* curve widget controls the shape of the arc.

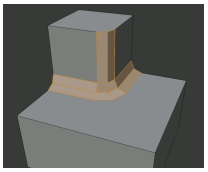
Diagrams of the miter patterns.



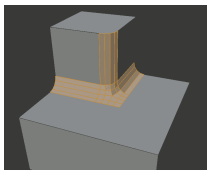
Sharp outer miter.



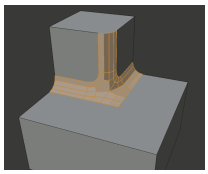
Patch outer miter.



Arc outer miter.



Sharp inner miter.



Arc inner miter.

Spread

The value used to spread extra vertices apart for non-sharp miters. This option is available when Miter Inner is set to Arc.

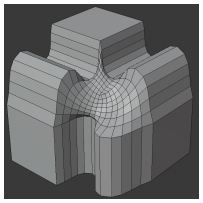
Intersections

When more than two beveled edges meet at a vertex, a mesh is created as a way to complete the intersection between the generated geometry. This option controls the method used to create that mesh.

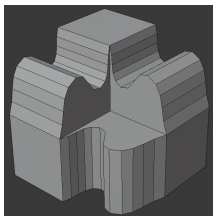
Grid Fill:: The default method for building intersections, useful when a smooth continuation of the bevel profile is desired. Without *Custom Profile* enabled, the curve of the profile continues through the intersection, but with a custom profile it just creates a smooth grid within the boundary of the intersection.

Cutoff:: Creates a cutoff face at the end of each beveled edge coming into the vertex. This is most useful for custom profiles when the new intersection is too complex for a smooth grid fill.
With a three way intersection, when the inner corners of the cutoff profiles faces meet at the same location, no center face is created.
The direction of the cutoff faces depends on the original vertex's normal.

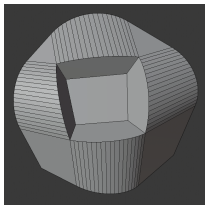
Intersection method options.



Grid fill intersection method.



Three way cutoff intersection where the inner vertices are merged.



Cutoff intersection method with a center face.

Clamp Overlap

Limits the width of each beveled edge so that edges cannot cause overlapping intersections with other geometry.

Loop Slide
: If there are unbeveled edges along with beveled edges into a vertex, the bevel tries to slide along those edges when possible. Turning the option off can lead to more even bevel widths.

Shading

Harden Normals
: When enabled, the per-vertex face normals of the bevel faces are adjusted to match the surrounding faces, and the normals of the surrounding faces are not affected. This will keep the surrounding faces flat (if they were before), with the bevel faces shading smoothly into them. For this effect to work, a mesh must have custom split normals. As a convenience, that option will be enabled for you if it is not already when you enable Harden Normals here.

Mark
: Seam
  : If a seam edge crosses a non-seam one and you bevel all of them, this option will maintain the expected propagation of seams.

  Sharp
  : Similar to Mark Seams, but for sharp edges.

Material Index
: The index of the material slot to use for the bevel. When set to -1, the material of the nearest original face will be used.

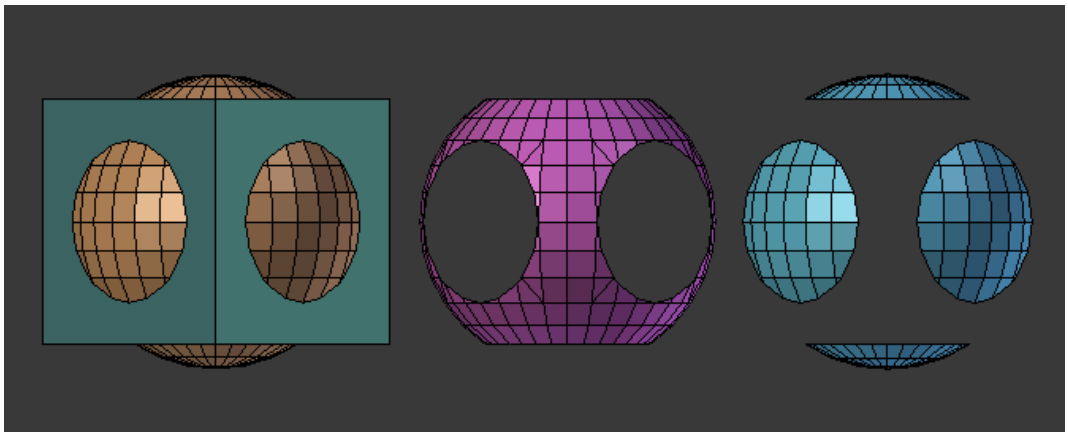
Face Strength
: Set *Face Strength* on the faces involved in the bevel, according to the mode specified here. This can be used in conjunction with a following Weighted Normals modifier (with the *Face Influence* option checked).

  None:: Do not set face strength.
  New::

	Set the face strength of new faces along edges to *Medium*, and the face strength of new faces at vertices to *Weak*.
Affected::	In addition to those set for the *New* case, also set the faces adjacent to new faces to have strength *Strong*.
All::	In addition to those set for the *Affected* case, also set all the rest of the faces of the model to have strength *Strong*.

Boolean Modifier

The *Boolean* modifier performs operations on meshes that are otherwise too complex to achieve with as few steps by editing meshes manually. It uses one of the three available Boolean operations to create a single mesh out of two mesh objects:



The Union, Intersection and Difference between a Cube and a UV Sphere, with the modifier applied to the sphere and using the cube as target.

This modifier needs a second mesh object, or collection of mesh objects, to be the target (the second operand) of the operation.

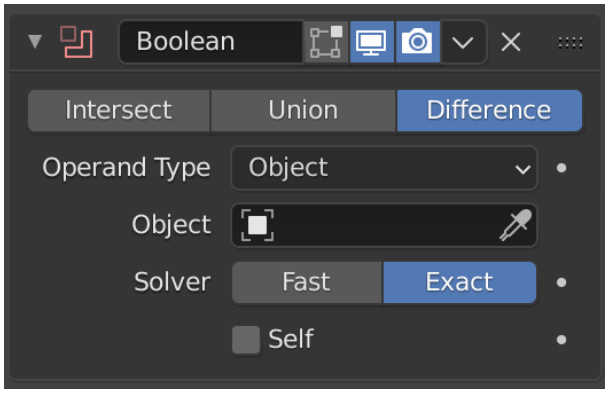
Warning

Only Manifold meshes are guaranteed to give proper results, other cases (especially "opened" meshes, Non-manifold but without any self-intersections) will usually work well, but might give odd glitches and artifacts in some cases.

Tip

If you have marked your objects to show the edges (in *Properties* ▸ *Object Properties* ▸ *Viewport Display*, enable *Wireframe*), you will see the edge creation process while you are moving your objects around. Depending on your mesh topology, you can also enable X-Ray and Transparency and see the topology being created in real-time.

Options



The Boolean modifier.

Operation

Intersect:: Everything inside both the target mesh and the modified mesh is kept. If the target is a collection, then only the inside of *all* meshes is kept.

Union:: The target mesh or collection is added to the modified mesh, removing any interior faces.

Difference:: The target mesh, or collection of meshes, is subtracted from the modified mesh (everything *outside* of the target mesh or collection is kept).

Operand Type

Choose the type of the operand (target).

Object:: The target is a mesh object.

Collection::

The target is a collection. When the target is a collection and the Solver is Fast, the Intersect operation is not allowed.

Object
: The name of the target mesh object.

Collection
: The name of the target collection (may be empty if Solver is Exact, which can be useful in combination with the Self option).

Solver
: Algorithm used to calculate the Boolean intersections.

Fast:: Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.

Exact:: Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Fast Solver*.

Solver Options

Materials *Exact Solver*
: Method for setting materials on the new faces.

Index Based:: Set the material on new faces based on the order of the material slot lists. If a material doesn't exist on the modifier object, the face will use the same material slot or the first if the object doesn't have enough slots.

Transfer:: Transfer materials from non-empty slots to the result mesh, adding new materials as necessary. For empty slots, fall back to using the same material index as the operand mesh.

Self Intersection *Exact Solver*

Correctly calculates cases when one or both operands have self-intersections, this involves more calculations making it slower.

Hole Tolerant *Exact Solver*

Optimizes the Boolean output for Non-manifold geometry at the cost of increased computational time. Because of the performance impact, this option should only be enabled when the *Exact* solver demonstrates errors with non-manifold geometry.

Overlap Threshold *Fast Solver*

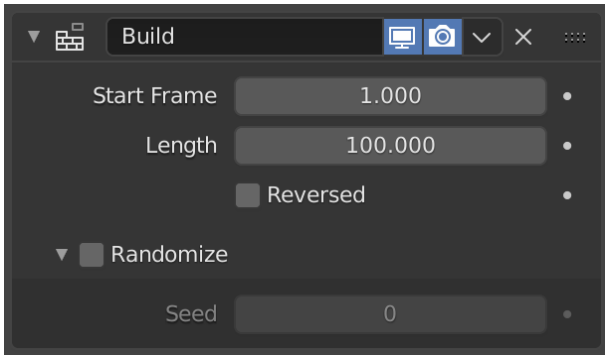
Maximum distance between two faces to consider them as overlapping. This helps solve the limitation of this solver, if the Boolean result seems unexpected try using the exact solver.

Build Modifier

The *Build* modifier causes the faces of the mesh object to appear or disappear one after the other over time.

By default, faces appear in the order in which they are stored in memory (by default, the order of creation). The face/vertex order can be altered in Edit Mode by using Sort Mesh Elements.

Options



The Build modifier.

Start Frame
: The start frame of the building process.

Length
: The number of frames over which to rebuild the object.

Reversed
: The modifier will operate in reverse, essentially allowing it to be used as a "deconstruction" effect. This is useful for making a set of instancing objects gradually disappear.

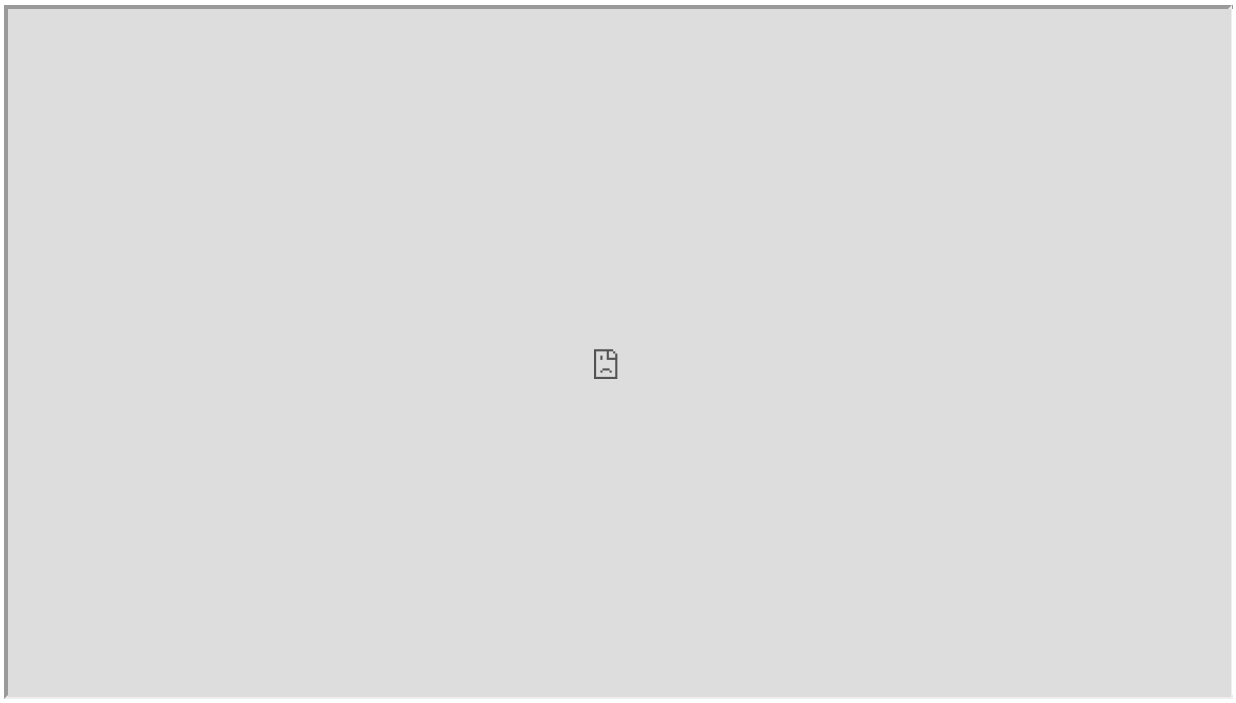
Randomize

Randomizes the order in which the faces are built.

Seed
: The random seed. Changing this value gives a different "random" order when *Randomize* is checked. This order is always the same for a given seed/mesh set.

Example

The Build modifier can be used to make a large number of items to progressively appear, without resorting to animating the visibility of each one by one. Examples of this include a mesh containing vertices only, which is used as an Instancing Vertex emitter, and has the Build modifier on it. Such a setup is a workaround/technique for being able to art-direct a semi-random layout of a collection of objects (e.g. leaves/balls forming a carpet). This can be preferable to particles e.g. due to undesirable distribution of items leaving random gaps and overlapping in other places.



Decimate Modifier

The *Decimate* modifier allows you to reduce the vertex/face count of a mesh with minimal shape changes.

This is not usually used on meshes which have been created by modeling carefully and economically (where all vertices and faces are necessary to correctly define the shape). But if the mesh is the result of complex modeling, sculpting and/or applied Subdivision Surface/ Multiresolution modifiers, the *Decimate* one can be used to reduce the polygon count for a performance increase, or simply remove unnecessary vertices and edges.

Unlike the majority of existing modifiers, this one does not allow you to visualize your changes in Edit Mode.

The modifier displays the number of remaining faces as a result of the *Decimate* modifier.

Options

Collapse

Merges vertices together progressively, taking the shape of the mesh into account.

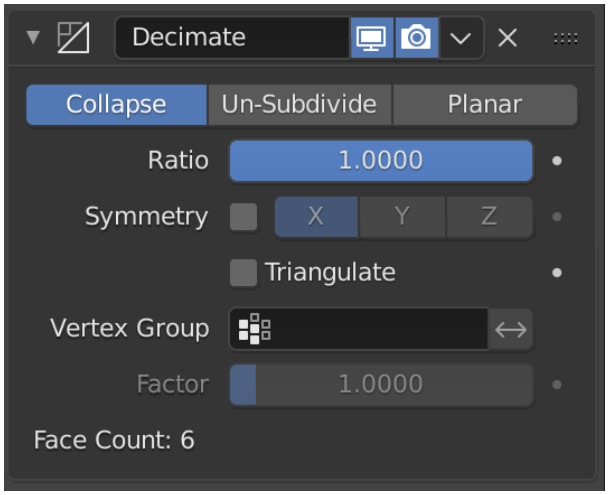
Ratio
: The ratio of faces to keep after decimation.

  - On 1.0: the mesh is unchanged.
  - On 0.5: edges have been collapsed such that half the number of faces remain (see note below).
  - On 0.0: all faces have been removed.

Note

Although the *Ratio* is directly proportional to the number of remaining faces, triangles are used when calculating the ratio.

This means that if your mesh contains quads or other polygons, the number of remaining faces will be larger than expected, because those will remain unchanged if their edges are not collapsed.



The Decimate modifier in Collapse mode.

This is only true if the *Triangulate* option is disabled.

Symmetry
: Maintains symmetry on a single axis.

Triangulate
: Keeps any resulting triangulated geometry from the decimation process.

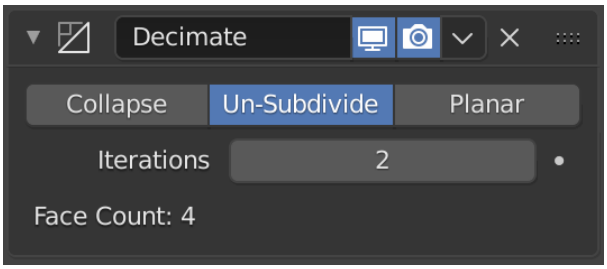
Vertex Group
: A vertex group that controls what parts of the mesh are decimated.

Factor
: The amount of influence the *Vertex Group* has on the decimation.

Un-Subdivide

It can be thought of as the reverse of subdivide. It attempts to remove edges that were the result of a subdivide operation. It is intended for meshes with a mainly grid-based topology (without giving uneven geometry). If

additional editing has been done after the subdivide operation, the results may be unexpected.

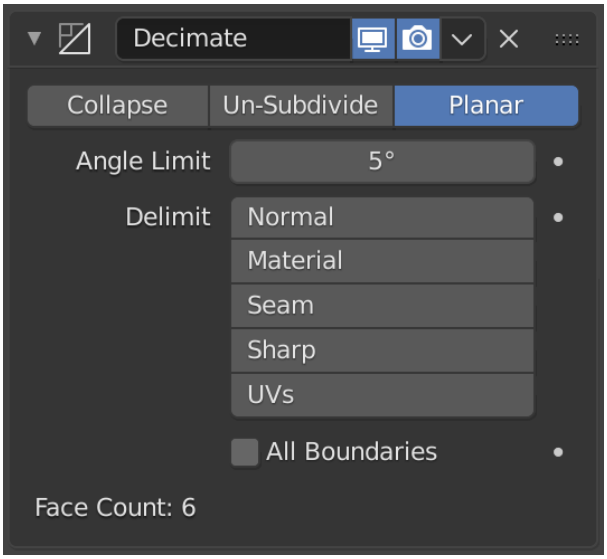


The Decimate modifier in Un-Subdivide mode.

Iterations
: The number of times to perform the un-subdivide operation.
Two iterations is the same as one subdivide operation, so you will usually want to use even numbers.

Planar

It reduces details on forms comprised of mainly flat surfaces.



The Decimate modifier in Planar mode.

Angle Limit
: Dissolve geometry which form angles (between surfaces) higher than this setting.

Delimit
: Prevent dissolving geometry in certain places.

Normal
: Does not dissolve edges on the borders of areas where the face normals are reversed.

Material
: Does not dissolve edges on the borders of where different materials are assigned.

Seam
: Does not dissolve edges marked as seams.

Sharp
: Does not dissolve edges marked as sharp.

UVs
: Does not dissolve edges that are part of a UV map.

All Boundaries
: When enabled, all vertices along the boundaries of faces are dissolved. This can give better results when using a high *Angle Limit*.

Edge Split Modifier

The *Edge Split* modifier splits, duplicates edges within a mesh, breaking 'links' between faces around those split edges.

The edges to split can be determined from the edge angle (i.e. angle between faces forming that edge), and/or edges marked as sharp.

Splitting an edge affects vertex normal generation at that edge, making the edge appear sharp. It can also be used for manual control of the smoothing process, where the user defines which edges should appear smooth or sharp (see Mesh Smoothing for other ways to do this). If desired, both modes can be active at once.

Note

This modifier is kept mostly for historical/compatibility reasons. Everything it can do in shading, and much more, can now be achieved using custom normals.

Unless you really need the topology changes it generates, it is not advised to use it in new projects.

Note

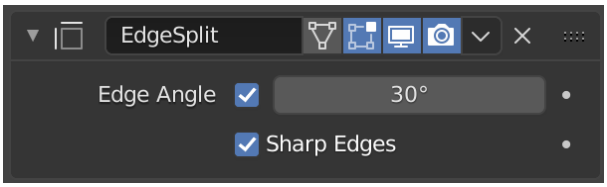
Splitting edges can also be performed manually in Edit Mode.

Options

Edge Angle
: When enabled, an edge will be split if the angle between its two adjacent faces is greater than the *Split Angle*.

Split Angle
: On 0: all edges are split. On 180: no edges are split.

Sharp Edges
: When enabled, edges will be split if they were marked as sharp.

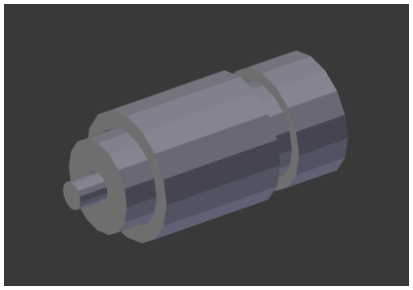


The Edge Split modifier.

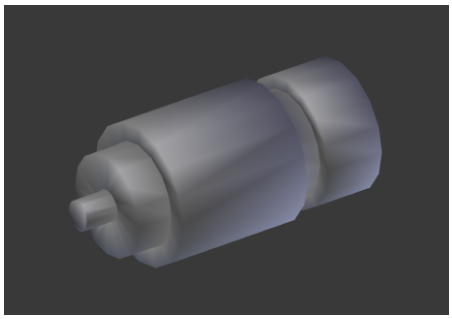
Note

Non-manifold edges will always be split.

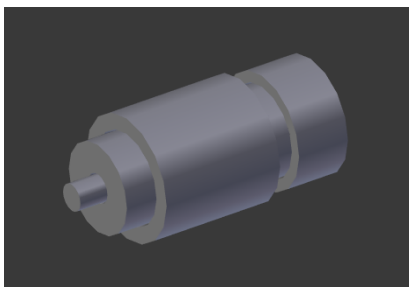
Examples



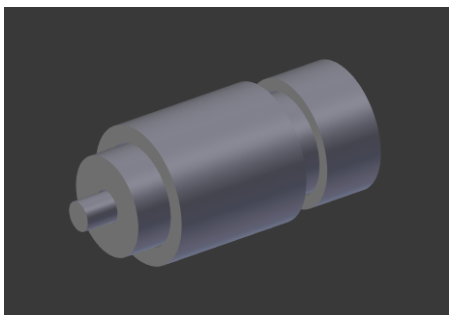
Flat shading.



Smooth shading.



Smooth shading with Edge Split.

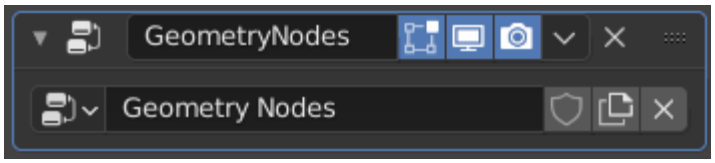


Smooth shading with Edge Split and Subdivision

Surface.

Geometry Nodes Modifier

The *Geometry Nodes* modifier creates a modifier with a node group which defines its functionality.



A new Geometry Nodes modifier with a new node group.

This modifier is supported by mesh, curve, text, and volume objects.

Options

Node Group

A Node Group with the geometry input and output. Those are respectively what is received and passed to the previous and next modifier in the stack. See Nodes for all available nodes.

Inputs

A list of the node group's inputs which can have unique values even if the group is shared among multiple modifiers.

If the input is connected to a Field socket, there will be a toggle to switch between using a single value for the input or using an attribute on the input geometry. Using an attribute for input means the value can be different for every element.

The attribute name used by default when using the node group in a modifier for the first time is defined in the node group inputs panel.

Note

The attribute domain and the used to access the attribute is defined by the node the input is connected to.

Warnings

Nodes that show a warning message in the node editor will also show that message here.

Custom warning messages can be created using the Warning Node.

Output Attributes

By connecting a field socket to the group output node, you can create custom Attributes from a Field output of any node in the node tree. The domain of the attribute must be specified in the group node's output properties. Note, this does not work with Instanced Data.

The attribute name used by default when using the node group in a modifier for the first time is defined in the node group outputs panel.

This panel is hidden unless output node has attribute sockets.

Manage

Bake

Bake Target

Specifies where baked data should be stored. This can be overridden for individual bakes.

Packed:: The baked data is packed into the .blend file. So no separate file is necessary.

Disk:: The baked data is stored in a separate directory on disk.

Bake Path

Location on disk where the baked data for Simulation Zones and Bake Nodes are stored.

See also

Geometry Node Baking

Named Attributes

This panel displays information about all custom named attributes used by the node group. More information is available in the geometry nodes inspection page.

Move to Nodes Operator

Creates a new geometry node tree with the name of the current node tree with `.wrapper` appended to the name. This operation moves all inputs and outputs from the old modifier into a new node group. In order for this operator to function, there **must** be a Group Input **and** a Group Output each with a Geometry socket attached to the node group. This action causes all *Output Attributes* to become *Internal Dependencies* utilizing the Store Named Attribute Node. All modifier "inputs" will then also become inputs of the newly created node group.

This operator is useful to easily allow a node tree to be reused in other trees or to mark it as an Asset to be reused in other projects.

Mask Modifier

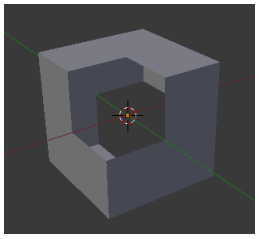
The *Mask* modifier allows vertices of an object to be hidden dynamically based on vertex groups.

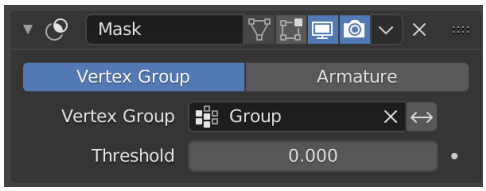
Options

Mode
: The Mask Modifier can hide parts of a mesh based on two different modes, selectable from this select menu.

  Vertex Group
  : Hides all vertices not included in the chosen vertex group.

    The Mask modifier in Vertex Group mode.





  Armature
  : When in Pose Mode, vertices belonging to the vertex group associated with the active bone (same names) will be visible. Vertices not in that group will be hidden.

    The Mask modifier in Armature mode.

Mesh to Volume Modifier

Reference

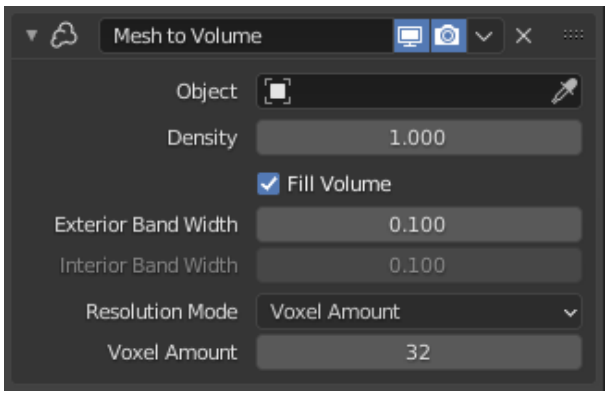
This modifier is only available for Volume Objects.

The *Mesh to Volume* modifier uses a mesh to create a new volume grid. All previously existing volume grids on the volume object are discarded. So this modifier is usually added to an empty volume object. The new volume grid is called “density”.

Tip

To copy and move the generated volume separately from the mesh object, use a collection instance.

Options



The Mesh to Volume modifier.

Object
: The mesh object that determines where the volume data will be generated.

Density
: Makes the generated volume appear denser or less dense when rendering.

Interior Band Width
: The maximum distance of the included voxels to the surface on the inside of the mesh.

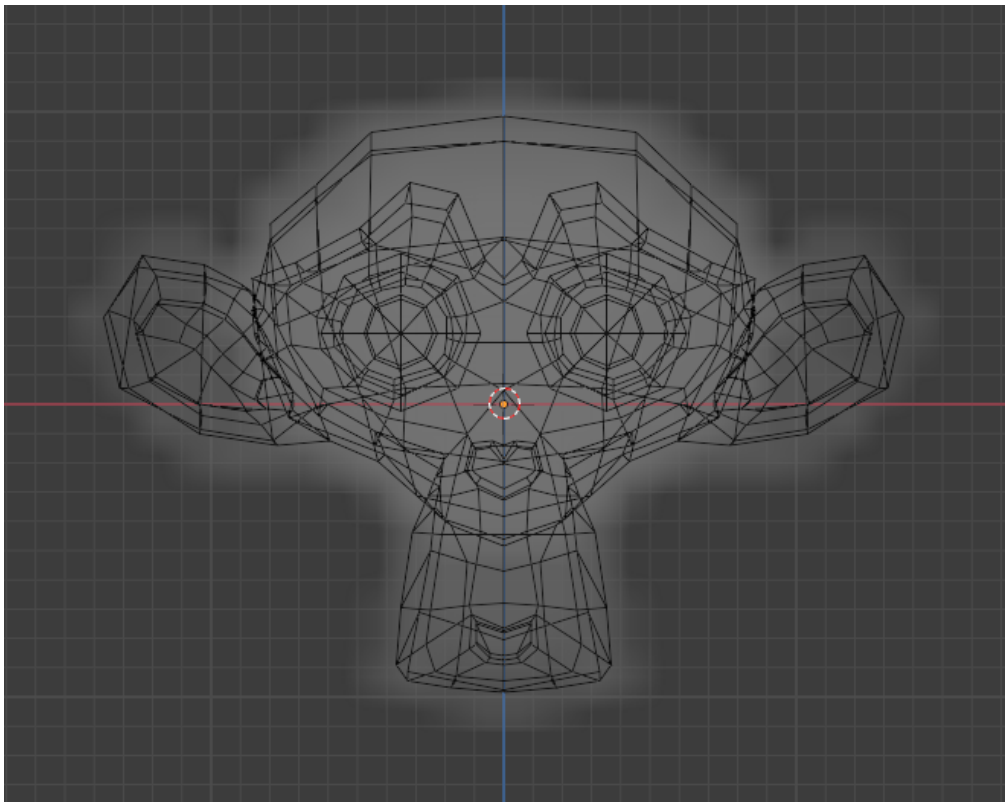
Resolution Mode

Mode for how the voxel size is specified.

Voxel Amount:: This allows setting an approximate number of voxels that will be used to represent mesh along its diagonal. When the dimensions of the mesh changes, the voxel size will change as well. For final rendering of animations, it's better to specify the voxel size explicitly to avoid artifacts.

Voxel Size:: This allows setting the exact voxel size that will be used. This is idea for rendering when the voxel size should not change between frames.

Example

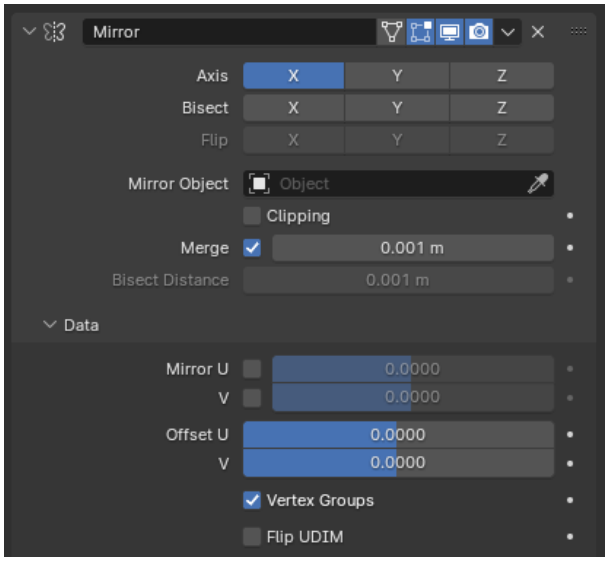


Converting Suzanne to a volume.

Mirror Modifier

The *Mirror* modifier mirrors a mesh along its local X, Y and/or Z axes, across the Object Origin. It can also use another object as the mirror center, then use that object's local axes instead of its own.

Options



The Mirror modifier.

Axis

The X, Y, Z axis along which to mirror, i.e. the axis perpendicular to the mirror plane of symmetry.

To understand how the axis applies to the mirror direction, if you were to mirror on the X axis, the positive X values of the original mesh would become the negative X values on the mirrored side.

You can select more than one of these axes. And will then get more mirrored copies. With one axis you get a single mirror, with two axes four mirrors, and with all three axes eight mirrors.

Bisect

If the mesh is already on both sides of the mirror plane, it is cut by that plane, and only one side (the "positive" one by default) is kept to perform the mirror process.

Flip

When *Bisect* is enabled on an axis, you can use this setting to switch the side kept and mirrored (i.e. when it is enabled, the "negative" side will be kept, instead of the "positive" one).

Mirror Object
: An Object Selector to select an object (usually an empty), which position and rotation will be used to define mirror planes (instead of using the ones from the modified object).

  You can animate it to move the mirror axis.

Clipping
: Prevents vertices from moving through the mirror plane(s) when you transform them in Edit Mode.

  If it is enabled but vertices are beyond the mirror plane and outside of the *Merge Distance*, the vertices will not be merged. But as soon as the vertices are within *Merge Distance* they are snapped together and cannot be moved beyond the mirror plane.

  Note

  Vertices on the mirror plane will be unable to move away from the mirror plane as long as *Clipping* is enabled. You must disable it to be able to move the vertices along the mirror axis again.

Merge
: Where a vertex is in the same place (within the *Merge Distance*) as its mirror it will be merged with the mirrored vertex.

  Merge Distance
  : The maximum distance between a vertex and its mirror copy at which they are merged together (being snapped on the mirror plane). Needs *Merge* to be enabled.

Bisect Distance
: Distance from the bisect plane within which vertices are removed.

Data

Flip UV
: With this option you can mirror the UV texture coordinates across the middle of the image.

  E.g. if you have a vertex with UV coordinates of (0.3, 0.9), its mirror copy will have UV coordinates of (0.7, 0.1).

UV Offsets
: Amount to shift mirrored UVs on the U/V axes.

  It's useful for baking (as overlapping UVs can cause artifacts to appear in the baked map), so the UVs can be moved outside the image and not used for baking, but still be used for display.

Vertex Groups
: Try to mirror existing vertex groups, with the following specific prerequisites:

  - The vertex groups you want to mirror must be named following the usual left/right pattern (i.e. with suffixes like ".R", ".right", ".L", etc.).
  - The mirror side vertex group must already exist (it will not be created automatically). It must also be completely empty (no vertices assigned to it).

Flip UDIM
: Mirror the texture coordinates around each tile center.

Hints

Many modeling tasks involve creating objects that are symmetrical. This modifier offers a simple and efficient way to do this, with real-time update of the mirror as you edit it. Once your modeling is completed you can either click *Apply* to make a real version of your mesh, or leave it as-is for future editing.

Accurately Positioning the Mirror Plane

To apply a *Mirror* modifier, it is common to have to move the object's origin onto the edge or face that is to be the axis for mirroring. This can be tricky when attempted visually.

A good technique to achieve an exact position is to select the edge, then snap *Cursor to Selection*. This will position the 3D Cursor in the center of the edge. Finally, use the Set Origin menu, and select *Origin to 3D Cursor*. This will move the object's origin (and thus, the mirror plane) to where the 3D cursor is located, and the mirroring will be exact.

An alternative is to use an empty as a *Mirror Object* that you move to the correct position.

Multiresolution Modifier

The *Multiresolution* modifier (often shortened to “Multires”) gives you the ability to subdivide a mesh similarly to the Subdivision Surface modifier, but also allows you to edit the new subdivision levels in Sculpt Mode.

Note

Multiresolution is the only modifier that cannot be repositioned in the stack after any modifier that will change geometry or other object data (i.e. all *Generate*, some *Modify* and some *Simulate* modifiers cannot come before the *Multiresolution*).

Deform modifiers will be applied onto the Multires subdivision levels instead of the base mesh, if they come after the Multires.

Tip

This is especially useful for re-projecting details from another sculpt with a Shrinkwrap modifier. For the best result make sure to set the wrap method to *Project*, snap mode to *Above Surface* and enable *Negative*.

Options

Levels Viewport
: Set the level of subdivisions to show in the viewport.

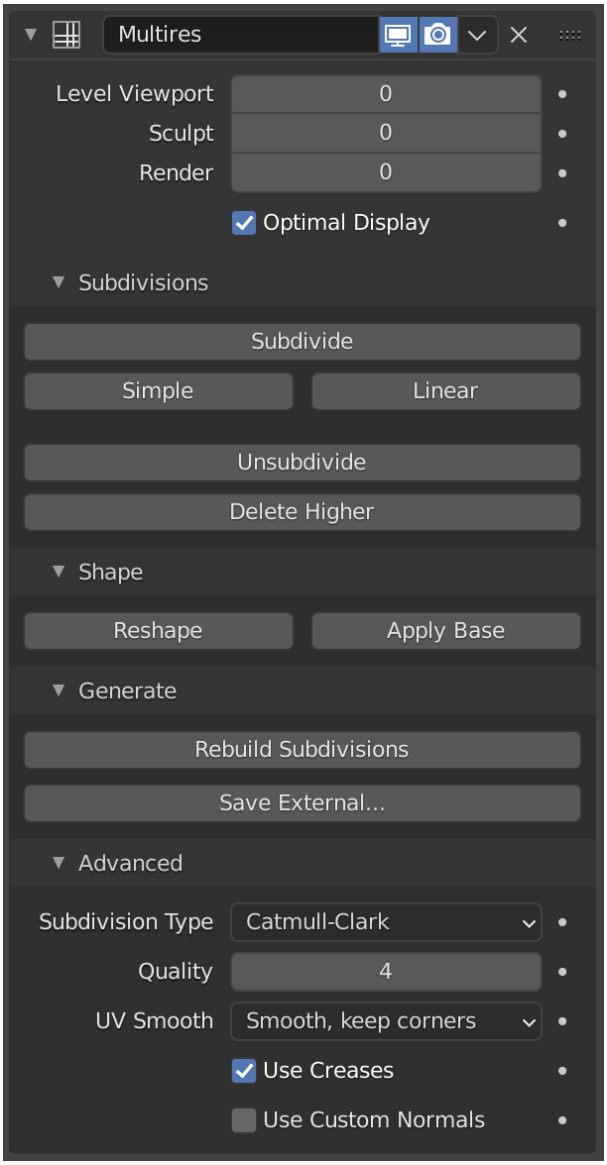
Sculpt
: Set the level of subdivisions to use specifically in Sculpt Mode. While in Sculpt mode use `Alt-1` to decrease the level or `Alt-2` to increase.

Render

Set the level of subdivisions to show when rendering.

Sculpt Base Mesh
: Deform the unsubdivided base mesh instead of the higher levels. Meanwhile the set level will be previewed. This allows you to make much broader changes in visual context to higher sculpted details without creating surface noise and artifacts.

Optimal Display
: Only display the edges of the original geometry. So when rendering the wireframe of this object, the wires of the subdivided edges will be skipped.



The Multiresolution modifier.

Subdivisions

Subdivide
: Creates a smooth level of subdivision (using the default Catmull-Clark algorithm).

Simple

Creates a level of subdivision with un-smoothed base mesh edges (using a simple interpolation by subdividing edges without any smoothing).

Linear
: Creates a completely un-smoothed level of subdivision (using linear interpolation of the current sculpted displacement).

Unsubdivide
: Rebuild a lower subdivision level of the current base mesh.

Delete Higher
: Deletes all subdivision levels that are higher than the current one.

Shape

Reshape
: Copy the shape of another object onto the multires levels by copying its vertex coordinates.

  To use it, first select a different mesh object with matching topology and vertex indices, then `Shift` select the object you wish to copy vertex coordinates to, and click *Reshape*.

Apply Base
: Modifies the original unsubdivided mesh to match the form of the subdivided mesh.

Generate

Rebuild Subdivisions
: Rebuilds all possible subdivisions levels to generate a lower resolution base mesh. This is used to create an optimized multiresolution version of a preexisting sculpt. This option is only available when no subdivision level have been created through the modifier.

Save External
: Saves displacements to an external `.btx` file.

Advanced

Quality
: How precisely the vertices are positioned (relatively to their theoretical position), can be lowered to get a better performance when working on high-poly meshes.

UV Smooth
: How to handle UVs during subdivision.

None:: UVs remain unchanged.
Keep Corners:: UV islands are smoothed, but their boundary remain unchanged.
Keep Corners, Junctions:: UVs are smoothed, corners on discontinuous boundary and junctions of three or more regions are kept sharp.
Keep Corners, Junctions, Concave:: UVs are smoothed, corners on discontinuous boundary, junctions of three or more regions and darts and concave corners are kept sharp.
Keep Boundaries:: UVs are smoothed, boundaries are kept sharp.
All:: UVs and their boundaries are smoothed.

Boundary Smooth
: Controls how open boundaries (and corners) are smoothed.

All:: Smooth boundaries, including corners.
Keep Corners:: Smooth boundaries, but corners are kept sharp.

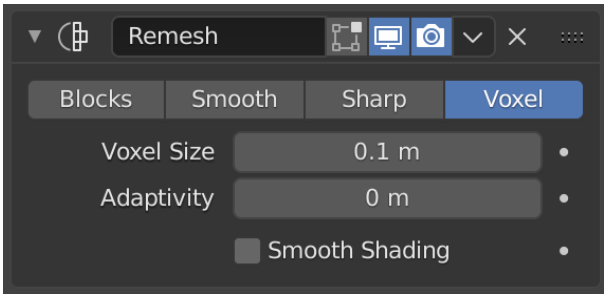
Use Creases
: Use the Weighted Edge Creases values stored in edges to control how smooth they are made.

Use Custom Normals
: Interpolates existing Custom Split Normals of the resulting mesh.

Remesh Modifier

The *Remesh* modifier is a tool for generating new mesh topology. The output follows the surface curvature of the input, but its topology contains only quads.

Options



The Remesh modifier.

Mode
: There are three basic modes available in the *Remesh* modifier. The output topology is almost identical between the three modes, what changes is the smoothing.

  Blocks
  : There is no smoothing at all.

  Smooth
  : Output a smooth surface.

  Sharp
  : Similar to *Smooth*, but preserves sharp edges and corners.

    Sharpness
    : Higher values produce edges more similar to the input, while lower values filter out noise.

  Voxel
  : Uses an OpenVDB to generate a new manifold mesh from the current geometry while trying to preserve the mesh's original volume.

Adaptivity
: Reduces the final face count by simplifying geometry where detail is not needed. This introduce triangulation to faces that do not need as much detail.

Smooth Shading
: Outputs faces with Smooth Shading instead of flat shading.

Octree Depth
: Sets the resolution of the output. Low values will generate larger faces relative to the input, higher values will generate a denser output.

Scale
: The result can be tweaked further by this, lower values effectively decrease the output resolution.

Remove Disconnected
: Filter out small disconnected pieces of the output.

  Thin parts of the input mesh can become loose, and generate small isolated bits of mesh. This option will remove those.

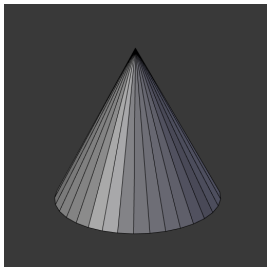
  Threshold
  : Use this to control how small a disconnected component must be to be removed.

Smooth Shading
: Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input faces is not preserved.

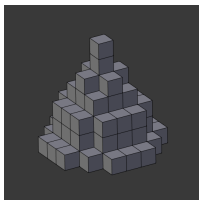
Note

The input mesh should have some thickness to it. If the input is completely flat, add a Solidify Modifier above the *Remesh* one.

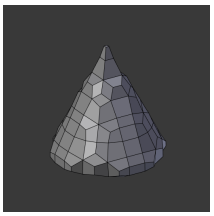
Examples



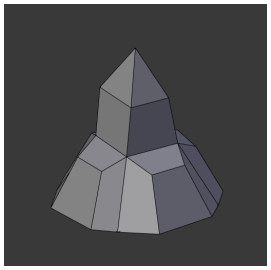
Unmodified mesh.



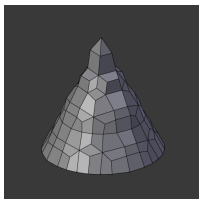
Blocks mode with Octree Depth 3.



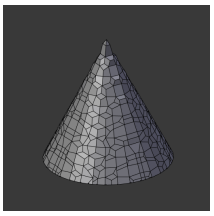
Smooth mode with Octree Depth 3.



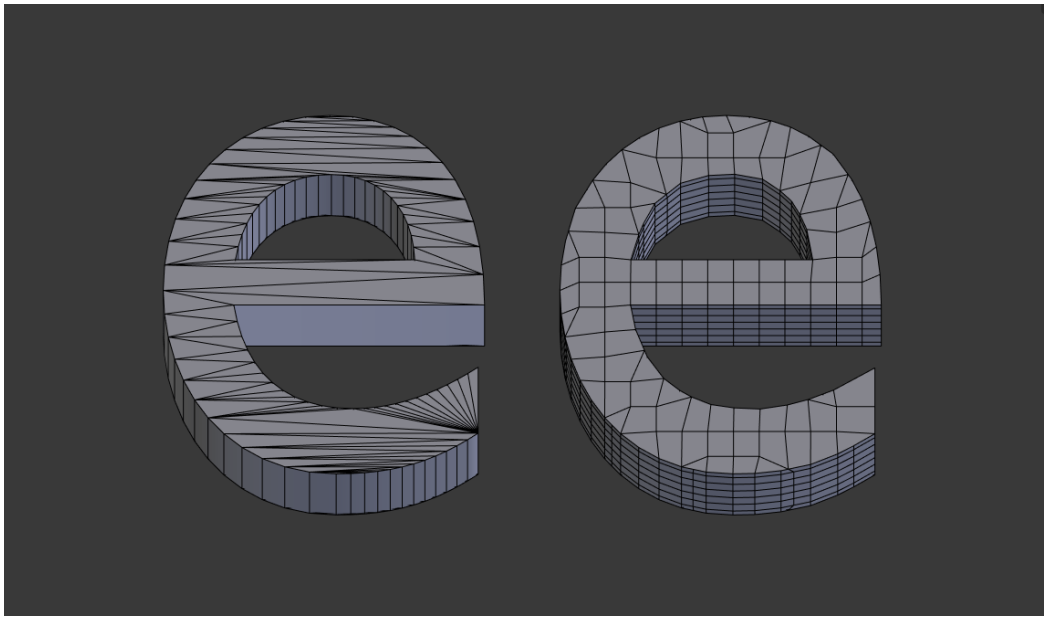
Sharp mode with Octree Depth 2.



Sharp mode with Octree Depth 3.



Sharp mode with Octree Depth 4.

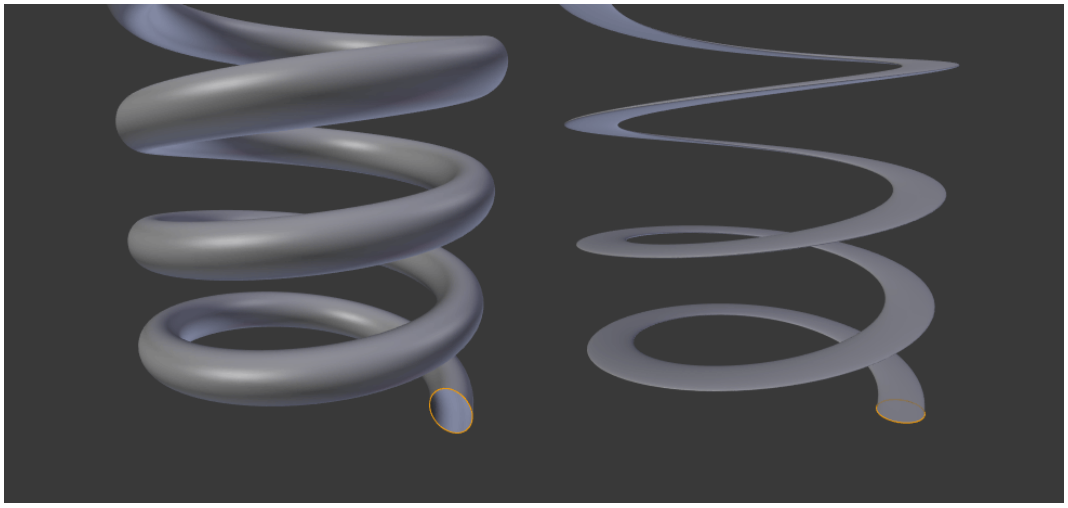


The Remesh Modifier applied to a text to improve its topology.

Animated Example

Screw Modifier

The *Screw* modifier is similar to the Screw tool in the Toolbar, in that it takes a profile object, a mesh or a curve, to create a helix-like shape.



Properly aligning the profile object is important.

The profile should be properly aligned to the cardinal direction of the object rather than to the screw axis.

Options

Angle
: Degrees for a single helix revolution.

Screw
: The height of one helix iteration.

Iterations
: Number of revolutions.

Axis

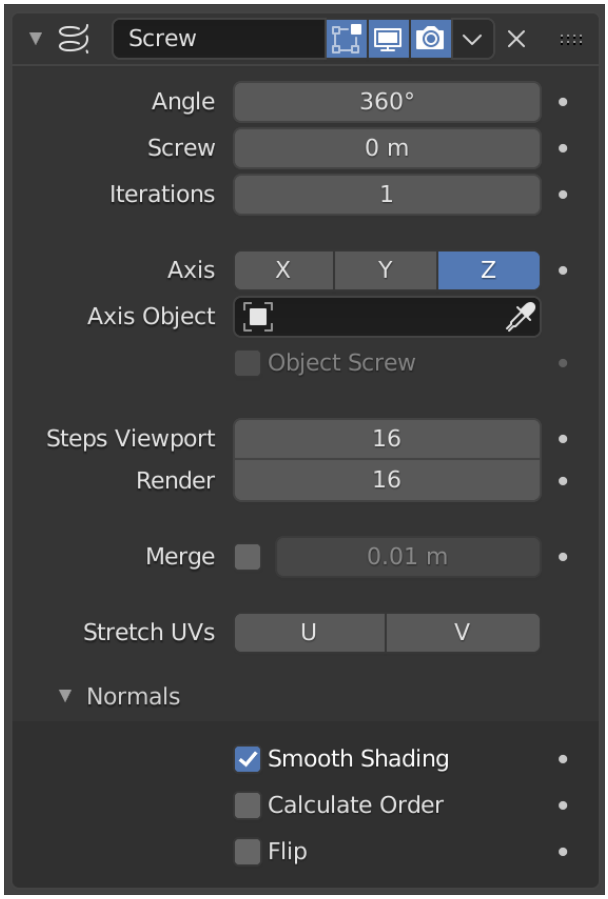
The axis along which the helix will be built.

Axis Object
: The name of an object to define the axis direction.

  Object Screw
  : Use the distance from the *Axis Object* to define the height of one helix iteration.

Steps Viewport
: Number of steps used for a single revolution displayed in the 3D Viewport.

Render
: As above, but used during render time. Increase to improve quality.



The Screw modifier.

Merge
: Merge vertices that lie on the axis of rotation. Use this to close off end points with a triangle fan.

  Merge Distance
  : Vertices under this distance to the axis are merged.

Stretch UVs
: Stretch the UV coordinates from (0.0 to 1.0) when UVs are present.

Normals

Smooth Shading

Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input geometry is not preserved.

Calculate Order

Order of edges is calculated to avoid problems with normals and shading. Only needed for meshes, not curves.

Flip

Flip normals direction.

Skin Modifier

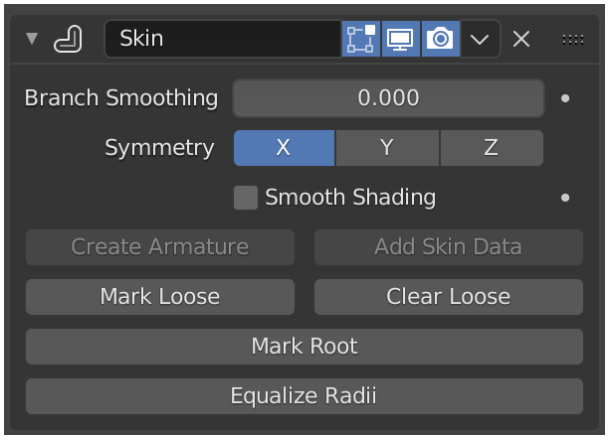
The *Skin* modifier uses vertices and edges to create a skinned surface, using a per-vertex radius to better define the shape. The output is mostly quads, although some triangles will appear around intersections.

It is a quick way to generate base meshes for sculpting and/or smooth organic shapes with arbitrary topology.

Note

Faces in the original geometry are ignored.

Options



The Skin modifier.

Branch Smoothing
: A branch point is a vertex with three or more connected edges. These areas tend to produce more complicated topology, some of which may overlap. This setting relaxes the surface around these points, with the side effect of shrinking it.

Symmetry
: These checkboxes are used to keep the output topology symmetrical in their respective axes. In other words, using it avoids merging triangles across an axis unless the triangles form a symmetric quad.

Note

They do not add geometry flipped across an axis. For that, the Mirror modifier should be used, typically placed above the Skin one.

Smooth Shading

Output faces with smooth shading rather than flat shading. The smooth/flat shading of the input geometry is not preserved.

Create Armature

Create an armature on top of the object. Each edge becomes a bone.

Note

If the root vertex has more than one adjacent edge, an extra bone will be created to serve as the root.

This tool does the following:

1. A new armature object is added with bones matching the input mesh. The active selection is switched to the new armature.
2. Weight groups are added to the input mesh. The Skin modifier propagates these weights to the output as well.
3. An Armature modifier is added directly below the Skin one. Note that the Armature modifier is being applied after the Skin one because it should only deform the output, whereas if it were above, it might change the resulting topology.

Add Skin Data

This modifier uses a custom set of data in the mesh, that is generated automatically when you add the modifier the first time.

However, you may remove that data, or loose it some way or the other. That operator will generate it again.

Mark/Clear Loose

By default, a branch vertex (vertex with three or more connected edges) will generate extra edge loops along adjacent edges in order to keep the output tight. Branches can be made loose by clicking *Mark Loose*,

which will allow the output to stretch between all adjacent vertices. This can be disabled again by clicking *Clear Loose*.

Mark Root
: Marking a vertex as root causes that vertex to be used for calculating rotations for connected limbs. Root vertices also affect the armature output, they will be used as the origin for the root bones.

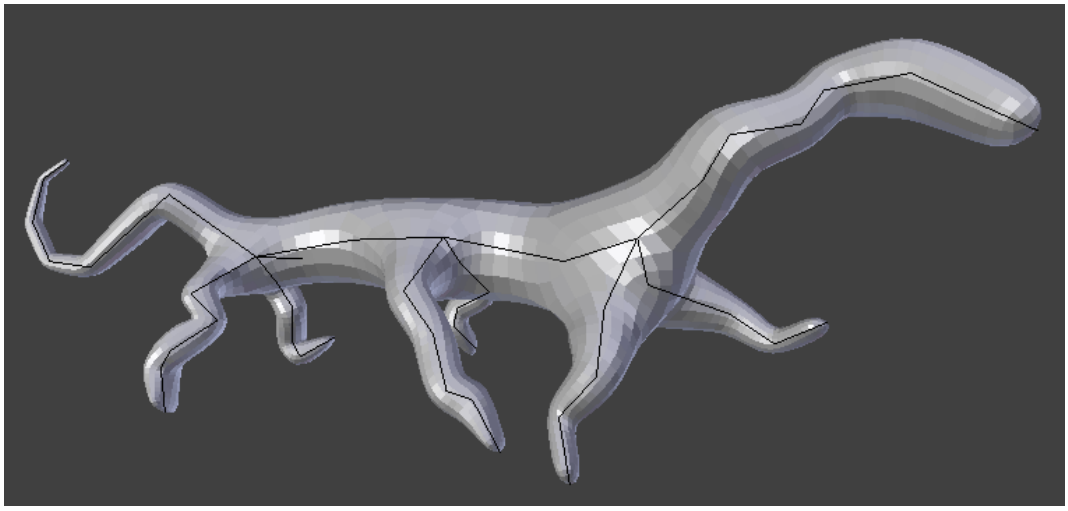
  Each set of connected vertices should have one root node (one is selected by default if you do not assign any manually). *Mark Root* enforces the one-root per set rule, so it is not necessary to manually unmark roots.

Equalize Radii
: Makes the skin radii of selected vertices equal on each axis.

Skin Mesh Data

That modifier needs a set of specific data in the original mesh to work properly. This data allows you to define the root vertices of each tree, which ones are loose, and the size (radius) of the skin at each vertex. The radii of input vertices can be individually scaled in Edit Mode with the Skin Resize.

Examples



Simple creature, made with only the Skin and Subdivision Surface modifiers.

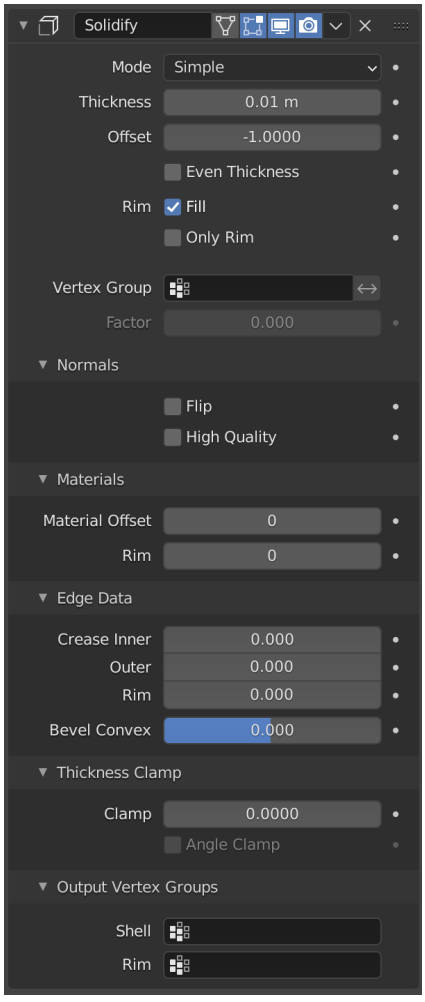
External Links

- Skin Modifier Development at Blender Nation – An early demonstration of the Skin Modifier by Nicholas Bishop (March 2011).
- Ji, Zhongping; Liu, Ligang; Wang, Yigang (2010). B-Mesh: A Fast Modeling System for Base Meshes of 3D Articulated Shapes, Computer Graphics Forum 29(7), pp. 2169-2178. – The work this modifier is based on (DOI 10.1111/j.1467-8659.2010.01805.x).
- Related thread on Blender artists.

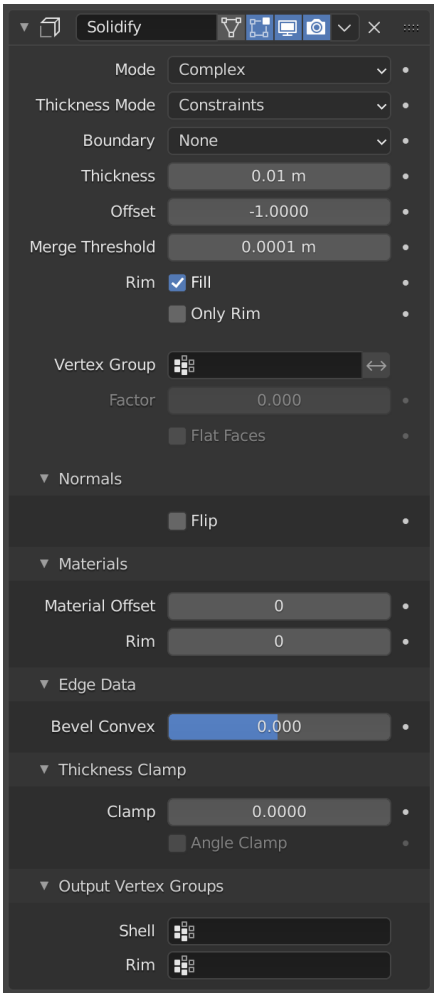
Solidify Modifier

The *Solidify* modifier takes the surface of any mesh and adds depth, thickness to it.

Options



The Solidify modifier in simple mode.



The Solidify modifier in complex mode.

Mode

Simple

This is the default solidify algorithm, which simply extrudes the geometry. This algorithm does not work on geometry where edges have more than two adjacent faces.

Important

If the normals of adjacent faces don't point into the same general direction, simple mode will not be able to solidify the boundary between those. This happens if the normals are not recalculated or for example on one-sided surfaces like a Möbius strip.

Complex

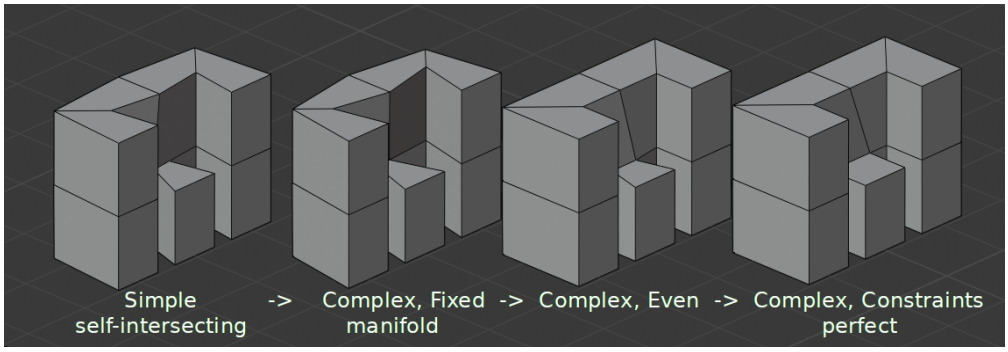
This is a solidify algorithm which can handle every geometric situation to guarantee a manifold output geometry. This algorithm is able to solidify shapes like Möbius strips, Klein bottles, architectural wall layouts and many more which the *Simple Mode* isn't able to do. If the special cases are not present it is recommended to choose *Simple* because the extra logic makes this algorithm much slower.

Note

There are no options for crease in the Modifier tab because crease is handled in a dynamic way. The modifier will transfer the creases of the original mesh in a smart way to the output mesh to work with the Subdivision Surface modifier.

Thickness Mode *Complex Mode*

Choose the kind of thickness handling (thickness solver).



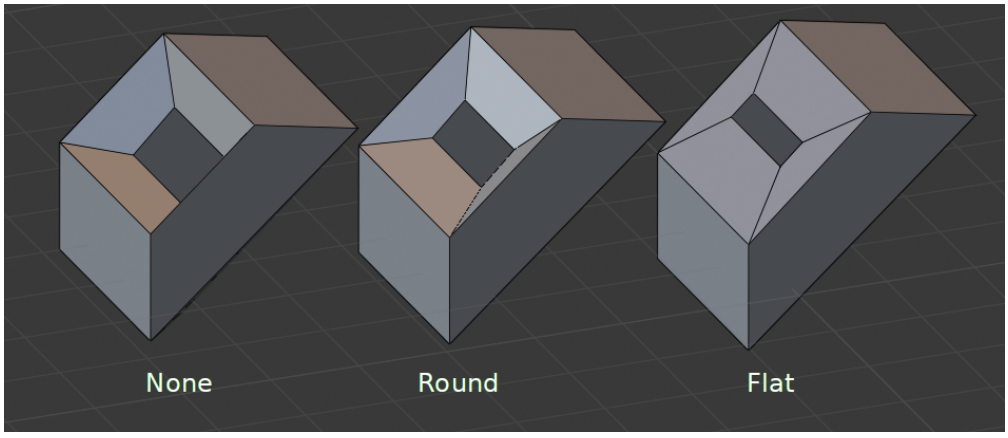
Different thickness options on a non-manifold mesh.

Fixed
: This is similar to *Simple Mode* without *Even Thickness*. The new vertices are always in a fixed distance to the old ones.

Even
: This is similar to *Simple Mode* with *Even Thickness* and *High Quality Normals*. It adjusts for sharp corners, but may not always work when more than three faces come together.

Constraints
: This is a more advanced model to try to always get the optimal thickness everywhere. For up to three faces it is always guaranteed to find an optimal solution.

Boundary *Complex Mode*
: Choose the kind of boundary that suits the model the most.



Different boundary options with a matCap.

None
: No boundary fix is applied. Results are stable.

Round
: Adjusts the boundary for an opening to face inwards (like a hole in an egg).

Flat
: Adjusts the boundary of a planar opening to be a flat (like a cut sphere).

Thickness
: The depth to be solidified.

Important

The modifier thickness is calculated using local vertex coordinates. If the object has a non-uniform scale, the thickness will vary on different sides of the object.

To fix this, either Apply or Clear the scale.

Offset
: A value between (-1 to 1) to locate the solidified output inside or outside the original mesh. The inside and outside is determined by the face normals. Set to 0.0, the solidified output will be centered on the original mesh.

Even Thickness *Simple Mode*
: Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

Merge Threshold *Complex Mode*
: Distance within which degenerated geometry is merged.

Rim
: Fill
  : Fills the gap between the inner and outer edges.

Only Rim

In *Simple Mode*: Will not extrude surfaces parallel to the original one, but instead will only add the perpendicular rim.

In *Complex Mode*: Will only leave the generated perpendicular rim.

Note

Fill and *Only Rim* only make a difference on Non-manifold objects, since the rims are generated from the borders of the original geometry.

Vertex Group

The weights of the selected vertex group are multiplied onto the *Thickness*, so vertices with lower weights will be less thick. The vertices which are not part of the vertex group will be used as if their weight was zero.

Invert

Reverses the vertex group weights, so that the used weight is one minus the actual weight.

Factor

How much the vertex weights are taken into account.

- On 0.0 , vertices with zero weight will have no thickness at all.
- On 0.5 , vertices with zero weight will be half as thick as those with full weight.
- On 1.0 , the weights are ignored and the *Thickness* value is used for every vertex.

Flat Faces *Complex Mode*

Use the minimal vertex weight assigned to the vertices of a face to make sure that new faces stay parallel to their original ones. This is slow, so disable it when it is not needed.

Note

If the final thickness of a vertex is zero, it will still be solidified. Therefore creating duplicate geometry, which sometimes needs extra

care.

Normals

Flip Normals
: Reverse the normals of all geometry (both the inner and outer surfaces).

High Quality Normals *Simple Mode*
: Normals are calculated to produce a more even thickness. Sometimes improves quality but also increases computation time.

Materials

Material Offset
: Choose a different material slot index to use for the new geometry. This is applied as an offset from the original material of the face from which it was solidified.

  - A value of 0 means it will use the same material.
  - A value of 1 means it will use the material immediately below the original material.
  - A value of -2 means the material two positions above the original material will be used.

  These are clamped to the top-most and bottom-most material slots.

Rim
: Similarly, you can give another material to the rim faces.

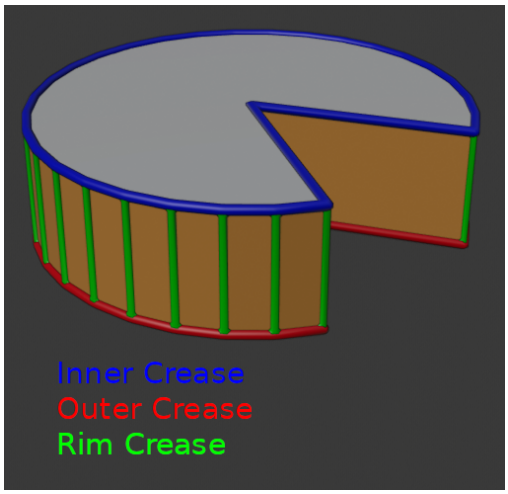
Edge Data

Inner *Simple Mode*
: Set a crease to the inner edges.

Outer *Simple Mode*
: Set a crease to the outer edges.

Rim *Simple Mode*
: Set a crease to the rim.

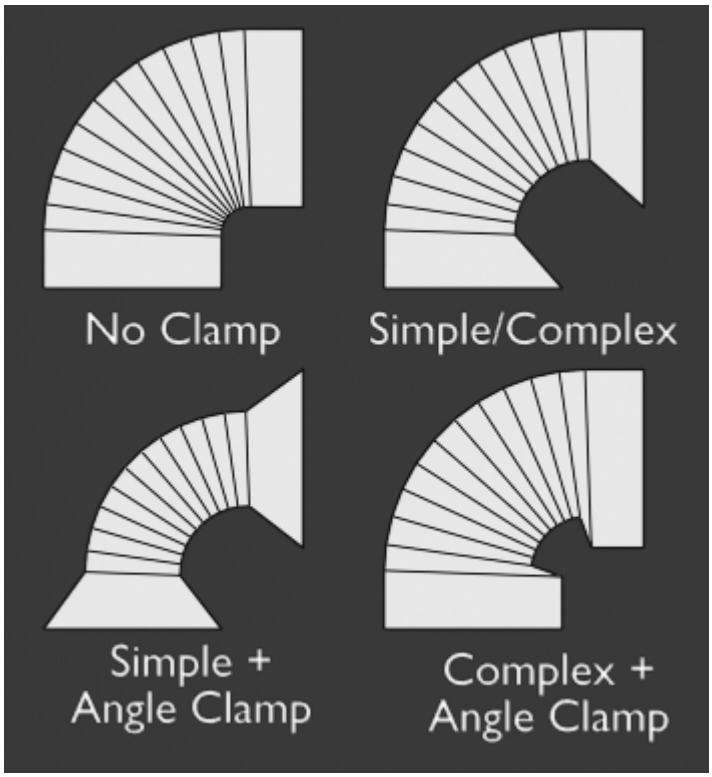
Bevel Convex
: Edge bevel weight to be added to outside edges.



: Edges which will get creases marked.

Thickness Clamp

Clamp
: A value between (0 to 2) to clamp offsets to avoid self-intersection. The amount is determined by the length of the shortest adjacent edge.



Clamp Offset.

Angle Clamp
: If enabled clamping will also consider angles in the geometry, not only lengths.

Output Vertex Groups

Shell
: Vertex group that the generated shell geometry will be weighted to. This allows you to use other modifiers to only affect the shell geometry by using a that modifier's vertex group influence control.

Rim
: Same as *Shell Vertex Group*, but for the generated rim geometry.

Known Limitations

Even Thickness

Solidify thickness is an approximation. While *Even Thickness* and *High Quality Normals* should yield good results, the final wall thickness is not guaranteed and may vary depending on the mesh topology. Especially for vertices with more than three adjacent faces.

In order to maintain a precise wall thickness in every case, we would need to add/remove faces on the offset shell, something this modifier does not do since this would add a lot of complexity. The best option to preserve wall thickness is complex mode with constraints thickness mode, but it is also not guaranteed to work perfect in every case.

Subdivision Surface Modifier

The *Subdivision Surface* modifier (often shorten to “Subdiv”) is used to split the faces of a mesh into smaller faces, giving it a smooth appearance. It enables you to create complex smooth surfaces while modeling simple, low-vertex meshes. It avoids the need to save and maintain huge amounts of data, and gives a smooth “organic” look to the object.

As with any modifier, order of execution (position in the modifier stack) has an important bearing on the results.

Keep in mind that this is a different operation than its companion, Smooth Shading. You can see the difference between the two in the grid image below.



Subdivision levels 0 to 3, without and with Smooth Shading.

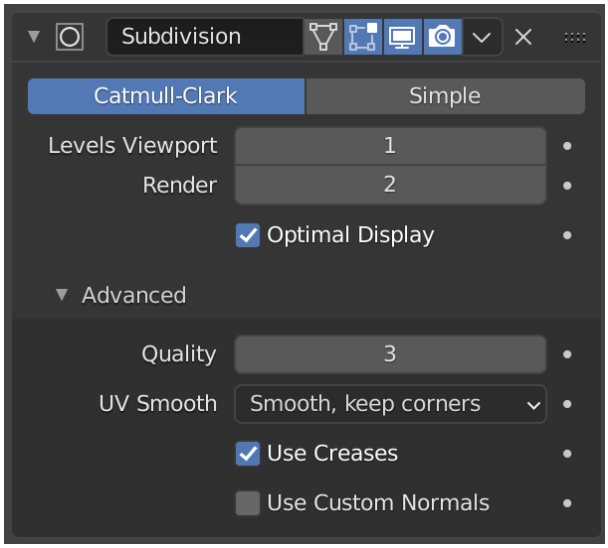
Tip

The *Subdivision Surface* modifier does not allow you to edit the new subdivided geometry without applying it, but the Multiresolution modifier does (in Sculpt Mode).

Note

This modifier uses the OpenSubdiv library as a backend.

Options



The Subdivision Surface modifier.

Catmull-Clark
: Subdivides and smooths the surfaces to create a more pleasant looking mesh. According to its Wikipedia page, the "arbitrary-looking formula was chosen by Catmull and Clark based on the aesthetic appearance of the resulting surfaces rather than on a mathematical derivation."

Simple
: Only subdivides the surfaces, this often does not provide any smoothing unless the surface is non-coplanar (the same as the Subdivide operator, in Edit Mode). To work around this behavior for non-coplanar geometry, triangulate to ensure all geometry is coplanar.

  Simple mode can be used, for example, to increase the base mesh resolution when using displacement maps.

Levels Viewport, Render

The number of subdivision levels shown in the 3D Viewport or the final render.

Warning

Higher levels of subdivisions results in more vertices, which means higher memory consumption (both system RAM, and video memory for display). This can cause Blender to hang or crash if not enough memory is available.

Tip

The right combination of these settings will allow you to keep a fast and lightweight approximation of your model when interacting with it in the 3D Viewport, but use a higher quality version when rendering.

Be careful not to set the *Viewport* subdivisions higher than the *Render* subdivisions, this would mean that the quality in the 3D Viewport will be higher than the rendered.

Optimal Display

When rendering the wireframe of this object, the wires of the new subdivided edges will be skipped (only displays the edges of the original geometry).

Advanced

Use Limit Surface

Places vertices at the surface that would be produced with infinite levels of subdivision (smoothest possible shape).

Quality

When *Use Limit Surface* is enabled this property controls how precisely vertices are positioned on the limit surface (relatively to their theoretical position of an infinitely subdivided mesh). It can be lowered to get a better performance.

Using higher values does not necessarily mean real improvement in quality, ideal results might be reached well before the maximum *Quality* value.

Note

This value can affect the accuracy of Edge Creases; using a higher *Quality* value will allow for a wider range of crease values to work accurately.

UV Smooth

Controls how subdivision smoothing is applied to UVs.

None:: UVs remain unchanged.

Keep Corners:: UV islands are smoothed, but their boundary remain unchanged.

Keep Corners, Junctions:: UVs are smoothed, corners on discontinuous boundary and junctions of three or more regions are kept sharp.

Keep Corners, Junctions, Concave:: UVs are smoothed, corners on discontinuous boundary, junctions of three or more regions and darts and concave corners are kept sharp.

Keep Boundaries:: UVs are smoothed, boundaries are kept sharp.

All:: UVs and their boundaries are smoothed.

Boundary Smooth

Controls how open boundaries (and corners) are smoothed.

All:: Smooth boundaries, including corners.

Keep Corners:: Smooth boundaries, but corners are kept sharp.

Use Creases

Use the Weighted Edge Creases values stored in edges to control how smooth they are made.

Use Custom Normals

Interpolates existing Custom Split Normals of the resulting mesh. Otherwise, new faces will have the overall normal orientation of that original face.

Keyboard Shortcuts

To quickly add a *Subdivision Surface* modifier to one or more objects, select the object(s) and press `Ctrl-1`. That will add a Subdivision Surface modifier with *Viewport* subdivisions set to 1. You can use other numbers too, such as `Ctrl-2`, `Ctrl-3`, etc, to add a modifier with that number of subdivisions. Adding a *Subdivision Surface* modifier in this fashion will not modify the *Render* subdivisions.

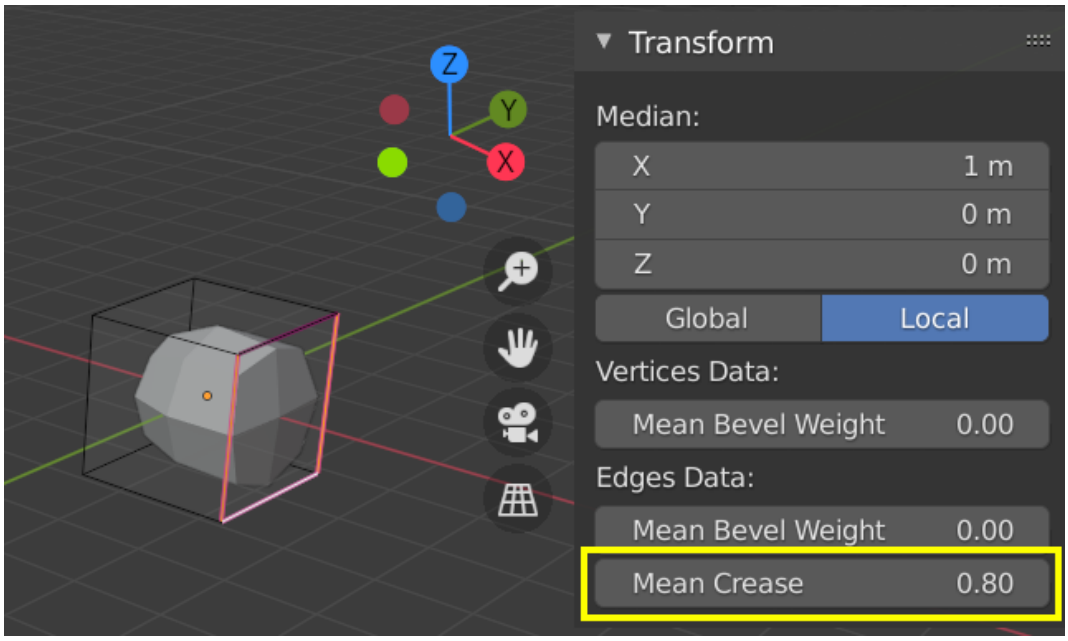
If an object already has a *Subdivision Surface* modifier, doing this will simply change its subdivision level instead of adding another modifier.

Control

Catmull-Clark subdivision rounds off edges, and often this is not what you want. There are several solutions that allow you to control the subdivision.

Weighted Edge Creases

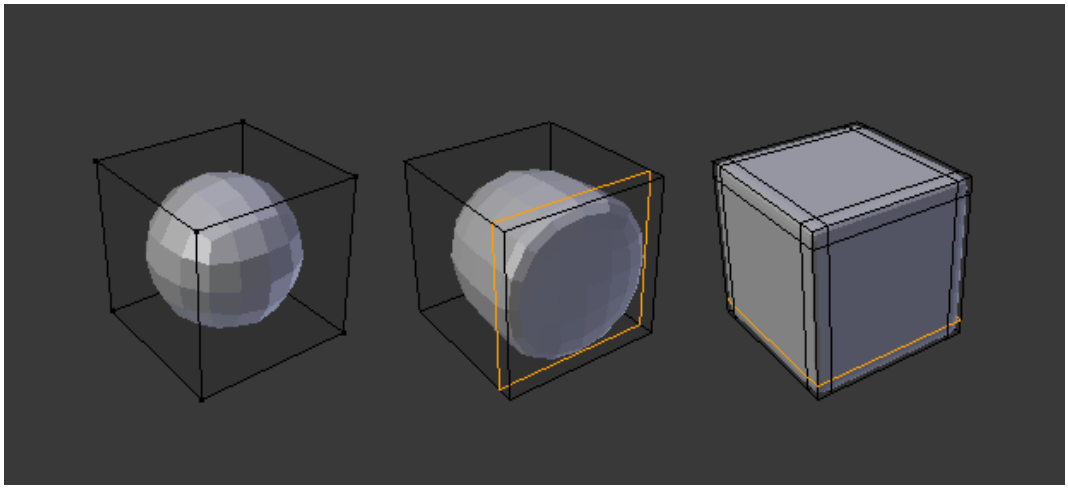
Weighted edge creases for subdivision surfaces allows you to change the way the *Subdivision Surface* modifier subdivides the geometry to give the edges a smooth or sharp appearance.



A subdivided cube with creased edges.

The crease weight of selected edges can be changed in the *Transform* panel, Sidebar of the 3D Viewport. The scale-like dedicated tool `Shift-E` can also be used to adjust the crease weight. A higher value makes the edge "stronger" and more resistant to the smoothing effect of subdivision surfaces.

Edge Loops



Subdivision Level 2 cube, the same with an extra Edge Loop, and the same with six extra Edge Loops.

The *Subdivision Surface* modifier demonstrates why good, clean topology is so important. As you can see in the figure, it has a drastic effect on a default cube. Until you add in additional loops (with e.g. Loop Cut and Slide), the shape is almost unrecognizable as a cube.

A mesh with deliberate topology has good placement of edge loops, which allow the placement of more loops (or their removal) to control the sharpness/smoothness of the resultant mesh.

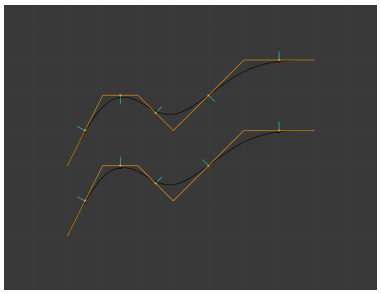
Known Limitations

Non-Contiguous Normals

Abrupt normal changes will prevent portions of the mesh from producing a smooth subdivision. Instead, these portions with non-contiguous normals will be subdivided using the “Simple” subdivision method.



Comparison of good normals and bad normals.

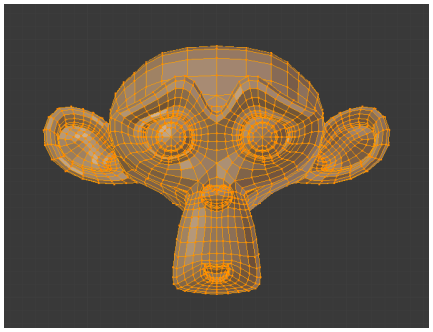


Side view of image on the left.

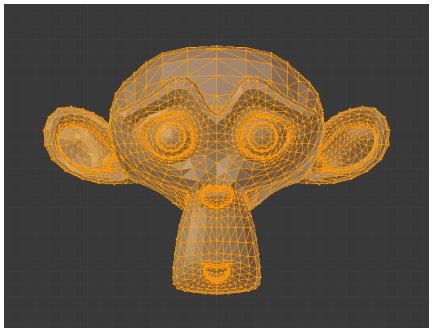
A quick way to fix this is to Recalculate Normals. If this does not work you may have to manually flip the normals.

Triangulate Modifier

The *Triangulate* modifier converts all faces in a mesh (quads and n-gons) to triangular faces. It fulfills the exact same function as the Triangulate tool in Edit Mode.



Mesh before Triangulate modifier.



Mesh after Triangulate modifier.

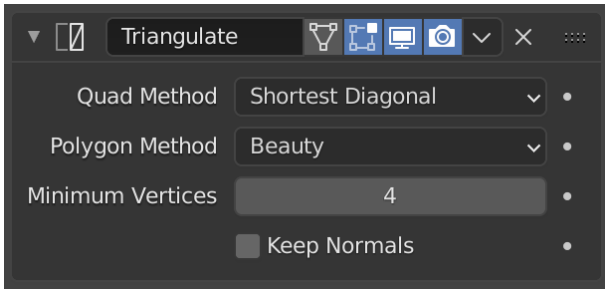
Options

Quad Method
: Beauty
: Split the quads in nice triangles, slower method.

: Fixed
: Split the quads on their 1st and 3rd vertices.

: Fixed Alternate
: Split the quads on their 2nd and 4th vertices.

: Shortest Diagonal



The Triangulate modifier.

Split the quads along their shortest diagonal.

Longest Diagonal
: Split the quads along their longest diagonal. This is the preferred mode for cloth simulations.

N-gon Method
: Beauty
  : Arrange the new triangles nicely, slower method.

: Clip
  : Splits n-gons using an ear-clipping algorithm (the same method of tessellation used for viewport display).

Minimum Vertices
: Minimum number of vertices a face must have to be triangulated. For example, setting this value to 5, will prevent triangulation of Quads and only triangulate N-gons.

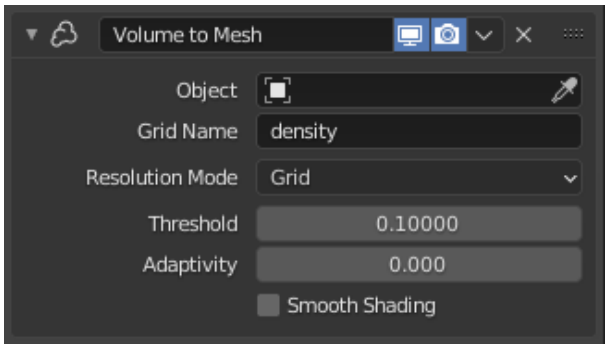
Volume to Mesh Modifier

This modifier is the inverse of the *Mesh to Volume* modifier. It takes an existing volume object and converts one of its grids to a mesh. Only scalar grids (such as the density grid) can be converted.

Tip

To copy and move the generated mesh separately from the volume object, use a collection instance.

Options



The Volume to Mesh modifier.

Object
: The source volume object.

Grid Name
: The name of the grid that will be converted. This has to be a scalar grid.

Resolution Mode
: Mode for how the resolution of the final mesh is controlled.

  Grid
  : This makes the resolution dependent on the resolution of the grid that is converted. Higher resolution grids result in a higher resolution mesh. In many cases, that is the most efficient mode.

  Voxel Amount
  : Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.

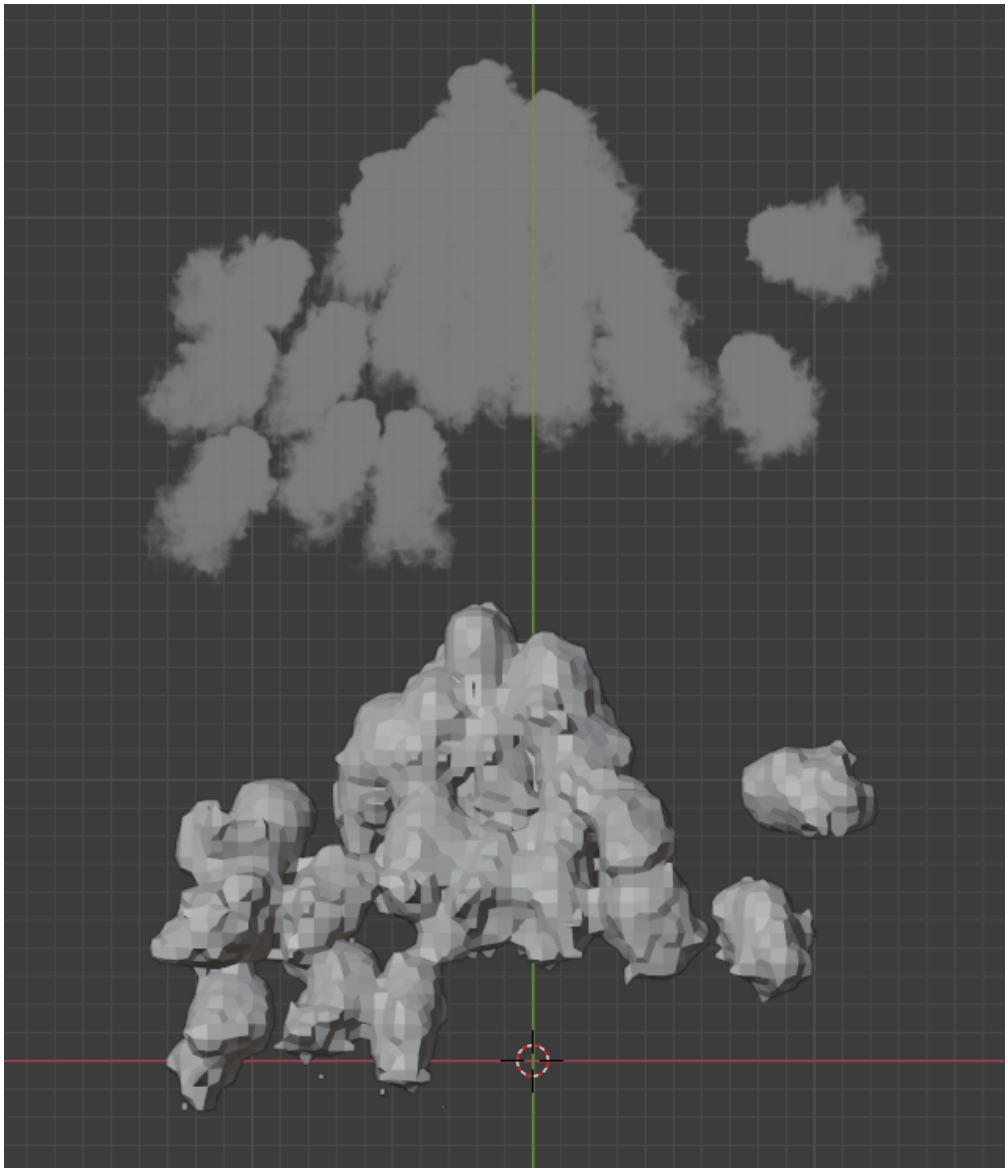
Voxel Size
: Use a fixed resolution that does not change when the volume changes.

Threshold
: Voxels with a larger value are considered to be inside the mesh and all other voxels outside. The mesh will be generated on the boundary of inside and outside voxels. This is sometimes also called the “iso value”.

Adaptivity
: This is similar to decimating the final to reduce resolution where it is not needed.

Smooth Shading
: Enables smooth shading on the generated mesh.

Example

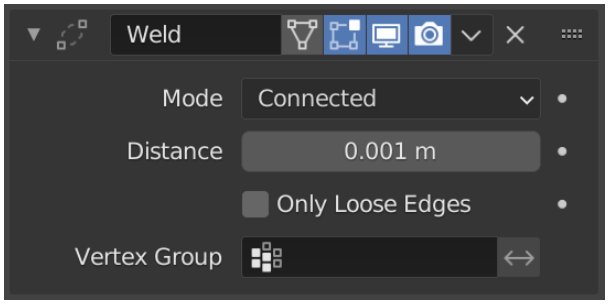


Converting a cloud-shaped volume to a mesh.

Weld Modifier

The *Weld* modifier looks for groups of vertices within a threshold and merges them, collapsing the surrounding geometry.

Options



The Weld modifier.

Mode
: Method for choosing which vertices are merged.

  All:: Merge includes all geometry including loose parts.

  Connected:: Merge only includes attached geometry i.e. the modifier will not merge loose parts together.

Distance
: Maximum distance that the vertices must have each other to be merged.

Only Loose Edges *Connected Mode*
: Only collapse short edges which are not adjacent to any face. This is useful for example to stitch the seams used in cloth simulations.

Vertex Group
: When the *Vertex Group* option is selected, only vertices with weight above zero will be affected by the modifier.

Invert <–>

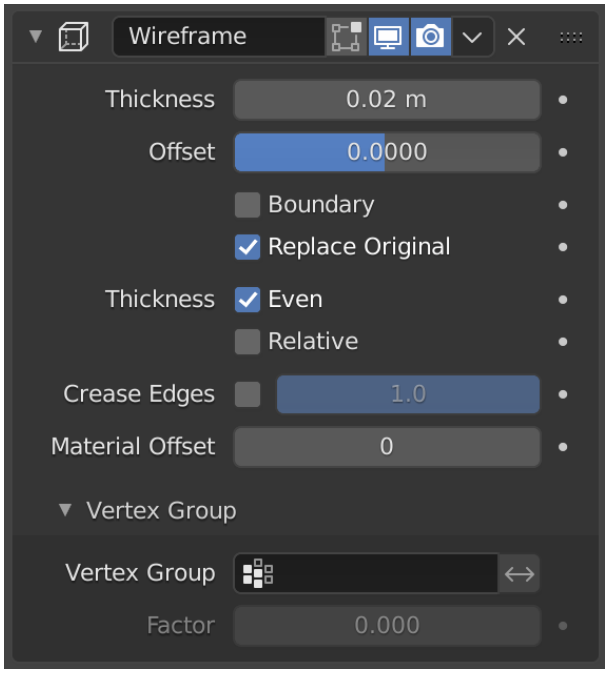
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be merged by the modifier.

The setting reverses the weight values of the group.

Wireframe Modifier

The *Wireframe* modifier transforms a mesh into a wireframe by iterating over its faces, collecting all edges and turning those edges into four-sided polygons. Be aware of the fact that your mesh needs to have faces to be wireframed. You can define the thickness, the material and several other parameters of the generated wireframe dynamically via the given modifier options.

Options



The Wireframe modifier.

Thickness
: The depth or size of the wireframes.

Offset
: A value between (-1 to 1) to change whether the wireframes are generated inside or outside of the original mesh. Set to zero, *Offset* will center the wireframes around the original edges.

Boundary
: Creates wireframes on mesh island boundaries.

Replace Original
: If this option is enabled, the original mesh is replaced by the generated wireframe. If not, the wireframe is generated on top of it.

Thickness

Even
: Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

Relative
: Determines the edge thickness by the length of the edge. Longer edges will be thicker.

Crease Edges
: This option is intended for usage with the Subdivision modifier. Enable this option to crease edges on their junctions and prevent large curved intersections.

Crease Weight
: Define how much crease (0 to 1, nothing to full) the junctions should receive.

Material Offset
: Uses the chosen material index as the material for the wireframe; this is applied as an offset from the first material.

Warning

Wireframe thickness is an approximation. While *Even Thickness* should yield good results in many cases, skinny faces can cause ugly spikes. In this case you can either reduce the extreme angles in the geometry or disable the *Even Thickness* option.

Vertex Group

The weights of the selected vertex group are multiplied onto the *Thickness*, so vertices with lower weights will be less thick. The vertices which are not part of the vertex group will be used as if their weight was zero.

Invert

Reverses the vertex group weights, so that the used weight is one minus the actual weight.

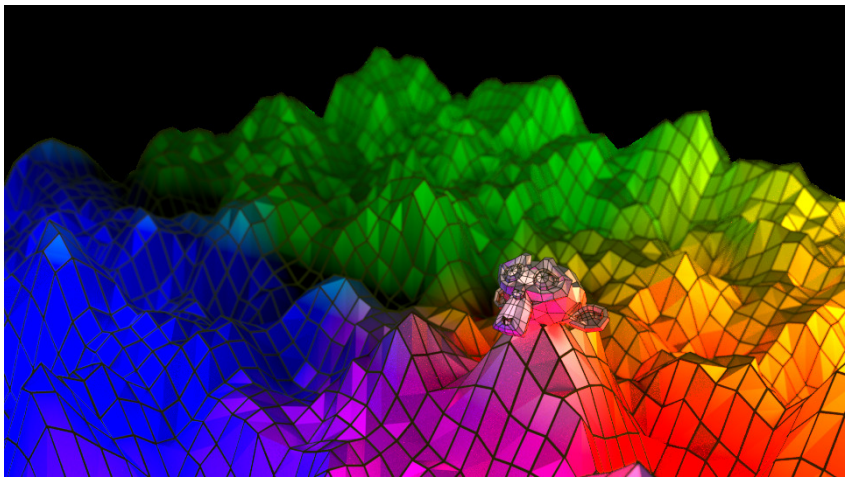
Factor
: How much the vertex weights are taken into account.

  - On 0.0 , vertices with zero weight will have no thickness at all.
  - On 0.5 , vertices with zero weight will be half as thick as those with full weight.
  - On 1.0 , the weights are ignored and the *Thickness* value is used for every vertex.

Note

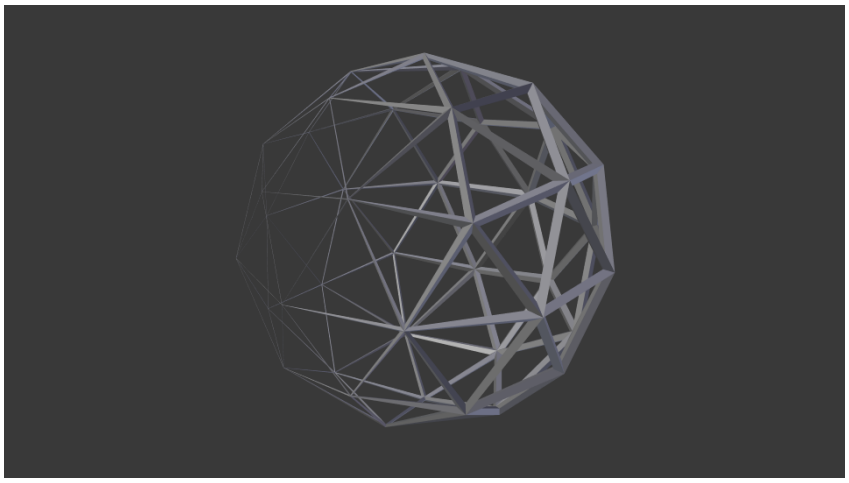
If the final thickness of a vertex is zero, it will still generate a wireframe. Therefore creating duplicate geometry, which sometimes needs extra care.

Examples



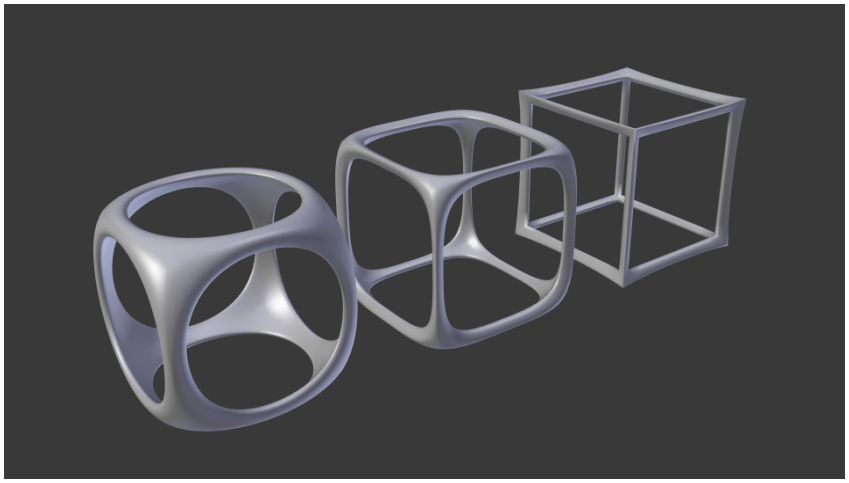
Wireframes on a displaced plane.

In this example, the wireframes carry a second (dark) material while the displaced plane uses its original one.



Vertex group weighting.

The weights of the vertex group gradually change from 0 to 1.



Wireframe and Subdivision Surface modifier.

Cube with enabled *Crease Edges* option. The *Crease Weight* is set to 0, 0.5 and 1.

Deform

- Armature Modifier
- Cast Modifier
- Curve Modifier
- Displace Modifier
- Hook Modifier
- Laplacian Deform Modifier
- Lattice Modifier
- Mesh Deform Modifier
- Shrinkwrap Modifier
- Simple Deform Modifier
- Smooth Modifier
- Smooth Corrective Modifier
- Smooth Laplacian Modifier
- Surface Deform Modifier
- Volume Displace Modifier
- Warp Modifier
- Wave Modifier

Armature Modifier

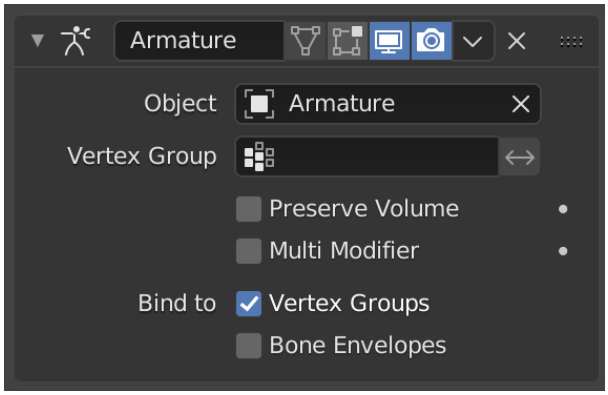
The *Armature* modifier is used for building skeletal systems (rigs) for animating the poses of characters and anything else which needs to be posed.

By adding an armature system to an object, that object can be deformed accurately so that geometry does not have to be animated by hand.

See also

For more details on armatures usage, see the armature section.

Options



The Armature modifier.

Object
: The name of the armature object used by this modifier.

Vertex Group
: A vertex group of the object, which weights will be used to determine the influence of this modifier's results when mixing it with the results from other *Armature* ones.

  Only meaningful when having at least two of these modifiers on the same object, with *Multi Modifier* activated.

  Invert <–>

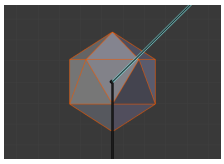
Inverts the influence set by the vertex group defined in previous setting (i.e. reverses the weight values of this group).

Preserve Volume

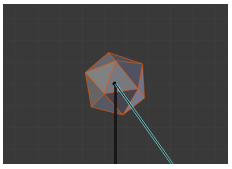
Use quaternions for preserving volume of object during deformation. It can be better in many situations.

Without it, rotations at joints tend to scale down the neighboring geometry, up to nearly zero at 180 degrees from rest position. With it, the geometry is no longer scaled down, but there is a “gap”, a discontinuity when reaching 180 degrees from rest position.

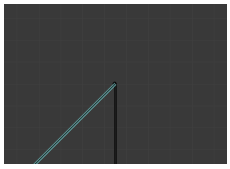
Example of *Preserve Volume* effects. Note that the icosphere is deformed using the envelopes weights.



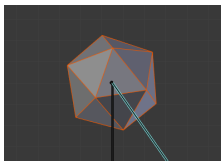
Initial state.



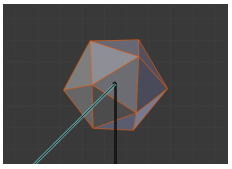
100° rotation, *Preserve Volume* disabled.



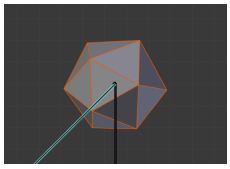
180° rotation, *Preserve Volume* disabled.



100° rotation, *Preserve Volume* enabled.



179.9° rotation, *Preserve Volume* enabled.



180.1° rotation, *Preserve Volume* enabled.

Multi Modifier

Use the same data as a previous modifier (usually also an *Armature* one) as input. This allows you to use several armatures to deform the same object, all based on the "non-deformed" data (i.e. this avoids having the second *Armature* modifier deform the result of the first one…).

The results of the *Armature* modifiers are then mixed together, using the weights of the *Vertex Group* as "mixing guides".

Tip

Armature modifiers can quickly be added to objects by parenting them to an armature.

Bind to

Methods to bind the armature to the mesh.

Vertex Groups

Meshes and lattices only. When enabled, bones of a given name will deform vertices which belong to vertex groups of the same name. E.g. a bone named "forearm", will only affect the vertices in the "forearm" vertex group.

The influence of one bone on a given vertex is controlled by the weight of this vertex in the relevant group. A much more precise method than *Bone Envelopes*, but also generally longer to set up.

Bone Envelopes

When enabled, bones will deform vertices or control points near them, defined by each bone's envelope radius and distance. This lets bone envelopes control the deformation (i.e. bones deform vertices in their neighborhood).

Example of skinning methods.

Cast Modifier

The *Cast* modifier shifts the shape of a mesh, curve, surface or lattice, towards any of a few predefined shapes (sphere, cylinder, cuboid).

It is equivalent to the To Sphere tool in Edit Mode, and what other programs call “Spherify” or “Spherize”, but, as written above, it is not limited to casting to a sphere.

Tip

The Smooth Modifier is a good companion to *Cast*, since the cast shape sometimes needs smoothing to look nicer or even to fix shading artifacts.

Note

For performance reasons, this modifier only works with local coordinates. If the modified object looks wrong, you may need to apply its transformations, especially when casting to a cylinder.

Options

Shape
: Menu to choose target shape of the projection: *Sphere*, *Cylinder* or *Cuboid*.

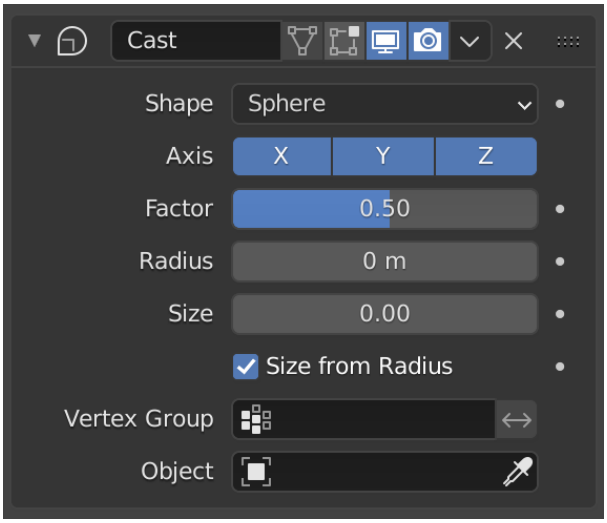
Axis
: Toggle buttons to enable/disable the modifier in the X, Y, Z axes directions (X and Y only for *Cylinder* cast type, since the Z axis remains unaffected).

Factor

The factor to control blending between original and cast vertex positions.

It is a linear interpolation: 0.0 gives original coordinates (i.e. modifier has no effect), 1.0 casts to the target shape.

Values below 0.0 or above 1.0 exaggerate the deformation, sometimes in interesting ways.



The Cast modifier.

Radius
: If nonzero, this radius defines a sphere of influence. Vertices outside it are not affected by the modifier.

Size
: Alternative size for the projected shape. If zero, it is defined by the initial shape and the control object, if any.

Size from Radius
: If activated, calculate *Size* from *Radius*, for smoother results.

Vertex Group
: If set, restrict the effect to the only vertices in that vertex group. This allows selective, real-time casting, by painting vertex weights.

  Invert <–>
  : Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

    The setting reverses the weight values of the group.

Object

The name of an object to control the effect. The location of this object's origin defines the center of the projection. Also, its size and rotation transform the projected vertices.

Hint

Animating (keyframing) this control object also animates the modified object's casting deformation.

Example



Top: Suzanne without modifiers. Middle: Suzanne with each type of Cast Modifier (Sphere, Cylinder and Cuboid). Bottom: Same as above, but now only X axis is enabled. Sample blend-file.

Curve Modifier

The *Curve* modifier provides a simple but efficient method of deforming a mesh along a curve object.

It works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction (by default, the X axis), the mesh will traverse along the curve, as if it was a train following and deforming along rails. Moving the mesh perpendicularly to this axis, the object will move closer or further away from the curve.

When you move the object beyond the curve's ends, the object will continue to deform based on the direction vector at those ends.

Note

This modifier works in global space, in other words, the actual position of the geometry relative to the curve is determinant to get a correct result.

Typically, you'll want your object's origin to be at the center of your geometry (not offset far away from it, you can e.g. Set Origin to Geometry).

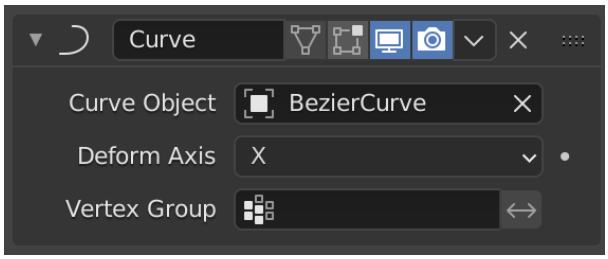
And then you'll want to start with your object's origin at the same location as your curve object's origin (you may use snap tools for that…).

If the curve is 3D, the *Tilt* value of its control points will be used to twist the deformed object. And the *Radius* property controls the size of the object as well. Those options are in the *Shape* panel, under Path/Curve-Deform.

Options

Curve Object

The name of the curve object that will affect the deformed object.



The Curve modifier.

Deformation Axis

This is the axis that the curve deforms along.

X/Y/Z/-X/-Y/-Z

Vertex Group

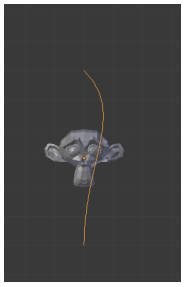
If set, restrict the effect to the only vertices in that vertex group.

Invert <–>

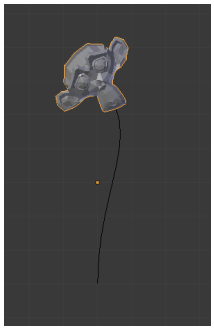
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

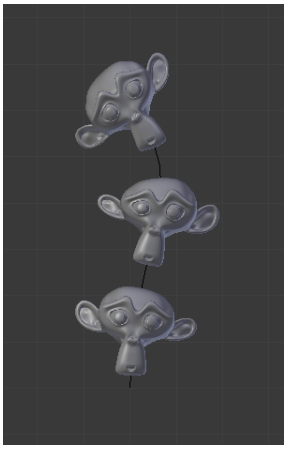
Example



Edit curve.



Monkey on a curve.



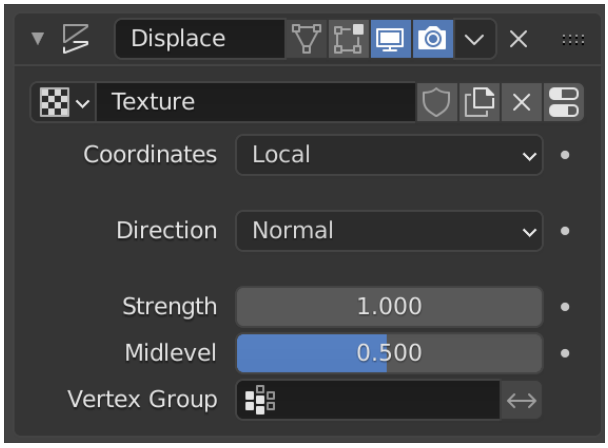
Monkey deformations.

Displace Modifier

The *Displace* modifier displaces vertices in a mesh based on the intensity of a texture. Either procedural or image textures can be used.

The displacement can be along a particular local axis, along the vertex normal, or the separate RGB components of the texture can be used to displace vertices in the local X, Y and Z directions simultaneously (sometimes referred to as *Vector Displacement*).

Options



The Displace modifier.

Texture
: The name of the texture from which the displacement for each vertex is derived. If this field is empty, the modifier defaults to 1.0 (white).

  Coordinates
  : The texture coordinate system to use when retrieving values from the texture for each vertex.

    See common masking options for a complete reference.

Direction
: The direction along which to displace the vertices. Can be one of the following:

  X, Y, Z
  : Displace along an axis.

Normal
: Displace along the vertex normal.

Custom Normal
: Displace along (averaged) custom normals, instead of vertex normals.

RGB to XYZ
: Displace along local XYZ axes individually using the RGB components of the texture (Red values displaced along the X axis, Green along the Y, Blue along the Z).

Space
: With a direction set to X, Y, Z, or XYZ the modifier can either displace along local or global axes.

Strength
: The strength of the displacement. After offsetting by the *Midlevel* value, the displacement will be multiplied by the *Strength* value to give the final vertex offset.

  $vertex_offset = displacement \times Strength$

  A negative strength can be used to invert the effect of the modifier.

Midlevel
: The texture value which will be treated as no displacement by the modifier. Texture values below this threshold will result in negative displacement along the selected direction, while texture values above it will result in positive displacement.

  $displacement = texture_value - Midlevel$

  Recall that color/luminosity values are typically between (0.0 to 1.0) in Blender, and not between (0 to 255).

Vertex Group

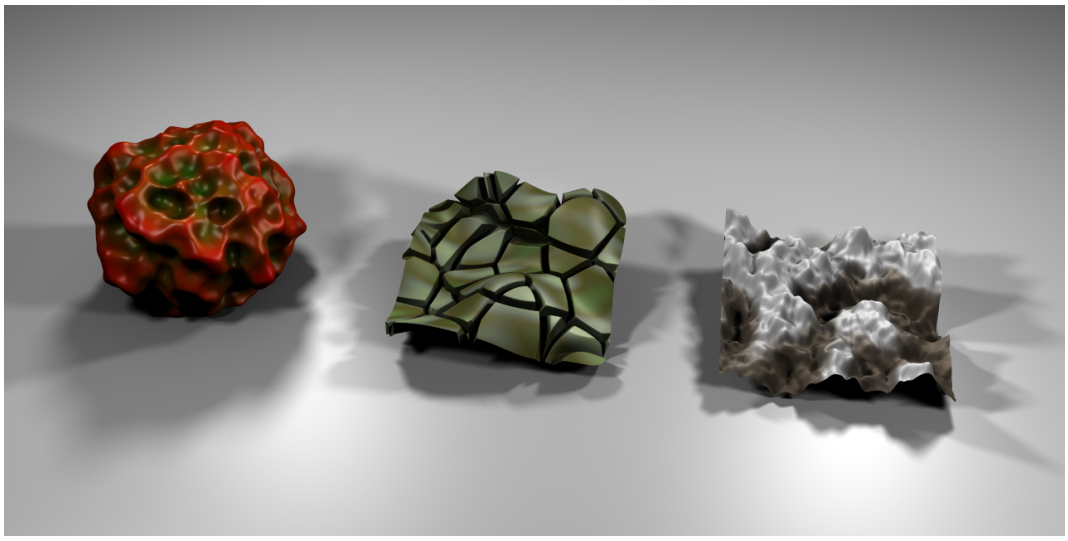
The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.

Invert <–>

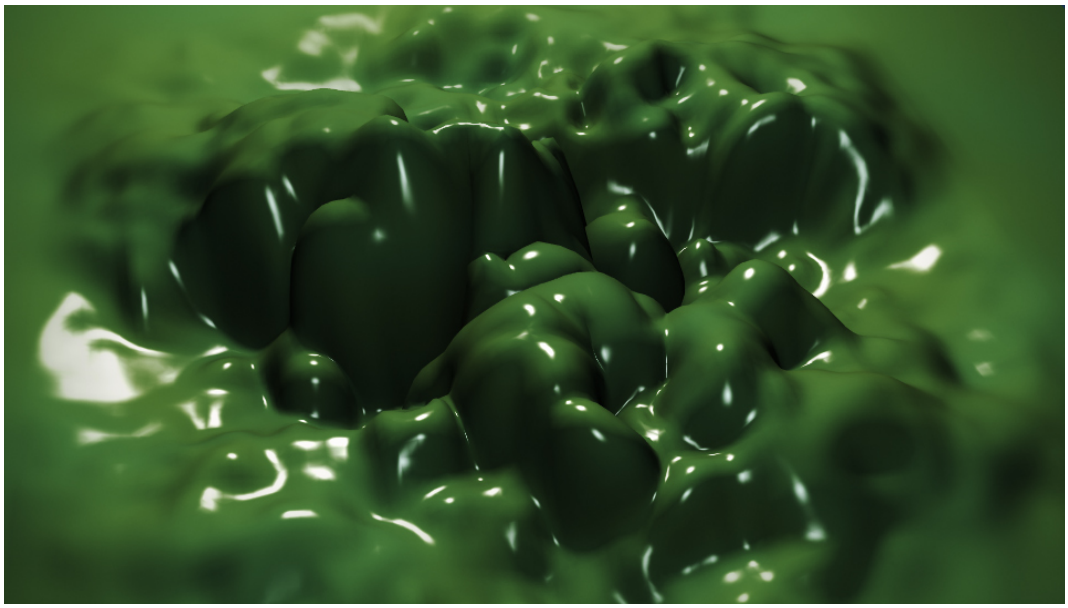
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Example



Three different objects created with the Displace modifier.
Sample blend-file.



A slime animation created with the Displace modifier. Sample blend-file.

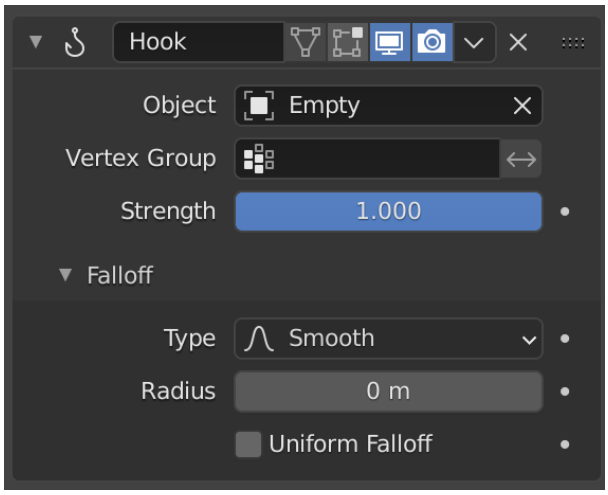
Hook Modifier

The *Hook* modifier is used to deform a mesh, curve or lattice using another object (usually an empty or a bone but it can be any object). As an object specified as hook moves, it pulls vertices or control points from the geometry with it. You can think of it as animated Proportional Editing.

While hooks do not give you the fine control over vertices movement that shape keys do, they have the advantage that you can select vertices directly for manipulation.

To assign selected vertices (in Edit Mode) you can use the *Assign* button on the modifier panel or use the Add Hook menu.

Options



The Hook modifier.

Object
: The name of the object to hook vertices to.

Vertex Group
: Allows you to define the influence per vertex.

  Useful when you want something other than a spherical field of influence.

  Invert <–>
  : Inverts the influence of the selected vertex group,
  meaning that the group now represents vertices that will not be deformed by the modifier.

    The setting reverses the weight values of the group.

Strength
: Adjust this hooks influence on the vertices, were (0.0 to 1.0) (no change to fully follows the hook).

  Since multiple hooks can work on the same vertices, you can weight the influence of a hook using this property.

The following settings are only available in Edit Mode:

Reset
: Recalculate and clear the offset transform of the hook.

Recenter
: Set the hook center to the 3D cursor position.

Select
: Select the vertices affected by this hook.

Assign
: Assigns selected vertices to this hook.

Warning

The Hook Modifier stores vertex indices from the original mesh to determine what to affect. This means that modifiers that generate geometry, like *Subdivision Surface*, should always be put **after** the *Hook* one in the stack. Otherwise, the generated geometry will be left out of the hook's influence.

Falloff

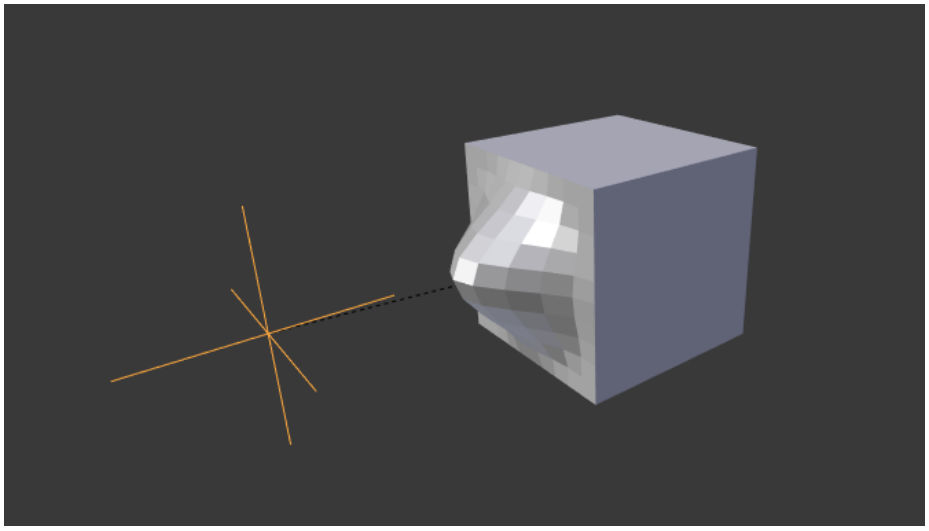
Type
: This can be used to adjust the kind of influence curve that the hook has on the mesh. You can also define a custom curve to get a much higher level of control.

Radius
: The size of the hooks influence.

Uniform Falloff
: This setting is useful when using hooks on scaled objects, especially in cases where non-uniform scale would stretch the result of the hook.

  This is especially useful for lattices, where it is common to use non-uniform scaling.

Example



Empty used as Hook to deform a subdivided cube.

Laplacian Deform Modifier

The *Laplacian Deform* modifier allows you to pose a mesh while preserving geometric details of the surface.

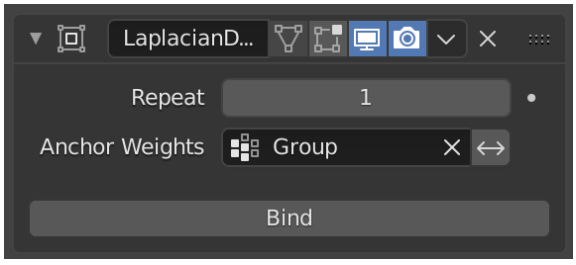
The user defines a set of "anchor" vertices, and then moves some of them around. The modifier keeps the rest of the anchor vertices in fixed positions and calculates the optimal locations of all the remaining vertices to preserve the original geometric details.

This modifier captures the geometric details with the use of differential coordinates. The differential coordinates capture the local geometric information, the curvature and direction of a vertex based on its neighbors.

Note

You must define an *Anchors Vertex Group*. Without it the modifier does nothing.

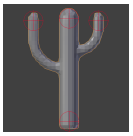
Options



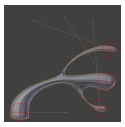
The Laplacian Deform modifier.

Repeat
: How many iterations to do to improve the found solution. The objective is to find the rotation of the differential coordinates preserving the best possible geometric details. Details are retained better if more iterations are used, however, it will take longer to calculate.

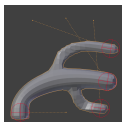
Deform horse example blend-file.



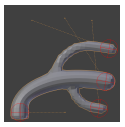
Original Model.



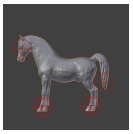
Repeat: 1.



Repeat: 2.



Repeat: 5.



Original Model.



Repeat: 1.



Repeat: 2.



Repeat: 10.

Anchor Weights

The group of vertices that the user will use to transform the model. The weight of each vertex does not affect the behavior of the modifier, the method only takes into account vertices with weight greater than 0.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Bind

The *Bind* button is what tells the *Laplacian Deform* modifier to actually capture the geometry details of the object, so that altering the anchor vertices actually alters the shape of the deformed object.

Unbind

After binding the modifier, you may later decide to make changes to the *Anchors Vertex Group*. To do so you will first need to *Unbind* the modifier before binding it again.

Error Messages

Vertex group *group_name* is not valid

This message is displayed when a user deletes the vertex group or changes its name.

Vertices changed from X to Y
: This message is displayed when a user adds or deletes vertices to/from the mesh.

Edges changed from X to Y
: This message is displayed when a user adds or deletes edges to/from the mesh.

The system did not find a solution
: This message is displayed if the solver could not find a solution for the linear system.

Note

If the mesh is dense, with a number of vertices greater than 100,000, then it is possible that the nonlinear optimization system will fail.

History

Laplacian Surface Editing is a method developed by Olga Sorkine and others in 2004. This method preserves geometric details as much as possible while the user makes editing operations. This method uses differential coordinates corresponding to the difference between a vector and the weighted average of its neighbors to represent the local geometric detail of the mesh.

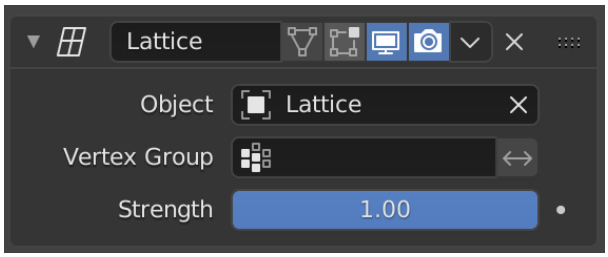
Lattice Modifier

The *Lattice* modifier deforms the base object according to the shape of a Lattice object. Objects to be deformed can be meshes, curves, surfaces, text, lattices and even particles.

Tip

A *Lattice* modifier can quickly be added to selected objects by parenting them using the *Lattice Deform* option.

Options



The Lattice modifier.

Object
: The Lattice object with which to deform the base object.

Vertex Group
: An optional vertex group name which lets you limit the modifier's effect to a part of the base mesh.

  Invert <–>
  : Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

    The setting reverses the weight values of the group.

Strength
: A factor to control blending between original and deformed vertex positions.

Hints

Why would you use a lattice to deform a mesh instead of deforming the mesh itself in Edit Mode? There are a couple of reasons for that:

- If your object has a large number of vertices, it would be difficult to edit portions of it quickly in Edit Mode. Using a lattice will allow you to deform large portions efficiently.
- The smooth deformation you get from a *Lattice* modifier can be hard to achieve manually.
- Multiple objects can use the same lattice, thus allowing you to edit multiple objects at once.
- Like all modifiers, it is non-destructive. Meaning all changes happen on top of the original geometry, which you can still go back to and edit without affecting the deformation.
- A lattice does not affect the texture coordinates of a mesh's surface.

Note

When using a lattice to deform particles, order in the modifier stack matters. You need to place the *Lattice* modifier after the *Particle System* one.

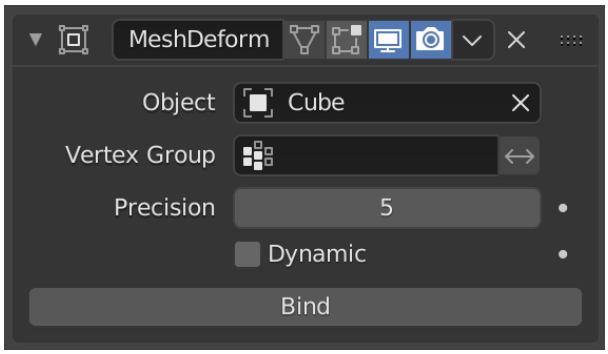
Mesh Deform Modifier

The *Mesh Deform* modifier allows an arbitrary mesh (of any closed shape) to act as a deformation cage around another mesh.

Note

This modifier is reasonably easy to use, but it can be very slow to compute the binding (the mapping between the deform mesh cage to the deformed object geometry).

Options



The Mesh Deform modifier.

Object
: The name of the mesh object to be used as the deforming cage.

Vertex Group
: An optional vertex group of the object's mesh to restrict the vertices that will be affected by this modifier. Vertices not in this group will not be deformed.

  Invert <–>
  : Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

    The setting reverses the weight values of the group.

Precision

Controls the accuracy with which the deform mesh cage alters the deformed object, when the points on the cage are moved. Raising this value higher can greatly increase the time it takes to complete the binding calculations, but it will get more accurate cage mapping to the deformed object.

This setting becomes unavailable once a cage has been bound.

Dynamic

When activated, other mesh altering features (such as other modifiers and shape keys) are taken into account when binding, increasing deformation quality.

The setting is deactivated by default to save memory and processing time when binding. Like with *Precision*, this setting is unavailable once a cage has been bound.

Bind

Links the current vertex positions of both the modified geometry and the deforming *Object* chosen together. An unbound *Mesh Deform* modifier has no effect, it must be bound so that altering the shape of the deform mesh cage actually alters the shape of the modified object.

Warning

Depending on the settings of the modifier and complexity of the deform mesh cage and/or deformed object, it can take a long time for this operation to complete. This can result in Blender not responding to user's actions until it has completed.

It is also possible that Blender will run out of memory and crash.

To be safe, save your blend-file before proceeding!

Unbind

When a deformed object has been associated to a deform mesh cage, it can later be disassociated by clicking the *Unbind* button which replaced

the *Bind* one.

When *Unbind* is clicked, the *deforming mesh cage* will keep its current shape, it will not reset itself back to its initial shape. If you need this original shape, you will have to save a copy of it before you alter it.

The deformed object will, however, reset back to its original shape that it had before it was bound to the deform mesh cage.

Warning

Significant changes to the entire cage mesh *(such as rotating the cage upside down)* can cause noticeable artifacts.

These can be reduced by binding with a higher *Precision*, however, it is a known limitation with this modifier and cannot be avoided entirely.

Hints

- Ensure that the normals on the cage mesh point to the outside (they are used to determine the inside and outside of the cage).
- Besides the outer cage, more faces within the cage, either loose or forming another smaller cage, can be used for extra control. Such smaller cages may also overlap with the main cage. For example, to get extra control over eyes, two small sphere cages could be added around them.

See also

- The Lattice modifier.
- Original paper

Shrinkwrap Modifier

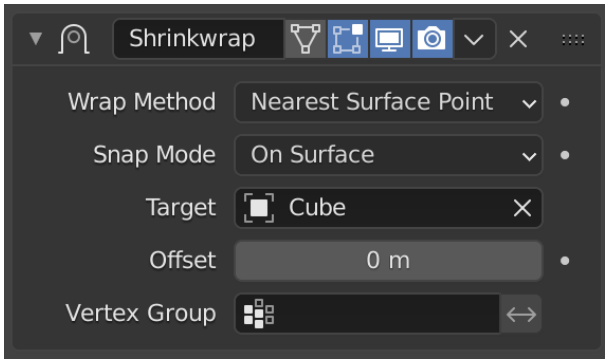
The *Shrinkwrap* modifier allows an object to “shrink” to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh (using one of the four methods available).

It can be applied to meshes, lattices, curves, surfaces and texts.

See also

Shrinkwrap Constraint.

Options



The Shrinkwrap modifier in Nearest Surface Point mode.

Wrap Method
: This selector specifies the method to be used to determine the nearest point on the target’s surface for each vertex of the modified object. Some options will add some extra, specific controls to the panel. See Wrap Methods for an explanation of each method.

Snap Mode

Most modes support an additional setting to control how the vertex is moved to the target point selected by the methods described above. Some of the choices only differ if *Offset* is not zero.

On Surface
: The vertex is always moved. The offset is applied along the projection line connecting the original vertex and selected target point towards the original position.

Outside Surface
: Like *On Surface*, but the offset is always applied towards the outside of the target.

Above Surface
: Like *On Surface*, but the offset is applied along the smooth normal of the target.

Inside
: The vertex is not moved if it is already inside the target. Offset shrinks the allowed volume towards the inside along the projection line.

Outside
: The vertex is not moved if it is already outside the target. Offset expands the exclusion volume towards the outside along the projection line.

Note

The *Inside* and *Outside* options can be used for very crude collision detection. The inside vs outside determination is done based on the target normal and is not always stable near 90 degree and sharper angles in the target mesh.

Target
: Shrink target, the mesh to shrink to/wrap around.

Offset

The distance that must be kept from the calculated target position.

Vertex Group
: The vertex group to control whether and how much each vertex is displaced to its target position. If a vertex is not a member of this group, it is not displaced (same as weight 0).

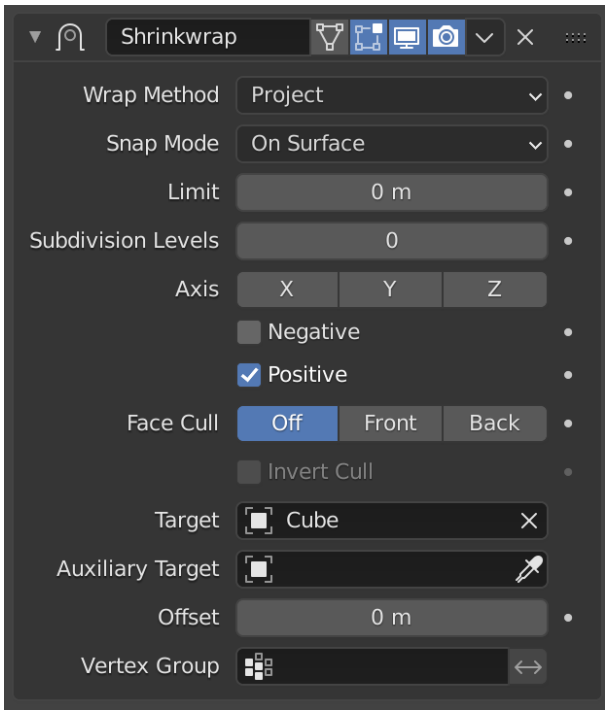
Wrap Methods

Nearest Surface Point

This will select the nearest point over the surface of the shrunk target.

Project

This will project vertices along a chosen axis until they touch the shrink target. Vertices that never touch the shrink target are left in their original position.



Project mode.

Limit
: This is a distance limit between original vertex and surface. If the distance is larger than this limit vertex would not be projected onto the surface.

Subdivision Levels
: This applies a (temporary) *Catmull-Clark* subdivision to the modified object's geometry, before computing the wrap.

Axis
: Along which local axis of the modified object the projection is done. These options can be combined with each other, yielding a "median

axis" of projection. If none are selected, the normal direction is used.

Negative/Positive
: This allows you to select the allowed direction(s) of the shrink along the selected axis. If both options are enabled, both ways are evaluated and the closest hit is selected.

Face Cull
: Allows you to prevent any projection over the "front side" (respectively the "back side") of the target's faces. The "side" of a face is determined by its normal (front being the side "from where" the normal "originates").

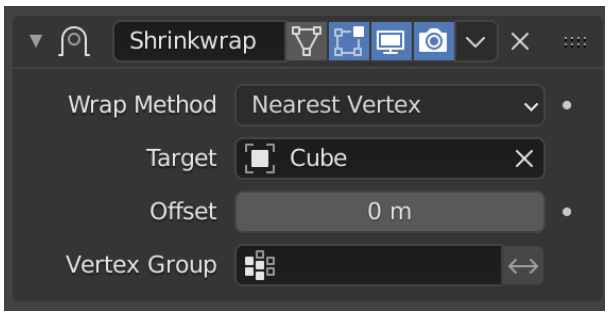
Invert Cull
: If *Cull Faces* is enabled, and *Negative* direction along axis is allowed, this option can be used to invert the *Front* or *Back* cull choice for the *Negative* direction. This is useful when projecting in both directions.

Auxiliary Target
: An additional object to project over.

Nearest Vertex

This will snap vertices to the nearest vertex of the shrunk target. It adds no extra options.

This method doesn't support the *Snap Mode* setting.

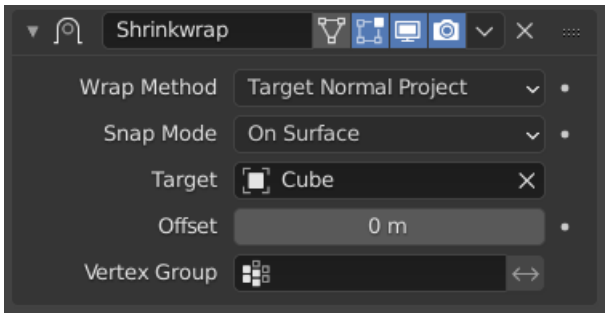


Nearest Vertex mode.

Target Normal Project

This mode is similar to *Nearest Surface Point*, but produces a much smoother projection in return for being significantly slower.

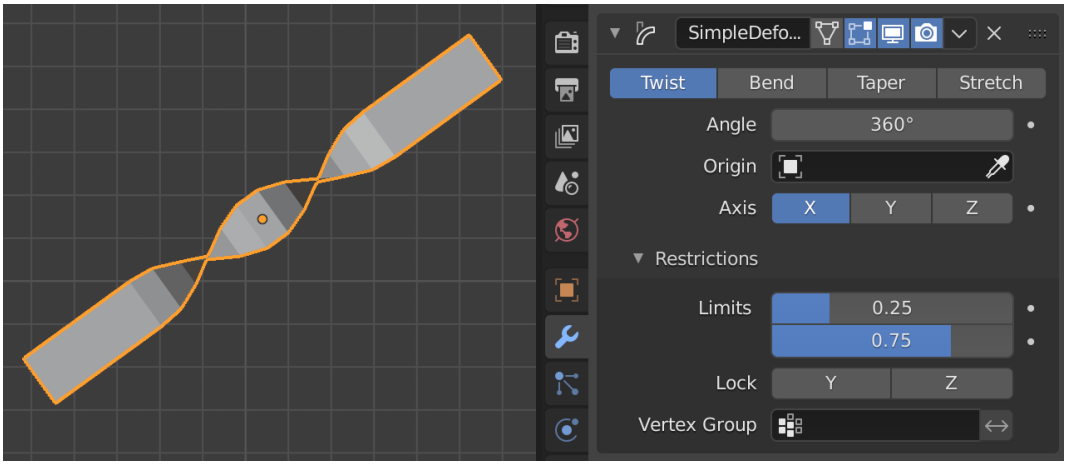
Instead of finding the closest point, it searches for the nearest point that has its interpolated smooth normal pointing towards or away from the original vertex position. Non-manifold boundary edges are specially handled as infinitely thin cylinders that emit normals in all perpendicular directions; ignores flat shading.



Target Normal Project mode.

Simple Deform Modifier

The *Simple Deform* modifier allows the application of a simple deformation to an object. Meshes, lattices, curves, surfaces and texts are supported objects. The deformation is either a rotation (Twist, Bend) or a scaling (Taper, Stretch). The amount of deformation is specified by the Deform Angle (rotation) or Deform Factor (scaling).



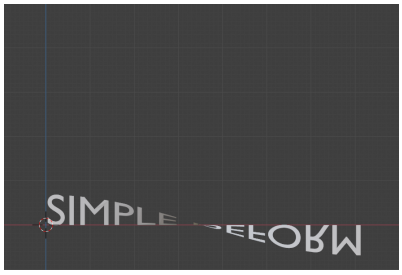
The Simple Deform modifier.

The deformation is calculated in the *local* coordinate space. Be aware that the local axes of an object can differ from the global ones. In the figure above, the global Z axis points up and the local Z axis points at 45°. The deformation is applied along a Deform axis, which can be set by selection from a list (X, Y or Z). By using the *Limits* field, the influence of the modifier can be restricted to a subset of the Deform axis. All distances are measured from the origin of the object. The vertices that are furthest away from the origin on the Deform axis represent the upper and lower limits. The origin of the object and the orientation of the local axes can be defined by an external Deform object (most of the time, an empty).

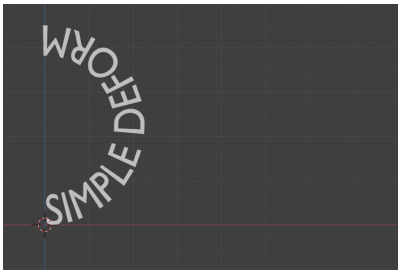
Options

Mode

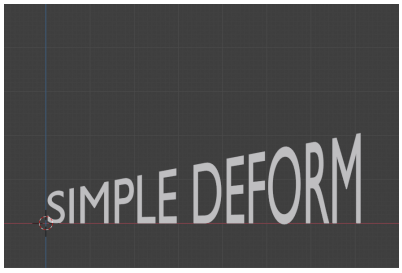
Defines the kind of deformation which will be applied. The figure below shows the four modes, applied to a text object. The origin of the object is at the very left of the text.



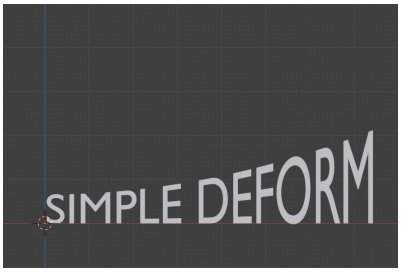
Twist around X axis (180°).



Bend around Z axis (180°).



Taper along X axis (factor = 2).



Stretch along X axis (factor = 0.3).

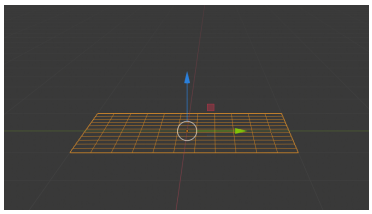
Twist

Rotates the mesh around the specified *Axis*. Each vertex along the Deform axis is rotated around the object's origin. If the origin is inside the object, this results in a twisted appearance. Below the origin, there is a negative rotation and above the origin, the rotation is positive or clockwise. Vertices in the same plane as the origin are not rotated.

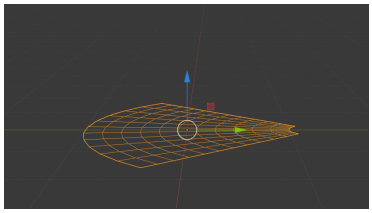
The total amount of rotation is specified by the angle and the rotation at each vertex is thus weighted by the distance of the vertex to the origin of the object. Vertices that are furthest away from the object origin have max rotation, positive or negative.

Bend

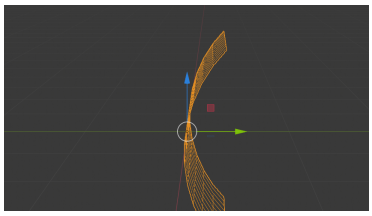
Bends the mesh over the specified *Axis*. The Bend mode is more complex and less intuitive. The picture below shows the same plane but with different Deform and Bending axes.



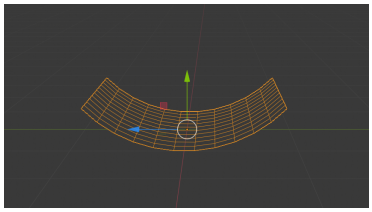
Deform axis X or Y (a).



Deform axis Z (b).



Deform axis X. Local axis Y points down (c).



Deform axis X. Local axis Y points up (d).

Using a mesh plane and setting the Deform axis to X or Y will not result in any deformation (Fig a). You expect that the bending should be something like Figure (c) or (d). In a 3D world however, selecting the X axis introduces an ambiguity because bending along the X axis could result in Figure (c) or (d). The following pairs describe the selected deform axis vs. the desired bending axis: X and Z, Y and Z, Z and X.

In Fig (a), because of the Deform axis X, the Bending is along the Z. All vertices however have the same Z coordinate equal to the local origin. So, no deformation occurs. In Fig (d) the local axes are rotated around the Deform axis X, so that the Bending axis Z points to the left. So, all vertices are bend for their Z coordinate. The further away from the local origin, the more bending.

This explains also the unexpected result of Fig (b). The Deform axis is set here to Z (pointing up). So, according to the pairs above, the Bending axis defaults to X. All vertices of the plane are bent in their X coordinate. The further away, the more rotation occurs. Negative X coordinates are rotated counterclockwise.

Taper

Linearly scales along the specified *Axis*. The scaling factor is weighted by the distance from the origin of the object in the deform axis. No scaling occurs in the plane of the origin of the object. The maximum scaling occurs at the vertices that are furthest away from the local origin. This can be a positive or negative scaling, depending on the location of the origin. If the local origin is within the object, the deformed object appears tapered.

Stretch

Stretches the object along the specified *Axis*. If the local origin is within the object, the deformed mesh looks stretched like pulling a rubber from both sides. With a positive factor, the mesh gets longer in the deformed axis, wider at the borders and thinner at the origin than the original mesh. If the factor is negative, then the mesh is squashed in the deformed axis, thicker at the origin and thinner at the borders.

Angle (Twist & Bend)/Factor (Taper & Stretch)

The total amount of deformation. Can be negative to reverse the deformation.

Axis, Origin

The name of an object that defines the origin and axis of deformation (usually an empty). This object can be:

- Rotated to control the axis (its local *Axis* is now used as the deformation one).
- Moved to control the origin of the deformation.
- Scaled to change the deformation factor.

Restrictions

Limits

You can set the lower and upper limits of the deformation. The upper limit cannot be lower than the lower one. These limits are mapped on the Deform axis.

Lock (Twist, Taper and Stretch modes only)

These controls whether the coordinates along the two other axes are allowed to change or not. E.g. if you *Stretch* your object along its Z axis, it is possible to squash along the X axis only, by locking the Y one.

Vertex Group

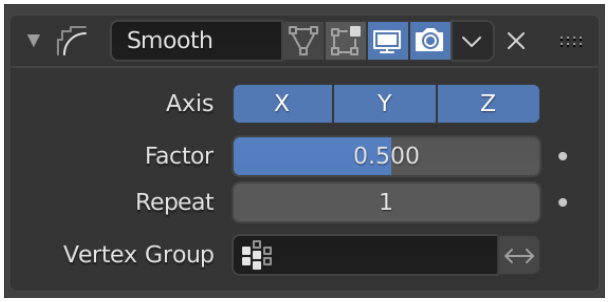
The name of the vertex group that indicates whether and how much each vertex is influenced by the deformation. The amount of influence is determined by the weight in the Weight Paint map.

Smooth Modifier

The *Smooth* modifier smooths a mesh by flattening the angles between adjacent faces in it, just like the Smooth tool in Edit Mode. It smooths without subdividing the mesh, the number of vertices remains the same.

This modifier is not limited to smoothing, though. Its control factor can be configured outside the (0.0 to 1.0) range (including negative values), which can result in interesting deformations.

Options



The Smooth modifier.

Axis
: Enable/disable the modifier in the X, Y and/or Z axes directions.

Factor
: Controls the smoothing amount. Higher values will increase the effect.

  Values outside expected range (above 1.0 or below 0.0) will distort the mesh.

Repeat
: The number of smoothing iterations, equivalent to executing the *Smooth* tool multiple times.

Vertex Group
: If set, restrict the effect to the only vertices in that vertex group. This allows for selective, real-time smoothing, by painting vertex weights.

  Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

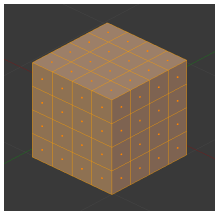
The setting reverses the weight values of the group.

Algorithm

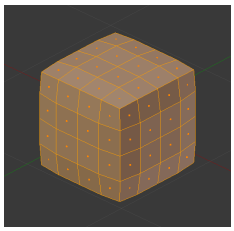
The calculation done by the *Smooth* modifier is a simple and logical one, and can be thought of as the geometric equivalent of blurring images.

Each new vertex position is simply moved towards the average position of all its neighbor vertices (topologically speaking, i.e. the vertices directly connected to it by an edge).

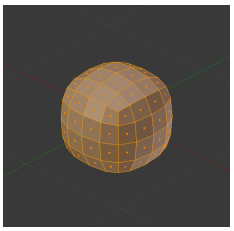
Example



Mesh before smoothing.



Mesh after one smoothing iteration.



Mesh after ten smoothing iterations.

Smooth Corrective Modifier

The *Smooth Corrective* modifier is used to reduce highly distorted areas of a mesh by smoothing the deformations.

This is typically useful *after* an Armature modifier, where distortion around joints may be hard to avoid, even with careful weight painting.

To use this modifier effectively, it is important to understand the basics of how it works.

Rest State
: Used as a reference to detect highly distorted areas. The original vertex locations are used by default.

Smoothing
: Many options for this modifier relate to smoothing which is used internally to correct the distorted regions.

Options

Factor
: The factor to control the smoothing amount. Higher values will increase the effect.

  Values outside expected range (above 1.0 or below 0.0) will distort the mesh.

Repeat
: The number of smoothing iterations, equivalent to executing the *Smooth* tool multiple times.

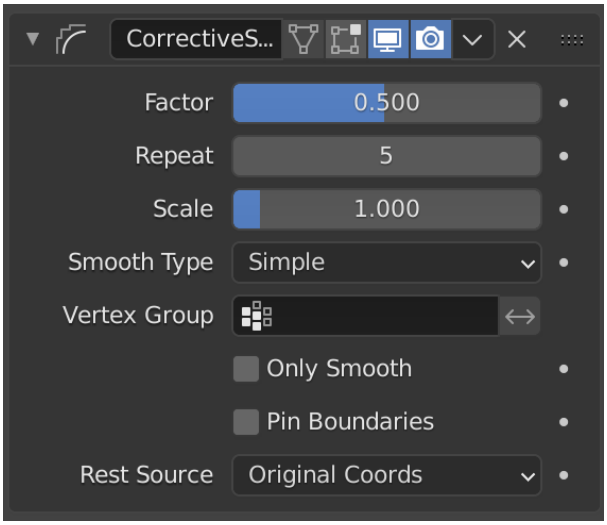
Scale
: Additional scaling factor to increase the size of the mesh. This is useful because sometimes the Smooth Corrective modifier will introduce

volume loss, especially when used with a rig.

Smooth Type
: Select the smoothing method used.

  Simple
  : This simply relaxes vertices to their connected edges.

  Length Weight
  : Uses a method of relaxing that weights by the distance of surrounding vertices. This option can give higher quality smoothing in some cases, by better preserving the shape of the original form.



The Smooth Corrective modifier.

Vertex Group
: If set, restrict the effect to the only vertices in that vertex group. This allows for selective, real-time smoothing, by painting vertex weights.

Only Smooth
: This option is included to preview the smoothing used, before correction is applied.

Pin Boundaries
: Prevent boundary vertices from smoothing.

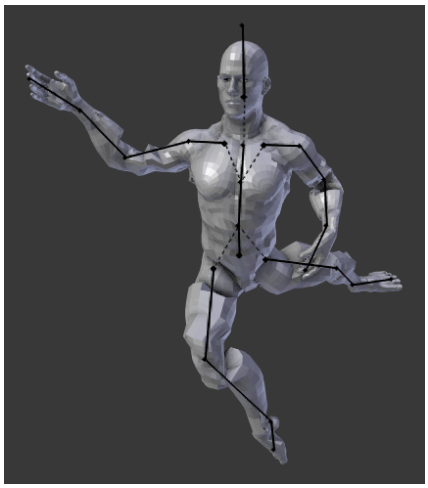
Rest Source
: Select the source for reference vertex positions that defines the undeformed state.

  Original Coordinates
  : Use the original input vertex positions. This relies on the original mesh having the same number of vertices as the input, modified mesh.

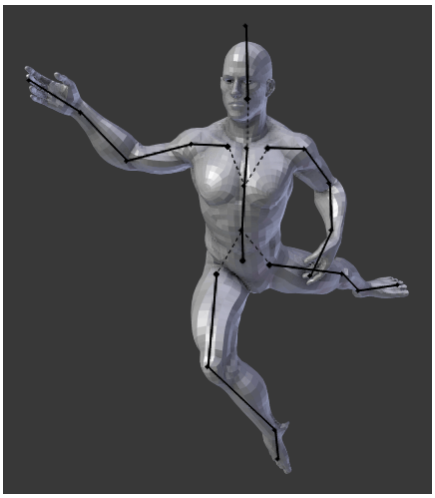
Bind Coordinates
: Optionally you may bind the modifier to a specific state. This is required when there are constructive modifiers such as *Subdivision Surface* or *Mirror* in the stack before this modifier.

Example

An example of a rig using bone envelopes and not weight painting.



Armature only.



Armature and Corrective Smooth.

Smooth Laplacian Modifier

The *Smooth Laplacian* modifier allows you to reduce noise on a mesh's surface with minimal changes to its shape.

It can also exaggerate the shape using a negative *Factor*.

The *Smooth Laplacian* is useful for objects that have been reconstructed from the real world and contain undesirable noise. It removes noise while still preserving desirable geometry as well as the shape of the original model.

This modifier is based on a curvature flow *Laplace Beltrami* operator in a diffusion equation.

Hint

Meshes with a great number of vertices, more than ten thousand (10,000), may take several minutes for processing. You can use small portions of the mesh for testing before executing the modifier on the entire model.

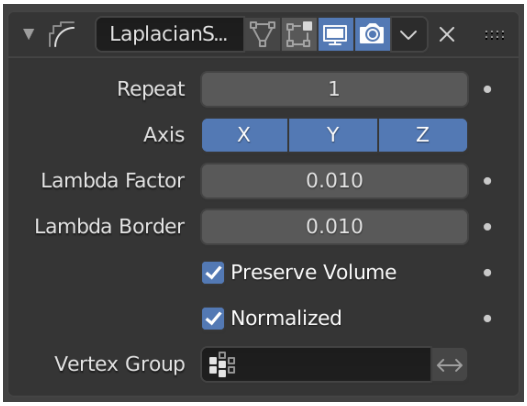
Options

Repeat
: Repetitions allow you to run the smoothing operation multiple times. Each repetition causes the flow curvature of the mesh to be recalculated again, and as a result it removes more noise with every new iteration using a small *Factor* < 1.0.

  When on 0, no smoothing is done.

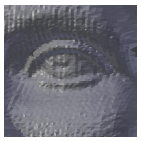
  Note

  More repetitions will take longer to calculate. So beware of doing so on meshes with a large number of vertices.



The Smooth Laplacian modifier.

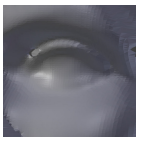
With a factor of 0.5.



Repeat:
0.



Repeat:
1.



Repeat:
5.



Repeat:
10.

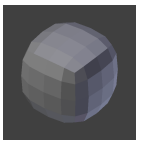
With a factor of 2.0.



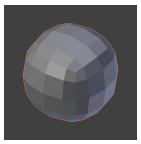
Repeat:
0.



Repeat:
1.



Repeat:
5.



Repeat:
10.

With a factor of -0.5.



Repeat:
0.



Repeat:
1.



Repeat:
5.



Repeat:
10.

Axis

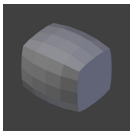
Toggle buttons to enable/disable deforming vertices in the X, Y and/or Z axes directions.



X, Y, Z:
Unselected.



X, Y, Z:
Selected.



X, Z:
Selected.

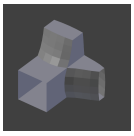


X:
Selected.







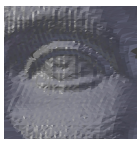


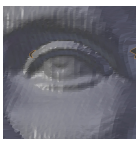
X, Y, Z: Unselected.	X, Y, Z: Selected.	X, Z: Selected.	X: Selected.

Lambda Factor

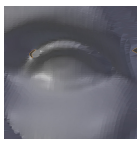
Controls the amount of displacement of every vertex along the flow curvature.

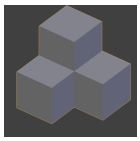
- Using a small *Factor*, you can remove noise from the shape without affecting desirable geometry.
- Using a large *Factor*, you get smoothed versions of the shape at the cost of fine geometry details.
- Using a negative *Factor*, you can enhance the shape, preserving desirable geometry.
- When the *Factor* is negative, multiple iterations can magnify the noise.


Factor: 0.0.


Factor: 0.5.


Factor: 2.5.


Factor: 5.0.


Factor: 0.0.


Factor: 1.0.


Factor: 10.0.


Factor: 50.0.


Factor: 0.0.


Factor: -20.0.


Factor: -50.0.


Factor: -300.0.

Lambda Border

Since there is no way to calculate the curvature flow on border edges, they must be controlled separately. Border edges are smoothed using a much simpler method, using this property to control the influence.

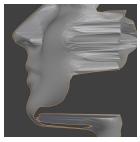
Positive values will smooth the vertex positions, while negative values will "enhance" them by transforming them in the opposite direction.

With a factor of 2.5.

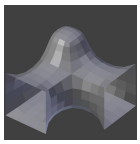

Border:
0.0.

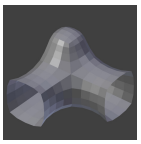

Border:
1.0.


Border:
2.5.


Border:
10.0.

With a factor of 20.0.


Border:
0.0.


Border:
1.0.


Border:
5.0.


Border:
20.0.

With a factor of -30.0.


Border:
0.0.

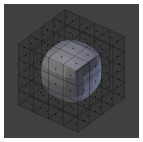

Border:
-20.0.

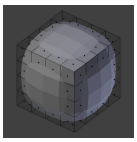

Border:
-50.0.

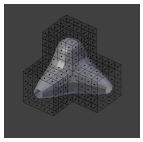

Border:
-200.0.

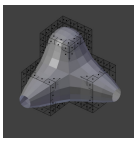
Preserve Volume

The smoothing process can produce shrinkage. That is significant for large *Factor* or large *Repeat* values. You can use that option to reduce that effect.


Off.


On.

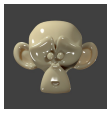

Off.


On.

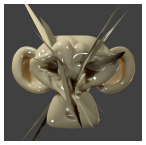
Normalized

When enabled, the results will depend on face sizes. When disabled, geometry spikes may occur.


Original Geometry.

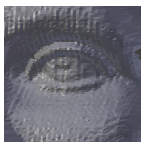
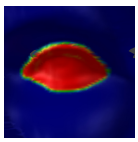
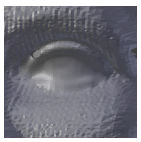
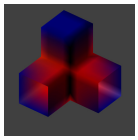
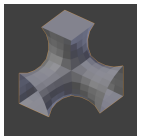

On.


Off.


Off, High Factor.

Vertex Group

A vertex group name, to constrain the effect to a group of vertices only. Allows for selective, real-time smoothing or enhancing, by painting vertex weights.

Original Geometry	**No Group Chosen**	**Vertex Weights**	**Result**
			
			
			

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

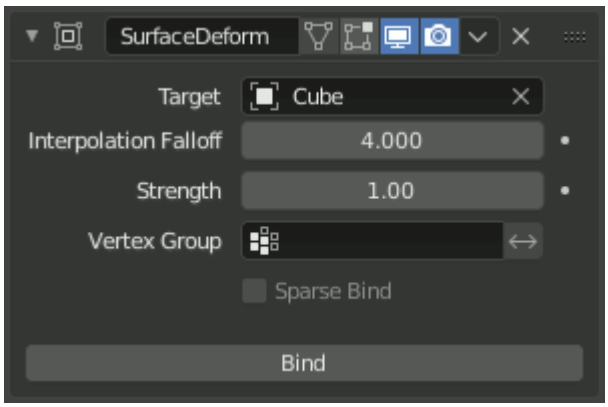
The setting reverses the weight values of the group.

Examples

Surface Deform Modifier

The *Surface Deform* modifier allows an arbitrary mesh surface to control the deformation of another, essentially transferring its motion/deformation. One great use for this is to have a proxy mesh for cloth simulation, which will in turn drive the motion of your final and more detailed mesh, which would otherwise not be suitable for simulation.

Options



The Surface Deform modifier.

Target

The object to which to bind (this setting is unavailable after binding).

Warning

Target Mesh Validity

While there are no restrictions with regard to the modified mesh, the target object's mesh has a few constraints, which if not followed, will prevent a successful binding:

- It must **not** contain edges with more than two faces.
- It must **not** contain concave faces.
- It must **not** contain overlapping vertices (doubles).
- It must **not** contain faces with collinear edges.

Interpolation Falloff

How much a vertex bound to one face of the target will be affected by the surrounding faces (this setting is unavailable after binding). This essentially controls how smooth the deformations are.

Note

While lower values result in smoother deformations, they may also introduce slight artifacts.

Strength

The overall amount of influence the modifier has on deforming the mesh.

Vertex Group

Allows you to define the influence per vertex.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Sparse Bind

Only record bind data for vertices that have nonzero weights in the group at the time of bind. This is an optimization, but adding new vertices to the group will require a rebind.

Bind

Bind the current state of the modified mesh to the current state of the target mesh, such that any later change in the target mesh will deform the modified mesh as well. Note that until the bind has been executed, this modifier will have no effect at all.

Unbind

Once the mesh is bound, the *Bind* button changes to *Unbind*. Executing this frees the modified mesh from the target, and resets it to its original shape.

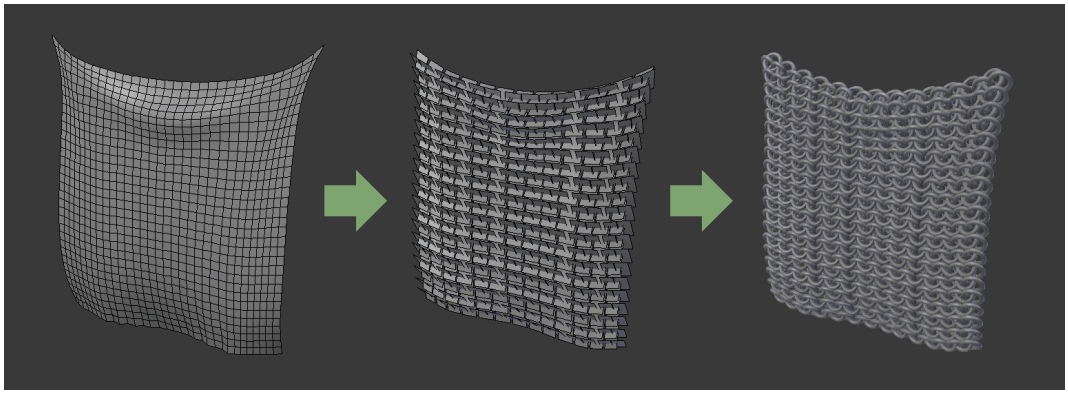
Note

The meshes are bound with regard to global coordinates, but later transformations on the objects are ignored. This means that one can freely transform the target or modified object after binding, without affecting the modified object. The modified mesh will only pick up changes to the target object's mesh itself.

Note

The further a mesh deviates from the target mesh surface, the more likely it is to get undesirable artifacts. This is an inherent characteristic of surface binding in general, so it is recommended to have reasonably well matching meshes, in order to get a good bind.

Example



Cloth simulation copied to an arbitrary mesh with rings as instancing faces.

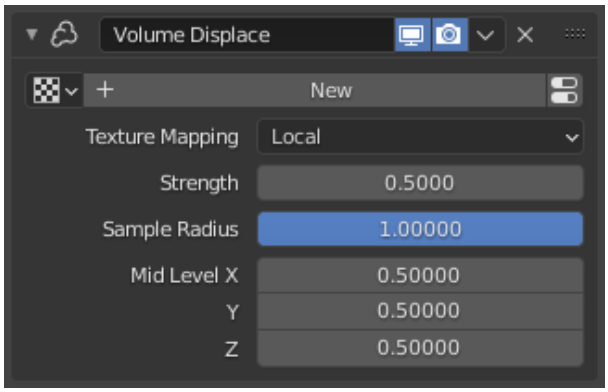
Volume Displace Modifier

Reference

This modifier is only available for Volume Objects.

The *Volume Displace* modifier displaces existing volume grids based on a 3D texture. It uses the RGB color channels of the texture to displace the volume into the X, Y and Z direction.

Options



The Volume Displace modifier.

Texture
: The texture that is evaluated at every voxel to determine how far and in what direction to displace.

  Note

  Grayscale textures lead to stretching along one axis. It's best to use a color texture.

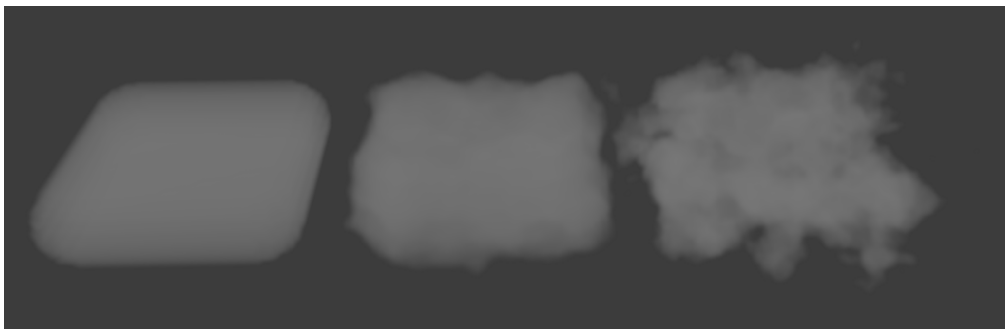
Strength
: Controls how far voxels are displaced.

Sample Radius
: Smaller values result in better performance, but might cut off the volume outside.

Mid Level

This should be modified if the texture offsets the entire volume in one direction and you want to center it again. For performance reasons, the displaced volume should stay close to its original position.

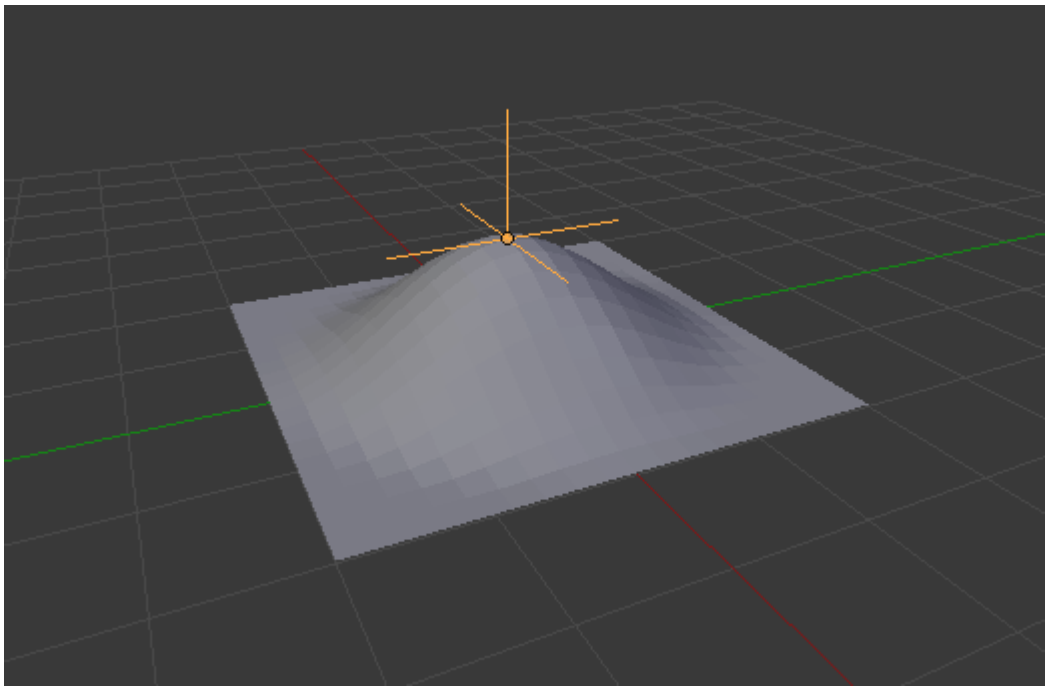
Example



A volume displaced with various strengths.

Warp Modifier

The *Warp* modifier can be used to warp parts of a mesh to a new location in a very flexible way, by using two objects to select the “from” and “to” regions.



A Warp modifier applied to a grid mesh.

This modifier is a bit tricky to understand at first. It requires two points, specified by the two target objects’ origins. The “from” point designates a point in space that is pulled toward the “to” point. It is akin to using the Proportional Editing in Edit Mode.

Options

Object From
: The object defining the origin transformation of the warp.

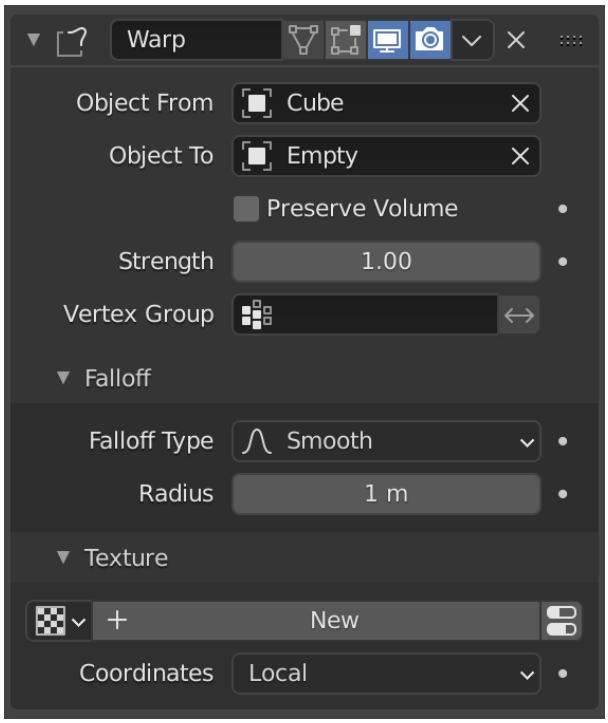
Object To

The object defining the destination transformation of the warp.

Preserve Volume
: Enables volume preservation when rotating one of the transforms.

Strength
: Sets how strong the effect is.

Vertex Group
: The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.



The Warp modifier.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group.

Falloff

Falloff Type
: Sets the way the strength of the warp change as it goes from the center of the transform to the *Radius* value. See Proportional Editing for descriptions of the falloff types.

Radius
: Sets the distance from the transforms that can be warped by the transform handles.

Texture

Texture
: You can finely control which vertices are affected by the warp, and to what extent, using a texture.

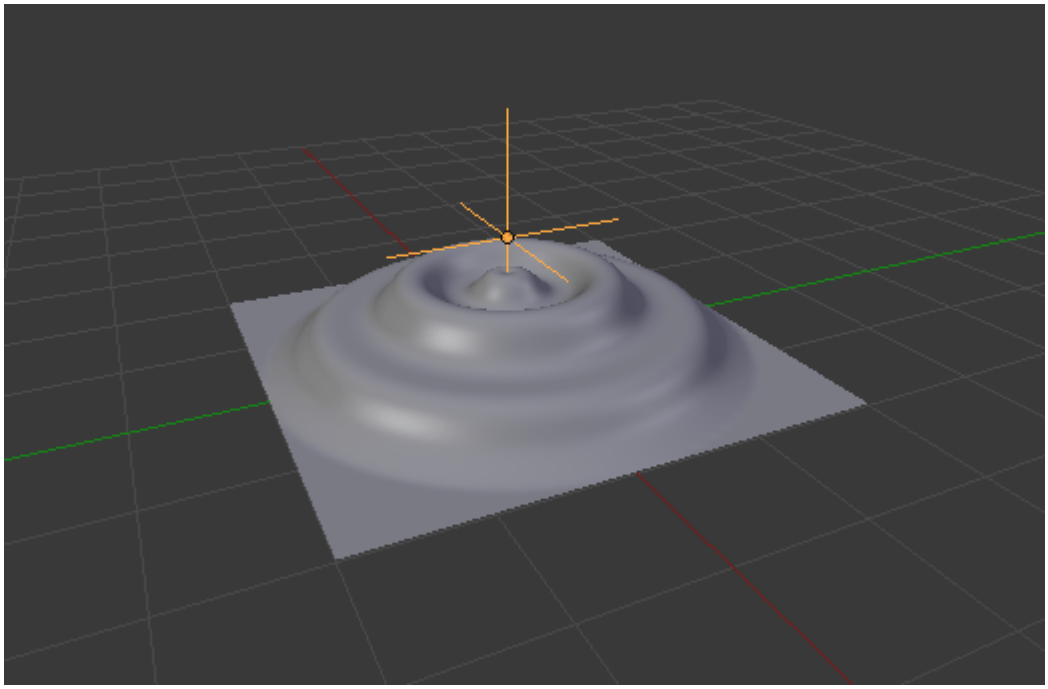
See common masking options for a complete reference.

Usage

The *Warp* modifier can be awkward to use sometimes, and its use case is rather small, but there are a few still. For example, it can be used to have an interactive Proportional Editing that can be used for animations.

Another way to use this modifier is similar to the Deform Modifier. This allows you to deform parts of the mesh without having to make a vertex group.

Examples

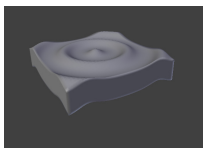


Warp Modifier with a custom falloff curve.

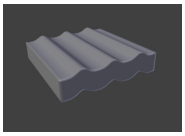
Wave Modifier

The *Wave* modifier adds a ripple-like motion to an object's geometry.

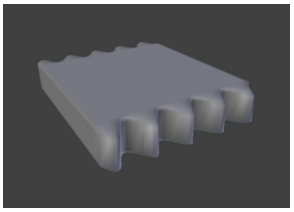
This modifier is available for meshes, lattices, curves, surfaces and texts.



Circular wave front.



Linear wave front.



Motion enabled for X, Normals enabled for Y.

Options

Motion

The wave effect deforms vertices/control points in the Z direction, originating from the given starting point and propagating along the object with circular wave fronts (if both X and Y are enabled), or with rectilinear wave fronts (if only one axis is enabled), then parallel to the axis corresponding to the X or Y button activated.

Cyclic

Repeats the waves cyclically, rather than a single pulse.

Along Normals

For meshes only. Displaces the mesh along the surface normals (instead of the object's Z axis).

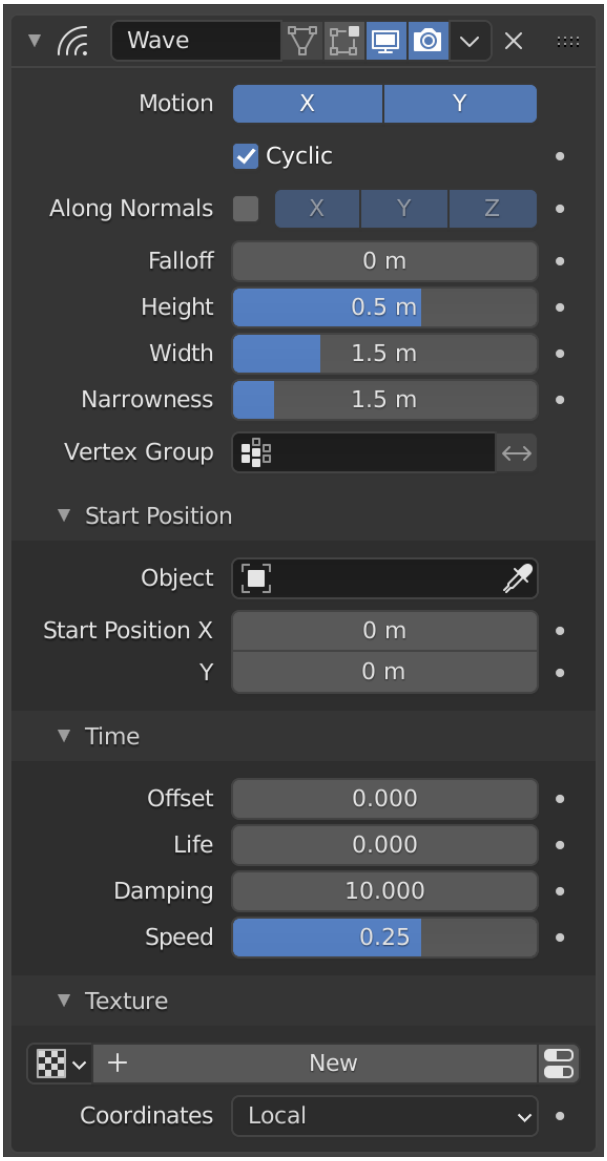
X/Y/Z

Restrict displacement along normals to the selected local axes.

Falloff
: Controls how fast the waves fade out as they travel away from the coordinates above (or those of the *Start Position Object*).

Height
: The height or amplitude of the ripple.

Width
: Half of the width between the tops of two subsequent ripples (if *Cyclic* is enabled). This has an indirect effect on the ripple amplitude. If the pulses are too near to each other, the wave may not reach the zero Z position, so in this case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude. See Technical Details and Hints for more information.



The Wave modifier.

Narrowness
: The actual width of each pulse: the higher the value the narrower the pulse. The actual width of the area in which the single pulse is apparent is given by `4 / Narrowness`. That is, if *Narrowness* is 1 the *pulse* is 4 units wide, and if *Narrowness* is 4 the *pulse* is 1 unit wide.

Vertex Group
: The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group.

Important

All the values described above are in local object space, i.e. they must be multiplied with the corresponding *Scale* values of the object to get the real dimensions.

Start Position

Object
: Use another object as the reference for the starting position of the wave. Note that you then can animate this object's position, to change the wave's origin across time.

Start Position X/Y
: Coordinates of the center of the waves, in the object's local space.

Time

Settings to control the animation.

Offset
: Time offset in frames. The frame at which the wave begins (if *Speed* is positive), or ends (if *Speed* is negative). Use a negative frame number to prime and pre-start the waves.

Life
: Duration of animation in frames. When set to zero, loops the animation forever.

Damping
: An additional number of frames in which the wave slowly damps from the *Height* value to zero after *Life* is reached. The dampening occurs for all the ripples and begins in the first frame after the *Life* is over. Ripples disappear over *Damping* frames.

Speed
: The speed per frame, of the ripple.

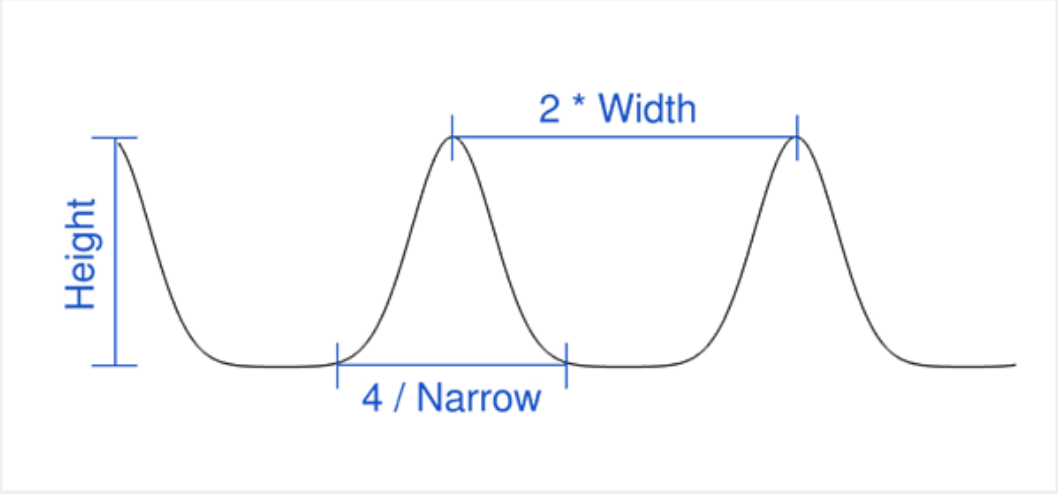
Texture

You can finely control which vertices are affected by the wave, and to what extent, using a texture.

See common masking options for a complete reference.

Technical Details and Hints

The relationship of the above values is described here:



Wave front characteristics.

To obtain a nice wave effect similar to sea waves and close to a sinusoidal wave, make the distance between following ripples and the ripple width equal. That is, the *Narrowness* value must be equal to `2 / Width`. E.g. for *Width* to be 1, set *Narrow* to 2.

Normals

- Normal Edit Modifier
- Weighted Normal Modifier

- Smooth By Angle Modifier

Normal Edit Modifier

The *Normal Edit* modifier affects (or generates) custom normals. It uses a few simple parametric methods to compute them (quite useful in game development and architecture areas), and mixes back those generated normals with existing ones.

Options

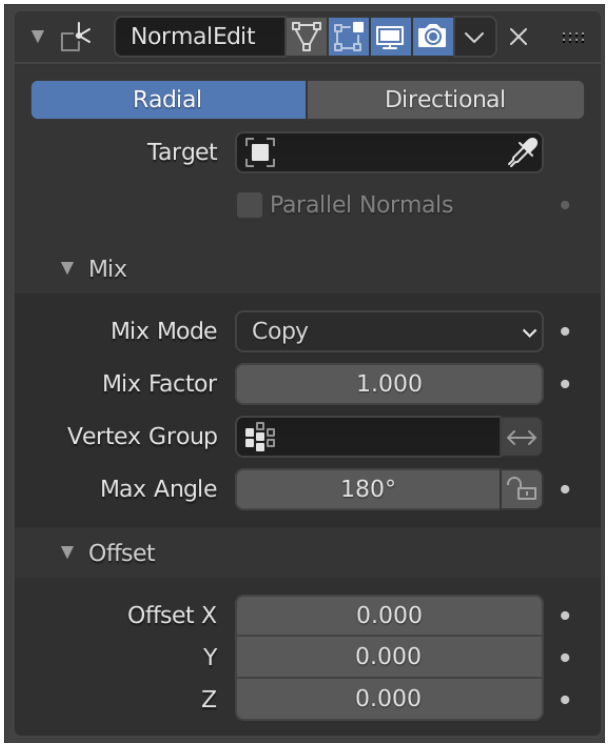
Radial
: Aligns normals with the `(origin, vertex_coordinates)` vector, in other words all normals seems to radiate from the given center point, as if they were emitted from an ellipsoid surface.

Directional
: Makes all normals point (converge) towards a given target object.

Target
: Uses this object's origin as reference point when generating normals.

  Optional in *Radial* mode, mandatory in *Directional* one.

Parallel Normals



Normal Edit Modifier.

Makes all normals parallel to the line between both objects' origins, instead of converging towards target's origin.

Only relevant in *Directional* mode.

Mix

Mix Mode
: How to affect existing normals with newly generated ones.

  Note that the *Multiply* option is **not** a cross product, but a faster component-by-component multiplication.

Mix Factor
: How much of the generated normals get mixed into existing ones.

Vertex Group
: Allows per-item fine control of the mix factor. The vertex group influence can be inverted by using the arrow button to the right.

Max Angle
: Forbids new generated normals to have an angle to the original normal above that given threshold. This is useful to prevent extreme changes, that can even lead to inverting the front/back sides of a face, and consequently to shading artifacts.

  Lock Polygon Normals (padlock icon)
  : Prevents flipping (reversing front/back sides) of polygons which normal does not match anymore the side to which point its corners' custom normals. Can also help to avoid shading issues.

Offset

Gives modified object's origin an offset before using it to generate normals.

Only relevant in *Radial* mode if no *Target Object* is set, and in *Directional* mode when *Parallel Normals* is set.

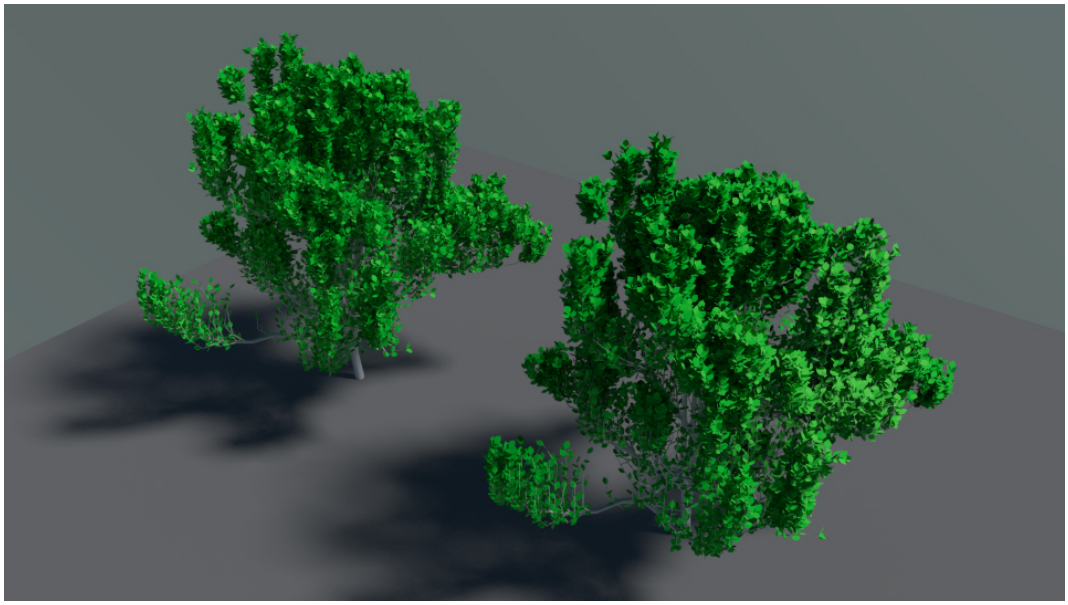
Usage

This modifier can be used to quickly generate radial normals for low-poly tree foliage or “fix” shading of toon-like rendering by partially bending default normals…

Tip

More complex normal manipulations can be achieved by copying normals from one mesh to another, see the Data Transfer Modifier. Some shading effects can also make use of the Weighted Normals modifier.

Example



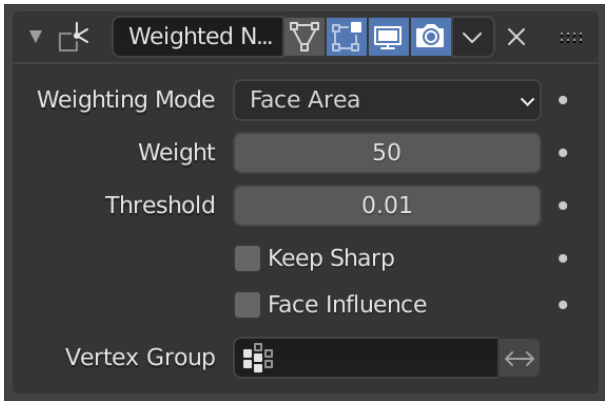
Editing custom normals to point towards a given direction (blend-file).

The left tree mesh has unmodified normals, while on the right one a *Normal Edit* modifier is used to bend them towards the camera. This shading trick is often used in games to fake scattering in trees and other vegetation.

Weighted Normal Modifier

This modifier changes the custom normals of a mesh, using various selectable methods. This can be useful to make some faces appear very flat during shading, among other effects. See Normals for a description of normals and custom normals.

Options



Weighting Mode
: The normals around a vertex will be combined to create a custom (per face corner) normal using various weights for each. The *Weighting Mode* defines how to compute the weights.

  Face Area
  : Weight according to the area of the face that the normal originates. A larger area means that the normal from that face will get a higher weight in final result.

  Corner Angle
  : Weight according to the angle each face forms at the vertex. This is the method Blender uses by default when combining face normals to compute a vertex one.

  Face Area and Angle
  : Weights are obtained by multiplying the face area and corner angle ones.

Weight

Determines how strongly the weights are biased according to the face areas and/or corner angles, a bit like a contrast setting for a picture.

A value of 50 means all faces are weighted uniformly. More than 50 means faces with higher area or angles are given even more weight (more "contrast"). Less than 50 means faces with higher area or angles are given lesser weights (less "contrast").

Threshold

A weight-rounding threshold which means that, if two angles or areas differ by less than that threshold, they will get equal weights.

Keep Sharp

Preserve sharp edges, though smoothing will still happen if there are multiple faces between any two sharp edges.

Face Influence

Use face weights (weak, medium, or strong) as assigned by the Set Strength tool or by the *Set Strength* mode of a Bevel modifier.

For example, if three faces meet at a vertex and have the face weights weak, medium, and strong, then only the normal associated with the strong face will be used to set the final result.

Vertex Group

If a vertex group is specified, the modifier will only affect those vertices. The "arrow" button to its right will invert the selection (only affect the vertices *not* in the vertex group).

Smooth By Angle Modifier

Sets the sharpness of mesh edges based on the angle between the neighboring faces.

Note

This is a geometry nodes asset that is included in the bundled "Essentials" asset library.

Tip

This modifier can easily be added to an object with Shade Auto Smooth or removed with Shade Smooth or Shade Flat.

Options

Angle
: Maximum angle between face normals that will be considered as smooth.

Ignore Sharpness
: Smooth all edges, even if they have been marked as sharp.

Physics

- Cloth Modifier
- Collision Modifier
- Dynamic Paint Modifier
- Explode Modifier
- Fluid Modifier
- Ocean Modifier
- Particle Instance Modifier
- Particle System Modifier
- Soft Body Modifier

Cloth Modifier

The Cloth modifier is a container for a Cloth Physics simulation. It can be useful for example, to simulate on a low-poly mesh then add a Subdivision Surface Modifier after the Cloth modifier to improve the visual quality of the cloth without drastically increasing simulation times.

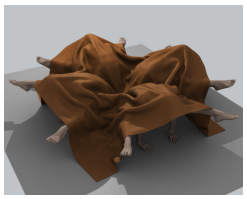
Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Cloth Physics Properties for more information.

Example



Cloth example.



Cloth on carved wooden men (made by motorsep).



Cloth example.

Collision Modifier

The Collision modifier is a container for a Collision Physics. Collision physics provide interaction between different physics simulations.

Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Collision Physics Properties for more information.

Example



Deflected particles.

Here is a *Meta* object, using Instancing Vertices to a particle system emitting downwards, and deflected by a mesh cube.

Dynamic Paint Modifier

The Dynamic Paint modifier is a container for a Dynamic Paint Physics simulation.

Options

As the modifier is only a container its actual options are configured in the *Physics Properties* tab. See the Dynamic Paint Physics Properties for more information.

Explode Modifier

The *Explode* modifier is used to alter the mesh geometry by moving/rotating its faces in a way that roughly tracks particles emitted by that object, making it look as if the mesh is being exploded (broken apart and pushed outward).

For this modifier to have any visible effect, there needs to be a particle system on its object. That particle system will control how the mesh is exploded.

Both the number of emitted particles and number of faces determine how granular the *Explode* modifier is. More of each faces and particles will mean more individual pieces.

Here is a demo video showing a cube with a particle system and *Explode* modifier. (blend-file).

Note

The *Explode* modifier must come after the *Particle System* one in the modifier stack, in order for the former to get required data from the later.

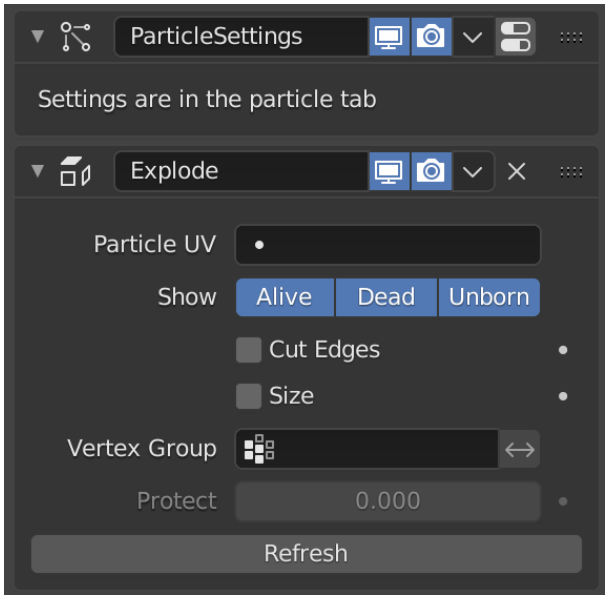
Options

Particle UV

If set, the U value of the coordinates in that UV Map will be overwritten with the age of the particle attached to the matching mesh face (in proportion, from 0 for not yet born particles, to 1 for dead ones).

The V value is set to a constant 0.5 value.

You can for example vary the color of a fragment (face) during the explosion phase, by using a texture with a color gradient along its *U* axis.



The Explode modifier, with a Particle System above it.

Show

Unborn
: Show faces when their attached particles are unborn.

Alive
: Show faces when their attached particles are alive.

Dead
: Show faces when their attached particles are dead.

Cut Edges
: Split the mesh in pieces based on location of emitted particles, instead of using existing faces. This will typically results in a splitting that appears more random.

Size
: Scale each face using the size of its attached particle, once that particle is alive.

Vertex Group
: Vertices in this group may not be affected by the *Explode* modifier. Vertices with full weight are not affected at all, while vertices with less

weight have a higher chance of being affected.

Vertices with null weight will be treated like those which do not belong to the group at all, and explode normally.

Invert <–>
: Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

  The setting reverses the weight values of the group.

Protect
: Clean vertex group edges. Depending on the weights assigned to that vertex group, either completely protect those faces from being affected by the *Explode* modifier (which would happen if the faces had a weight value of 1), or completely remove protection from those faces (which would happen if the faces had a weight value of 0).

Refresh
: Refresh data in the *Explode* modifier.

Known Limitations

Dynamic Vertex Weights

This modifier uses the initial vertex weights. Modifiers that dynamically change weights will not influence the explosion as these values are only used once.

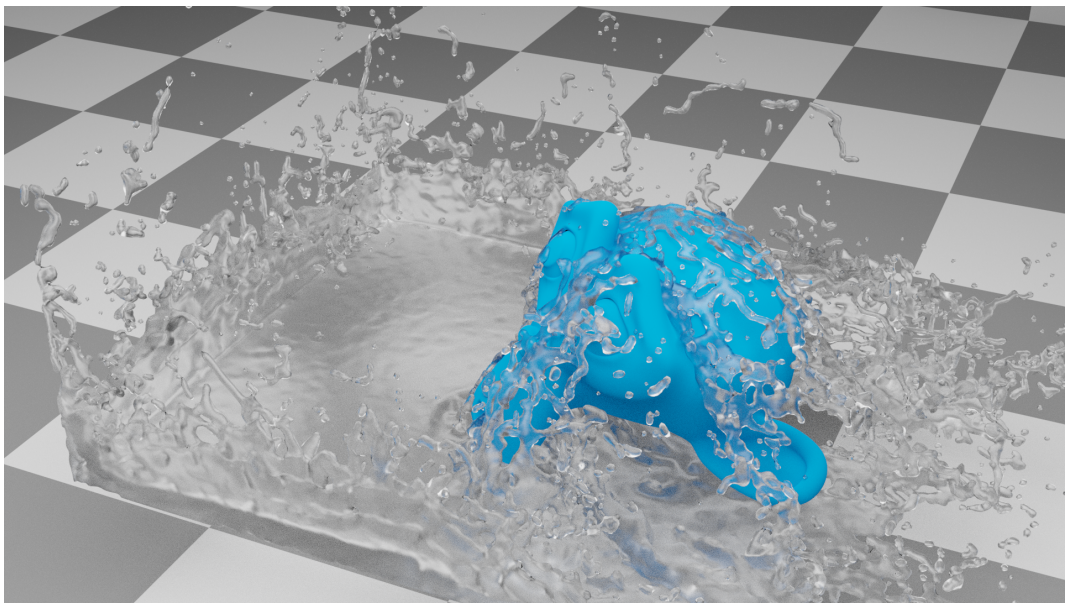
Fluid Modifier

The Fluid modifier is a container for a Fluid Physics simulation. It can be useful for example, to simulate on a low-poly mesh then add a Subdivision Surface Modifier after the Fluid modifier to improve the visual quality of the fluid without drastically increasing simulation times.

Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Fluid Physics Properties for more information.

Example



Example of a liquid simulation.

Ocean Modifier

The *Ocean* modifier is a tool to simulate and generate a deforming ocean surface, and associated texture, used to render the simulation data. It is intended to simulate deep ocean waves and foam.

It is a port from the open source Houdini Ocean Toolkit.

Options

Geometry

Generate

Creates a tiled mesh grid that exactly corresponds with the resolution of the simulation data.

When generating a mesh surface, the existing mesh object is completely overridden with the ocean grid; this also includes any data generated from previous modifiers in the stack. A UV channel is also added, mapping the (0.0 to 1.0) UV space to the simulation grid.

Repeat X, Y

Controls the number of times the grid is tiled in X and Y directions. UVs for these tiled mesh areas continue outside of the (0.0 to 1.0) UV space.

Displace

Uses the existing geometry rather than replacing it. Vertices are displaced along the local Z axis.

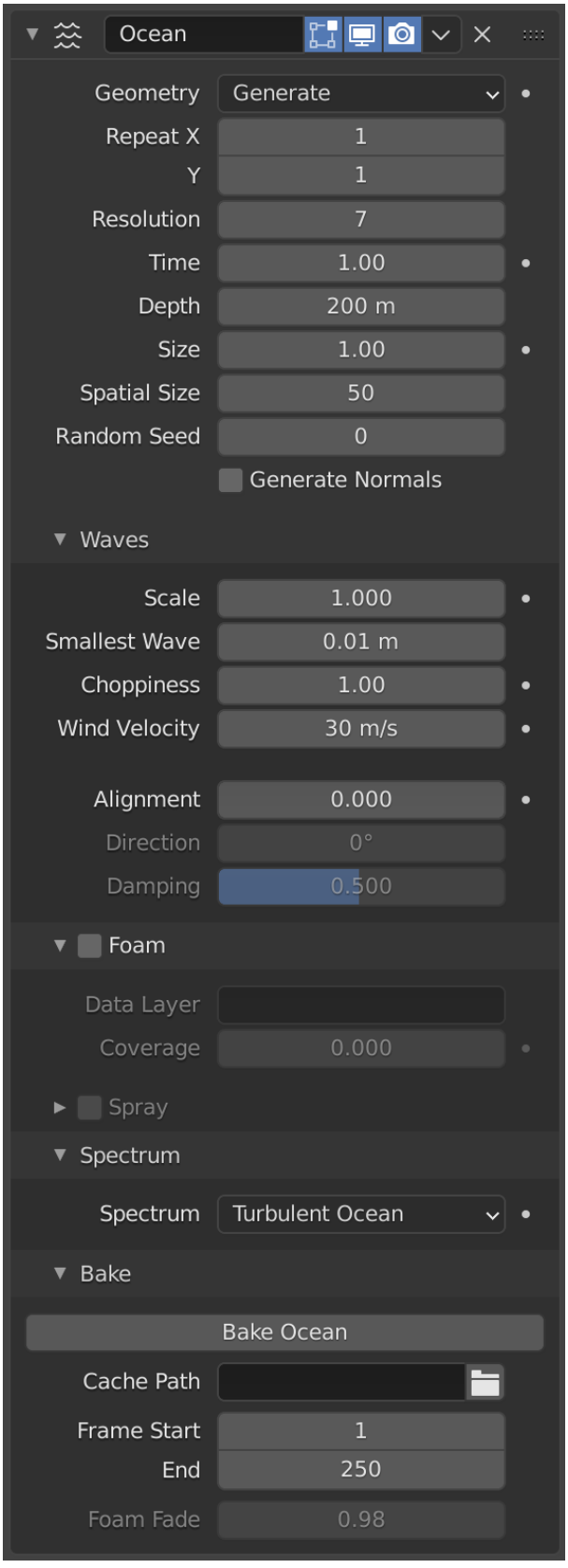
Resolution Viewport, Render

The main control of quality vs speed in the simulation engine. This determines the resolution of the internal 2D grids generated by the simulation for the 3D Viewport or the final render.

The internal grids are powers of two of the resolution value, so a resolution value of `16`, will create simulation data of size `256×256`. The higher the resolution, the more details will be produced, but the slower it will be to calculate.

Note

When using the *Generate* modifier geometry option, this resolution value also determines the resolution of the generated mesh surface, equal to the resolution of the internal simulation data.



Time

The time at which the ocean surface is being evaluated. To create an animated ocean, you will need to animate this value. The speed that the time value is changing will determine the speed of the wave animation.

Depth

The constant depth of the ocean floor under the simulated area. Lower values simulate shallower waters by producing higher frequency details and smaller waves.

Size

A simple scaling factor that does not affect the height of the waves or behavior of the simulation.

The Ocean modifier.

Spatial Size
: The width of the ocean surface area being simulated, in meters. This also determines the size of the generated mesh, or the displaced area. Of course, you can scale the object with the *Ocean* modifier in Object Mode to tweak the apparent size in your scene.

Random Seed
: A different Seed will produce a different simulation result.

Generate Normals
: Simulates extra normal map data. This can be used by the Ocean texture, when mapped to Normals, as a bump map, and enables generating normal map image sequences when baking.

Waves

Scale
: An overall scale control for the amplitude of the waves. It approximates the height or depth of the waves above or below zero. Rather than just scaling the ocean object in Z, it scales all aspects of the simulation, displacement in X and Y, and corresponding foam and normals too.

Smallest Wave
: A minimum limit for the size of generated waves. Acts similarly to a low-pass filter, removing higher frequency wave detail.

Choppiness
: The choppiness of the wave peaks. With a choppiness of 0, the ocean surface is only displaced up and down in the Z direction, but with higher choppiness, the waves are also displaced laterally in X and Y, to create sharper wave peaks.

Wind Velocity
: Wind speed in meters/second. With a low velocity, waves are restricted to smaller surface waves.

Alignment
: The directionality of the wave shapes due to wind. At a value of 0, the wind and waves are randomly, uniformly oriented. With higher *Alignment* values, the wind is blowing in a more constant direction, making the waves appear more compressed and aligned to a single direction.

Direction
: When using *Alignment*, the direction in degrees that the waves are aligned to (using local X axis as reference).

Damping
: When using *Alignment*, this will define the amount that inter-reflected waves are damped out. This has the effect of making the wave motion more directional (not just the wave shape).

  With a *Damping* of 0.0, waves are reflected off each other in every direction, with a *Damping* of 1.0, these inter-reflected waves are damped out, leaving only waves traveling in the direction of the wind.

Foam

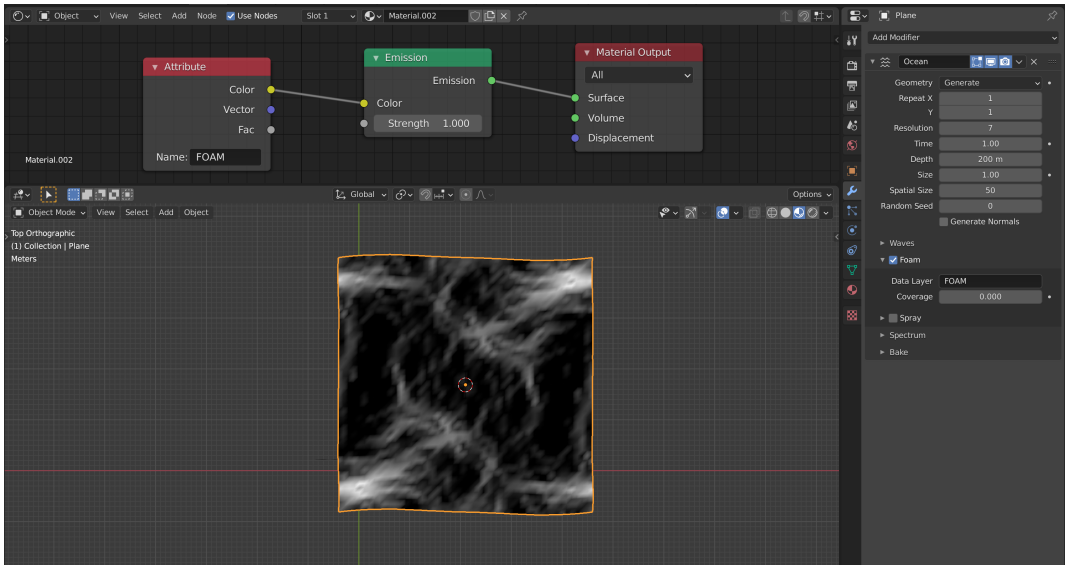
Simulates extra foam data.

This can be retrieved by the Ocean texture for use in texturing (perhaps as a mask), and enables generating foam map image sequences when baking.

Data Layer
: Optional name for the vertex data layer, used by the Ocean Modifier to store foam maps as a Color Attribute. This is required for accessing the foam data in the renderer.

Coverage

Tweaks the amount of foam covering the waves, negative values will reduce the amount of foam (leaving only the topmost peaks), positive values will add to it. Typically ranges from (-1.0 to 1.0).



Using foam Color Attributes with a named data layer.

Spray

Generate map of spray direction as a Color Attribute. This map can be used to define the velocities for spray particles.

Spray Map
: Name of the Color Attribute used for the spray direction map.

Invert
: Inverts the spray direction map.

Spectrum

Spectrum
: Used to choose the wave spectrum model to use. Wave spectra are used to describe how energy moves through the waves at different frequencies. Energy travels through waves differently depending on the depth of the water and the wind speed.

Turbulent Ocean

Use for turbulent seas with foam (Phillips).

Established Ocean

Use for a large area, established ocean where the ocean would extend for miles with wind blowing for days allowing the waves to reach a point of equilibrium (Pierson-Moskowitz method).

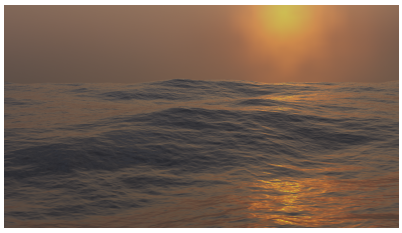
Established Ocean (Sharp Peaks)

Similar to regular *Established Ocean* however, waves will continue to grow with time creating sharper peaks (JONSWAP and Pierson-Moskowitz method). An extra parameter is used to define the sharpness of these peaks.

Shallow Water

Use for shallow water with depths less than about 10 meters which makes it great for small lakes and ponds without heavy wind (JONSWAP and TMA – Texel-Marsen-Arsloe methods).

Examples of different spectra, settings adjusted for each.



Turbulent Ocean.



Established Ocean.



Established Ocean (Sharp Peaks).



Shallow Water.

Sharpness Peak
: An artificial factor to control how sharp the peaks of the waves are in the *Established Ocean (Sharp Peaks)* and *Shallow Water* spectrum models.

Fetch
: Distance from a lee shore, called the fetch, or the distance over which the wind blows with constant velocity. Used by *Established Ocean (Sharp Peaks)* and *Shallow Water* spectrum models.

Bake

Rather than simulating the ocean data live, it can be baked to files in a given directory. When a simulation is baked, the simulator engine is completely bypassed, and all information for the modifier or texture is retrieved from the baked files.

Baking has the following advantages:

- It is faster to use the stored data rather than recalculating it.
- It allows rendering of ocean data in external renderers.
- It enables more advanced foam maps.

Simulation data is stored as sequences of `OpenEXR` image maps, one for each of displacement, normals, and foam (if enabled to be generated). Upon loading the data from these baked files, when a frame of the bake sequence is read, it is cached in memory. This means that accessing loaded frames subsequent times is fast, not incurring the overhead of drive access.

Since these baked files are plain `OpenEXR`'s, they can also be opened and rendered in any other application or renderer that supports them.

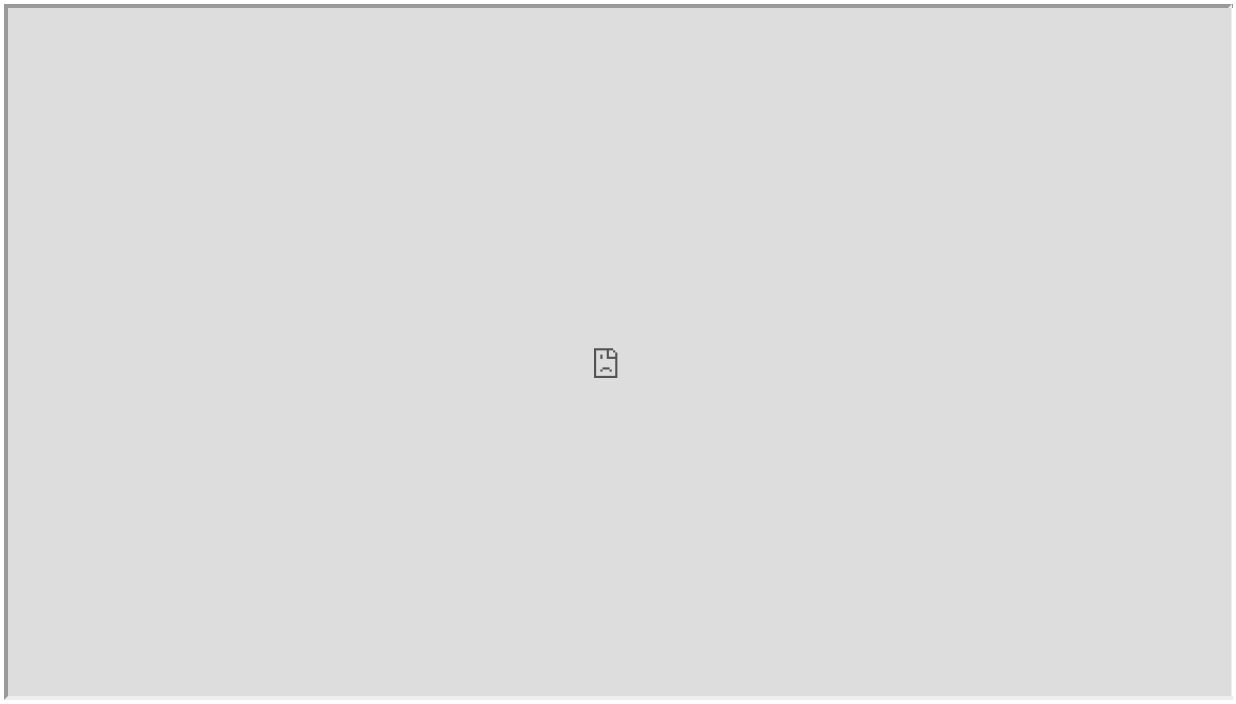
Cache Path
: Folder to store the baked EXR files in. The sequences will be in the form `disp_####.exr`, `normal_####.exr`, and `foam_####.exr`, where `####` is the four digit frame number. If the cache path folder does not exist, it will be created.

Frame Start, End
: Frames of the simulation to bake (inclusive). The start and end frames of the bake are repeated when accessing frames outside of the baked range.

Foam Fade
: Baking also provides improved foam capabilities. When simulating live, the ocean simulator retrieves data for that current frame only. In the case of the foam map, this represents the tips of wave crests for that given frame. In reality, after foam is created by wave interactions, it remains sitting on the top of the wave surface for a while, as it dissipates. With baking, it is possible to approximate that behavior, by accumulating foam from previous frames, leaving it remaining on the surface.

Example

The following example was created and rendered in Blender, notice how the peaks of the waves are white; an effect generated from the foam data.



Particle Instance Modifier

When a Particle Instance modifier is added to an object, the mesh of this object will be duplicated at the location of the particles of the selected particle system from another target object. This means that to use this modifier, you must have at least one other object that has a Particles System on it.

Because of the correlation in which the Particle Instance modifier is influenced by the underlying particle systems on other objects, some of the apparent effects generated by the modifier can look and act vastly different, depending on the underlying settings of the particle systems it is associated with. This is worth taking account of, when it seems that the Particle Instance modifier settings do not return the expected results.

Options

Object
: The target object which has a particle system associated with it.

Particle System
: Which particle system from the target *Object* to apply this modifier to.

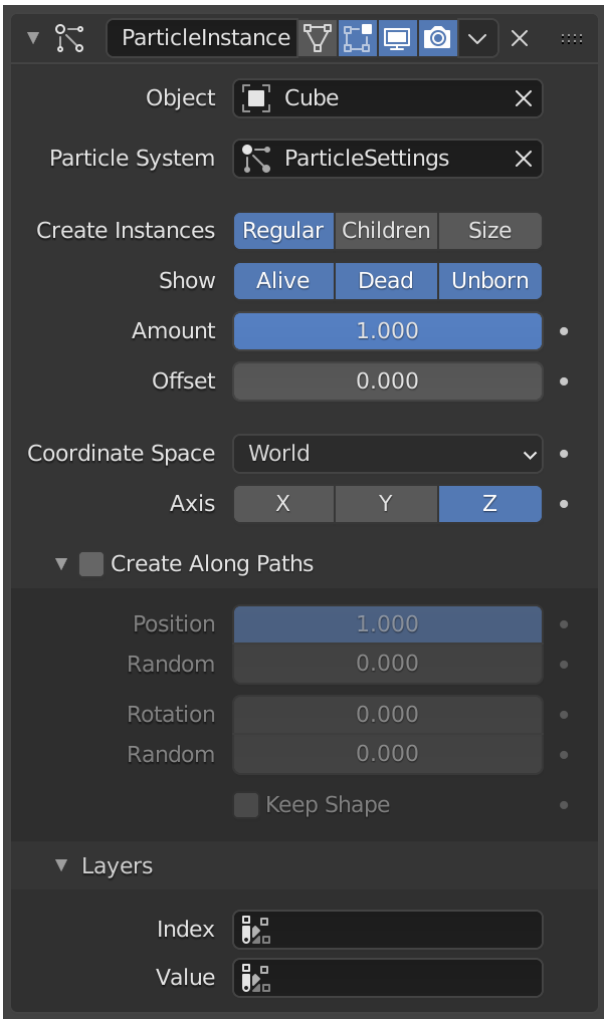
Create Instances
: Regular
  : When enabled, the modifier will use the regular (parents) particles to duplicate the mesh of the modified object.

: Children
  : When enabled, the modifier will use the children particles to duplicate the mesh of the modified object.

: Size

Scale the instanced copies of the mesh by the particle size attribute. When this is disabled, all the copies appear the same size as the origin.

See the particle system's Render and Children panels for particle's size options.



The Particle Instance modifier.

Show

Unborn

When enabled, the modifier will use the unborn particles to duplicate the mesh of the modified object.

Alive

When enabled, the modifier will use the alive particles to duplicate the mesh of the modified object.

Dead

When enabled, the modifier will use the dead particles to duplicate the mesh of the modified object.

Amount

The proportion of particles to be used. Allows you to **randomly** skip particles to adjust the amount of instances.

Warning

The random algorithm used currently only ensures that relative amount to be respected *statistically*. The actual amount of instances generated will differ from the theoretical one, depending on the *Seed*

value of the target particle system (and the *Offset* value described below, too).

That deviation is not significant with high number of particles, but it will be highly noticeable with low numbers (e.g. with 100 particles in the target system, and an *Amount* value of `0.1`, it can generate either up to 15 or 5 instances, instead of the 10 expected).

Offset

A relative offset in the range of particles used for instantiation. Allows you to avoid overlapping of the used particles, when the same particle system is used in multiple modifier instances.

Tip

If you want to fully avoid overlaps, your *Offset* value must be at least as high as your *Amount* value.

Coordinate Space

World, Local

Use World Space, or Local Space of the target object (that the particle system is assigned to).

- World space means that the locations of the copies of the modified mesh will depend on the location of the modified object **and** of the target object.
- Local space means that the locations of the copies of the modified mesh will depend only on the location of the modified object.

Axis

Specify which axis of the modified object to use as pole axis to apply the rotation from the instantiated particles.

Create Along Paths

By default, the instances are placed depending on the particles position in the current frame. By enabling *Create Along Paths*, the instance of the modified object follows deforms its shape along the particle path (or the hair strand). This allows you to select the position along the particles path regardless of the current frame.

Tip

You can adjust the particles' path (using the *Path* visualization type) on the Render panel of the *Particle System* tab.

Note

The particle system must be baked, except for *Hair* type or *Keyed* physics.

Position
: Specify what percentage of the path that the instance fills, or the position on the path if the *Keep Shape* option is enabled.

Random
: Adds randomness to the *Position* value of each instance.

Rotation
: Specifies the rotation around the path.

Random
: Adds randomness to the *Rotation* value of each instance.

Keep Shape
: Enabling this prevents the instance from being deformed, and places it on the path according to the *Position* value.

Layers

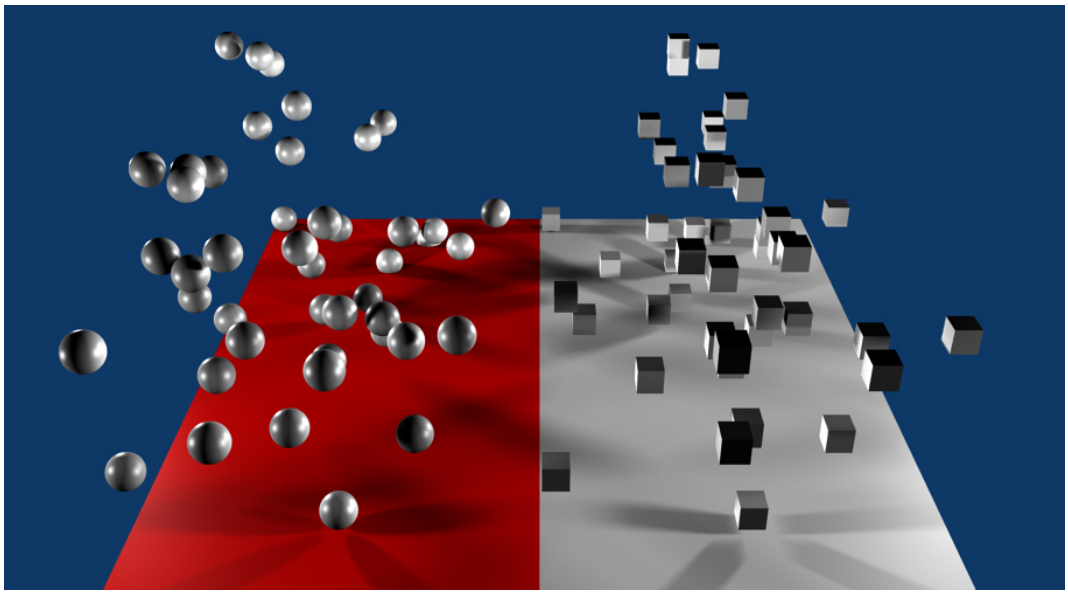
With these fields you can select the Color Attribute, which will be filled with colors based on the particles information. These Color Attributes can

be used, for example, in a shader to add variance to a material.

Index
: A Color Attribute for values based on the particles index.

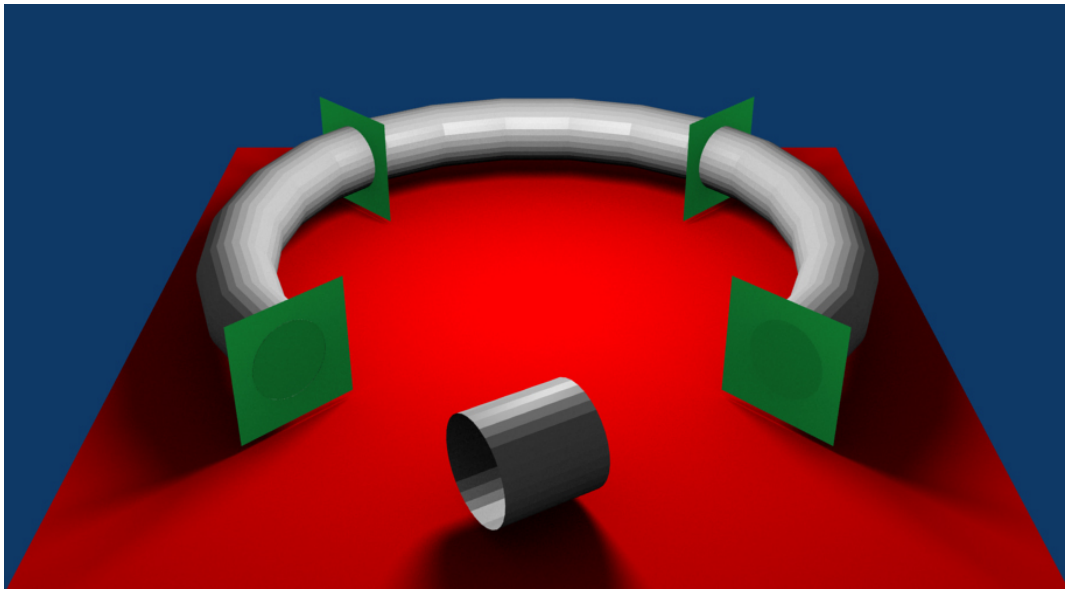
Value
: A Color Attribute for random per-particle values.

Examples



Particle Instance modifier example.

The render above shows a single plane mesh object assigned to two different vertex groups and each of those vertex groups is assigned to a separate and independent particle system, with each particle system being assigned to a different Particle Instance modifier. In the case shown the Particle Instance modifiers are added to a sphere and a cube. See example blend-file.



Create Along Path example.

In this example, a single *Keyed* particle travels through four points (green planes), on an elliptical path. The Particle Instance modifier is added to a cylinder object and then associated with that *Keyed* particle system.

When the *Create Along Paths* is activated, instead of the cylinder location just following the position of the particle, the cylinder mesh is fitted to the shape of the path followed by the particle. The mesh geometry of the object which is deformed can have an impact on how well the deformation is carried out. In the case of the cylinder, it has many loop cuts along its length so that it can bend at those points to deform along the particle path.

The Particle Instance modifier *Create Along Paths* option works for hair (strand) particles as well as with keyed particles. In this case, the mesh of the modifier will follow the length and profile of the hair strands paths.

Note

Strands, when they are generated, instantly die when created, so for the *Create Along Paths* checkbox to be of any use, you must also have the *Dead* checkbox enabled.

Particle System Modifier

The Particle System modifier is a container for Particle Systems.

Note

By default the Particle System modifier does not take into account the modifier stack. Make sure to enable Use Modifier Stack in the Particle properties if you want Particle System modifier to take other modifiers into account.

Options

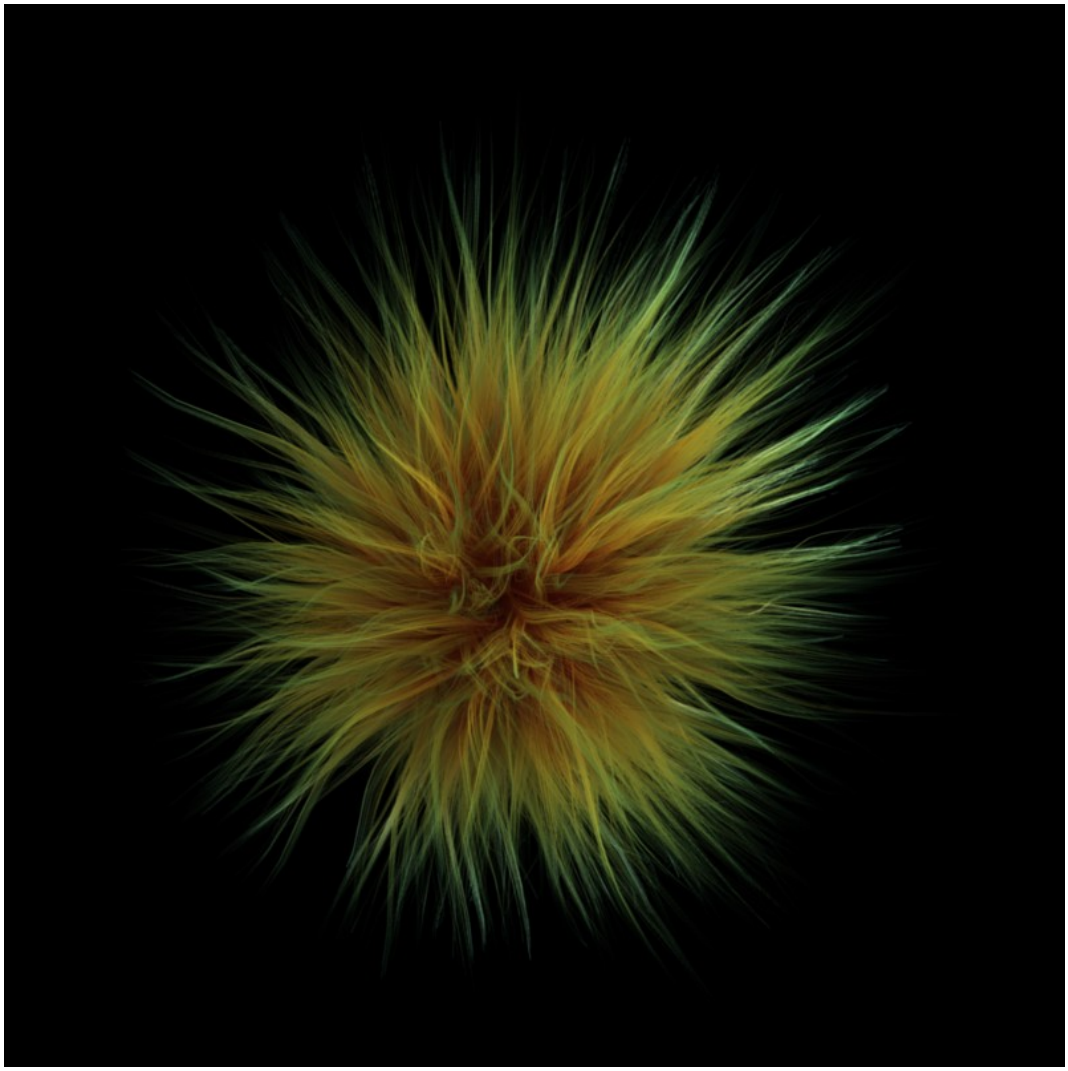
As the modifier is only a container its actual options are configured in the *Particle Properties* tab. See the Particle Systems Properties for more information.

Converting Particle Systems

Make Instances Real
: Creates a new object of each instanced object or collection. See Make Instances Real for more information.

Convert to Mesh
: Converts path particles to mesh objects. See Convert for more information.

Example



Fur made from particles.

Soft Body Modifier

The Soft Body modifier is a container for a Soft Body Physics simulation.

Options

As the modifier is only a container its actual options are configured in the *Physics Properties* tab. See the Soft Body Physics Properties for more information.

Example



The wind cone is a soft body, as the suspension.

Geometry Nodes

- Introduction
- Inspection
- Attributes
- Fields
- Instances
- Baking
- Node-Based Tools
- Gizmos

Node Types

- Attribute Nodes
  - Attribute Statistic Node
  - Domain Size Node
  - Blur Attribute Node
  - Capture Attribute Node
  - Remove Named Attribute Node
  - Store Named Attribute Node
- Input Nodes
  - Constant
    - Boolean Node
    - Color Node
    - Image Node
    - Integer Node
    - Material Node
    - Rotation Node
    - String Node
    - Value Node
    - Vector Node
  - Gizmo
    - Dial Gizmo
    - Linear Gizmo

        - Transform Gizmo
    - Scene
        - 3D Cursor Node
        - Active Camera Node
        - Collection Info Node
        - Image Info Node
        - Is Viewport Node
        - Mouse Position Node
        - Object Info Node
        - Scene Time Node
        - Self Object Node
        - Viewport Transform Node
- Output Nodes
    - Warning Node
    - Viewer Node

- Geometry Nodes
    - Read
        - ID Node
        - Index Node
        - Named Attribute Node
        - Normal Node
        - Position Node
        - Radius Node
        - Selection Node
        - Active Element Node
    - Sample
        - Geometry Proximity Node
        - Index of Nearest
        - Raycast Node
        - Sample Index Node
        - Sample Nearest Node
    - Write
        - Set Geometry Name Node
        - Set ID Node
        - Set Position Node
        - Set Selection Node

- Operations
  - Bake Node
  - Bounding Box Node
  - Convex Hull Node
  - Delete Geometry Node
  - Duplicate Elements Node
  - Merge by Distance Node
  - Split To Instances Node
  - Sort Elements Node
  - Transform Geometry Node
  - Separate Components Node
  - Separate Geometry Node
- Geometry to Instance Node
- Join Geometry Node

- Curve Nodes
  - Read
    - Curve Handle Positions Node
    - Curve Length Node
    - Curve Tangent Node
    - Curve Tilt Node
    - Endpoint Selection Node
    - Handle Type Selection Node
    - Is Spline Cyclic Node
    - Spline Length Node
    - Spline Parameter Node
    - Spline Resolution Node
  - Sample
    - Sample Curve Node
  - Write
    - Set Curve Normal Node
    - Set Curve Radius Node
    - Set Curve Tilt Node
    - Set Handle Positions Node
    - Set Handle Type Node
    - Set Spline Cyclic Node
    - Set Spline Resolution Node

        - Set Spline Type Node
    - Operations
        - Curve to Mesh Node
        - Curve to Points Node
        - Curves to Grease Pencil Node
        - Deform Curves on Surface Node
        - Fill Curve Node
        - Fillet Curve Node
        - Grease Pencil to Curves Node
        - Interpolate Curves Node
        - Merge Layers Node
        - Resample Curve Node
        - Reverse Curve Node
        - Subdivide Curve Node
        - Trim Curve Node
    - Primitives
        - Arc Node
        - Bézier Segment Node
        - Curve Circle Node
        - Curve Line Node
        - Curve Spiral Node
        - Quadratic Bézier Node
        - Quadrilateral Node
        - Star Node
    - Topology
        - Curve of Point Node
        - Offset Point in Curve Node
        - Points of Curve Node
- Instances Nodes
    - Instance on Points Node
    - Instances to Points Node
    - Realize Instances Node
    - Rotate Instances Node
    - Scale Instances Node
    - Translate Instances Node
    - Set Instance Transform Node
    - Instance Transform Node

    - Instance Rotation Node
    - Instance Scale Node
- Mesh Nodes
    - Read
        - Edge Angle Node
        - Edge Neighbors Node
        - Edge Vertices Node
        - Edges to Face Groups Node
        - Face Area Node
        - Face Group Boundaries Node
        - Face Neighbors Node
        - Face Set Node
        - Is Face Planar Node
        - Is Edge Smooth Node
        - Is Face Smooth Node
        - Mesh Island Node
        - Shortest Edge Paths Node
        - Vertex Neighbors Node
    - Sample
        - Sample Nearest Surface Node
        - Sample UV Surface Node
    - Write
        - Set Face Set Node
        - Set Shade Smooth Node
    - Operations
        - Dual Mesh Node
        - Edge Paths to Curves Node
        - Edge Paths to Selection Node
        - Extrude Mesh Node
        - Flip Faces Node
        - Mesh Boolean Node
        - Mesh to Curve Node
        - Mesh to Points Node
        - Mesh to Volume Node
        - Scale Elements Node
        - Split Edges Node
        - Subdivide Mesh Node

        - Subdivision Surface Node
        - Triangulate Node
    - Primitives
        - Cone Node
        - Cube Node
        - Cylinder Node
        - Grid Node
        - Icosphere Node
        - Mesh Circle Node
        - Mesh Line Node
        - UV Sphere Node
    - Topology
        - Corners of Edge Node
        - Corners of Face Node
        - Corners of Vertex Node
        - Edges of Corner Node
        - Edges of Vertex Node
        - Face of Corner Node
        - Offset Corner in Face Node
        - Vertex of Corner Node
    - UV
        - Pack UV Islands Node
        - UV Unwrap Node
- Point Nodes
    - Distribute Points in Volume
    - Distribute Points on Faces
    - Points Node
    - Points to Curves Node
    - Points to Vertices Node
    - Points to Volume Node
    - Set Point Radius Node
- Volume Nodes
    - Operations
        - Volume to Mesh Node
    - Primitives
        - Volume Cube Node

- Simulation Zone

- Material Nodes
  - Replace Material Node
  - Material Index Node
  - Material Selection Node
  - Set Material Node
  - Set Material Index Node
- Texture Nodes
  - Brick Texture Node
  - Checker Texture Node
  - Gabor Texture Node
  - Gradient Texture Node
  - Image Texture Node
  - Magic Texture Node
  - Musgrave Texture Node
  - Noise Texture Node
  - Voronoi Texture Node
  - Wave Texture Node
  - White Noise Texture Node
- Utilities Nodes
  - Color
    - Blackbody Node
    - Color Ramp Node
    - Combine Color Node
    - Mix Color Node
    - RGB Curves Node
    - Separate Color Node
  - Text
    - Join Strings Node
    - Replace String Node
    - Slice String Node
    - Special Characters Node
    - String Length Node
    - String to Curves Node
    - Value to String Node

- Vector
  - Vector Curves Node
  - Vector Math Node
  - Vector Rotate Node
  - Combine XYZ Node
  - Mix Vector Node
  - Separate XYZ Node
- Field
  - Accumulate Field Node
  - Evaluate at Index Node
  - Evaluate on Domain Node
- Math
  - Boolean Math Node
  - Clamp Node
  - Compare Node
  - Float Curve
  - Float To Integer Node
  - Hash Value Node
  - Integer Math Node
  - Map Range Node
  - Math Node
  - Mix Node
- Matrix
  - Combine Matrix Node
  - Combine Transform Node
  - Invert Matrix Node
  - Matrix Determinant Node
  - Multiply Matrices Node
  - Project Point Node
  - Separate Matrix Node
  - Separate Transform Node
  - Transform Direction Node
  - Transform Point Node
  - Transpose Matrix Node
- Rotation
  - Align Rotation to Vector Node
  - Axes to Rotation Node

        - Axis Angle to Rotation Node
        - Euler to Rotation Node
        - Invert Rotation Node
        - Rotate Rotation Node
        - Rotate Vector Node
        - Rotation to Euler Node
        - Rotation to Quaternion Node
        - Quaternion to Rotation Node
    - Deprecated
        - Align Euler to Vector Node
        - Rotate Euler Node
    - For Each Geometry Element Zone
    - Index Switch Node
    - Menu Switch Node
    - Random Value Node
    - Repeat Zone
    - Switch Node

- Group

- Hair Nodes
    - Deformation
        - Blend Hair Curves
        - Displace Hair Curves
        - Frizz Hair Curves
        - Hair Curves Noise
        - Roll Hair Curves
        - Rotate Hair Curves
        - Shrinkwrap Hair Curves
        - Smooth Hair Curves
        - Straighten Hair Curves
        - Trim Hair Curves
    - Generation
        - Duplicate Hair Curves
        - Generate Hair Curves
        - Interpolate Hair Curves

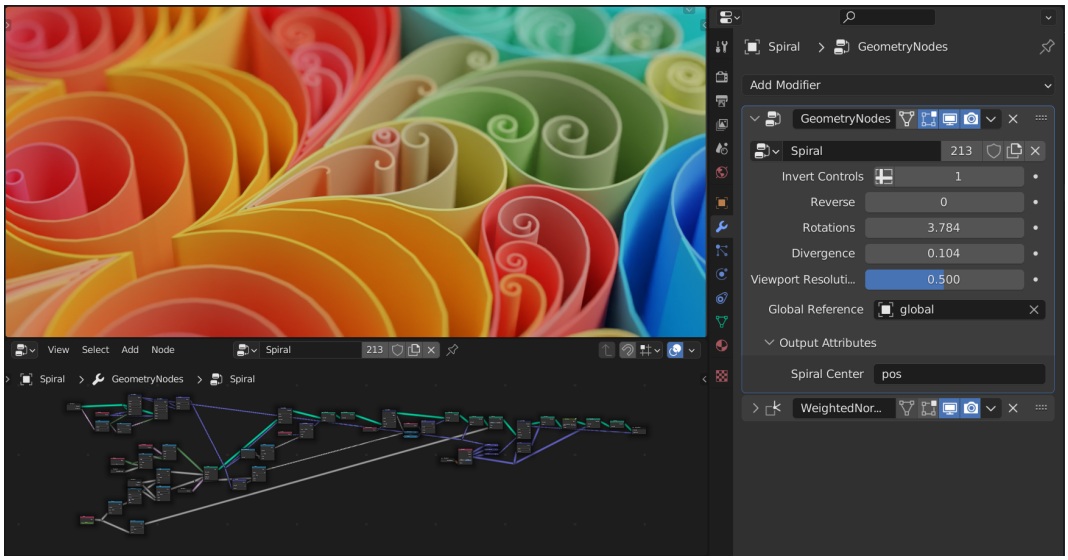
    - Guides
        - Braid Hair Curves
        - Clump Hair Curves
        - Create Guide Index Map
        - Curl Hair Curves
    - Read
        - Curve Info
        - Curve Root
        - Curve Segment
        - Curve Tip
        - Hair Attachment Info
    - Utility
        - Attach Hair Curves to Surface
        - Redistribute Curve Points
        - Restore Curve Segment Length
    - Write
        - Set Hair Curve Profile
- Normals Nodes
    - Smooth By Angle Node Group

Tip

Asset Catalogs that contain geometry node groups will also appear in the add menu.

Introduction

Geometry Nodes is a system for modifying the geometry of an object with node-based operations. It can be accessed by adding a Geometry Nodes Modifier.



The properties of a Geometry Nodes modifier in the modifier stack.

The geometry node tree connected to a modifier is a Node Group. The geometry from the state before the modifier (the original geometry or the result of the previous modifier) will be passed to the *Group Input* node. Then the node group can operate on the geometry and pass an output to the *Group Output* node, where it will be passed to the next modifier.

Geometry nodes can modify different types of geometry:

- Meshes
- Curves
- Point Clouds
- Volumes
- Instances

The interface of the modifier is described in the Modifier page.

To expand Blender with node-group operators see the description in the Node-Based Tools page.

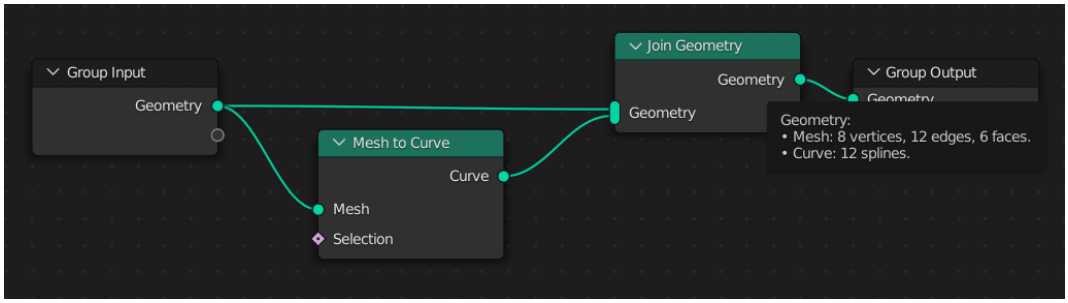
Inspection

Inspecting intermediate values in a geometry node tree is useful while building/understanding one or when trying to figure out why something is not working. Blender provides multiple tools to understand how a node tree is working or why it is not working.

Note

Generally, the inspection tools display data from the last time the node tree has been evaluated. If it has not been evaluated, no information is available.

Socket Inspection

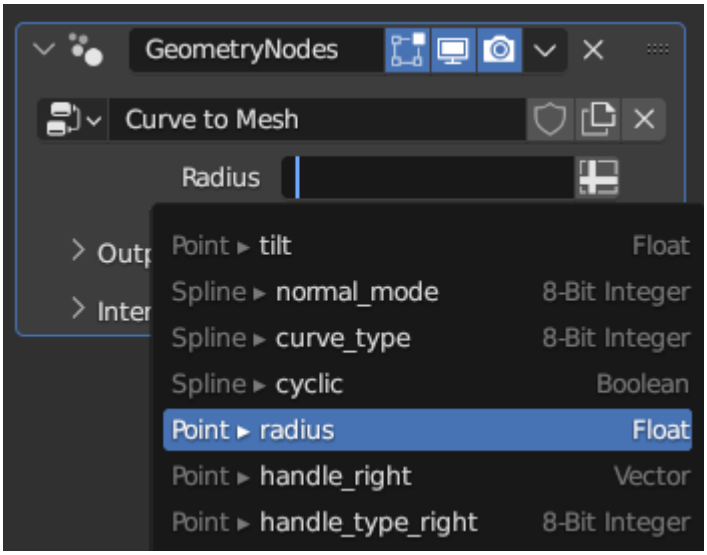


Socket Inspection.

Socket inspection shows information about the value in a socket during the last evaluation. For primitive data types such as integers, vectors, and strings the actual value is shown. For geometry sockets only some data about the geometry is stored, including the set of data types the geometry contains, and a count of their elements.

Socket values are only logged from when the node tree was executed, so a node must be connected to the *Group Output* to have a value for inspection. Values are not logged during rendering, to improve performance.

Attribute Search



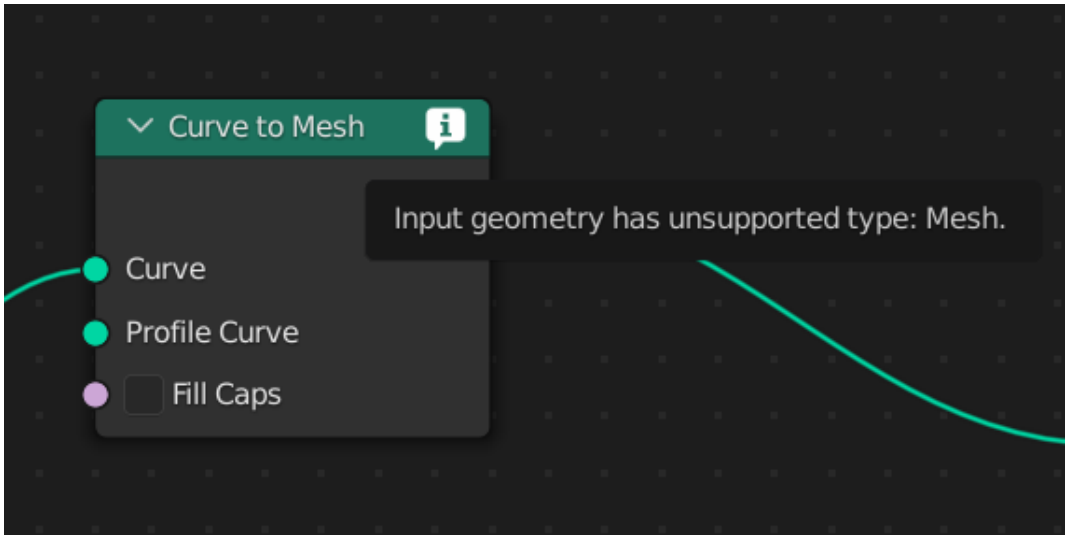
Attribute Search.

The attribute search is shown when clicking on an attribute input in the modifier. It contains a list of all the attributes that were available at that point in the modifier or node execution.

Viewer Node

The Viewer node is used to display intermediate geometry in the Spreadsheet Editor and the Viewport. For more information see Viewer Node.

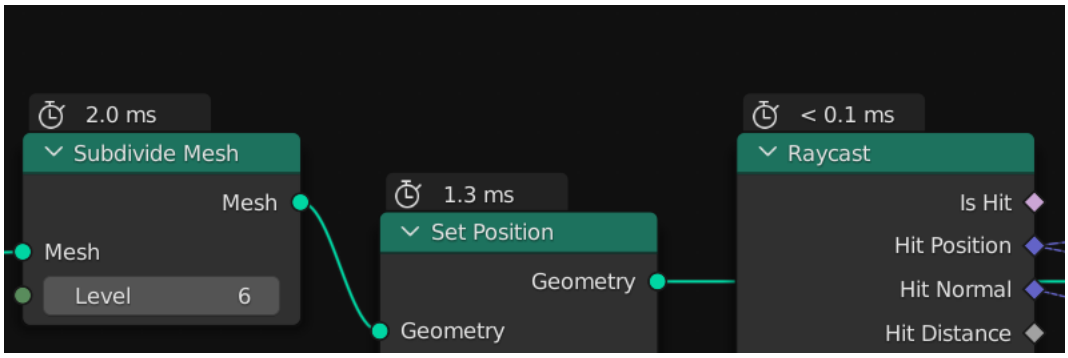
Node Warnings



Node Warning.

When the inputs to a node are invalid, it displays a warning in the title. Hovering over the warning icon shows the error message. These warnings are only generated when the node is executed, so a node must be connected to the *Group Output* to have a warning.

Node Timings Overlay



The node timings overlay.

Node timings show how long a node took to execute the last time the node group was evaluated. They can be turned on in the overlays popover on the top right of the node editor. When a node group is used in multiple places, the timings depend on the context of the node editor, which is displayed in the path on the top left.

Frame nodes display the total time from all of the contained nodes and the *Group Output* node displays the total time for the entire node group.

The displayed timings should only be considered an approximation, since they can also take into account actions like copying or deleting a geometry input that aren't part of the node's operation. Also, when a node uses multiple CPU cores, the evaluation system might work on other nodes at the same time. It's also important to remember that field nodes generally don't do work by themselves, so their execution time is only added to the data-flow nodes they are connected to.

Named Attributes Overlay

The "Named Attributes" overlay allows displaying when a custom named attribute is used by a node or a node group. Named attributes can be used by the Capture Attribute Node, the Named Attribute Node, and the Remove Named Attribute Node, and can be written to, read, or removed.

Using named attributes (as opposed to Anonymous Attributes) can be problematic when the original geometry already has attributes with the specified names. In that case a geometry node group might mistakenly overwrite some essential data. The overlay helps to make detecting that situation easy.

The same data is also available in the Named Attributes panel in the modifier's UI.

Geometry Randomization

Many nodes don't guarantee the order of elements in which they output things. For example, the order of edges coming out of the **Triangulate** node is deterministic but not well defined. The order may change between Blender versions. Therefor, if node setups depend on a specific order, they may break when the Blender implementation changes. Changing the order can often be necessary in order to fix bugs or improve performance.

“Geometry randomization” can be temporarily enabled to see if a blend-file depends on the indices in unstable ways. When enabled, various internal algorithms shuffle the result geometry elements so that any dependence on it would not work anymore. When building setups that are supposed to last a long time, it is recommended to check if they still work with randomization enabled.

To enable it, first enable **Developer Extras** in the preferences. Then search for **Set Geometry Randomization**. The popup allows enabling and disabling the randomization.

Attributes

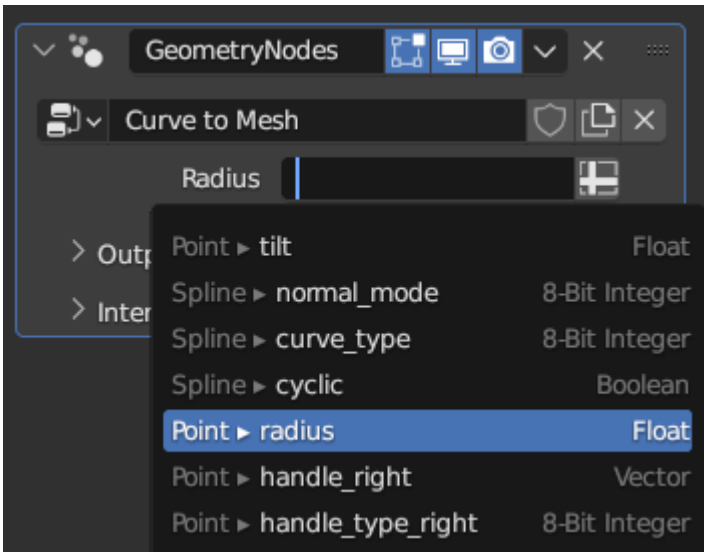
An *attribute* is a generic term to describe data stored per-element in a geometry data-block. For example, every vertex can have an associated number or vector. Attributes can be altered by connecting a value to the *Group Output* node, but also many nodes can change the values of specific attributes.

Note

Attribute data types and domains are converted implicitly where possible, just like node sockets.

Named Attributes

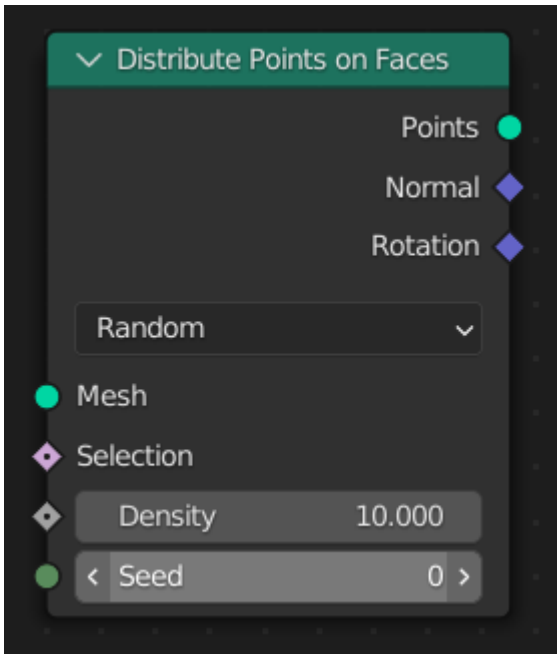
Named attributes are created and used in other areas of Blender like shaders, painting, and UV mapping. In the modifier panel, a named attribute can be used for input by clicking the icon to the right of the value button. The string input input allows you to search and choose existing attributes from the modifier's input geometry.



Attribute Search.

The attribute search gives a bit of context about each attribute. To the left of the menu, the attribute domain is shown followed by the attribute name. To the right of the menu, the attribute data type is shown.

Anonymous Attributes



The *Normal* and *Rotation* outputs are examples of attribute fields, which refer to an attribute stored on a geometry.

An anonymous attribute is a set of generic data stored on a geometry that doesn't have a name. Usually, attributes exposed in Blender's interface all have names. However, in geometry nodes, attributes can be passed around with node sockets. In these cases, an *Attribute Field* output is created, which is used by nodes to find attribute data in an input geometry.

Anonymous attributes are still stored on the geometry like other attributes, and they are even automatically interpolated when the geometry changes with other nodes, except for a few cases. So generally, if the node link is still accessible, the attribute it references will be available too. However, anonymous attributes cannot be connected to a completely separate

geometry that was created from a different source. To transfer attributes between separate geometries, the Sample Index Node or other similar nodes like the Sample Nearest Surface Node can be used.

Attribute Data Types

The type of an attribute is the kind of data stored at each element.

Boolean::	True or false value.
Integer::	32-bit integer.
8-Bit Integer::	Smaller integer with a range from -128 to 127.
Float::	Floating-point value.
Vector::	3D vector with floating-point values.
2D Vector::	2D vector with floating-point values.
Color::	RGBA color with 32-bit floating-point values.
Byte Color::	RGBA color with 8-bit positive integer values.
Quaternion::	Floating point Quaternion rotation.

The above list is in the order of least to most “complex” (An integer can contain more data than a boolean, so it is more complicated). When joining separate geometries together, the more complex data type is preferred when there are matching names. This is particularly important when joining geometry with named attributes with the Join Geometry Node.

To store 2D Vectors (UV maps) and Byte Colors the Store Named Attribute Node must be used because there are no sockets for those types.

Data Conversion

Through the use of Geometry Nodes, data can be converted between types.

Valid conversions:

- Between color and vector – mapping between color channels and vector components.
- Between color and float – the color data is converted to its grayscale equivalent.

- Between float and integer – integers simply become floats, floats are truncated.
- Between float and vector – when a float becomes a vector the value is used for each component. When a vector becomes a float the average of the components is taken.
- Between float and boolean – values greater than 0 are true, true maps to 1, and false maps to 0.

Attribute Domains

The domain of an attribute refers to what type of geometry element the attribute corresponds to. Knowing the domain of an attribute is important because it defines how it may be interpolated and used in nodes and shading. You can use the Spreadsheet Editor to determine the domains of attributes.

- **Point** domain attributes are associated with single locations in space with a position:
    - Vertices of a mesh
    - Points of a point cloud
    - Curve control points

- **Edge** domain attributes are associated with the edges of a mesh.
- **Face** domain attributes are associated with the faces of a mesh.
- **Face Corner** domain attributes are associated with the corners of the faces of the mesh. An example is a UV map attribute.
- **Spline** domain attributes are associated with a group of connected curve control points.
- **Instance** domain attributes exist on the Instances in a geometry. They can be used to store different values on copies of geometry data. Instance domain attributes are only supported in geometry nodes.
- **Layer** domain attributes are associated with a Grease Pencil Layer.

Attributes are automatically interpolated to other domains. For example, when the Position Node is connected to the selection input of the Set Material Node node, the values are interpolated from the *Point* domain to

the *Face* domain. Normally, domain conversions use simple averages for values, but *Boolean* data type attributes have special rules for interpolation:

Boolean Domain Interpolation

From	To	Conversion
Point	Edge	An edge is selected if both of its vertices were selected.
Point	Face	A face is selected if all of its vertices were selected too.
Point	Corner	Each corner's value is simply a copy of the value at its vertex.
Point	Spline	A spline is selected if all of its control points were selected.
Edge	Point	A vertex is selected if any connected edge was selected.
Edge	Face	A face is selected if all of its edges are selected.
Edge	Corner	A corner is selected if its two adjacent edges were selected.
Face	Point	A vertex is selected if any of the connected faces were selected.

From	To	Conversion
Face	Edge	An edge is selected if any connected face was selected.
Face	Corner	Each corner’s value is simply a copy of the value at its face.
Corner	Point	A vertex is selected if all connected face corners were selected and it is not a loose vertex.
Corner	Edge	An edge is selected if all corners on adjacent faces were selected.
Corner	Face	A face is selected if all of its corners were selected.
Spline	Point	Each point’s value is simply a copy of the corresponding value of the spline.

Built-In Attributes

Built-in attributes always exist, and cannot be removed. Their data type and domain cannot be changed.

Name	Type	Domain	Notes
`position`	*Vector*	*Point*	Built-in attribute describing vertex or point locations, in the local space of a geometry. Any node that changes the location of points will adjust this attribute, like the Transform Geometry Node and the Set Position Node.
`radius`	*Float*	*Point*	A built-in attribute on point clouds used to set the size for the points in the viewport. Also built-in on curves, where it controls the size of each curve control point when converted to a mesh, or for other operations.
`id`	*Integer*	*Point*	Created by the Distribute Points on Faces to provide stability when the shape of the input mesh changes, and used on instances to create motion blur. Values are expected to be large, with no order. This attribute is used by nodes that generate randomness, like the Random Value Node. Unlike other built-in attributes, this attribute is not required, and can be removed.
`material_index`	*Integer*	*Face*	Used to specify the material slot for every face in a mesh.

Name	Type	Domain	Notes
`sharp_edge`	*Boolean*	*Edge*	Attribute determining if an edge should have flat (rather than smooth) shading enabled in the viewport or a render.
`sharp_face`	*Boolean*	*Face*	Attribute determining if a face should have flat (rather than smooth) shading enabled in the viewport or a render.
`resolution`	*Integer*	*Spline*	Determines the number of evaluated points between two control points of a spline. Only NURBS and Bézier splines have this attribute, for poly splines, the value is always one.
`cyclic`	*Boolean*	*Spline*	Determines whether the spline has a segment that connects its first and last control points.
`handle_left`	*Vector*	*Point*	Describes the location of the left handle of a curve control point, on the side of the curve's start. Only exists when the curve contains a Bézier spline.
`handle_right`	*Vector*	*Point*	Describes the location of the right handle of a curve control point, on

Name	Type	Domain	Notes
			the side of the curve's end. Only exists when the curve contains a Bézier spline.

Naming Conventions

These attributes do not exist by default, but are used implicitly by certain parts of Blender. The data type of these attributes can be changed, just like any attribute besides the built-in attributes. However, the attributes might be expected by Blender to have a certain type.

Name	Type	Domain	Notes
`velocity`	*Vector*	*Point*	Used to create motion blur when rendering animations.
`rest_position`	*Vector*	*Point*	Holds the position of points or vertices from before a geometry is deformed procedurally. Can be created automatically before Shape Keys and Modifiers are evaluated with the Add Rest Position option.
`surface_uv_coordinate`	*2D Vector*	*Curve*	Used to describe curve attachment locations on a

Name	Type	Domain	Notes
			mesh surface, typically used for the hair system.
`crease_vert`	*Float*	*Point*	Vertex attribute used by the Subdivision Surface modifier. The values are expected to be in a range of 0 and 1.
`crease_edge`	*Float*	*Edge*	Edge attribute used by the Subdivision Surface modifier. The values are expected to be in a range of 0 and 1.
`uv_seam`	*Boolean*	*Edge*	True if an edge is considered a boundary between UV islands when unwrapping.
`bevel_weight_vert`	*Float*	*Point*	Used as vertex control for the bevel modifier.
`bevel_weight_edge`	*Float*	*Edge*	Used as edge control for the bevel modifier.
`sculpt_face_set`	*Integer*	*Face*	Used by the Sculpt Face Sets Feature.

Name	Type	Domain	Notes
`sculpt_mask`	*Float*	*Point*	Used by the Sculpt Masking Feature.
`sculpt_face_set`	*Integer*	*Face*	Used by the Sculpt Face Sets Feature.

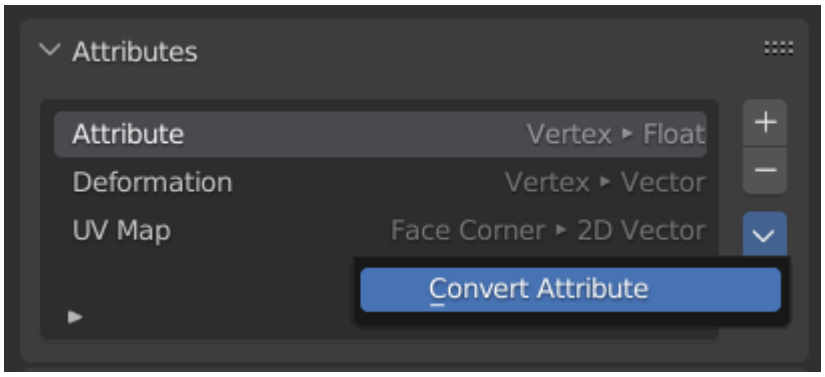
Custom Attributes

Vertex groups, UV maps and Color Attributes are available as attributes in geometry nodes. They are referred to by their name. Naming collisions (e.g. a vertex group and a UV map with the same name) should be avoided. If there is a naming collision, only one of the attributes is accessible in geometry nodes.

Attributes with any other name can also be created by nodes, when the name is used for the first time.

Note that geometry nodes does not always produce e.g. vertex groups if a node like Join Geometry is used. Similarly, if the data type of a vertex group attribute is changed from the initial “Float” type, the attribute will no longer be a vertex group.

Attribute Conversion Operator



This operator found in the *Attributes* panel of the property editor can change the domain or data type of an attribute.

Due to ongoing development in the area of attributes, many areas of Blender can not yet work with the generic (identified with a name, stored on any domain with any data type) attributes used by geometry nodes. That makes this operator an essential workaround in some cases where existing tools must be used with data generated from geometry nodes.

Mode

Generic:: Interpolate and convert the attribute between the domains and data types described on this page.

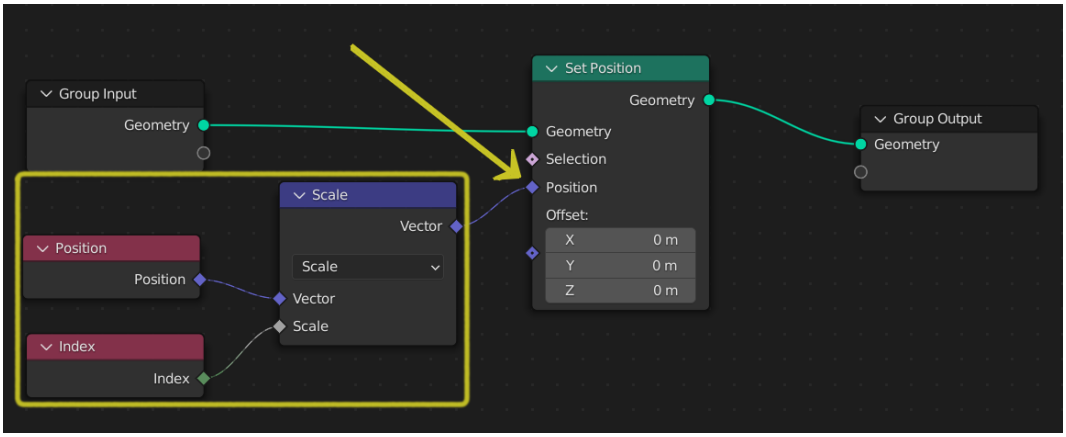
Vertex Group:: Create a Vertex Group from the attribute, which corresponds to a float attribute on the point domain.

Note

This operator only works on *original* object data, not including the results of modifiers, so any attributes added or changed by geometry nodes will not be affected. To change the type of an attribute generated procedurally, modifiers must be applied.

Fields

Fundamentally, a field is a function: a set of instructions that can transform an arbitrary number of inputs into a single output. A field's result can then be calculated many times with different input data. They are used all over geometry nodes to allow calculations that have different results for every element (mesh vertices, faces, etc.).



A field input to a node.

For example, in the figure above, the field connected to the "Set Position" node depends on two inputs, Position and Index, and transforms them into a vector using a single instruction.

Field Visualization

Socket shapes are used to convey which sockets are fields and which are regular data. There are three possible socket shapes, each visualizing its "field status":

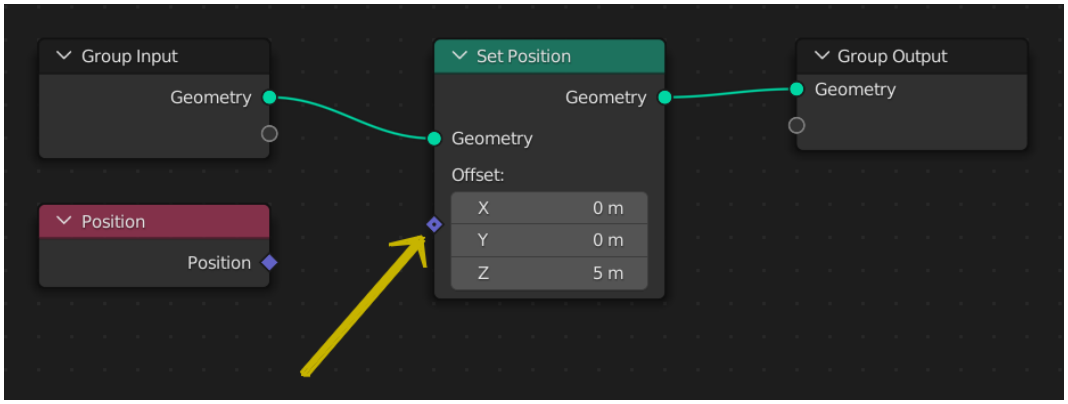
Circle:: The socket requires a single real value, it cannot accept a field input. For output sockets, this means the node always outputs a single value.

Diamond::

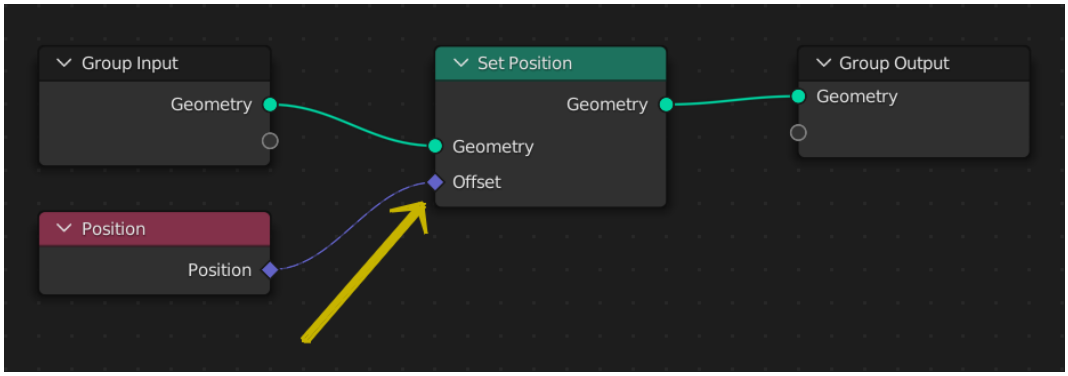
The socket can accept a field input, or it outputs a field. A constant single value can be connected to these sockets, but then the output will often not vary per element.

Diamond with Dot::

The socket can be a field, but it is currently a single value. This is helpful because it allows tracking where single values are calculated, instead of a field with many different results. It also means that Socket Inspection will show the value instead of field input names.



The socket shape is a diamond with a dot, meaning the field has the same value for every element. Every point will be moved up by 5 m.



The socket shape is a diamond and the field input now has a varying input. In other words, the value can be different for every

element. In this case, the position will be doubled, since the offset for every point is the point's position.

Tip

Often it is desired to extract a single value from a field. While it doesn't make sense conceptually to simply change a field into a single value, the Sample Index Node or the Attribute Statistic Node can be used to retrieve a single value from a field evaluated on a geometry.

When a connection is made between two node sockets that support fields the node connection will be drawn as a dashed line. If you make the mistake of connecting a non-field socket to a field socket, the connection will be drawn as a solid red line indicating that there is an error.

Node Types

Nodes can be separated into two categories: data flow nodes that usually pass geometry, and field nodes that operate on data per-element. Field nodes can be input nodes that bring geometry data into the node tree, or function nodes that operate on that data.

Data Flow Nodes

Nodes with a geometry input and a geometry output will almost always be data flow nodes. Meaning that they actually change geometry data that will be outputted from the Geometry Nodes modifier.

Function Nodes

Nodes with diamond socket inputs and outputs are field nodes, and resemble the instructions that will be evaluated by data flow nodes. Examples of function nodes are the math nodes and also more complex nodes like the Geometry Proximity Node.

Input Nodes

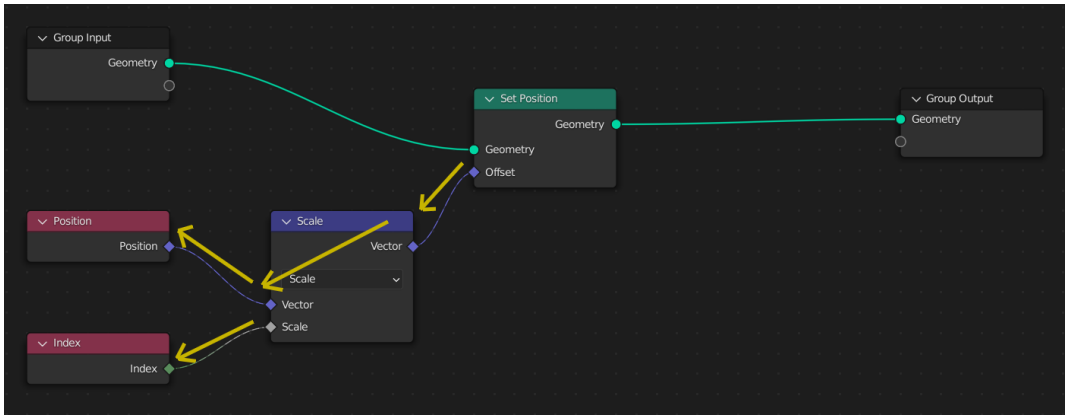
Input nodes provide data to the field evaluation process. By themselves, they mean nothing; they must be evaluated within the context of a data flow node (geometry) to actually output a value. Examples of input nodes are the built-in attribute input nodes like Position and ID, but also selection nodes like Endpoint Selection.

Field inputs may also come from other nodes that process geometry like the Distribute Points on Faces, in the form of Anonymous Attributes.

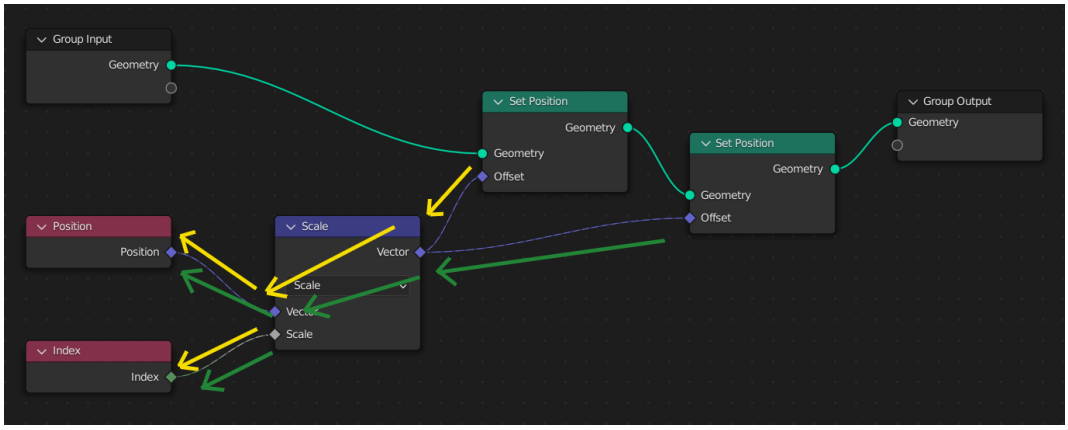
Field Context

All field nodes work in the context of the data flow node they are connected to. The context usually consists of a geometry component type and an attribute domain, so it determines what data is retrieved from input nodes.

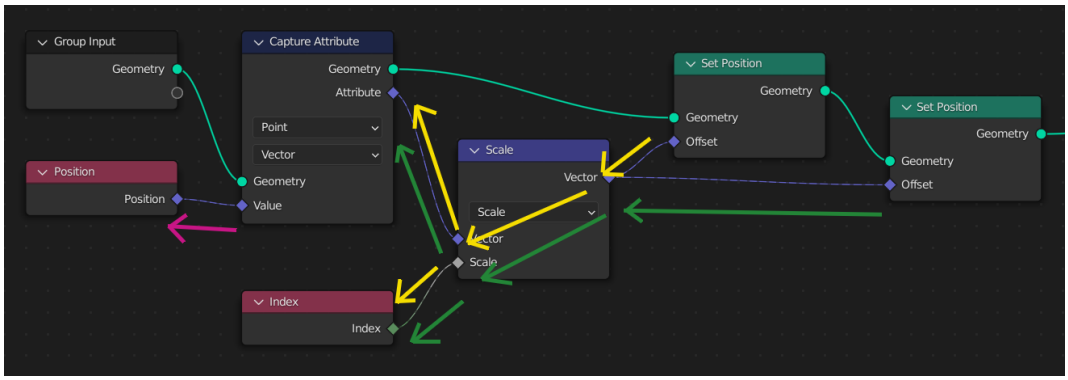
One common misunderstanding is that the same field node tree used in multiple places will output the same data. This is not necessarily true, because the field node tree will be evaluated for every data flow node, potentially retrieving data from a different or changed geometry.



Here, the Set Position node's input field is evaluated once. To evaluate the field, the node traverses backwards to retrieve the inputs from the field input nodes.



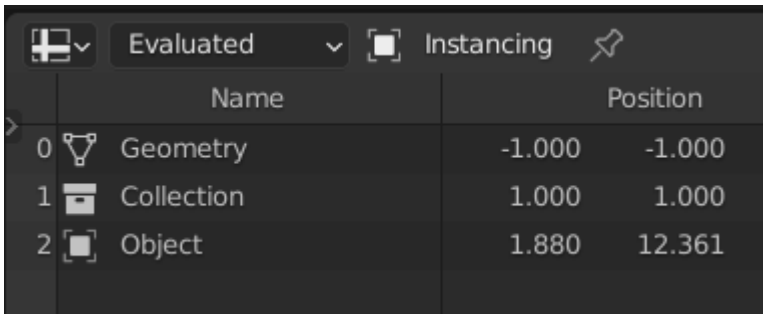
When a second Set Position node is added, the same field node tree is evaluated twice, once for each data flow node. At the second Set Position node, the results will be different since its geometry input will already have the changed position from the first node.



However, often it's necessary to use the same field values even after changing the geometry. The Capture Attribute Node evaluates a field, copying the result to an anonymous attribute on the geometry.

Here, a Capture Attribute node stores a copy of the initial position. Notice that evaluating the field input of the Capture Attribute node is an entirely different step. Later on, the input fields to the Set Position nodes don't use the actual position, but the anonymous attribute copy of it.

Instances



The three types of instances.

In addition to storing real data like a mesh or a curve, objects can store instances, which themselves can reference more geometry, an object, or a collection. The purpose of instancing is to allow duplicating geometry and storing it in an object, without duplicating the actual data. This optimization allows render engines like Cycles to handle the same geometry data in many different locations better than when the data is duplicated.

Each instance keeps track of which geometry it corresponds to and how the instanced is transformed compared to it's source geometry. Instances can also store the `id` attribute, which is used to correct motion blur when instances move in an animation.

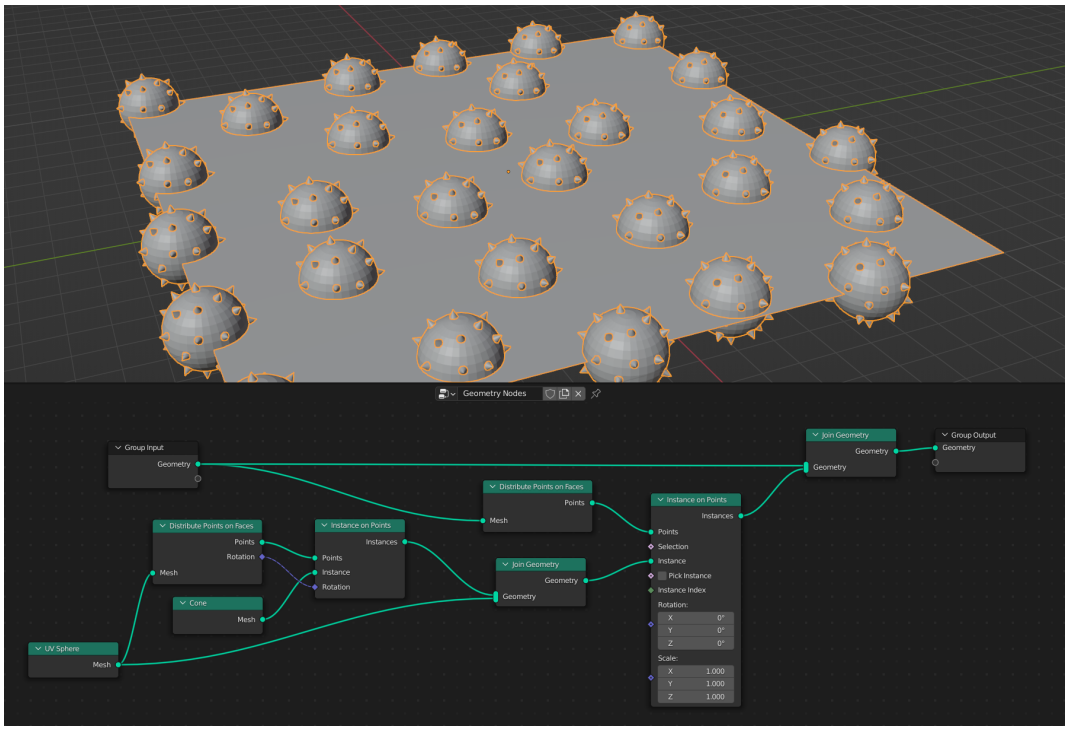
Instances can be created with geometry nodes using the Instance on Points Node.

Warning

Currently instancing from geometry nodes cannot be mixed with instancing from the Instancing panel in the property editor.

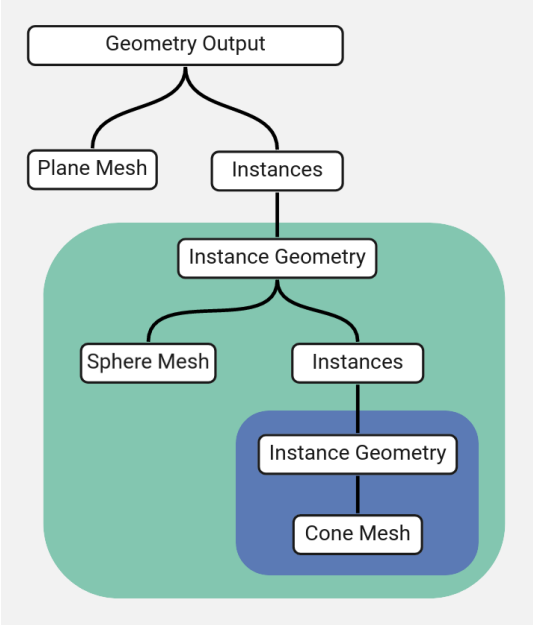
Nested Instancing

Since instances can store a geometry, and a geometry can contain instances, nested instancing is possible. In other words, it is possible to instance an instance, or even a collection of instances. For example, by default, the Instance on Points Node will create nested instances by instancing instances on the points of real geometry.



A node group that creates nested instancing by chaining Instance on Points nodes.

Here, nested instancing is used to distribute geometry that contains both a mesh and instances. The output geometry contains a “real” mesh and a group of instances. Each instance contains a sphere mesh and many instances of a cone geometry.



The tree of instanced geometry for the example above.

What makes this method helpful is the output geometry only contains three unique meshes: the plane, the sphere, and the cone. This would make the performance much better if the meshes were more complicated.

Warning

Only eight levels of nested instancing are supported for rendering and viewing in the viewport. Though deeper trees of instances can be made inside geometry nodes, they must be realized at the end of the node tree.

Realizing Instances

The term “realizing” instances referes to converting the instances into unique geometry. When instances are realized they will take up more memory and manipulation to geometry will have to be processed individually rather the once per instancing geometry.

To realize instance use the Realize Instances Node.

Instance Processing

Almost all nodes that process geometry do so by processing each unique geometry separately rather than realized geometry. For example, if a Subdivision Surface Node was placed at the end of the example above, it would only have to subdivide three meshes, rather than each instance of a mesh. Another important example is processing with the output of the String to Curves Node, where each unique character only has to be processed once.

This method can improve performance a lot, but it means that the result of an operation will be the same for every instance of a certain geometry. In order to have unique results for every instance, the Realize Instances Node node can be used.

Baking

Baking allows saving and loading intermediate geometries. Baking parts of the node tree can be used for better performance.

The data format used to store geometry data is not considered to be an import/export format. Volume objects, however, are saved using the OpenVDB file format which can be used interoperably.

Data can be baked using two methods:

- Bake Node – used to bake any portion of the node tree.
- Simulation Zone Baking – used to bake animations where the result of one geometry state can influence the next state.

Data is not stored to disk until the Bake operation is ran.

Important

- Blend-files must be saved to a disk before data can be baked.
- It's not guaranteed that data written with one Blender version can be read by another Blender version.

Bake Geometry Node

Bakes the simulations and bake nodes in all modifiers for the selected objects.

Tip

When data is baked the number of baked frames is displayed above the simulation zone or bake nodes.

Data-Block References

Reference

Editor:: Geometry Node Editor
Panel:: *Sidebar ▸ Node ▸ Data-Block References*

Baked geometries that reference other data-blocks such as materials are listed here.

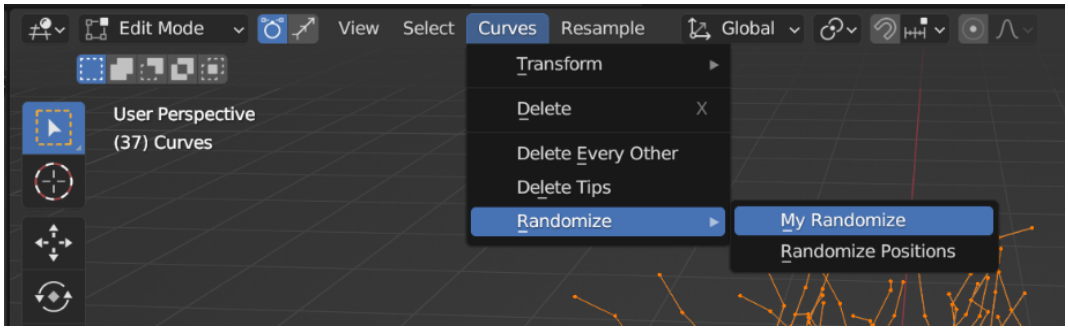
This panel allows changing these references after the data has been baked.

Note

Currently only material data-blocks are supported.

Node-Based Tools

Geometry Nodes can be used to expand the core functionality of Blender via node-group-defined tools. They can be shared as any regular node group assets.



Node group tools integrated in the Selection menu.

Tool Context

The way to create Node-based tools is by switching the Geometry Nodes editor context to *Tool*.

New node groups created in the tool context will be enabled as Tool by default, although users still need to set them as Assets if they want to share them (see below).

Note

The Inspection features are not supported in the *Tool* context: Viewer Node and Socket Inspection.

Asset

For a node group to be shared as a tool, it has to be an Asset first. The asset catalog is used to determine on which menus the node group will show, similar to the regular node group assets. If the catalog name matches an existing menu, the tool will be added to the end of it. Assets that have no catalog assigned to them, or local tools are exposed in the "Unassigned" menu.

The asset options need to be set on the Asset Browser.

Tool Settings

The node group inputs will be exposed as in the Adjust Last Operation panel.

Supported Modes & Data-Types

Node groups must specify which modes and object types they support. This helps to determine where the tool is available in the user interface. These properties can be configured in popover menus in the Geometry Node editor when in the *Tool* context.

Currently only Object, Edit, and Sculpting modes are supported, and only for the Mesh and Hair Curves object types.

Important

For mesh objects, Shape Keys are not supported. Operating a node tool on a mesh with shape keys will remove the shape key data.

Tool-specific Nodes

The following nodes are only supported in the *Tool* context:

- 3D Cursor Node
- Mouse Position Node

- Viewport Transform Node
- Active Element Node
- Selection Node
- Set Selection Node
- Face Set Node
- Set Face Set Node

Note

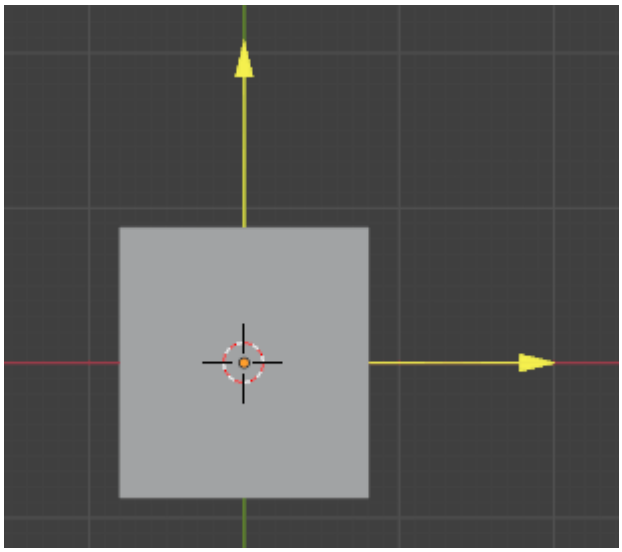
The Self Object node returns the Active object when inside a *Tool* node group.

Non-supported Nodes

These nodes are only supported in the *Modifier* context:

- Simulation Zone
- Viewer Node

Gizmos

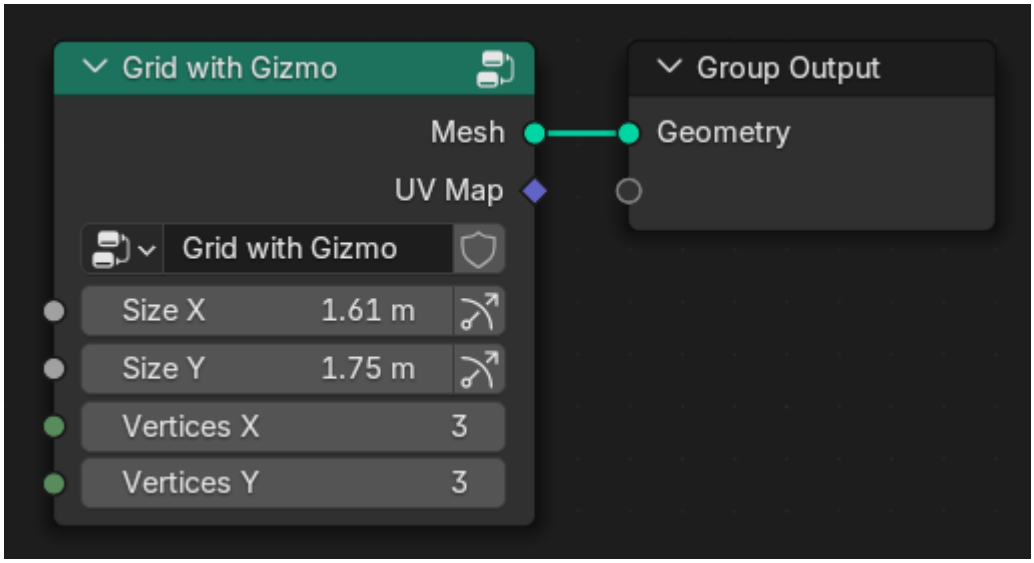


Grid with gizmo in the 3D viewport.

Gizmos allow changing inputs of Geometry Nodes directly in the 3D viewport. This is often more intuitive than controlling the inputs in the modifier or node editor.

Using Gizmos

Node inputs that can be controlled with a gizmo have an additional icon. The gizmos of that node are shown if it is selected. Clicking on the icon pins the gizmo, so that it is shown even if the node is not selected. Modifying the gizmo in the 3D viewport, modifies the value in the socket.

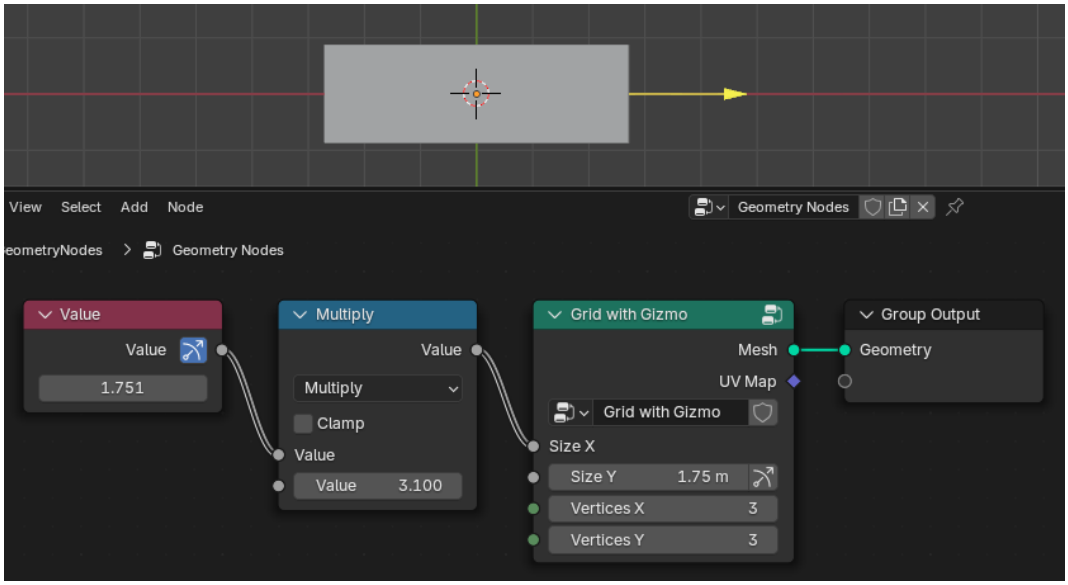


Example of a node group that has gizmos.

Note

Built-in nodes do not have their own gizmos yet. It's possible to create node groups that have gizmos though.

Gizmos are often automatically propagated when an input socket with a gizmo is linked. The gizmo then controls the value that it is propagated to, instead of the input of the node group directly. Not all nodes support propagating gizmos, but many math operations do. A double link indicates that the propagation was successful.



The gizmo is propagated from the Size X input socket to the Value node.

Gizmos can also be propagated to Group Inputs, in which case they are also available on the parent group node. If the current group is used by a modifier directly, the gizmo will also be available on the modifier. Gizmos that are propagated to the modifier always show when the modifier is active, independent of whether any node is visible or selected.

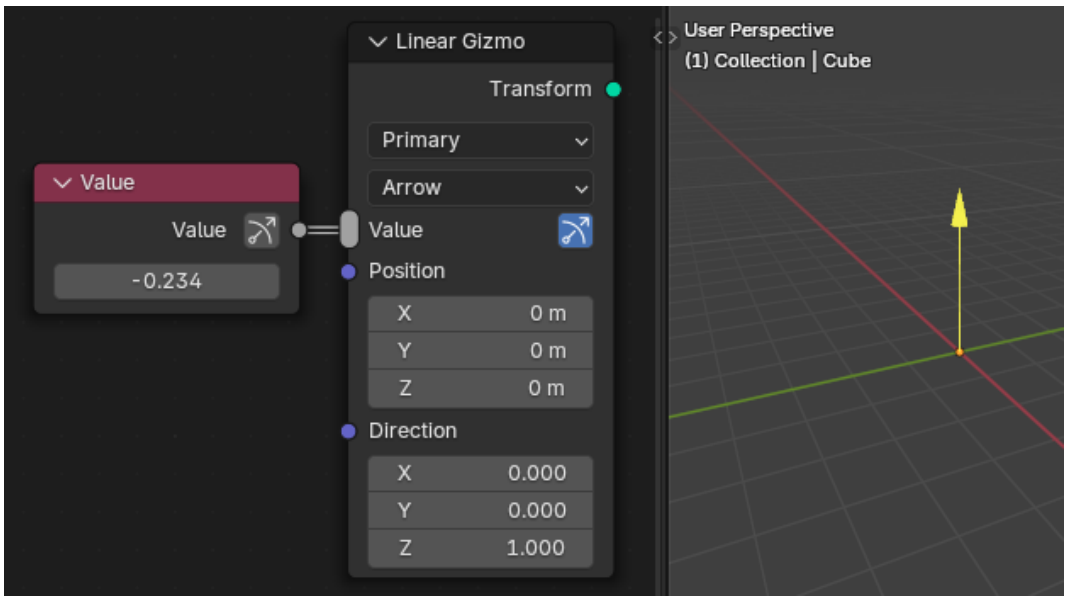
Creating Custom Gizmos

Adding custom gizmos to a node group that generates or modifies geometry can make it more convenient to use.

To add a gizmo to a node group, one has to use one of the gizmo nodes. The main aspect that makes gizmos unintuitive at first is that there is a **bidirection dependency**: changing the gizmo position changes the controlled value and vice versa.

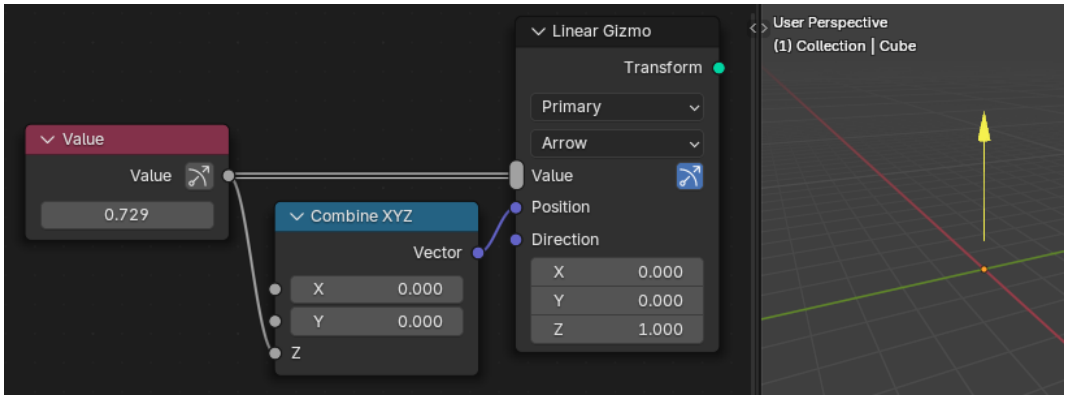
The most simple custom gizmo setup is shown below. The Linear Gizmo node adds a gizmo that is drawn in the 3D viewport. The gizmo controls the value that is plugged into it. When trying this, you will notice that the gizmo always jumps back to the origin while the value is still changed. That

is because the Position of the gizmo node does not dependend on the value yet.



The simplest custom gizmo setup.

When the gizmo position is made dependent on the value, the gizmo works more than one would expect. It now also works in both directions: changing the value moves the gizmo and moving the gizmo changes the value.

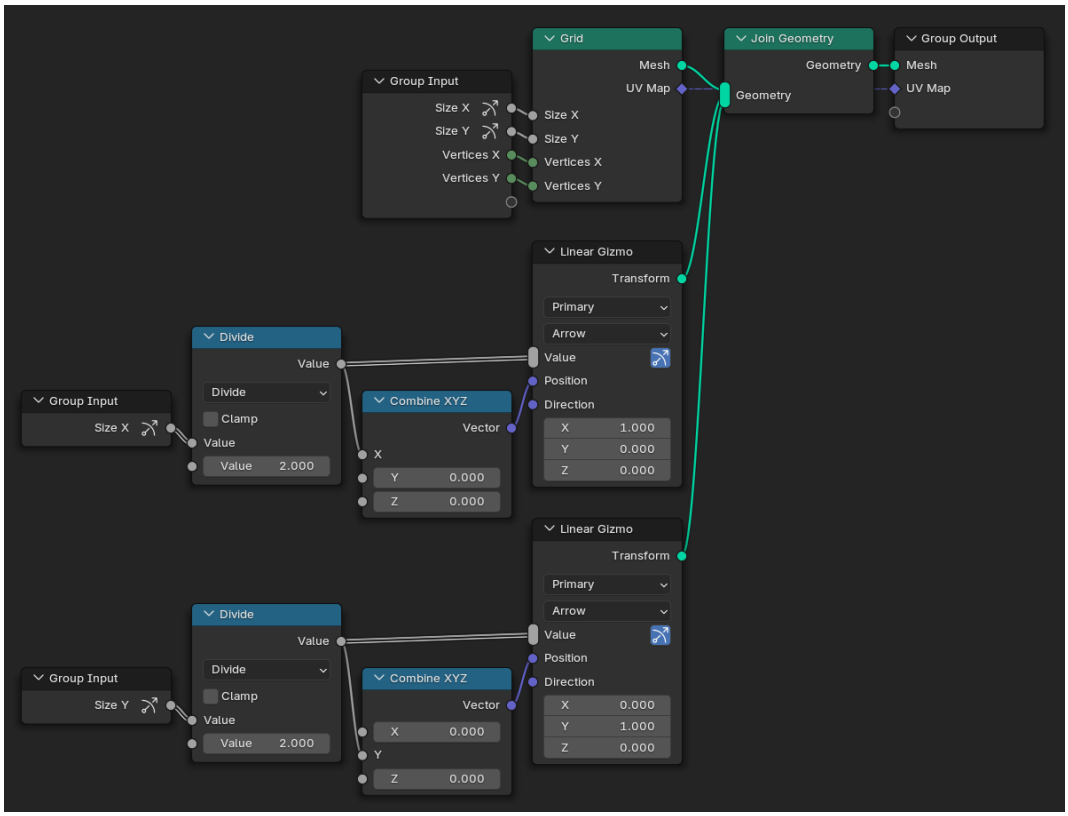


Simple gizmo setup where the gizmo position depends on the controlled value.

Multiple values can be plugged into the Value input of gizmo nodes at once. In that case, all these values are modified at the same time when moving the

gizmo. Multiply or divide nodes can be used if the values should change at different rates.

The Transform output of gizmo nodes should be joined into the geometry that the gizmo controls. This helps Blender to understand that the gizmos should be transformed together with the geometry later on.



Example showing how to add simple gizmos to the built-in Grid node.

Note

Generally, it is possible to build the entire node group functionality first and to add gizmos afterwards.

Attribute Nodes

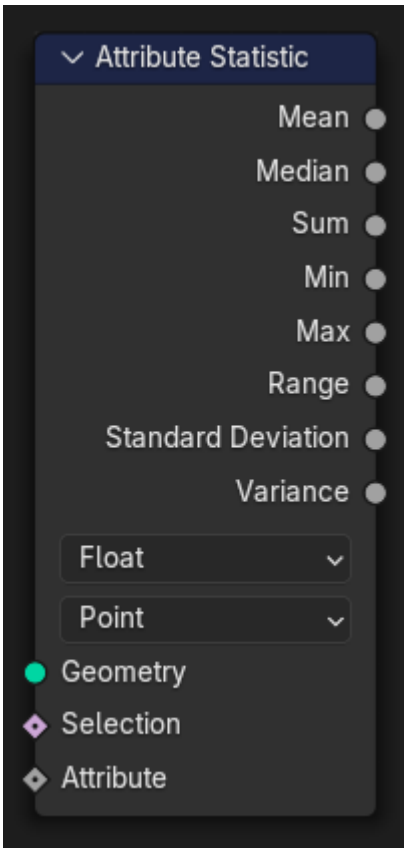
Nodes for working with data stored per object element, e.g. vertex groups.

- Attribute Statistic Node
- Domain Size Node

- Blur Attribute Node
- Capture Attribute Node
- Remove Named Attribute Node
- Store Named Attribute Node

Attribute Statistic Node

The *Attribute Statistic* node evaluates a field on a geometry and outputs a statistic about the entire data set.



Inputs

Geometry
: Standard geometry input.

Selection
: A boolean field input for each element indicating whether to include its value in the statistics result If the boolean is false, the corresponding value from the *Attribute* input will be ignored.

Attribute
: The attribute field to query a statistic from.

Properties

Data Type

Float:: The output will be a single floating-point value.

Vector:: The output will be a vector of three floating-point values. All calculations are elementwise.

Domain

The attribute domain used for the statistics and to evaluate the input *Attribute* field on.

Outputs

Mean

The average value of all data.

Median

The median value of all data.

Sum

The sum value of all data.

Min

The min value of all data.

Max

The max value of all data.

Range

The difference between the max and min value.

Standard Deviation

How much values differ from the mean. A low standard deviation indicates that the values are grouped tightly together at the mean. A high standard deviation indicates that the values are spread out over a large range.

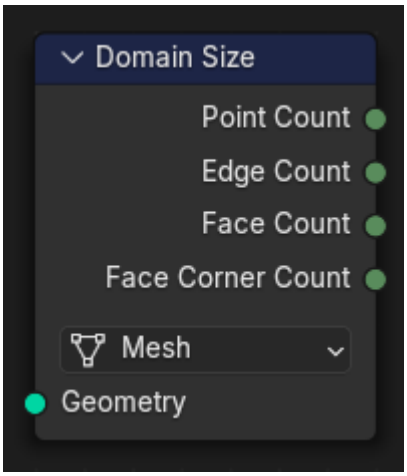
Variance

The variance of all data, defined as the square of the standard deviation.

Domain Size Node

The *Domain Size* outputs the size of an attribute domain on the selected geometry type, for example, the number of edges in a mesh, or the number of points in a point cloud.

For more information about attribute domains, see the geometry attributes page.



Inputs

Geometry
: Standard geometry input.

Properties

Component
: Which geometry type to retrieve the domain sizes from.

Outputs

Point Count
: The size of the *Point* domain on any supported geometry.

Edge Count
: The size of the *Edge* domain on meshes.

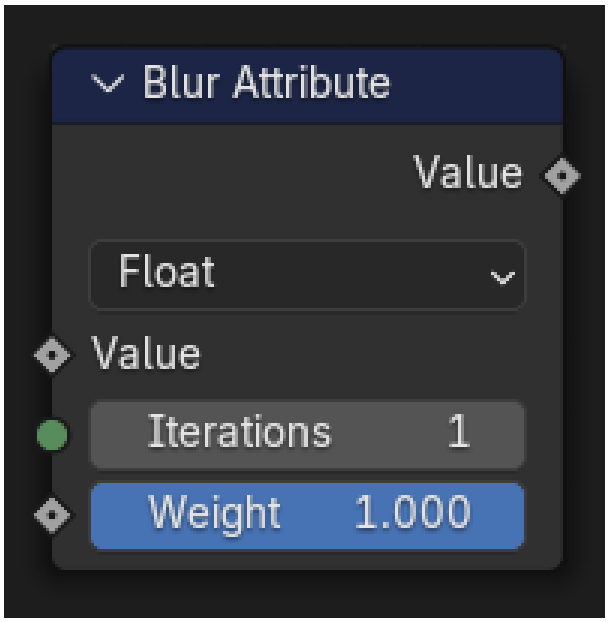
Face Count
: The size of the *Face* domain on meshes.

Face Corner Count
: The size of the *Face Corner* domain on meshes.

Spline Count
: The size of the *Spline* domain on curves.

Instance Count
: The number of top-level instances in the geometry. Nested instances are not considered.

Blur Attribute Node



The *Blur Attribute* node smooths attribute values between neighboring geometry elements.

The goal of each step is mixing values of each element with its neighbors. The weight for element is factor for multiplying all neighbor's values before accumulating them as new primitive value.

Blurring will only work with certain geometry types and attribute domains. Therefore, the attribute can only be affected on the Meshes and Curves components.

The domains this node works on is based on the field context of the node's evaluation. Only domains with explicit relations with their neighbors will work with this node. Explicit relations for correct blurring are vertices, edges, and faces of meshes, and curve control points.

Note

Blurring of face corner attributes is not handled by this node, because the ideal behavior for mixing face corner values is not clear.

All attribute data types are supported except for boolean attributes.

Inputs

Value
: The immediate value of each primitive to blur.

Iterations
: Number of repetitions of mixing value with neighbors. Each iteration is independent. Until one iteration is completed, its results are not used as a source for next blurring.

Weight
: Weight of each primitive.

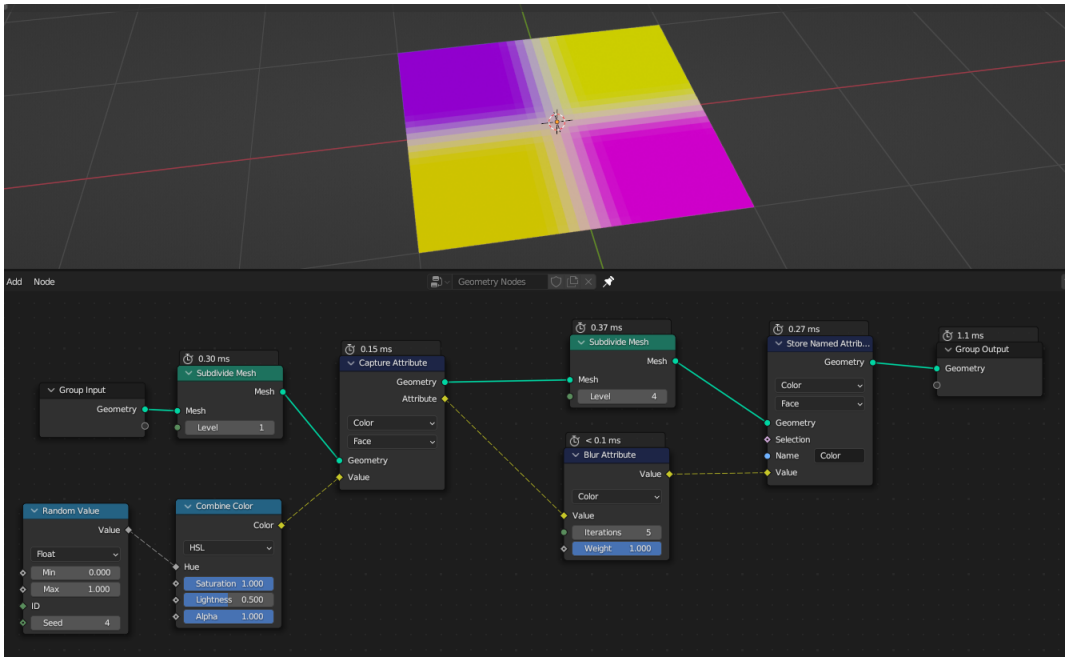
Properties

Data Type
: The data type used for the evaluated data.

Outputs

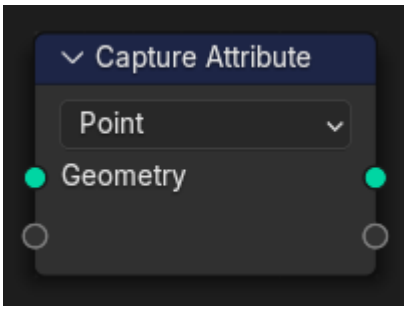
Value
: Values, mixed with neighbors defined number of times.

Examples



Input is Mesh Plane. First Subdivide Mesh Node add some faces for capture color with Random Value Node used as hue in Combine Color Node on this. Now second Subdivide Mesh Node split each face on a lot of new. Each one new duplicates original attribute. Blur Attribute node mixes all attributes for each face. Due to this, the result is smoothed.

Capture Attribute Node



The *Capture Attribute* node stores the result of a field on a geometry, and outputs the data as a node socket so it can be used by other nodes.

The result is stored on the geometry just like a regular attribute with a name, but instead of referencing it with a name, it is retrieved whenever the socket is connected to the input of a node. Later on when evaluating the node tree, the attribute will be removed automatically if it is no longer used.

This node is essential because field input nodes like the Radius Node work in the context of the node they are connected to. Meaning that in order to pass data like `radius` to a geometry that doesn't have radius, an explicit node link with the output of this node must be used.

Note

Because this node stores an anonymous attribute in the geometry, it's essential to use the geometry output for further operations in the node tree. The anonymous attribute will not exist for any other geometry besides the output.

Inputs

Geometry
: Standard geometry input.

Capture Items
: The input field to evaluate. More fields can be added by dragging sockets into the blank socket or in the *Capture Items* panel. Items can be

renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

Domain
: Which attribute domain to store the evaluated data on.

Capture Items

Reference

Menu:: *Sidebar ▸ Node ▸ Properties ▸ Capture Items*

Manages the input fields sockets of the node. Field sockets can be added, removed, reorganized, and renamed from the List View.

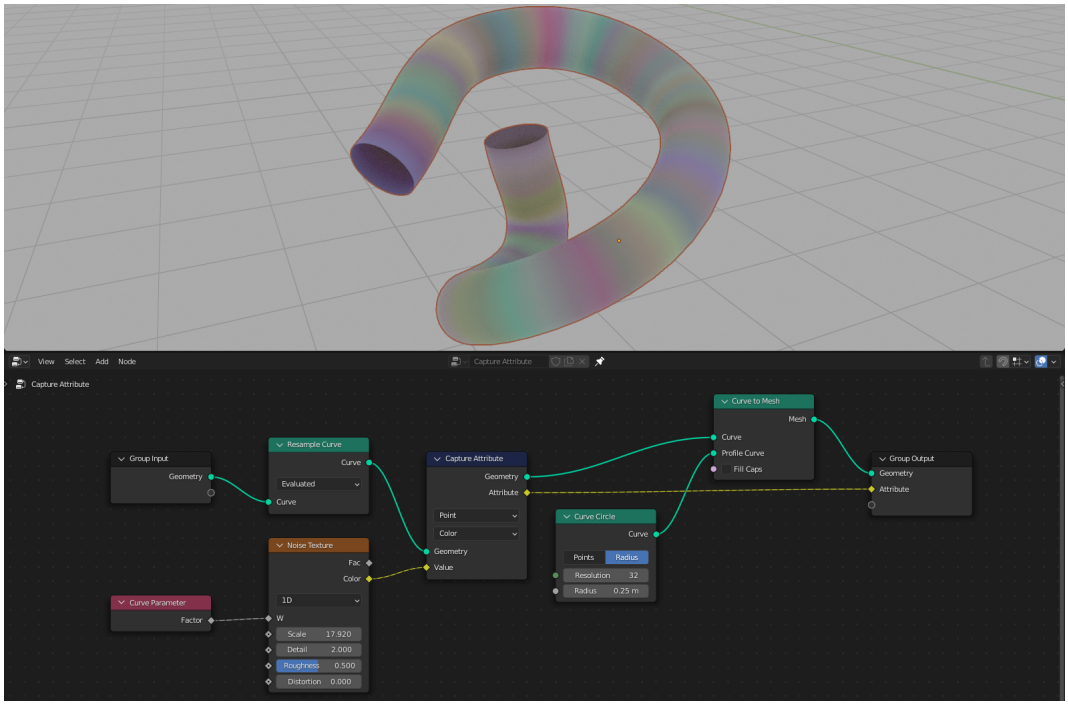
Data Type
: The data type used for the evaluated data.

Outputs

Geometry
: Standard geometry output.

Attribute
: The result of the evaluated field, stored on the geometry.

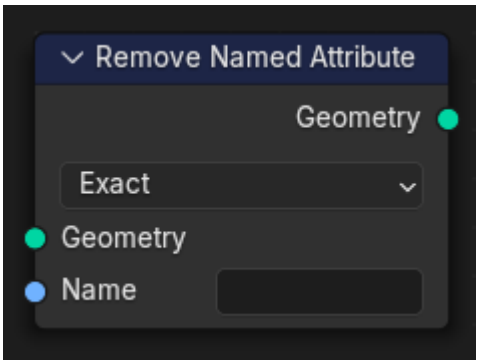
Examples



Here, a noise texture is evaluated in along the path of the curve in one dimension and rendered with a shader. The capture node is required because the output of the Curve to Mesh Node does not have a “curve parameter”, since it is a mesh and not a curve. So, the Spline Parameter Node must be evaluated while the geometry is still a curve.

Internally, after the noise texture is evaluated on the curve, it is automatically copied to the mesh result of the Curve to Mesh node. This means that anywhere *Attribute* output of this node can be connected along the same stream of geometry nodes, the internal attribute will be available.

Remove Named Attribute Node



The *Remove Named Attribute* node deletes an attribute with a certain name from its geometry input. Any attribute that exists on geometry data will be automatically propagated when the geometry storing it is changed, which can be an expensive operation, so using this node can be a simple way to optimize the performance of a geometry node tree or even to lower the memory usage of the entire scene.

Almost all named attributes can be removed. For certain Built-In Attributes, removing it will mean that a default value will be used instead. For example, removing the cyclic attribute on curves means that all curves will be non-cyclic.

Inputs

Geometry
: Standard geometry input.

Name
: The name of the attribute to remove.

Properties

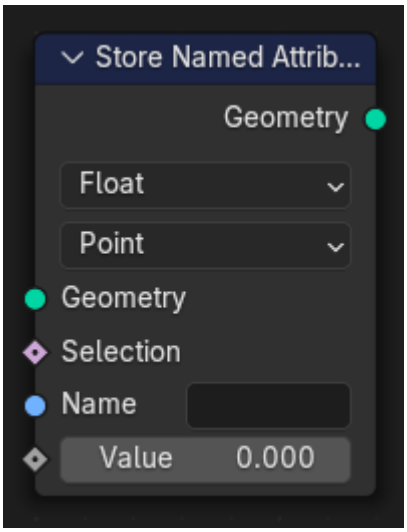
This node has no properties.

Outputs

Geometry

Standard geometry output.

Store Named Attribute Node



The *Store Named Attribute* node stores the result of a field on a geometry as an attribute with the specified name. If the attribute already exists, the data type and domain will be updated to the values chosen in the node. However, keep in mind that the domain and data type of Built-In Attributes cannot be changed.

Compared with the Capture Attribute Node, this node basically does the same thing, but the attribute gets a name instead of an anonymous reference. For reusing the data in the same node tree, the *Capture Attribute* node might be preferable since it does not create the chance for name conflicts in the input geometry.

Note

If the input geometry contains multiple geometry component types, the attribute will be created on each component that has the chosen domain.

Inputs

Geometry
: Standard geometry input.

Selection
: A boolean field input for each element indicating whether to store the attribute value for the given index. If the attribute does not exist, unselected parts are filled with zero values; otherwise, only the selected parts are filled.

Value
: The input field to evaluate.

Name
: The name to give the stored data.

Properties

Data Type
: The data type used for the evaluated data.

Domain
: Which attribute domain to store the evaluated data on.

Outputs

Geometry
: Standard geometry output.

Input Nodes

Nodes used mainly as input to other nodes.

- Constant
- Gizmo
- Scene

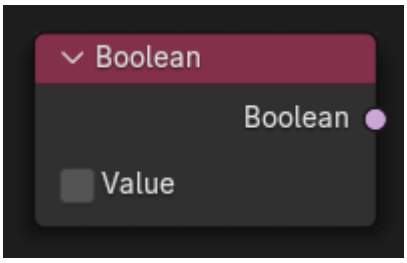
Input Constant Nodes

Nodes used mainly as input to other nodes.

- Boolean Node
- Color Node
- Image Node
- Integer Node
- Material Node
- Rotation Node
- String Node
- Value Node
- Vector Node

Boolean Node

The *Boolean* node provides a Boolean value.



Inputs

This node has no input sockets.

Properties

Single Boolean value (true/false).

Outputs

Boolean
: Standard Boolean output.

Color Node

The *Color* node outputs the color value chosen with the color picker widget.



Inputs

This node has no inputs.

Properties

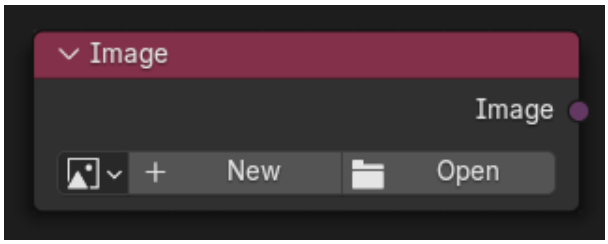
This node has no properties.

Outputs

Color
: Color value indicated by the color picker widget.

Image Node

The *Image* node provides access to a image file which allows you to conveniently enter and switch images for multiple nodes in the tree.



See also

Image node.

[Image Info Node]

Inputs

This node has no inputs.

Properties

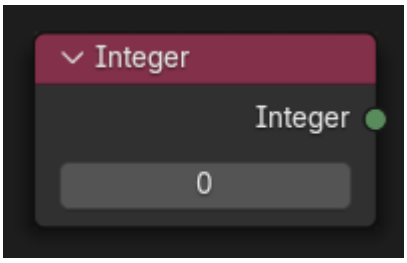
Image Data-Block
: The data-block selector to select an existing image or open an image from the file browser.

Outputs

Image
: The image file chosen from the data-block selector.

Integer Node

The *Integer* node provides an integer value.



Inputs

This node has no input sockets.

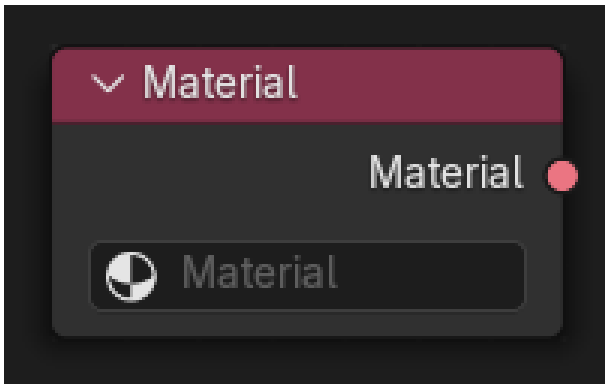
Properties

Single integer value.

Outputs

Integer
: Standard integer output

Material Node



The *Material* input node outputs a single material. It can be connected to other material sockets to make using the same material name in multiple places more convenient.

Tip

The Material node can also be added by dragging and dropping a material data-block into the node editor. This will add the node and select the dropped material in the Data-Block Menu.

Inputs

This node has no inputs.

Properties

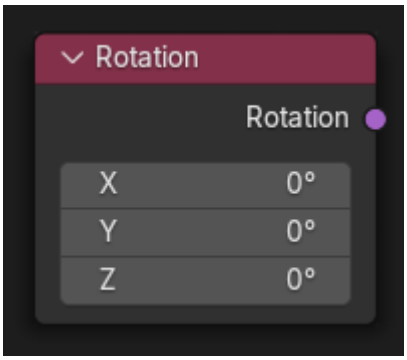
- Material

Output

Material
: A reference to the selected material.

Rotation Node

The *Rotation* input creates a rotation from euler rotation values.



Inputs

This node has no inputs.

Properties

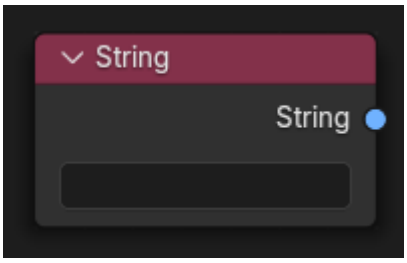
X, Y, Z
: The amount of rotation about each axes.

Output

Vector
: Standard rotation output.

String Node

The *String* input node creates a single string. It can be connected to attribute name sockets to make using the same attribute name in multiple places more convenient.



Inputs

This node has no inputs.

Properties

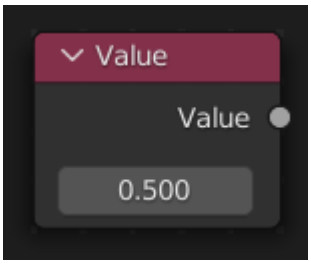
- String

Output

String
: Standard string output.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.



Inputs

This node has no input sockets.

Properties

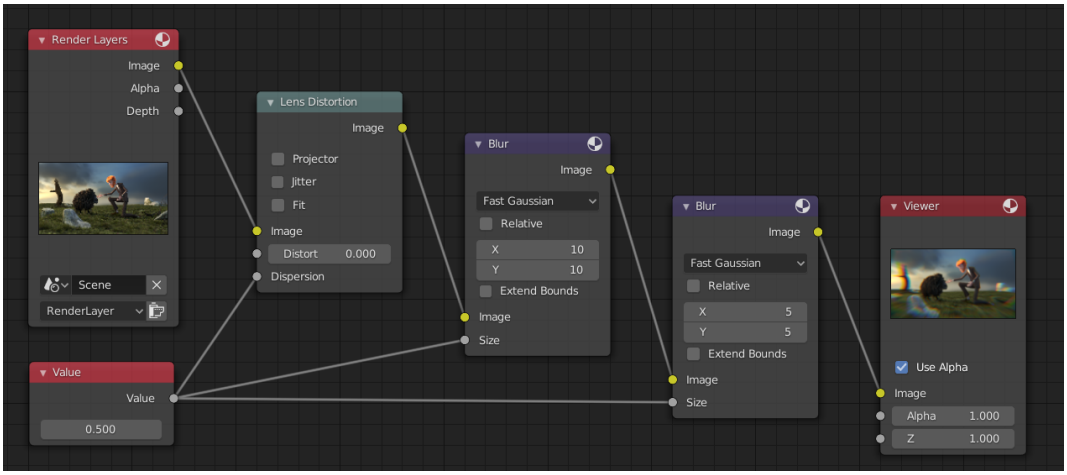
Single numerical value (floating-point).

Outputs

Value
: The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.



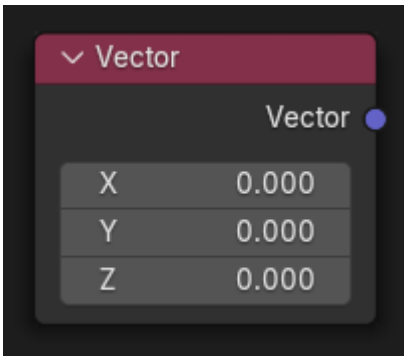
Example of the *Value* node.

Tip

From this you can also make different values proportional to each other by adding a Math Node in between the different links.

Vector Node

The *Vector* input node creates a single vector.



Inputs

This node has no inputs.

Properties

- X
- Y
- Z

Output

Vector
: Standard vector output.

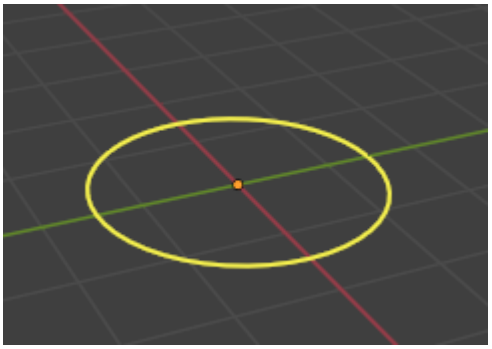
Gizmo Nodes

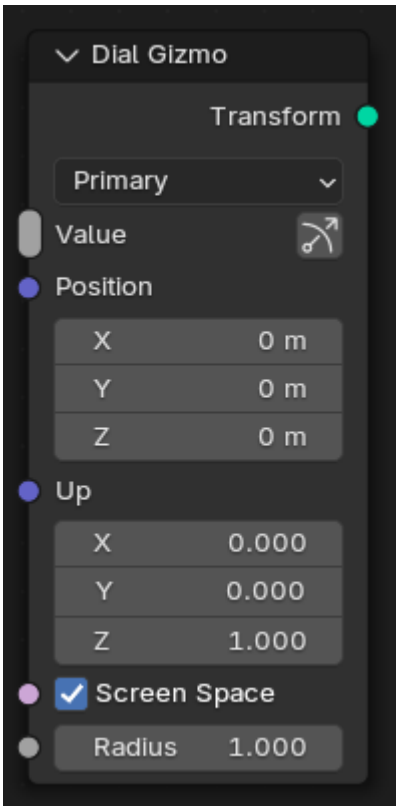
See this for a general guide on how to use gizmo nodes.

- Dial Gizmo
- Linear Gizmo
- Transform Gizmo

Dial Gizmo

The *Dial Gizmo* node is ideal for creating gizmos that control angles.





Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is rotated.

Position
: Position of the gizmo in the local space of the object.

Up
: Up or normal direction of the gizmo in the viewport.

Screen Space

If enabled, the gizmo will always have the same size in the viewport independent of the zoom level. This affects the meaning of the radius input.

Radius
: In screen space mode, this is a factor on top of the default radius. Otherwise, this is the radius of the gizmo in Blender units.

Properties

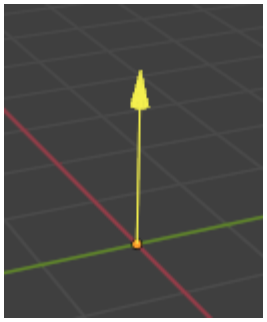
Color
: Controls which theme color is used for this gizmo.

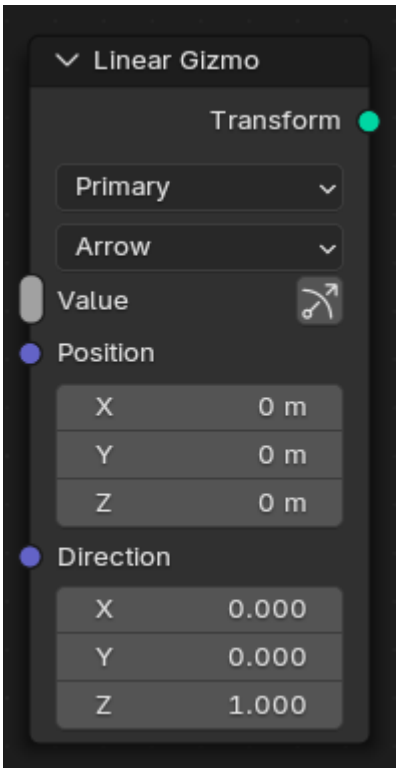
Outputs

Transform
: Should be joined into the geometry that is controlled by this gizmo.

Linear Gizmo

The *Linear Gizmo* node provides the most widely applicable gizmo. It can e.g. be used to control the height of something.





Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is moved.

Position
: Position of the gizmo in the local space of the object.

Direction
: Specifies the direction in with the gizmo points or is moved.

Properties

Color
: Controls which theme color is used for this gizmo.

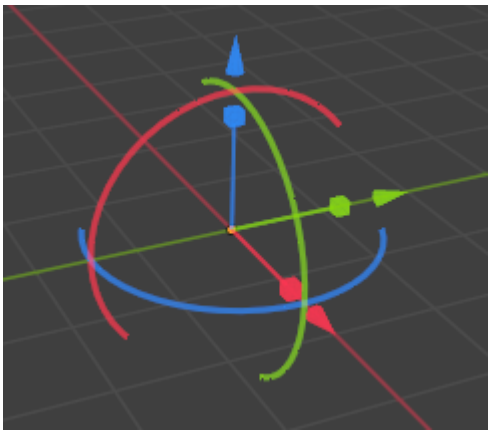
Draw Style
: Allows choosing between different styles of the gizmo.

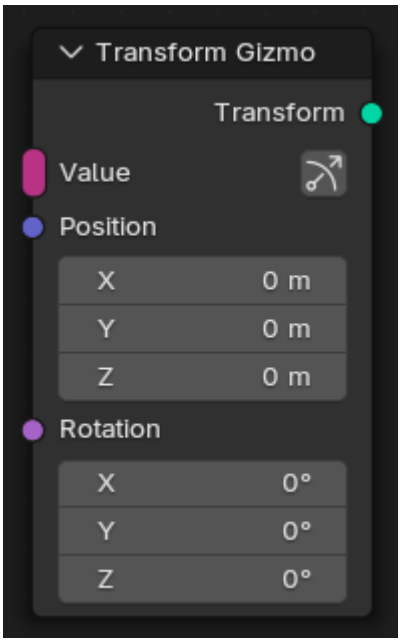
Outputs

Transform
: Should be joined into the geometry that is controlled by this gizmo.

Transform Gizmo

The *Transform Gizmo* node provides a compound gizmo that can control a position, rotation and scale.





Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is modified.

Position
: Position of the gizmo in the local space of the object.

Rotation
: Local orientation of the gizmo.

Note

The rotation input is ignored by the 3D viewport if the transform orientation is set to global.

Properties

The node has properties in the sidebar which allow disabling parts of the gizmo. This can be useful when e.g. controlling only a translation or only a rotation.

Outputs

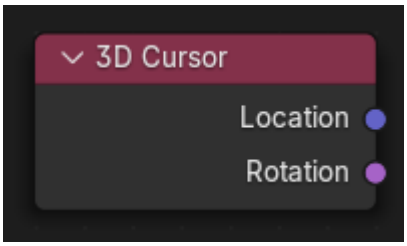
Transform
: Should be joined into the geometry that is controlled by this gizmo.

Input Scene Data Nodes

- 3D Cursor Node
- Active Camera Node
- Collection Info Node
- Image Info Node
- Is Viewport Node
- Mouse Position Node
- Object Info Node
- Scene Time Node
- Self Object Node
- Viewport Transform Node

3D Cursor Node

The *3D Cursor* node outputs the position and orientation of the 3D cursor in the scene.



Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

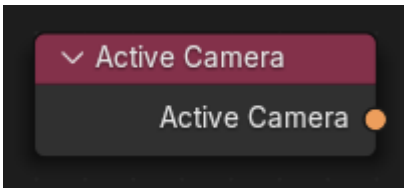
Outputs

Location
: The position of the 3D cursor.

Rotation
: The orientation of the 3D cursor as a standard rotation value.

Active Camera Node

The *Active Camera* node outputs the scene's current active camera.



Inputs

This node has no inputs.

Properties

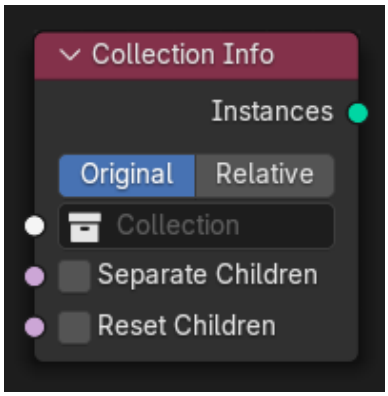
This node has no properties.

Outputs

Active Camera
: Current active camera.

Collection Info Node

The *Collection Info* node gets information from collections. This can be useful to control parameters in the geometry node tree with an external collection.



Tip

A *Collection Info* node can be added quickly by dragging a collection into the node editor.

Inputs

Collection
: Collection to get the properties from.

Separate Children
: Output each child of the collection as a separate instance. The list of instances will be sorted alphabetically with the objects and child collections sorted together. This can be used with the *Pick Instance* option in the Instance on Points Node to choose between collection children at each point.

Reset Children
: Remove the transform of each of the collection's children when converting them to instances. This is useful in order to keep child objects visually separate in the viewport, while keeping every instance located directly at the location of the point it was added for.

Properties

Transform Space

The transformation of the output instances. The instances are transformed, but not the geometry of the collection in them.

Original:: Output the instances relative to the collection offset.

Relative:: Join the input collection instances with the modified object as geometry, maintaining the relative position between the objects in the scene.

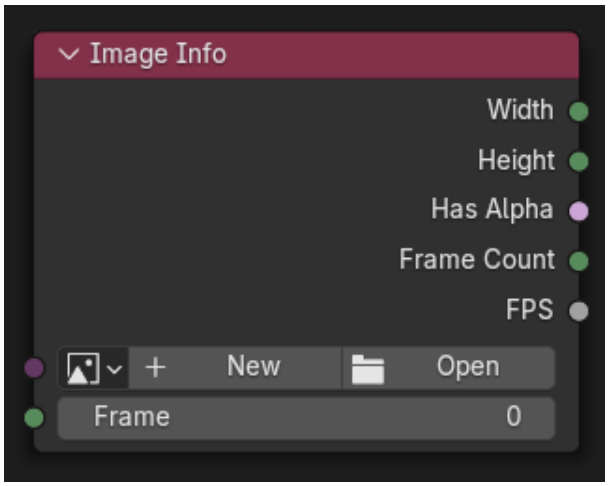
Outputs

Instances

Instances of the collection in world space with all modifiers applied and represented as geometry in instances.

Image Info Node

The *Image Info* node gets information from image and animation. This can be useful to generate parameters in the geometry node for arbitrary images. Image information can be either general or frame-specific.



Inputs

Image
: Source image to get parameters from.

Frame
: Frame index for frame-specific outputs.

Properties

This node has no properties.

Outputs

Width
: The number of pixels along the X axis. Specific to each frame.

Height
: The number of pixels along the Y axis. Specific to each frame.

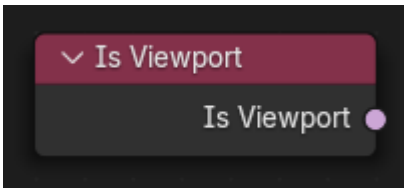
Has Alpha
: Whether the transparency channel be different from 1 for the pixels of this image frame. Specific to each frame.

Frame Count
: The number of frames in an image or video frame sequence. For a static image, always 1.

FPS
: The number of frames per second. For static image is always 0.

Is Viewport Node

The *Is Viewport* node outputs true when geometry nodes are evaluated for the viewport. For the final render the node outputs false.



Inputs

This node has no inputs.

Properties

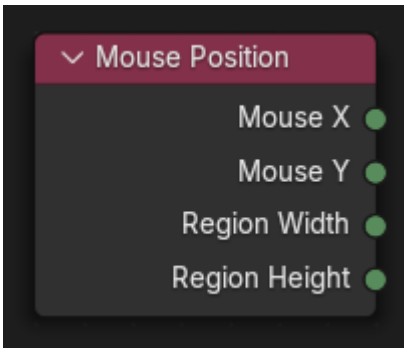
This node has no properties.

Outputs

Is Viewport
: Boolean value that indicates whether geometry nodes are evaluated for preview.

Mouse Position Node

The *Mouse Position* node returns information about the mouse cursor such as its position and the region's dimensions.



Tip

When using this node, enable Wait for Click. to wait for a mouse click input (LMB) before running the operator from a menu.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Mouse X
: The region-space mouse X location, in pixels, increasing from 0 at the left.

Mouse Y

The region-space mouse Y location, in pixels, increasing from 0 at the bottom.

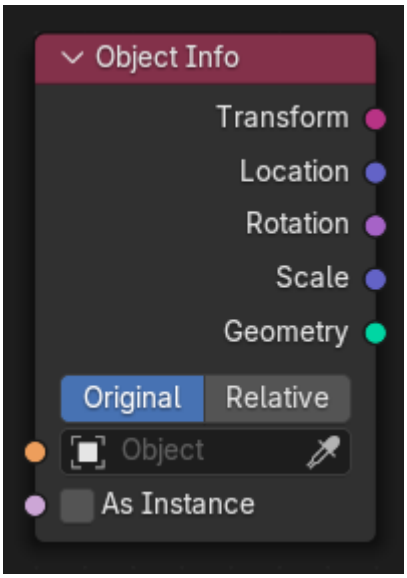
Region Width
: The total X size of the region in pixels.

Region Height
: The total Y size of the region in pixels.

Object Info Node

The *Object Info* node gets information from objects. This can be useful to control parameters in the geometry node tree with an external object, either directly by using its geometry, or via its transformation properties.

An *Object Info* node can be added quickly by dragging an object into the node editor.



Inputs

Object
: Object to get the properties from.

As Instance
: Output the entire object as single instance instead of realized geometry. This allows instancing non-geometry object types, because the output will contain an instance of the object.

Properties

Transform Space
: The transformation of the vector and geometry outputs.

Original:: Output the geometry relative to the input object transform, and the location, rotation and scale relative to the world origin.

Relative:: Bring the input object geometry, location, rotation and scale into the modified object, maintaining the relative position between the two objects in the scene.

Outputs

Transform
: Transformation Matrix containing the location, rotation and scale of the object.

Location
: Location of the object in world space.

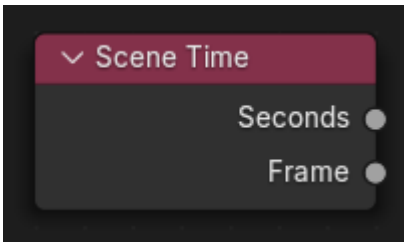
Rotation
: Rotation of the object in world space.

Scale
: Scale of the object in world space.

Geometry
: Geometry of the object in world space with all its modifiers applied.

Scene Time Node

The *Scene Time* node outputs the current time in the scene's animation in units of seconds or frames.



Inputs

This node has no inputs.

Properties

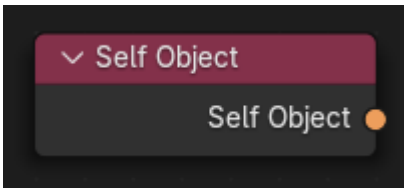
This node has no properties.

Outputs

Seconds
: Current scene time in seconds.

Frames
: Current scene frame. As an input in geometry nodes, this output may also output non-round numbers, in order to support higher quality motion blur.

Self Object Node



The *Self Object* node outputs the object that contains the geometry nodes modifier currently being executed. This can be used to retrieve the original transforms.

When evaluated in the Tool context, this node returns the Active object.

Note

The geometry cannot be retrieved from this object with the Object Info Node, since its final geometry is still being evaluated.

Inputs

This node has no inputs.

Properties

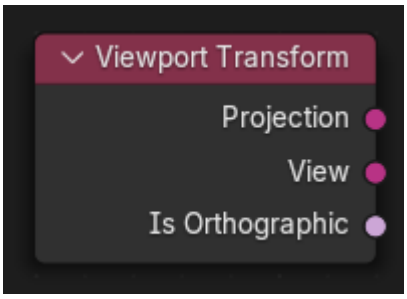
This node has no properties.

Outputs

Self Object
: The object currently being evaluated.

Viewport Transform Node

The *Viewport Transform* node retrieves the view direction and location of the 3D Viewport.



Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Projection
: The 3D Viewport's perspective or orthographic projection matrix.

View
: The view direction and location of the 3D viewport.

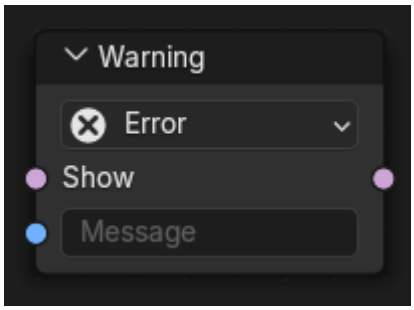
Is Orthographic
: Whether the viewport is using orthographic projection.

Output Nodes

Nodes that output to outside the node tree.

- Warning Node
- Viewer Node

Warning Node



Outputs a custom message that can be referenced in the Warnings panel of the Geometry Nodes Modifier.

This allows node groups to communicate expectations about input values.

By default, warnings are propagated through all parent node groups. However, this can be controlled using the Warning Propagation setting on each node.

Inputs

Show
: Present the warning in the Warnings panel.

Properties

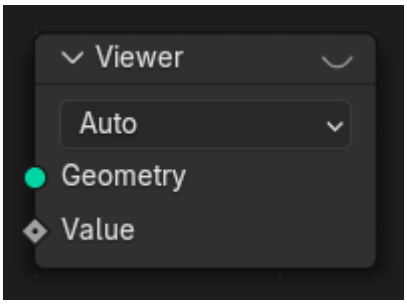
Warning Type
: The type of message to display, the type affects the icon displayed.

Outputs

Show
: A passthrough of the *Show* input.

Viewer Node

The *Viewer* node allows viewing data from inside a geometry node group in the Spreadsheet Editor and the 3D Viewport.



Any geometry connected can be visualized in the viewport and its attribute values can be read in the spreadsheet.

Note

This node cannot be used in the Tool context—only in the *Modifier* context.

Usage

Activation and Deactivation

Using `Shift-Ctrl-LMB` on any node or socket connects it to the viewer and makes it active. Using the same shortcut on empty space in the node editor makes deactivates the active viewer. When the active viewer is not visible anymore (e.g. another object is selected, or the current node group is exited), it is deactivated. The icon in the viewer node header can also be used to activate and deactivate it.

In the viewport, the *Show Viewer* option can turn off the viewer node visualization completely in order to see the final output of the object's evaluation instead.

Attribute Visualization

When the viewer has a geometry and a separate value input connected, the values can be visualized with a viewport overlays. When possible, the attribute domain is used to visualize the data is determined automatically. Otherwise, the viewer node falls back to the face corner domain on meshes and the point domain on curves. When necessary, the domain can be chosen manually.

The spreadsheet now only shows the "Viewer" column for the domain that is selected in the Viewer node.

Pinning

It can be helpful to pin a specific viewer node in the spreadsheet. When pinned, the spreadsheet still references the viewer node even when it becomes inactive.

Inputs

Geometry
: Geometry that will be displayed in the Spreadsheet.

Value
: Field to be evaluated on the geometry. The type for this value is chosen automatically when the keyboard shortcut to link an output is pressed. However, if the type must be adjusted manually, it is available in the node editor Sidebar.

Properties

Data Type
: The data type used to evaluate the *Value* input, visible in the node sidebar.

Domain
: The attribute domain used to evaluate the *Value* input. The *Auto* option chooses the domain automatically based on the connected nodes.

Outputs

This node has no outputs.

Geometry Nodes

Nodes that can operate on different geometry types (volume, mesh).

- Read
  - ID Node
  - Index Node
  - Named Attribute Node
  - Normal Node
  - Position Node
  - Radius Node
  - Selection Node
  - Active Element Node
- Sample
  - Geometry Proximity Node
  - Index of Nearest
  - Raycast Node
  - Sample Index Node
  - Sample Nearest Node
- Write
  - Set Geometry Name Node
  - Set ID Node
  - Set Position Node
  - Set Selection Node

- Operations
  - Bake Node
  - Bounding Box Node
  - Convex Hull Node
  - Delete Geometry Node
  - Duplicate Elements Node
  - Merge by Distance Node
  - Split To Instances Node
  - Sort Elements Node

    - Transform Geometry Node
    - Separate Components Node
    - Separate Geometry Node

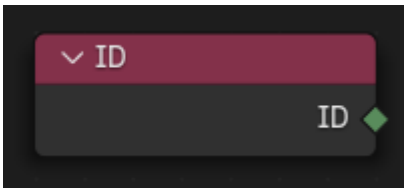
- Geometry to Instance Node
- Join Geometry Node

Read Geometry Nodes

- ID Node
- Index Node
- Named Attribute Node
- Normal Node
- Position Node
- Radius Node
- Selection Node
- Active Element Node

ID Node

The *ID* node gives an integer value indicating the stable random identifier of each element on the point domain, which is stored in the `id` attribute.



The node to set this data is the Set ID Node node.

Note

Unlike other built-in attributes, the `id` attribute does not always exist. In that case, this node will output the index.

Inputs

This node has no inputs.

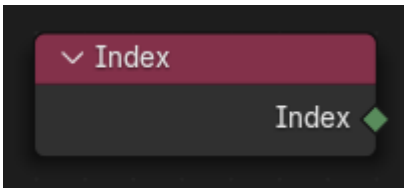
Properties

This node has no properties.

Outputs

ID
: Integer value.

Index Node



The *Index* node gives an integer value indicating the position of each element in the list, starting at zero. This depends on the internal order of the data in the geometry, which is not necessarily visible in the 3D Viewport. However, the index value is visible in the left-most column in the Spreadsheet Editor.

Note

Indices in geometry data are often defined by the internals of complex algorithms that create it. If no inputs change, indices will be the same when the same node tree is executed multiple times. However, they may not be predictable when inputs to nodes that generate geometry or change its topology are adjusted. Additionally, updates to algorithms in newer versions of Blender may change the order of generated elements.

To avoid relying on consistent indices, it is recommended to calculate them locally, or to avoid operations that change topology when they must be consistent over time.

Inputs

This node has no inputs.

Properties

This node has no properties.

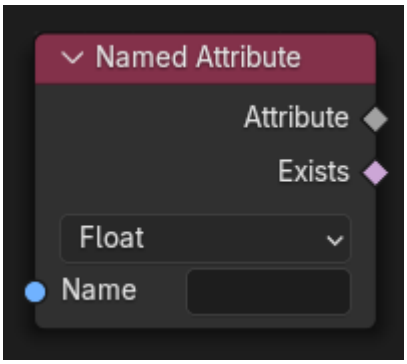
Outputs

Index

Integer value which enumerates each point on the geometry.

Named Attribute Node

The *Named Attribute* node outputs the data of an attribute based on the context of where it is connected (the Field Context).



Inputs

Name
: The name of the attribute to read.

Properties

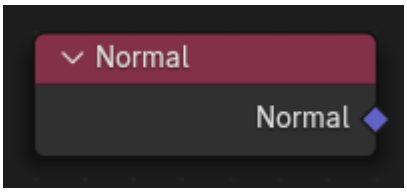
Data Type
: The data type used for the retrieved data. Attribute Search can be used to give a basic list of possible attribute names and data types. When a value is chosen from the search menu, the data type is set to automatically choose the data type from the geometry nodes result.

Outputs

Attribute
: The attribute data stored on the geometry.

Exists
: True if the attribute accessed by the node is present in the connected context.

Normal Node



The *Normal* node returns a vector for each evaluated point indicating the normal direction. The output can depend on the attribute domain used in the node evaluating the field, but the output is always a normalized unit vector.

Face:: On the face domain, the normal is the "up" direction of the face.

Mesh Vertices:: For mesh vertices, the normal is an average of the surrounding face normals. If the vertex does not have any connected faces, the output is simply the normalized position of that vertex.

Edge:: The normal output for each edge is the average of the edge's two vertex normals.

Face Corner:: The output for each face corner is the same as the face normal of the corresponding face.

Curve Control Points:: The output of this node when used for curve geometry is the evaluated normal of the curve, which depends on the twist method. The normal vector is always perpendicular to the direction of the curve's path at every point.

Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node can be used to create a poly spline, where there is a control point for every evaluated point.

Inputs

This node has no inputs.

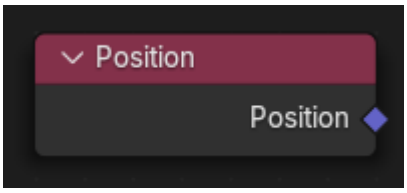
Properties

This node has no properties.

Outputs

Normal
: Vector indicating the normal of each geometry element.

Position Node



The *Position* node outputs a vector of each point of the geometry the node is connected to.

The node can work on geometry domains besides points. In that case, the position data will be automatically interpolated to the new domain. For example, when used as part of the input to the Split Edges Node, the position for each edge will be the average position of the edge's two vertices.

For instances themselves, the output is the origin of each instance. However, if the node is for a geometry node that adjusts data inside instances, the position output of this node will be in the local space of each instance. See the Instance Processing page for more details.

Inputs

This node has no inputs.

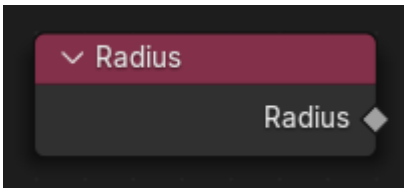
Properties

This node has no properties.

Outputs

Position
: Vector that indicates the location of each element of the geometry.

Radius Node



The *Radius* node outputs the radius value at each point on the evaluated geometry. For curves, this value is used for things like determining the size of the mesh created in the Curve to Mesh node. For point clouds, the value is used for the display size of the point in the viewport.

Inputs

This node has no inputs.

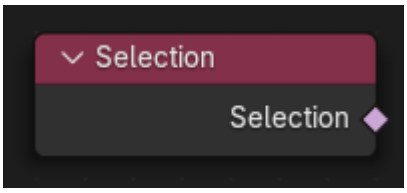
Properties

This node has no properties.

Outputs

Radius
: Float value indicating radius at each point on the geometry.

Selection Node



The *Selection* node outputs true for geometry that is selected, and false elsewhere.

The corresponding data flow node is the Set Selection Node.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

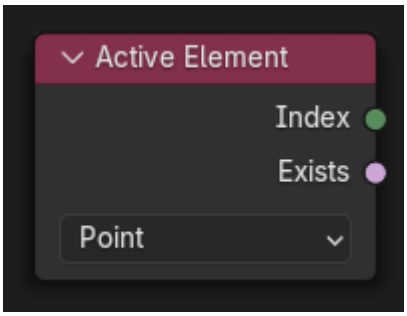
This node has no properties.

Outputs

Selection
: Boolean field set to true for geometry that is selected in edit mode.

Active Element Node

The *Active Element* node outputs the index of the Active vertex, edge, or face.



Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

Domain
: Which domain to return the index of.

Outputs

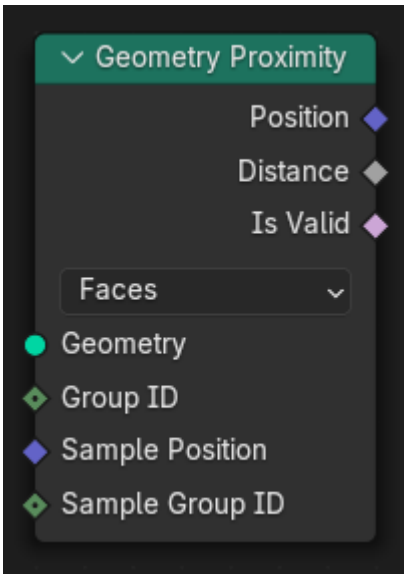
Index
: Index of the active element in the specified domain.

Exists
: True if an active element exists in the mesh, false otherwise

Sample Geometry Nodes

- Geometry Proximity Node
- Index of Nearest
- Raycast Node
- Sample Index Node
- Sample Nearest Node

Geometry Proximity Node



The *Geometry Proximity* node computes the closest location on the target geometry.

Tip

The Map Range Node is often helpful to use with the distance output of this node to create a falloff with a maximum distance.

Inputs

Geometry
: Standard geometry input.

Group ID
: Splits the elements of the input geometry into groups which can be sampled individually.

Sample Position

The given position to calculate the closest location on the target.

Sample Group ID
: Determines in which group the closest nearest element is detected.

Properties

Target Element

Faces:: Calculate the closest point anywhere on the faces of the target's mesh geometry.

Edges:: Calculate the closest point anywhere on the edges of the target's mesh geometry.

Points:: Calculate the closest point or vertex on the target geometry. This mode is usually the fastest. This mode works for both point cloud and mesh geometry, the other modes only work for meshes.

Outputs

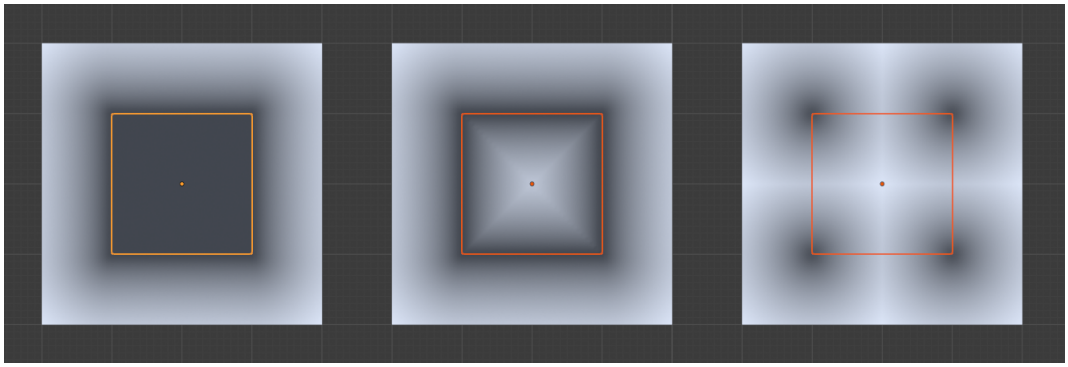
Position
: Closest location on the surface of the target mesh, or the closest point in the target point cloud in *Points* mode.

Distance
: Distance (as floating-point value) from the source position to the closest location in the target.

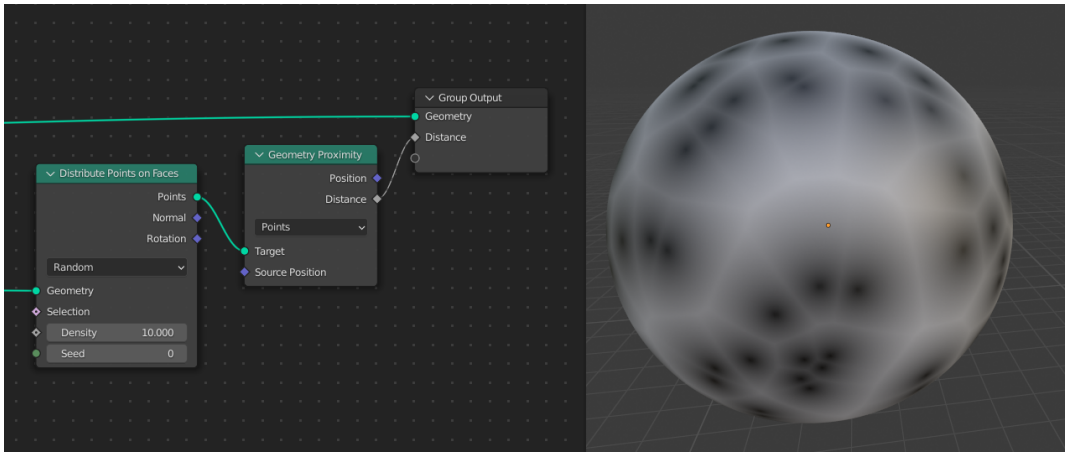
Is Valid
: Whether the sampling was successful. It can fail when the sampled group is empty.

Examples

The different modes of the node: faces, edges and points. In this example the Geometry Nodes modifier is added on the target plane. Note that the larger plane is subdivided and the smaller plane is not.



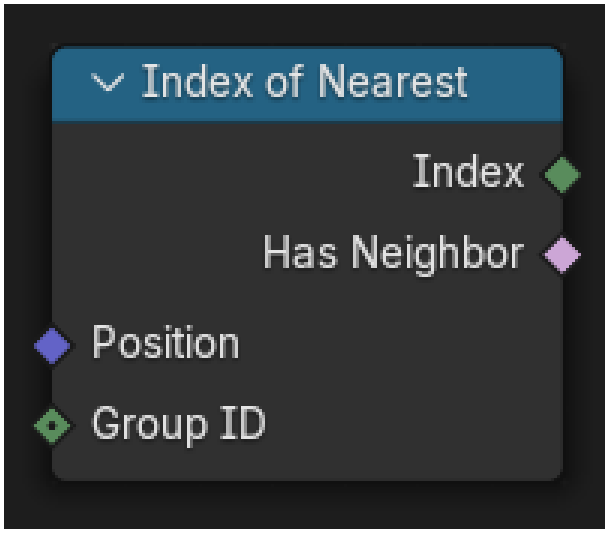
The three target element modes: faces, edges, and points.



Points distributed on a sphere used as a target for a distance used in a shader.

Index of Nearest

The **Index of Nearest** node is a way to find other close elements in the same geometry. If needed you can use Group ID to determine the group of neighbors to be analyzed together.



This is an alternative to the Sample Nearest Node node. The main difference is that this node does not require a geometry input, because the geometry from the field context is used.

Tip

This is often combined with the Evaluate at Index Node or the Sample Index Node node.

Inputs

Position
: The position for each element to search. By default, this is the same as if the Position Node was connected.

Group ID
: ID to group elements together.

Outputs

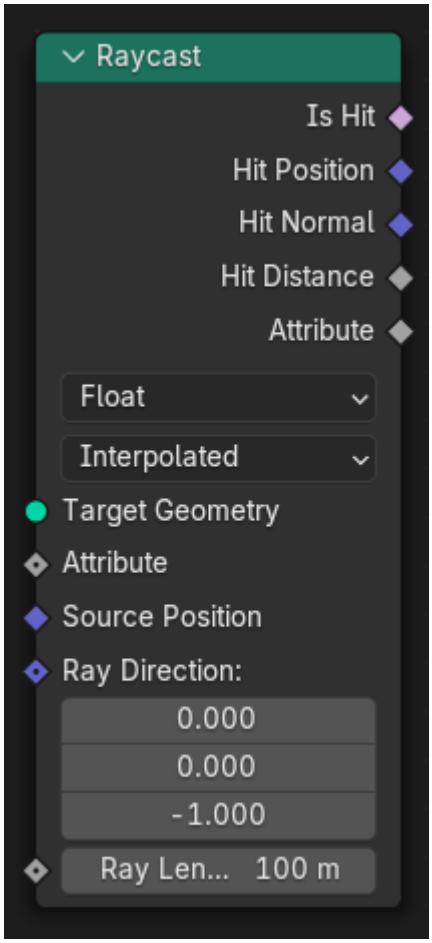
Index

The index of the closest element in the same geometry component.

Has Neighbor

This is true when the group of the element has at least two elements. This is only relevant when using *Group ID*.

Raycast Node



The *Raycast* node intersects rays from one geometry onto another. The source geometry is defined by the context of the node that the *Raycast* node is connected to. Each ray computes hit points on the target mesh and outputs normals, distances and any surface attribute specified.

Inputs

Target Geometry
: Geometry that rays are tested against.

Attribute

An optional field input evaluated on the *Target Geometry* that will be interpolated at the hit points. The resulting values are outputted with the *Attribute* output.

Source Position
: The position from where to start each ray. By default, this is the same as if the Position Node was connected.

Ray Direction
: Direction of each ray from the starting position. The field is evaluated on the geometry from the context of the field evaluation, not the *Target Geometry*.

Ray Length
: Maximum distance a ray can travel before being considered “no hit”.

Properties

Mapping
: How attributes of the target mesh are mapped to the attribute values on the result geometry.

Interpolated::	Vertex and corner attributes are interpolated smoothly, with a bilinear function.
Nearest::	Choose the value of the closest vertex without interpolating.

Outputs

Is Hit
: Boolean output that is true for each ray which has hit the *Target Geometry*.

Hit Position
: The location of the intersection point with the target mesh.

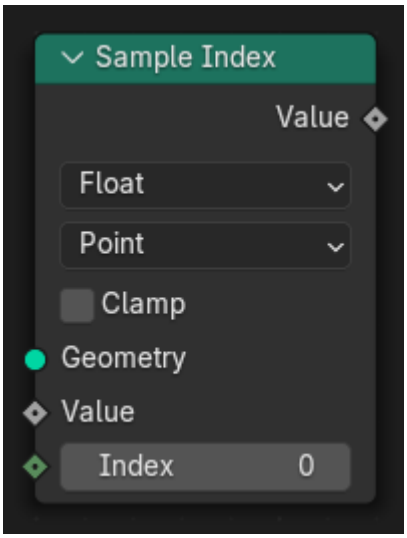
Hit Normal

The surface Normal vector at the hit location.

Hit Distance
: The distance from the *Source Position* to the *Hit Position*. If the ray does not hit, the *Ray Length* is returned.

Attribute
: Interpolated values of the *Attribute* input sampled at the *Hit Position*.

Sample Index Node



The *Sample Index* node retrieves values from a source geometry at a specific index.

Tip

If the *Geometry* used for the input is the same as the geometry from the field context, this node is equivalent to the Evaluate at Index Node. Using that node is usually preferable since avoiding the geometry socket makes the whole setup easier to use in other situations and share.

Tip

Different components can have same attribute domain (Points). This node simply uses first component that not empty for such domain, checked in the order of: Mesh, Point Cloud, Curve. The Separate Components Node can be used to sample directly from a specific component.

Inputs

Geometry
: The geometry to retrieve the attribute from.

Value
: A field to evaluate on the source *Geometry*. The values are then retrieved from specific indices for the output.

Index
: Which index to use when retrieving the data from the input *Value* field. Any index can be connected, resulting in a “shuffling” of the values.

Properties

Data Type
: The data type to use for the retrieved values.

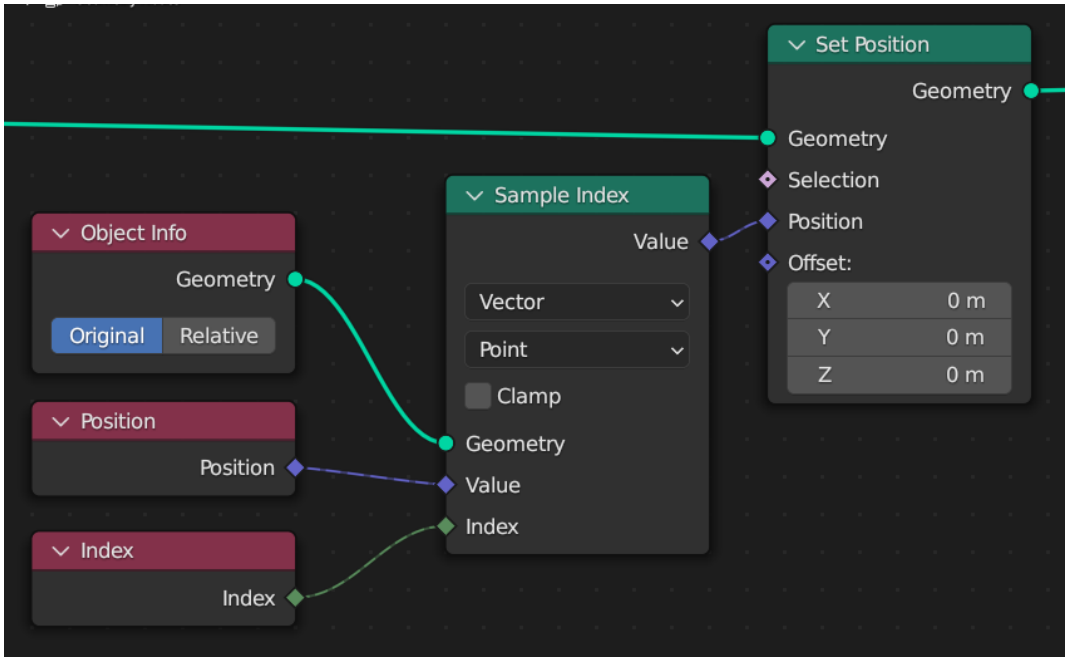
Domain
: The attribute domain that the attribute is transferred from, or in other words, the domain used to evaluate the *Attribute* input. For example, it is possible to transfer data from the faces of one geometry to the points of another.

Clamp
: Clamp the indices to the size of the attribute domain instead of outputting a default value for invalid indices.

Outputs

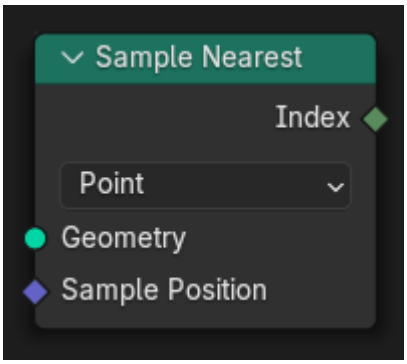
Value
: The data retrieved from the source *Geometry* input.

Examples



Here the node is used to copy the positions of one object to another. This recreates the behavior of the *Transfer Attribute* node from Blender versions before 3.4. This works best when their geometries have the same number of points and the same Topology.

Sample Nearest Node



The *Sample Nearest* node retrieves the index of the geometry element in its input geometry that is closest to the input position.

This node is similar to the Geometry Proximity Node, but it outputs the index of the closest element instead of its distance from the current location.

Tip

If you want to find nearest to each point in same geometry, its better to use the Index of Nearest node.

Inputs

Geometry
: The geometry to sample.

Note

This node only supports point cloud and mesh inputs.

Sample Position

The position to start from when finding the closest location on the target geometry. By default, this is the same as if the Position Node was connected.

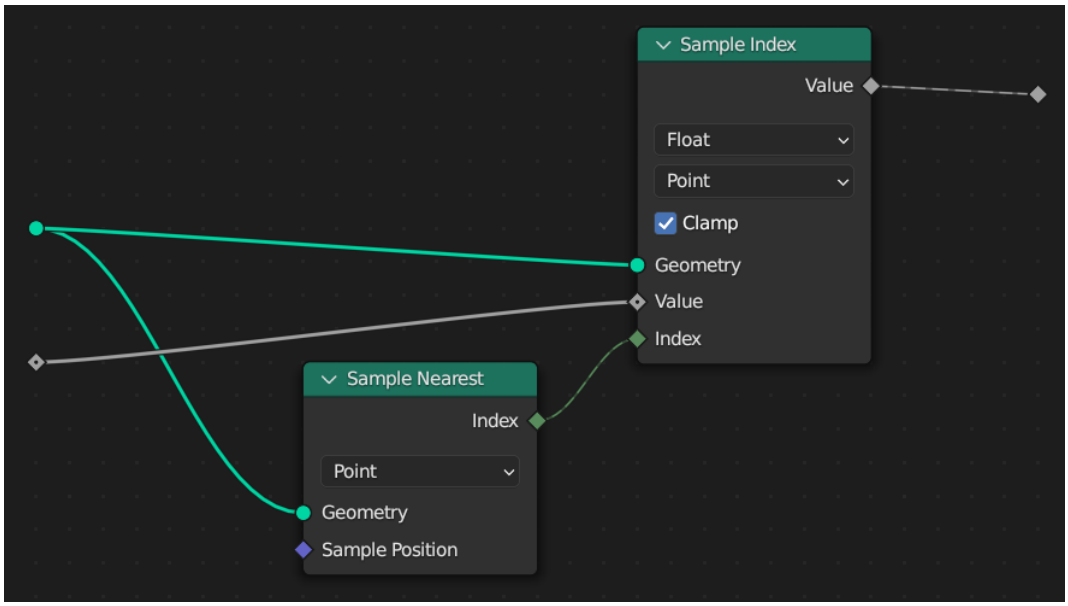
Properties

Domain
: The attribute domain to consider the distance from.

Outputs

Index
: The index of the closest geometry element of the chosen domain.

Examples



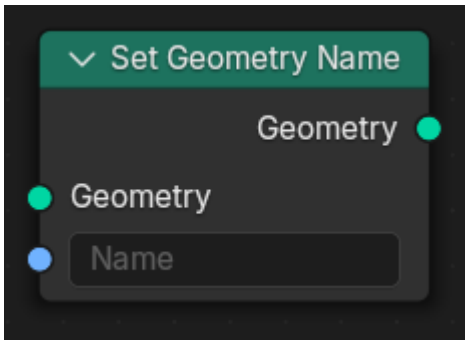
Combining this node with the Sample Index Node gives a setup that can retrieve the closest attribute value from another geometry. This is the same behavior as the *Transfer Attribute* node in versions of Blender before 3.4.

Write Geometry Nodes

- Set Geometry Name Node
- Set ID Node
- Set Position Node
- Set Selection Node

Set Geometry Name Node

The *Set Geometry Name* node stores a custom name on the geometry, overriding the name which might come from the Object Info Node or a Grease Pencil to Curves Node. The name is displayed in the spreadsheet and can helful for debugging purposes.



Inputs

Geometry
: Standard geometry input.

Name
: The new name for the geometry.

Properties

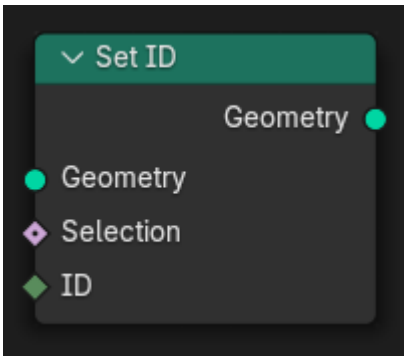
This node has no properties.

Outputs

Geometry
: Standard geometry output.

Set ID Node

The *Set ID* node fills the `id` attribute on the input geometry. If the attribute does not exist yet, it will be created with a default value of zero. The ID is also created by the Distribute Points on Faces, and it is used in the Random Value Node and other nodes if it exists.



The input node for this data is the ID Node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether or not to change the value on each point or instance. True values mean the value will be changed, false values mean the value will remain the same.

ID
: The value for each element. By default, this input uses the index, which is useful when stable IDs are desired when deleting a dynamic number of instances.

Properties

This node has no properties.

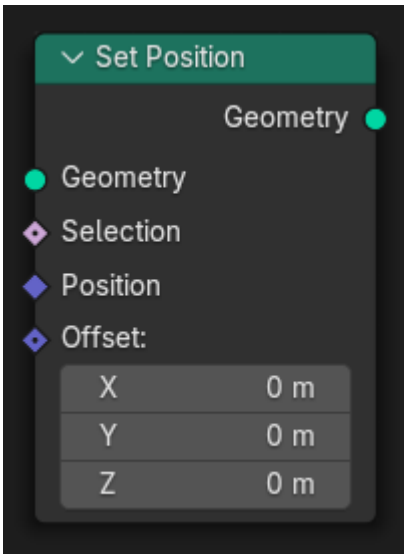
Outputs

Geometry

Standard geometry output.

Set Position Node

The *Set Position* node controls the location of each point, the same way as controlling the `position` attribute. If the input geometry contains instances, this node will affect the location of the origin of each instance.



The input node for this data is the Position Node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether or not to change the position of each point or instance. True values mean the position will be changed, false values mean it will remain the same.

Position
: The new position for selected elements. By default, this is the same as if the Position Node was connected, meaning the node will do nothing.

Offset
: An optional translation for each point. This is evaluated at the same time as the *Position* input, meaning that fields evaluated for it will not reflect the changed position.

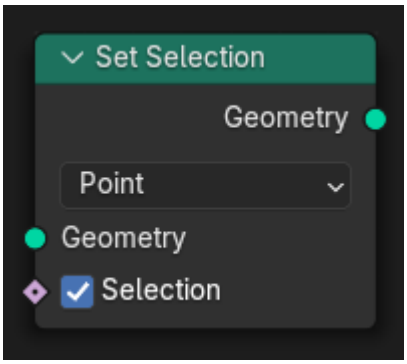
Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Set Selection Node



The *Set Selection* node controls which geometry is selected.

The input node for this data is the Selection Node.

Note

This node can only be used in the Tool context.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field for specifying which elements should be selected in the output geometry. Elements for which this field evaluates to false are implicitly de-selected.

Properties

Domain
: Which domain to set the selection on.

Outputs

Geometry
: Standard geometry output.

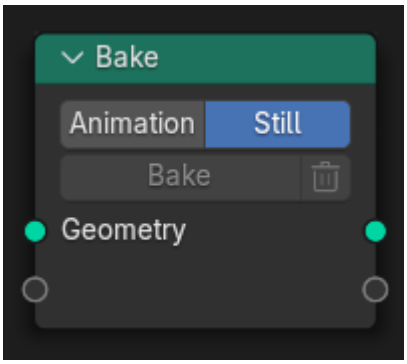
Geometry Operation Nodes

- Bake Node
- Bounding Box Node
- Convex Hull Node
- Delete Geometry Node
- Duplicate Elements Node
- Merge by Distance Node
- Split To Instances Node
- Sort Elements Node
- Transform Geometry Node

- Separate Components Node
- Separate Geometry Node

Bake Node

The *Bake* node allows saving and loading intermediate geometries. This node bakes parts of the node tree for better performance.



The data format used to store geometry data is not considered to be an import/export format. Volume objects, however, are saved using the OpenVDB file format which can be used interoperably.

Important

- It's not guaranteed that data written with one Blender version can be read by another Blender version.

Inputs

Geometry
: Standard geometry input, which is used as the default bake item. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Bake Items

Reference

Editor:: Geometry Node Editor
Panel:: *Sidebar* ▸ *Node* ▸ *Bake Items*

The *Bake Items* panel is used to manage the input sockets of the node thus also managing what data is baked.

Bake Items List
: Used to manage the inputs and outputs of the bake node. Items can be added, removed, renamed, and sorted.

Socket Type
: The data (Sockets) of the input/output.

Attribute Domain
: The attribute domain used to evaluate the input field on.

Is Attribute
: Bake item is an attribute stored on a geometry.

Properties

Note

Some properties can only be edited in the Properties panel (*Sidebar* ‣ *Node* ‣ *Properties*).

Bake Mode
: The *Bake* node can calculate the geometry of a single frame or an animation.

Animation::	Bakes the geometry data for multiple frames. By default the scene frame range is used, however, a *Custom Range* can also be defined.
Still::	Bakes the geometry data of the current frame.

Bake
: Preforms all necessary geometry calculations and saves the data on disk or packed into the .blend file.

Pack/Unpack
: Packs or unpacks already baked data. When unpacking, there are different options to control where the data will be stored. The options are the same as for the Unpack Resources operator.

Delete (Trash Icon)
: Deletes the bake data.

Bake Target
: Specifies where the baked data should be stored.

Inherit from Modifier:: The setting is copied from the modifier that contains the bake node. This is the default.

Packed:: The baked data is packed into the .blend file. So no separate file is necessary.

Disk:: The baked data is stored in a separate directory on disk.

Custom Path
: Specify a path where the baked data should be stored manually.

Bake Path
: Location on disk where the baked data is stored. Note, this path is also used for simulation zones.

Custom Range *Animation*
: Override the simulation frame range from the scene.

Start, End
: The start and end frame numbers for the custom range.

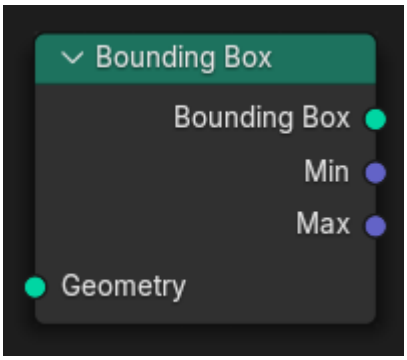
Outputs

For each input, the same output is added to act as a pass through.

Geometry

Standard geometry output, which is used as the default bake item. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl–LMB` on the socket name or in the nodes *Properties* panel.

Bounding Box Node



The *Bounding Box* node creates a box mesh with the minimum volume that encapsulates the geometry of the input. The node also can output the vector positions of the bounding dimensions.

The mesh output and the *Min* and *Max* outputs do not take instances into account. Instead, for instanced geometry, a bounding box is computed for each instance rather than the whole geometry. To compute the bounding box including the instances, a Realize Instances Node can be used.

Inputs

Geometry
: Standard geometry input.

Properties

This node has no properties.

Outputs

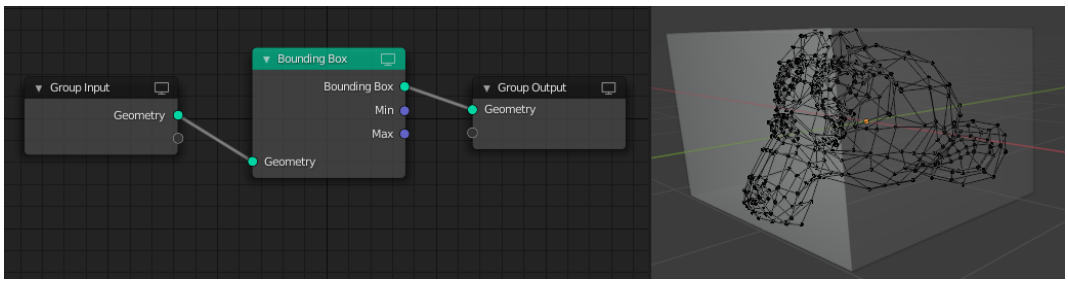
Bounding Box
: The resulting box that encapsulate the input geometry.

Min
: The coordinates corresponding to the box's -X, -Y, -Z position values, i.e. how far the box extends in each of the negative axes directions.

Max

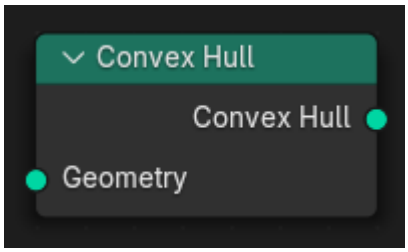
The coordinates corresponding to the box's +X, +Y, +Z position values, i.e. how far the box extends in each of the positive axes directions.

Example



Bounding Box node used to create a box that encapsulates the geometry of the monkey mesh.

Convex Hull Node



The *Convex Hull* node outputs a convex mesh that is enclosing all points in the input geometry.

Note

When the node is used on a geometry with instances, the algorithm will run once per instance, resulting in many convex hull meshes in the instance geometries. The *Realize Instances* node can be used to get a convex hull of an entire geometry.

Important

Volumes are not supported by this node, and attributes are not automatically transferred to the result.

Inputs

Geometry
: Standard geometry input.

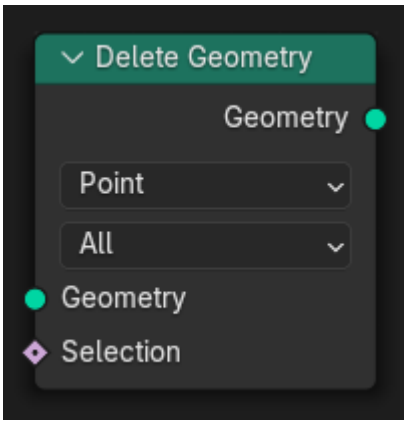
Properties

This node has no properties.

Outputs

Convex Hull
: Mesh that encloses all points in the input.

Delete Geometry Node



The *Delete Geometry* node removes the selected part of a geometry. It behaves similarly to the Delete tool in Edit Mode. The type of elements to be deleted can be specified with the domain and mode properties.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field that is true for parts of the geometry to be deleted.

Properties

Domain
: The domain on which the selection field is evaluated.

Point::	The selection is on the points, control points, and vertices of the geometry.
Edge::	The selection is on the edges of the mesh component. The other components are not modified.

Face:: The selection is on the faces of the mesh component. The other components are not modified.

Spline:: The selection is on the splines in the curve component. For each spline, it will either be deleted entirely or not at all. The other components are not modified.

Instance:: The selection is on the top-level instances, and the entire instance is removed if it is selected. If the input also contains realized geometry, that will be unaffected.

Mode

The type of elements to be affected. This only applies to the mesh component.

All:: Vertices, edges, and faces in the selection will be deleted.

Only Edges & Faces:: Vertices won't be deleted, even if they are in the selection.

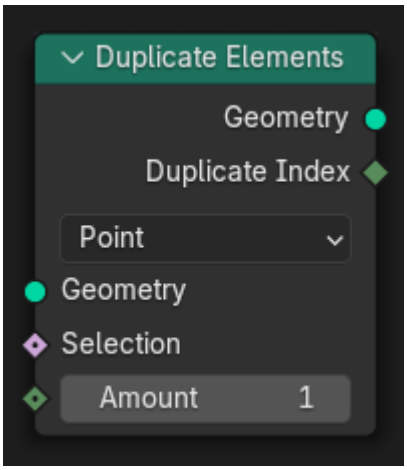
Only Faces:: Only faces in the selection will be deleted.

Output

Geometry

Standard geometry output.

Duplicate Elements Node



The *Duplicate Elements* node creates a new geometry with the specified elements from the input duplicated an arbitrary number of times. The positions of elements are not changed, so all of the duplicates will be at the exact same location.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field that is true for parts of the geometry to be deleted.

Amount
: Field indicating how many times each input element should be duplicated. If the value is zero for an element, it will not be included in the output at all.

Properties

Domain

The type of geometry element to duplicate

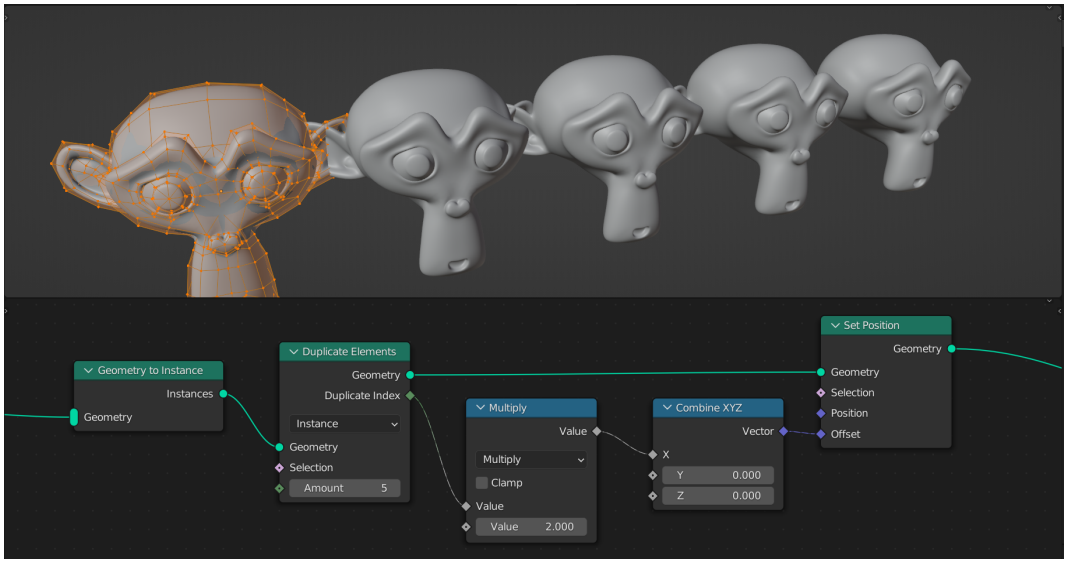
Point::	Duplicate the points of meshes, curves, or point clouds. Any other elements will not be included in the output.
Edge::	Duplicate mesh edges. Faces will not be included in the output.
Faces::	Duplicate mesh faces. Each duplicated face will be separate, in other words they will not share edges with other faces.
Spline::	Individual curves from the input curves component will be duplicated.
Instances::	Input top-level instances will be duplicated.

Output

Geometry
: Standard geometry output.

Duplicate Index
: An attribute field with a value for every output element describing which duplicate of the corresponding input. The value for every input element will start at 0 and increase to `Amount - 1`.

Examples

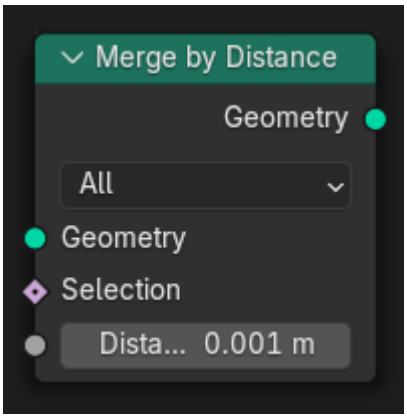


Combined with the Geometry to Instance Node, this can be used to create a basic efficient "Array" operation. This should be more efficient because the duplicates are instances.

The "Duplicate Index" is used to move each instance in the result a different amount.

Merge by Distance Node

The *Merge by Distance* node merges selected mesh vertices or point cloud points within a given distance, merging surrounding geometry where necessary. This operation is similar to the Merge by Distance operator or the Weld Modifier.



Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field that is true for parts of the geometry to be deleted. Unselected points will be completely unused for the operation– they will not be merged into other points, and no points will merge into them either.

> **Tip**
>
> When possible, using the selection input can be a simple way to speed up the node, since searching for nearby points is a relatively expensive operation that gets even more expensive when more points are involved.

Distance
: The distance to use for searching for nearby points.

Properties

Mode
: Method for choosing which vertices are merged.

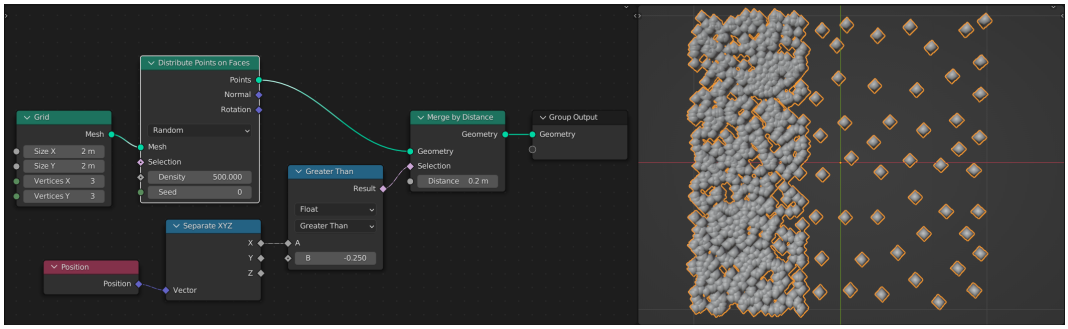
All:: Merge includes all geometry including loose parts.

Connected:: Merge only includes attached geometry i.e. the modifier will not merge loose parts together.

Output

Geometry
: Standard geometry output.

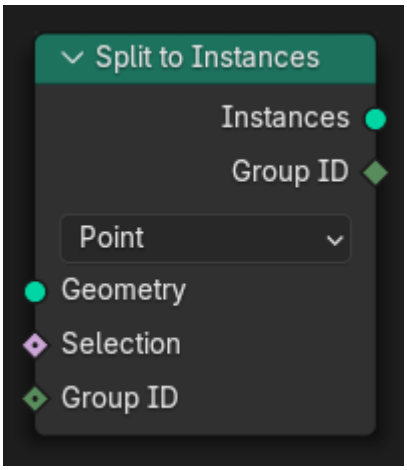
Examples



Using the selection input to only merge some of the points in a point cloud.

Split To Instances Node

Splits a selection of geometry elements (such as faces) into groups, then turns each group into an instance.



Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field indicating which geometry elements to include.

Group ID
: Integer field indicating which group each element belongs to. Elements with the same ID will be moved into the same output instance.

Properties

Domain
: The type of geometry to extract and split. This is also the domain on which the *Selection* and *Group ID* fields are evaluated.

Point::	Points, spline control points, and vertices.
Edge::	Mesh edges.
Face::	Mesh faces.

Spline::	Curve splines.
Instance::	Top-level instances. Realized instances are ignored.
Layer::	Grease Pencil layers.

Note

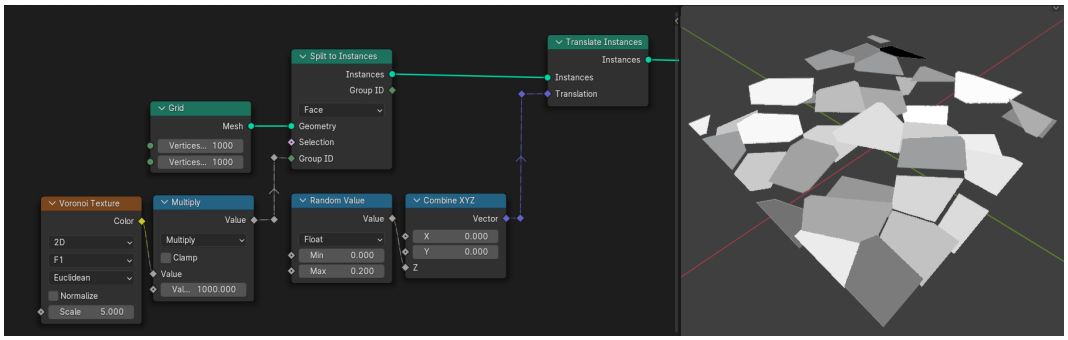
Geometry that doesn't match the selected domain will be removed. For example, if you choose *Edge*, any faces, splines, and instances in the input geometry will be lost.

Output

Instances
: The instances containing the grouped geometry elements.

Group ID
: Group ID of each instance.

Examples

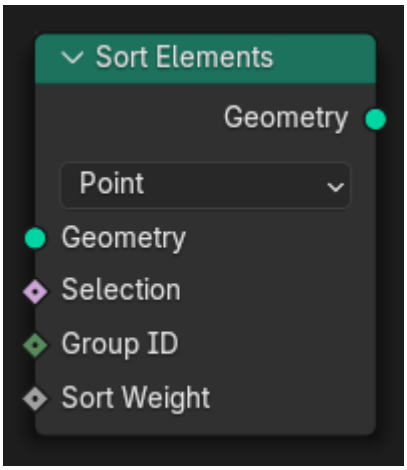


In the example above, we start with a grid of 1000x1000 square faces serving as "pixels." Then, we group these faces into patches by assigning them a group ID sampled from a Voronoi texture, and move each resulting instance by a random amount along the Z axis.

Note that, because the texture outputs floating point values between 0 and 1 while the group ID is an integer, all the values would be rounded to 0 or 1 and we would only get two groups. To get more variation, we multiply the texture value by 1000.

Sort Elements Node

The *Sort Elements* node rearranges geometry elements by changing their indices.



Inputs

Geometry
: Standard geometry input.

Selection
: The selection of elements to sort, if left blank, all elements are sorted. Non selected elements will be keep their current indices.

Group ID
: Elements with the same group ID are sorted together. If this is not a field, the node has no affect.

Sort Weight
: The sorted values used to do the reordering. If this is not a field, the node has no affect.

Properties

Domain
: The domain on which the selection and group ID fields are evaluated.

Point::	The fields are evaluated on points, control points, and vertices.
Edge::	The fields are evaluated on the edges of the mesh component.
Faces::	The fields are evaluated on the faces of the mesh component.
Spline::	The fields are evaluated on the splines in the curve component.
Instance::	The fields are evaluated on the top-level instances. Realized instances are ignored.

Outputs

Geometry
: Standard geometry output.

Transform Geometry Node

The *Transform Geometry Node* allows you to move, rotate or scale the geometry. The transformation is applied to the entire geometry, and not per element. The Set Position Node is used for moving individual points of a geometry. For transforming instances individually, the instance translate, rotate, or scale nodes can be used.



Inputs

Geometry
: Standard geometry input.

Translation
: Translation of the entire geometry in the local space of the modified object.

Rotation
: Euler rotation in the local space of the modified object.

Scale

Scale for the geometry in the local space of the modified object.

Transform
: A Transformation Matrix, available when using *Matrix* mode.

Properties

Mode
: How the transformation is specified.

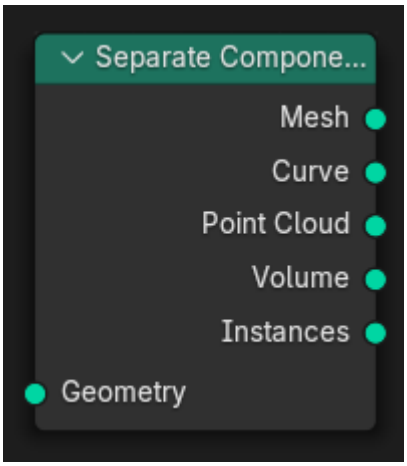
  Components:: Provide separate inputs for location, rotation and scale.

  Matrix:: Use a transformation matrix.

Output

Geometry
: Standard geometry output.

Separate Components Node



The *Separate Components* node splits a geometry into a separate output for each type of data in the geometry.

Inputs

Geometry
: Standard geometry input.

Properties

This node has no properties.

Outputs

Mesh
: Mesh component of the input geometry.

Curve
: Curve component of the input geometry.

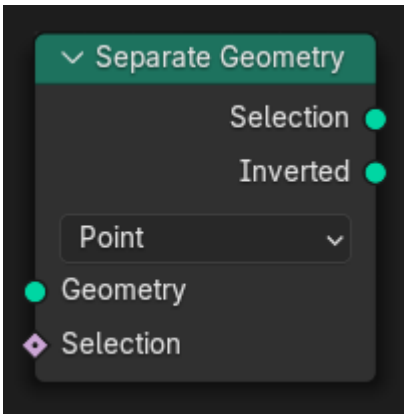
Point Cloud
: Point cloud component of the input geometry.

Volume
: Volume component of the input geometry.

Instances
: Instances component of the input geometry. Even if the instances contain geometry data with one of the other types, all instances will be added to this output. A Realize Instances Node can be added to move the data from geometry instances to their corresponding outputs.

Separate Geometry Node

The *Separate Geometry* node produces two geometry outputs. Based on the *Selection* input, the input geometry is split between the two outputs.



Tip

This node can be combined with the Compare Node for a more precise control of which parts are separated to a given output geometry.

Inputs

Geometry
: Standard Geometry input.

Selection
: Boolean field used to calculate which output each part of the geometry will go to. Parts in the selection will move to the *Selection* output. Parts not in the selection will move to the *Inverted* output.

Properties

Domain
: The domain on which the selection field is evaluated.

Point::	The selection is on the points, control points, and vertices of the geometry.
Edge::	The selection is on the edges of the mesh component. The other components are not modified.

Faces::	The selection is on the faces of the mesh component. The other components are not modified.
Spline::	The selection is on the splines in the curve component. For each spline, it is either entirely in the selection or not at all. The other components are not modified.

Note

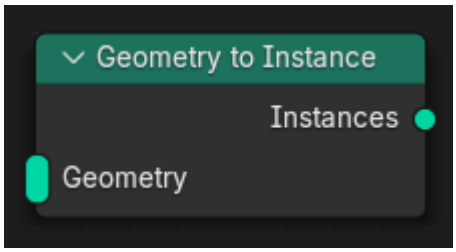
When selecting a domain that doesn't modify all components, the unmodified components will appear in both outputs.

Outputs

Selection
: The parts of the geometry in the selection.

Inverted
: The parts of the geometry not in the selection.

Geometry to Instance Node



The *Geometry to Instance* node turns every connected input geometry into an instance. Visually, the node has a similar result as the Join Geometry Node, but it outputs the result as separate instances instead. The geometry data itself isn't actually joined.

The node can be used in combination with the *Pick Instances* option in the Instance on Points Node, as a way to pick between geometry generated in the node tree (as opposed to picking from separate instances from the Collection Info Node, for example).

Tip

This node can be much faster than the join geometry node when the inputs are large geometries. This is because the join geometry node must actually create a larger mesh, or a larger curve. Even though the operation is simple, just creating a large mesh can have a significant cost. This node can be better, because instead of merging large geometries, it just groups them together as instances.

Inputs

Geometry
: Geometry that will be joined. Multiple inputs are allowed. When the node is muted, only the first link will be passed through.

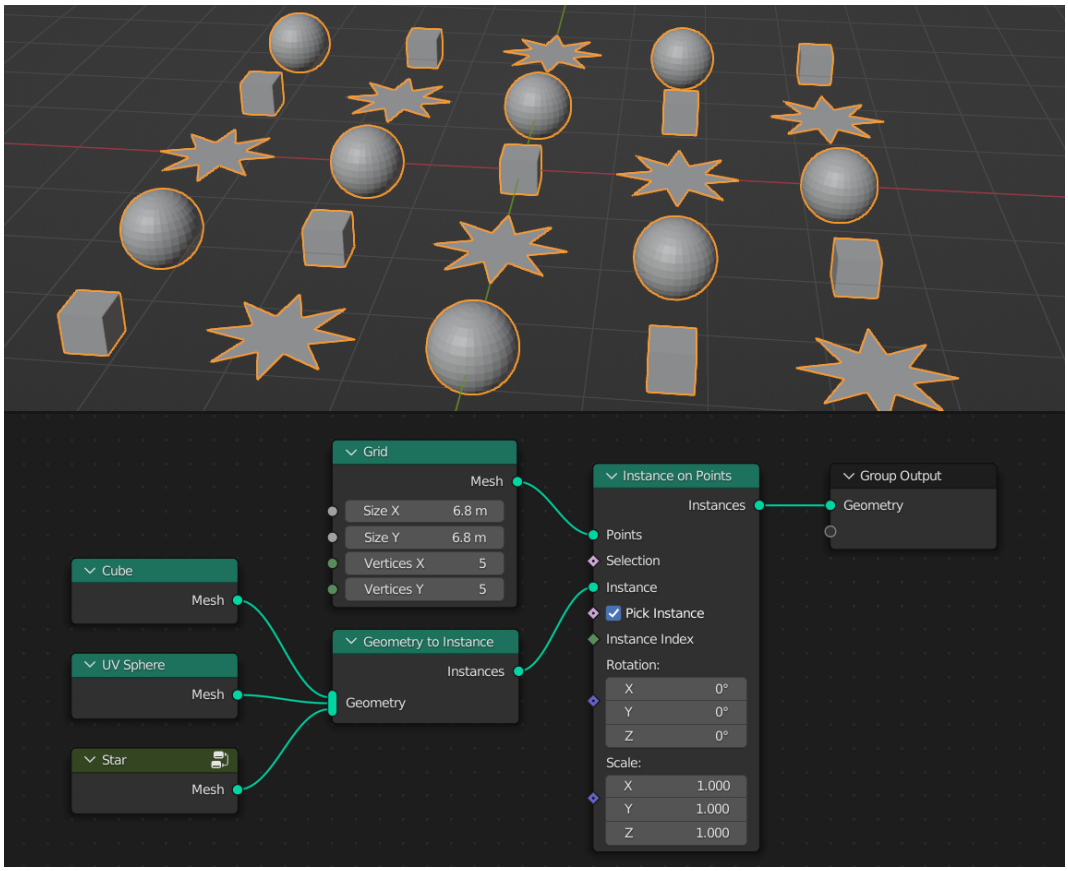
Properties

This node has no properties.

Output

Geometry
: Standard geometry output.

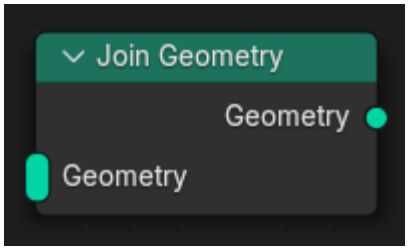
Examples



The node used in combination with the Instance on Points Node to choose between multiple primitives for instancing.

Join Geometry Node

The *Join Geometry* node merges separately generated geometries into a single one. If the geometry inputs contain different types of data, the output will also contain different data types.



Note

The node cannot handle the case when more than one geometry input has a volume component.

Materials

When multiple mesh inputs contain different materials, the material slots from each mesh geometry are merged so that the output mesh will contain all the input materials.

Attributes

When merging attributes from multiple geometry inputs, the highest complexity data type is chosen for the output attribute. In other words, if a `weight` attribute has a Boolean type on one geometry input and a vector data type on another geometry, the `weight` attribute on the output geometry will have a vector data type. The same heuristic is used for attribute domains, the domain with the most information will be used for the output.

Warning

Like other geometry nodes, this node always outputs generic typed attributes. So instead of a Vertex Group attribute, it will create a “Float” attribute on the result, and it will create a generic 2D vector attribute instead of a special “UV Map” attribute. Some other areas of Blender don’t properly handle generic attributes in version 3.0.

Custom face corner normals are also not transferred currently.

Inputs

Geometry
: Geometry that will be joined. Multiple inputs are allowed. When the node is muted, only the first link will be passed through.

Properties

This node has no properties.

Output

Geometry
: Standard geometry output.

Curve Nodes

Nodes that only operate on curves.

Note

If any modifiers precede the Geometry Nodes modifier, the curve will be seen as a mesh internally, and the Curve Nodes won't work. To get around this, first run the geometry through a Mesh to Curve Node.

- Read
  - Curve Handle Positions Node
  - Curve Length Node
  - Curve Tangent Node
  - Curve Tilt Node
  - Endpoint Selection Node
  - Handle Type Selection Node
  - Is Spline Cyclic Node
  - Spline Length Node
  - Spline Parameter Node
  - Spline Resolution Node
- Sample
  - Sample Curve Node
- Write
  - Set Curve Normal Node
  - Set Curve Radius Node
  - Set Curve Tilt Node
  - Set Handle Positions Node
  - Set Handle Type Node
  - Set Spline Cyclic Node
  - Set Spline Resolution Node
  - Set Spline Type Node

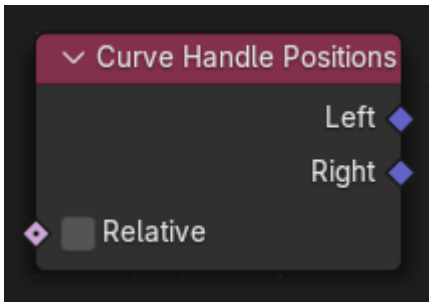
- Operations
  - Curve to Mesh Node
  - Curve to Points Node
  - Curves to Grease Pencil Node
  - Deform Curves on Surface Node
  - Fill Curve Node
  - Fillet Curve Node
  - Grease Pencil to Curves Node
  - Interpolate Curves Node
  - Merge Layers Node
  - Resample Curve Node
  - Reverse Curve Node
  - Subdivide Curve Node
  - Trim Curve Node
- Primitives
  - Arc Node
  - Bézier Segment Node
  - Curve Circle Node
  - Curve Line Node
  - Curve Spiral Node
  - Quadratic Bézier Node
  - Quadrilateral Node
  - Star Node
- Topology
  - Curve of Point Node
  - Offset Point in Curve Node
  - Points of Curve Node

Read Curve Nodes

- Curve Handle Positions Node
- Curve Length Node
- Curve Tangent Node
- Curve Tilt Node
- Endpoint Selection Node
- Handle Type Selection Node
- Is Spline Cyclic Node
- Spline Length Node
- Spline Parameter Node
- Spline Resolution Node

Curve Handle Positions Node

Gets the two handle positions of each control point in a Bézier spline.



You can use the Set Handle Positions Node to change these positions.

Inputs

Relative
: Output the handle positions relative to the control point instead of in the local space of the geometry.

Properties

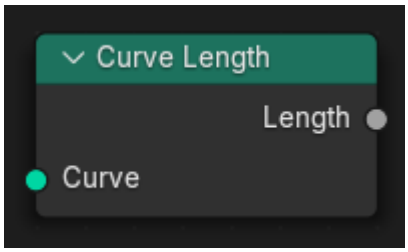
This node has no properties.

Outputs

Left
: The position of the control point's left handle.

Right
: The position of the control point's right handle.

Curve Length Node



The *Curve Length* node outputs the length of all splines added together.

Inputs

Curve
: Standard geometry input.

Properties

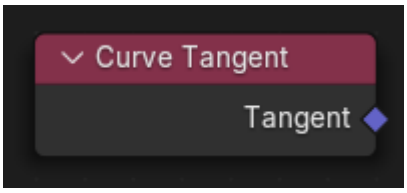
This node has no properties.

Output

Length
: Accumulated length of all splines of the curve.

Curve Tangent Node

The *Curve Tangent* node outputs the direction that a curve points in at each control point, depending on the direction of the curve (which can be controlled with the Reverse Curve Node). The output values are normalized vectors.



Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For example, a Bézier spline might have 48 evaluated points, but only four control points, if its resolution is 12. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node can be used to create a poly spline, where there is a control point for every evaluated point.

Inputs

This node has no inputs.

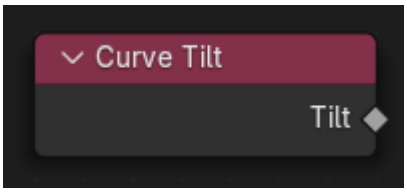
Properties

This node has no properties.

Outputs

Tangent
: The direction of the curve at every control point.

Curve Tilt Node



The *Curve Tilt* node outputs the angle used to turn the curve normal around the direction of the curve tangent in its evaluated points. Keep in mind that the output is per control point, just like the values that can be controlled in curve Edit Mode. For NURBS and Bézier splines, the values will be interpolated to the final evaluated points.

The input node for this data is the Set Curve Tilt node.

Inputs

This node has no inputs.

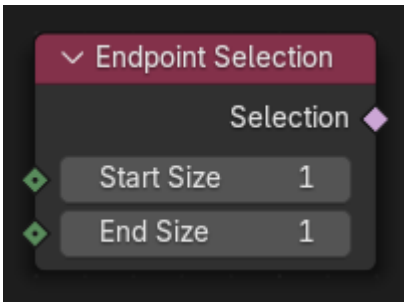
Properties

This node has no properties.

Outputs

Tilt
: The tilt angle for the normal in radians.

Endpoint Selection Node



The *Endpoint Selection* node provides a selection for an arbitrary number of endpoints in each spline in a curve.

Note

The selection operates for every control point. This may not correspond to the evaluated points displayed in the viewport for NURBS and Bézier splines, where one control point may correspond to many evaluated points.

Tip

To use this data after the curve has been converted to another data type like mesh or a point cloud, the Capture Attribute Node can be used.

Inputs

Start Size
: The number of points to select from the start.

End Size
: The number of points to select from the end.

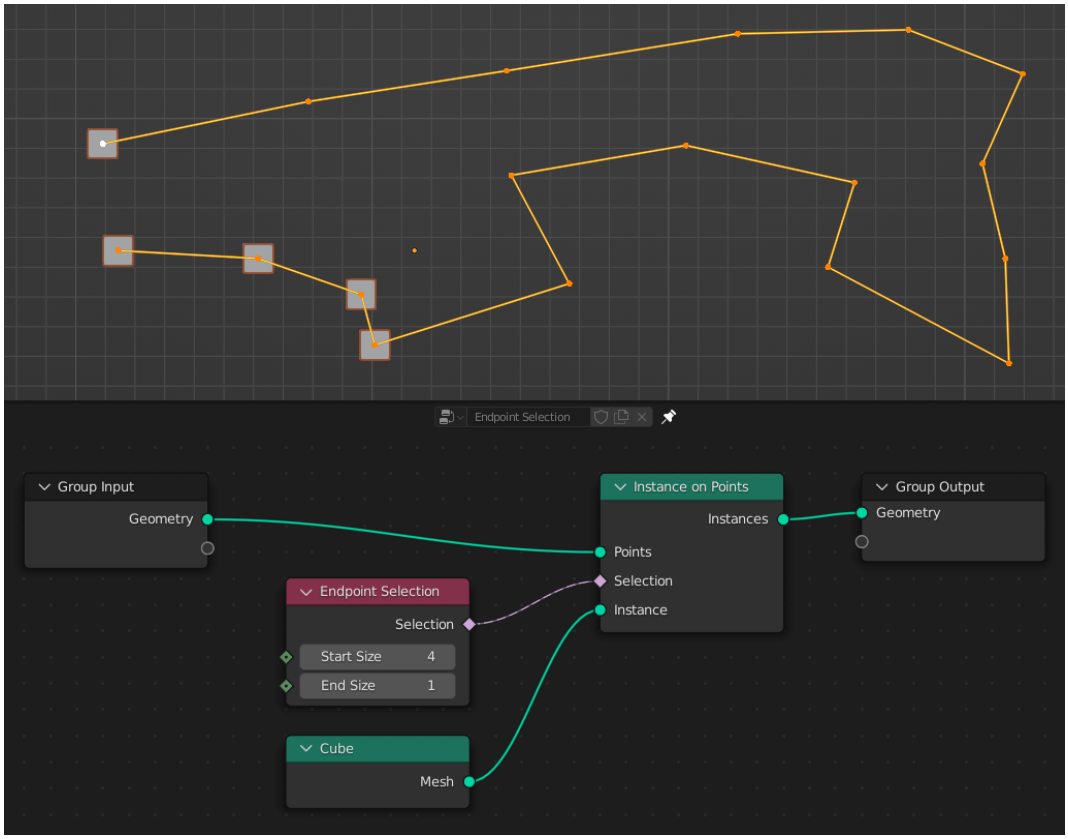
Properties

This node has no properties.

Outputs

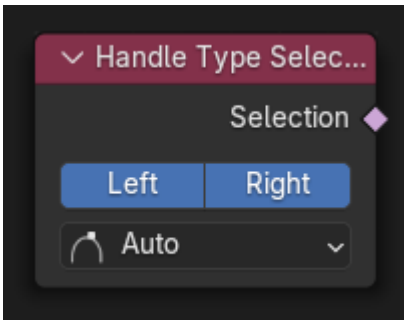
Selection
: Selection of the start and end of each spline of the curve.

Examples



Anywhere the geometry is a curve, this node can be used to generate a selection of only the first and last points of each spline. Here, the *Points* input of the Instance on Points Node is a curve consisting of the poly spline shown in Edit Mode.

Handle Type Selection Node



Creates a selection based on the handle types of the control points.

The handle type of each control point can be changed with the Set Handle Type Node.

Inputs

This node has no inputs.

Properties

Mode
: Whether to consider left and/or right handles. When both are selected, the output value is true if either of the handles are of the chosen type.

  Left:: Consider the left handle.
  Right:: Consider the right handle.

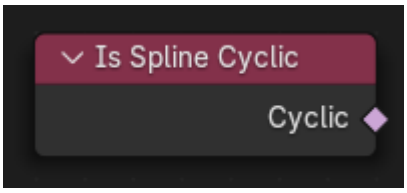
Handle Type
: Handle type for which the selection will be true. See the Bézier curves page for more details on the different handle types.

Outputs

Selection

Boolean field set to true wherever the handle type matches.

Is Spline Cyclic Node



The *Is Spline Cyclic* controls whether each of the curve splines start and endpoints form a connection. Its output corresponds to the built-in `cyclic` attribute on the curve spline domain.

The node to set this data is the Set Spline Cyclic Node.

Inputs

This node has no inputs.

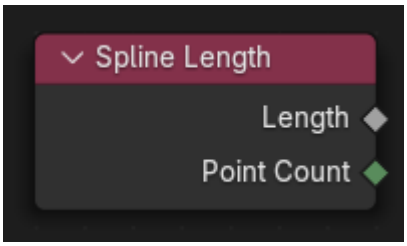
Properties

This node has no properties.

Outputs

Cyclic
: Whether the spline is cyclic.

Spline Length Node



The *Spline Length* node outputs the total length of each spline, as a distance, or a number of points. This is different than the Curve Length node, which adds up the total length for all of the curve's splines.

The output values correspond to the spline domain, but the node can be used to output a value for every curve control point as well.

Inputs

This node has no inputs.

Properties

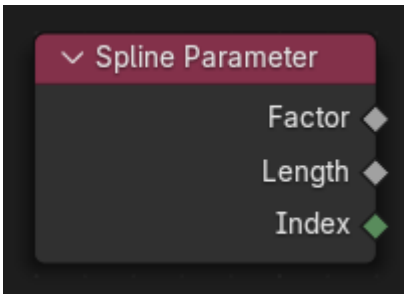
This node has no properties.

Outputs

Length
: The length of each spline.

Point Count
: The number of control points in each spline.

Spline Parameter Node



The *Spline Parameter* node outputs how far along each spline a control point is. The *Factor* output is different from dividing the index by the total number of control points, because the control points might not be equally spaced along the curve.

The first value is zero, so the output corresponds to the length at the control point rather than including the length of the following segment.

When used on the spline domain, the node outputs the portion of the total length of the curve (including all splines) has been traversed at the start of each spline. The order of the curve's splines is visible in the Spreadsheet Editor.

Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node node can be used to create a poly spline, where there is a control point for every evaluated point.

Note

When the *Length* is zero, the *Factor* is arbitrary. In this case the result is exceptionally calculated dividing the index by the total number of control points or curves.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

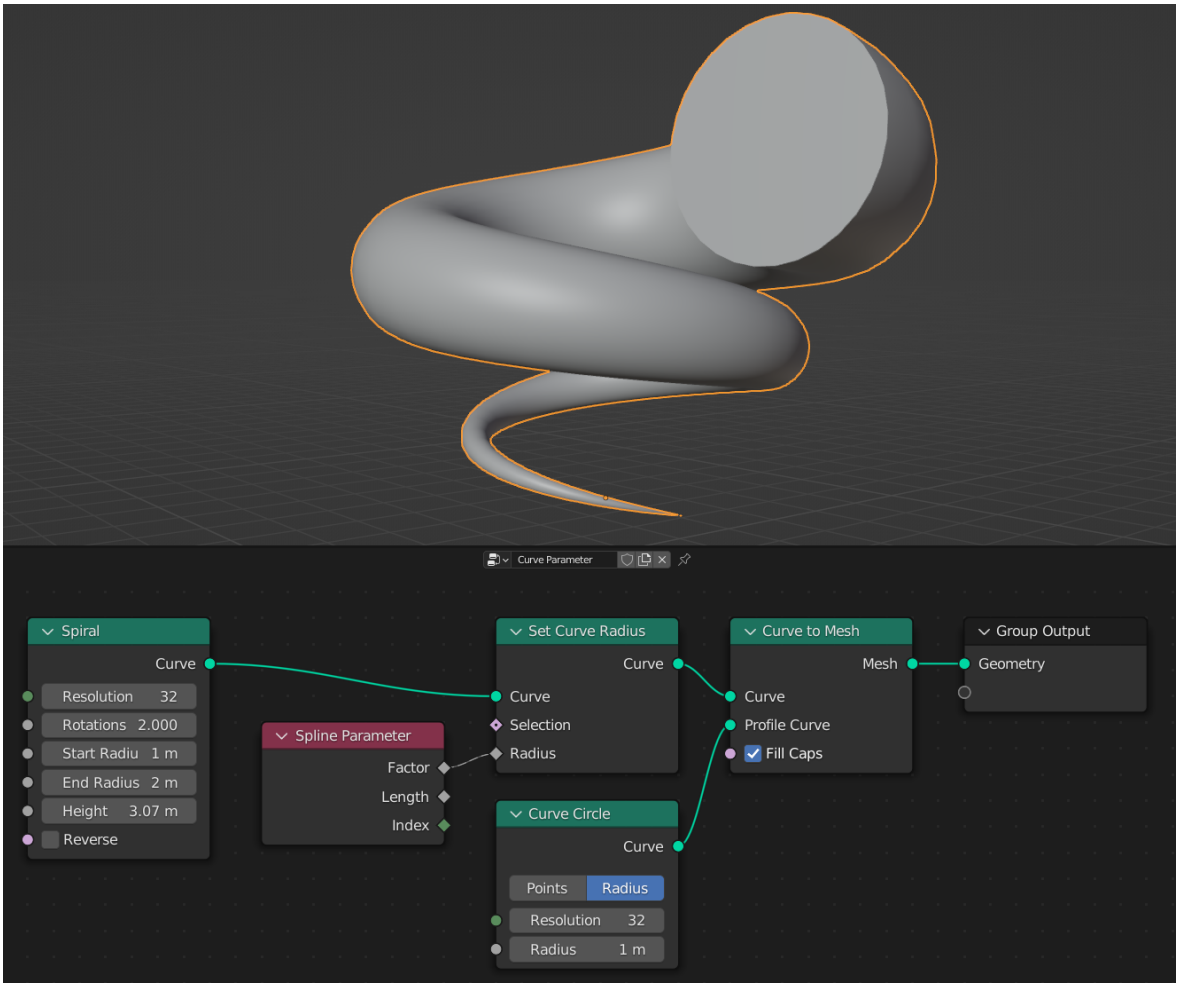
Factor
: When the node is used on the point domain, the value is the portion of the spline's total length at each control point. On the spline domain it is the portion of the curve's total length at the start of the spline.

Length
: When the node is used on the point domain, the value is the distance along the spline to each control point. On the spline domain it is the length along the entire curve at the start of the spline.

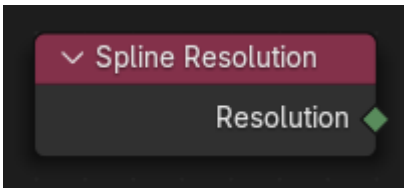
Index
: Each control point's index on its spline. This is different from the output of the index node, which also counts the control points in all previous splines.

Examples

The parameter used to control the radius of the curve. The beginning of the spline has a radius of 0, the end has a radius of 1.


Curve Parameter
Spiral
Curve
Resolution 32
Rotations 2.000
Start Radiu 1 m
End Radius 2 m
Height 3.07 m
Reverse
Spline Parameter
Factor
Length
Index
Set Curve Radius
Curve
Curve
Selection
Radius
Curve Circle
Curve
Points
Radius
Resolution 32
Radius 1 m
Curve to Mesh
Mesh
Curve
Profile Curve
Fill Caps
Group Output
Geometry

Spline Resolution Node



The *Spline Resolution* outputs the number of evaluated curve points that will be generated for every control point on the spline. This node works for NURBS, Bézier, and Catmull Rom splines.

For poly splines, there is a one-to-one correspondence between original points and evaluated points, so the resolution does not have an effect.

On Bézier splines, the resolution does not have an effect on segments between vector handles, since there are no extra evaluated points between the neighboring control points.

The node to set this data is the Set Spline Resolution Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

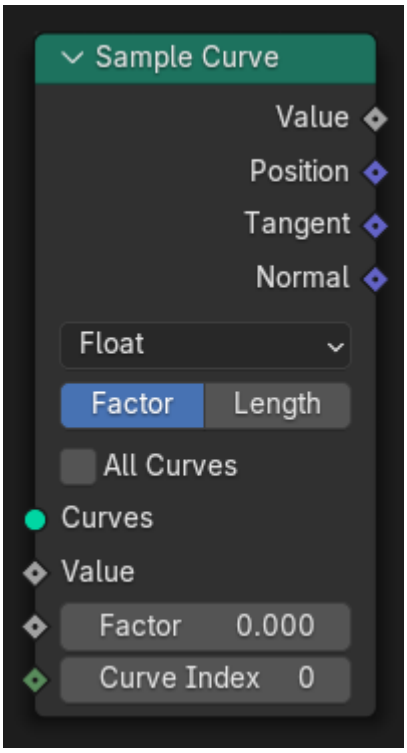
Resolution
: The integer resolution value for each spline.

Curve Sample Nodes

- Sample Curve Node

Sample Curve Node

The *Sample Curve* calculates a point on a curve at a certain distance from the start of the curve, specified by the length or factor inputs. It also outputs data retrieved from that position on the curve. The sampled values are linearly interpolated from the values at the evaluated curve points at each side of the sampled point.



Note

When the curve contains multiple splines, the sample position is found based on the total accumulated length, including the lengths of all previous splines. The order of the splines is the same order as displayed in the Spreadsheet Editor.

Inputs

Curves
: Standard geometry input with a curve component.

Value
: A field input to evaluate custom attributes. The evaluation is outputted through the *Value* output.

Factor *Factor mode*
: The portion of the total length used to determine the sample position.

Length *Length mode*

A length in distance units used to determine how far along the curve to travel before sampling.

Curve Index
: An index to only evaluate specific splines, these indices can be specified manually or from the Index Node. This input is ignored when the *All Curves* property is enabled.

Properties

Data Type
: The data type used for the evaluated data.

Mode
: How to find endpoint positions for the trimmed spline. The option acts the same as it does in the Trim Curve Node.

Factor::	Find the endpoint positions using a factor of each spline's length. The input values should be between 0 or 1.
Length::	Find the endpoint positions using a length from the start of each spline. The input values should be between 0 and the length of the splines.

All Curves
: Sample lengths based on the total length of all curves, rather than using a length inside each selected curve.

Outputs

Value
: The value of the input *Value* at the sample point.

Position
: The position at the sample along the spline.

Tangent

The normalized curve tangent at the sample.

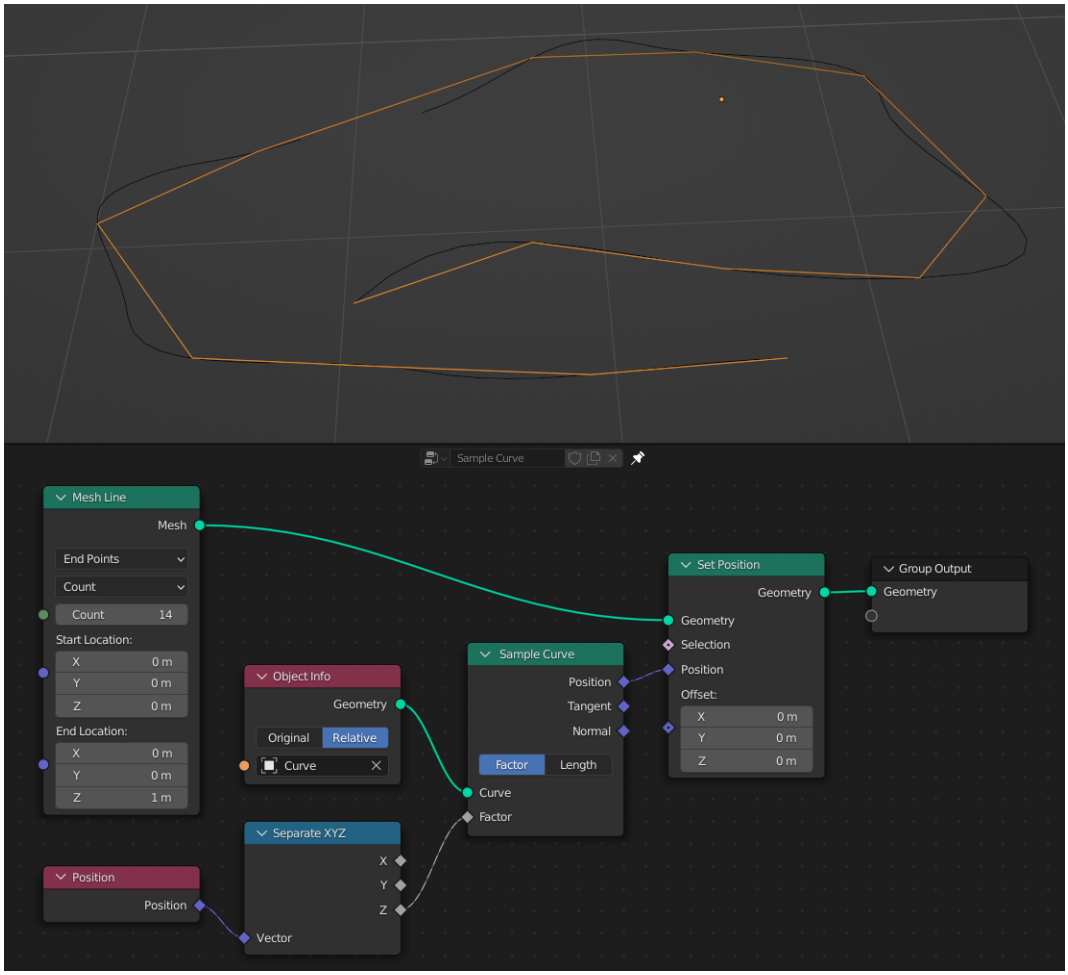
Tip

This output can be combined with the Align Rotation to Vector Node to create a rotation that lines up with direction of the curve. Including the *Normal* output in a second align node after can align the other rotation axis.

Normal

The normalized curve normal at the sample.

Examples

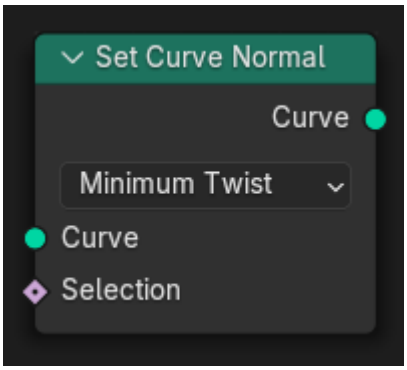


Here, the *Count* mode of the Resample Curve Node is recreated, except a mesh is used for the result instead of a curve. The Z axis of the position can be used as the sample factor because the position is between zero and one for the entire line.

Write Curve Nodes

- Set Curve Normal Node
- Set Curve Radius Node
- Set Curve Tilt Node
- Set Handle Positions Node
- Set Handle Type Node
- Set Spline Cyclic Node
- Set Spline Resolution Node
- Set Spline Type Node

Set Curve Normal Node



The *Set Curve Normal* controls the method used to calculate curve normals for every curve.

The node doesn't set the normals directly, those are calculated later as necessary. Combined with the tilt attribute value at each control point, this will define the final normals accessible with the Normal Node.

Internally this node adjusts the values of the `normal_mode` attribute on each curve.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the value on each curve.

Normal
: Input for the custom normal attribute (`custom_normal`) when using *Free* mode.

Properties

Mode
: The method for evaluation of the curve's normals

  Minimum Twist:: The final normals are calculated to have the smallest twist around the curve tangent across the whole curve.

Z-Up:: The final normals are calculated so that they are perpendicular to the Z axis and the tangent. If a series of points is vertical, the X axis is used.

Free:: Use the stored custom normal attribute (`custom_normal`) as the final normals.
This mode adds a *Normal* input that can be used to set the value of the custom normal.

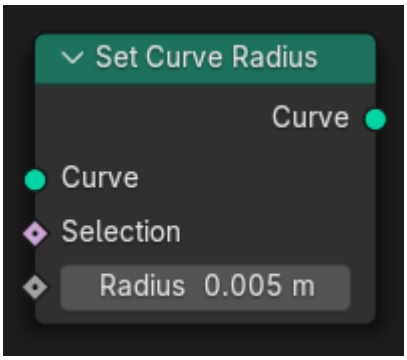
Note
Custom normals are not rotation invariant, meaning normals must be set **after** any rotation transformations; i.e. at the end of the node tree or at the bottom of the modifier stack.

Outputs

Curve
: Standard geometry output.

Set Curve Radius Node

The *Set Curve Radius* controls the radius of the curve, used for operations like the size of the profile in the Curve to Mesh node. The value is set for every control point, and is then interpolated to each evaluated point in between the control points.



The input node for this data is the Radius node.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the value on each control point. True values mean the value will be changed, false values mean the value will remain the same.

Radius
: The radius value for each control point.

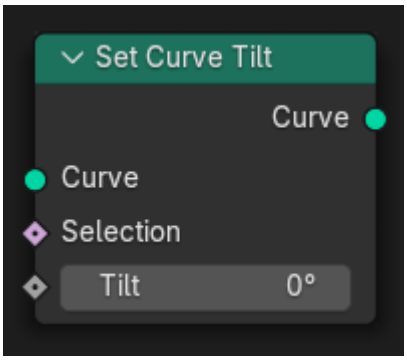
Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Set Curve Tilt Node



The *Set Curve Tilt* controls the tilt angle at each curve control point. That angle rotates normal vector which is generated at each point when evaluating the curve. The normal then can be retrieved with the Normal Node.

The rotation of the normal vector is an Axis Angle rotation. It is the same as the Vector Rotate Node operation with the tangent vector as the axis, the raw evaluated normal is used as the original vector, and the tilt as the rotation angle.

The input node for this data is the Curve Tilt Node.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the value on each control point. True values mean the value will be changed, false values mean the value will remain the same.

Tilt
: The tilt angle for each control point.

Properties

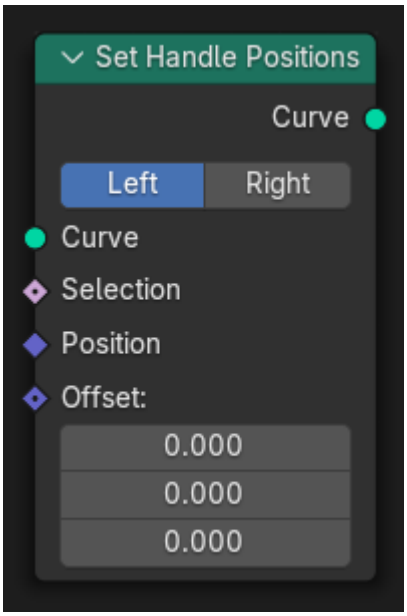
This node has no properties.

Outputs

Curve
: Standard geometry output.

Set Handle Positions Node

The *Set Handle Positions* node sets the positions for the handles of Bézier curves. They can be used to alter the generated shape of the curve. The input node for this data is the Curve Handle Positions Node. See the Bézier curves page for more details.



Note

When the position is changed, *Auto* handle types will be converted to *Aligned*, and *Vector* handle types will be converted to *Free*.

Note

The left and right handles cannot be changed at the same time with this node. That is because it would break the alignment for left and right handles at the same control point.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the handle position on each control point. True values mean the resolution will be changed, false values mean the resolution will remain the same.

Position

The new handle position.

Note

The handle positions are the global position of the handle, they are not relative to the position of the corresponding control point.

Offset
: An optional translation for each handle. This is evaluated at the same time as the *Position* input, meaning that fields evaluated for it will not reflect the changed position.

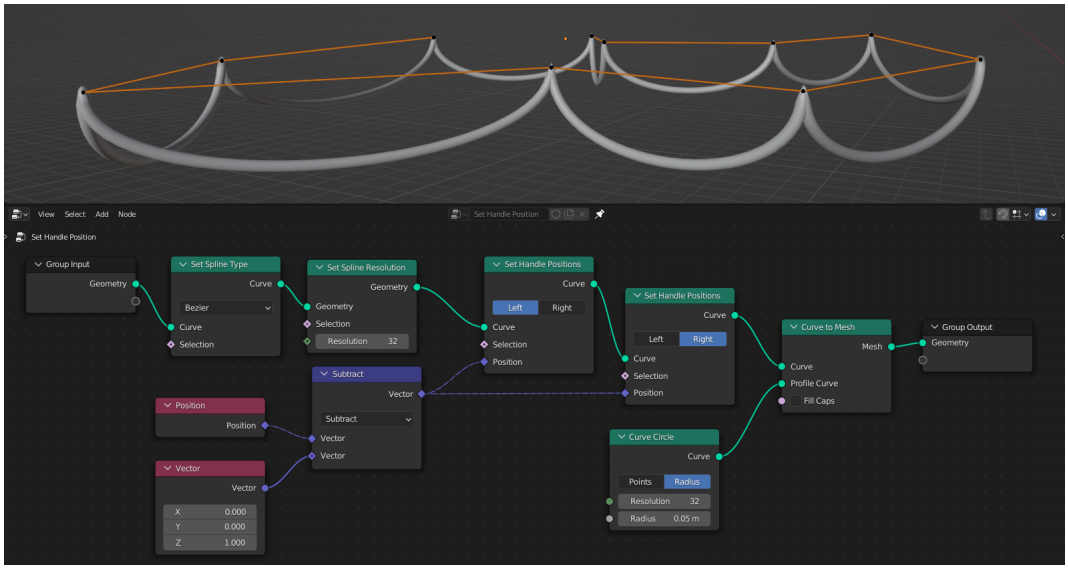
Properties

Left / Right
: Whether to set the handle position of the left or right handle. The *Left* handle is closer to the start of the spline, and the *Right* handle is closer to the end.

Outputs

Curve
: Standard geometry output.

Examples

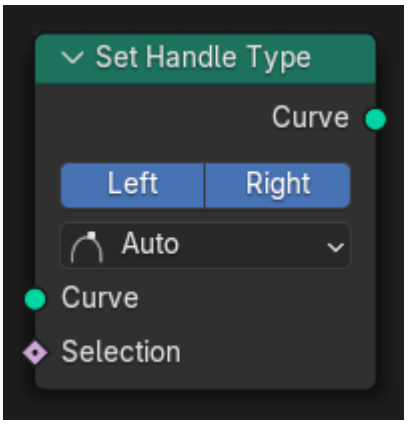


Here, the handles are adjusted to the same position as the control points, but offset down in the Z direction slightly. With the Set Spline Type Node, the curve can be a poly spline in Edit Mode, to make adjustment simpler for the user of the node group.

Set Handle Type Node

Sets the handle type for the points on the Bézier curve that are in the selection.

A selection for a certain handle type can be retrieved with the Handle Type Selection Node.



Inputs

Curve
: Bézier curve.

Selection
: The points whose handle types will be changed.

Properties

Mode
: Whether to update left or right or both handles. When both are selected, both the left and the right handles will be updated.

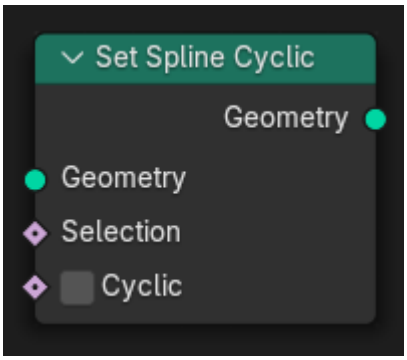
  Left:: Update the left handles.
  Right:: Update the right handles.

Handle Type
: The handle type to switch to. See the Bézier curves page for more details on the different handle types.

Outputs

Curve
: Standard curve output.

Set Spline Cyclic Node



The *Set Spline Cyclic* controls whether each spline will loop back on itself. Each spline has the same number of control points whether or not it is set as cyclic. But when displaying in the viewport or for operations with other nodes, a connection will be made between the first and last control points.

The input node for this data is the Is Spline Cyclic Node.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to change the cyclic value on each spline. True values mean the value will be changed, false values mean the value will remain the same.

Cyclic
: Whether or not to join the first and last points of each spline.

Properties

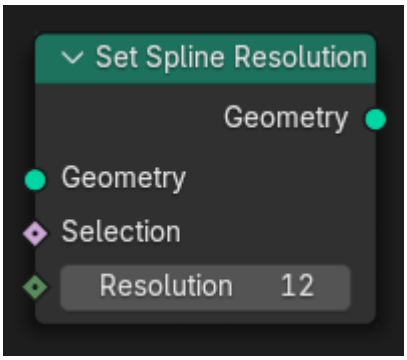
This node has no properties.

Outputs

Curve

Standard geometry output.

Set Spline Resolution Node



The *Set Spline Resolution* node sets the value for how many evaluated points should be generated on the curve for every control point. It only has an effect on NURBS, Bézier, and Catmull Rom splines. In case of Bézier splines, the resolution does not have an effect on segments between vector handles.

The evaluated points are displayed in the viewport, used in the Curve to Mesh Node node, and optionally used in the Resample Curve Node.

The input node for this data is the Spline Resolution Node.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to change the resolution value on each spline. True values mean the value will be changed, false values mean the resolution will remain the same.

Resolution
: The number of evaluated points generated for each control point in NURBS and Bézier splines. It controls the accuracy of operations like trimming or sampling a curve. Higher resolutions are more accurate, but slower.

Properties

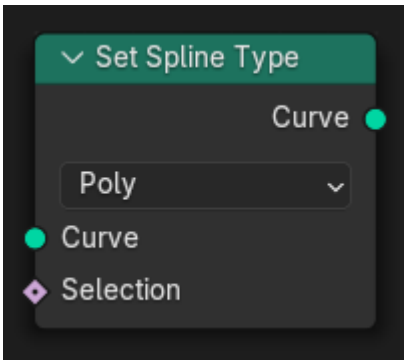
This node has no properties.

Outputs

Curve
: Standard geometry output.

Set Spline Type Node

Sets the spline type for the splines in the curve component that are in the selection.



Inputs

Curve
: Standard geometry input with a curve component.

Selection
: The splines whose spline types will be changed.

Properties

Spline Type
: The type to convert the splines in the selection to. Read the Spline Types page for more details on the different spline types.

  Bézier:: Convert to a Bézier spline. A spline converted from a poly spline gets vector handles, while one converted from NURBS or Catmull Rom spline gets auto handles.

  Note
  When converting from a NURBS spline to a Bézier spline, at least six points are needed. When the number of points is not a multiple of

three a full conversion is not possible and the spline has to be truncated.

NURBS:: Convert to a NURBS spline.

Poly:: Convert to a poly spline.

Catmull Rom:: Convert to a Catmull Rom spline.

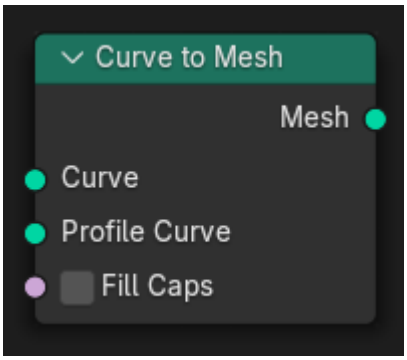
Outputs

Curve
: Standard curve output.

Curve Operation Nodes

- Curve to Mesh Node
- Curve to Points Node
- Curves to Grease Pencil Node
- Deform Curves on Surface Node
- Fill Curve Node
- Fillet Curve Node
- Grease Pencil to Curves Node
- Interpolate Curves Node
- Merge Layers Node
- Resample Curve Node
- Reverse Curve Node
- Subdivide Curve Node
- Trim Curve Node

Curve to Mesh Node



The Curve to Mesh node converts all splines of a curve to a mesh. Optionally, a profile curve can be provided to give the curve a custom shape. This profile curve is scaled by the radius at each control point; see the Set Curve Radius Node.

The node transfers attributes to the result. Attributes that are built-in on meshes but not curves, like `sharp_face`, will be transferred to the correct domain as well.

Tip

The output mesh has sharp edges set from the profile curve tagged automatically. If any splines in the profile curve are Bézier splines and any of the control points use *Free* or *Vector* handles, the corresponding edges will be shaded sharp.

Inputs

Curve
: Standard geometry input. All non-curve components are ignored.

Profile Curve

If a profile curve is provided, it will be extruded along all splines. Otherwise the generated mesh will just be a chain of edges.

Fill Caps
: If the profile spline is cyclic, fill the ends of the generated mesh with n-gons. The resulting mesh is Manifold, the two new faces for each spline are simply connected to existing edges.

Properties

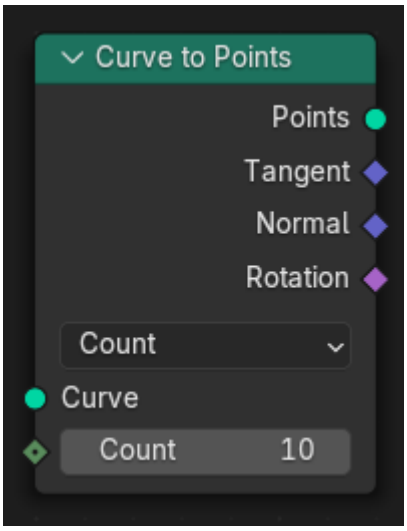
This node has no properties.

Outputs

Mesh
: Standard geometry output.

Curve to Points Node

The *Curve to Points* node generates a point cloud from a curve.



Inputs

Curve
: Standard curve input.

Count
: Number of points generated. This input is only available for *Count* mode.

Length
: Length of the curve. This input is only available for *Length* mode.

Properties

Mode

Evaluated:: Creates points from the curve's evaluated points based on the resolution attribute for NURBS and Bézier splines. This mode will generally be the fastest, since the second step of sampling equal lengths is not necessary.

Count:: Sample each spline by evenly distributing the specified number of points.

Length:: Sample each spline by splitting it into segments with specified length. The length will be rounded down so that a whole number of samples will fit in each input spline. To counteract jumping when the length of the spline changes, the Trim Curve Node can be used with a multiple of this length.

Outputs

Points
: Generated point cloud.

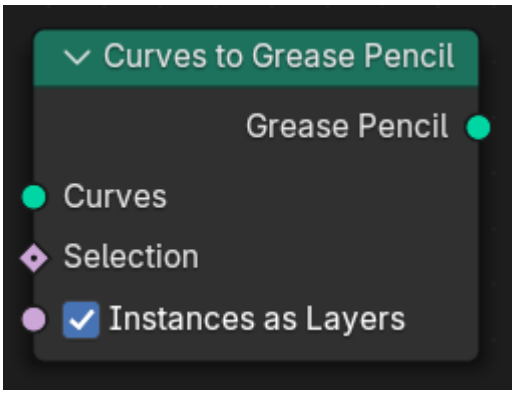
Tangent
: The normalized curve tangent at the sampled position, or the direct evaluated normal in *Evaluated* mode.

Normal
: The normal value from the evaluated curve at each result point. This is the same value from the Normal Node at those positions.

Rotation
: The Euler rotation build from the *Tangent* and *Normal* outputs, for convenience.

Curves to Grease Pencil Node

The *Curves to Grease Pencil* node Converts top-level curve instances into Grease Pencil layers



Inputs

Curves
: Either plain curves or curve instances.

Selection
: Either a curve or instance selection.

Instances as Layers
: Create a separate layer for each instance. If instances are used, realized curve geometry will be ignored. Layer names will use the instance geometry name. If real curve geometry is used, a single layer is created with the input geometry's name.

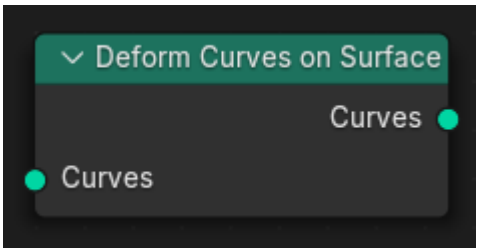
Properties

This node has no properties.

Outputs

Grease Pencil
: Standard Grease Pencil geometry.

Deform Curves on Surface Node



The *Deform Curves on Surface* node translates and rotates each curve based on the difference in its root position. The root position is defined by UV coordinates stored on each curve and the UV Map selected for the purpose in the Curves surface settings.

The transformation is calculated based on the difference of the original mesh (before shape keys and modifiers are evaluated), and the final mesh.

Unlike other geometry nodes, this node has quite a few implicit inputs:

- The original and evaluated mesh are retrieved from the modifier object's surface property. This means the node only works for curves objects.
- The original and evaluated UV map are also retrieved from the object's surface property.
- A 3D vector attribute named `rest_position`, used for calculating tangents for rotating curves that are consistent with the tangents calculated on the original mesh (the rotation needs to be calculated from the normal and tangent of the original and evaluated meshes).
- A 2D vector attribute on the curve domain named `surface_uv_coordinate` to store the location of the root positions on the surface mesh's UV map.

In future development, this node will be generalized so the setup is more flexible.

Parts of the internal operation are similar to the Sample UV Surface Node.

Warning

In order to achieve consistent deformation after the Subdivision Surface Modifier, the *UV Smooth* option of the modifier should be set to *None*. Otherwise the surface UV map will be subdivided in a way that may invalidate the curve UV attachement points stored in the `surface_uv_coordinate` attribute.

Inputs

Curves
: Standard curves input.

Properties

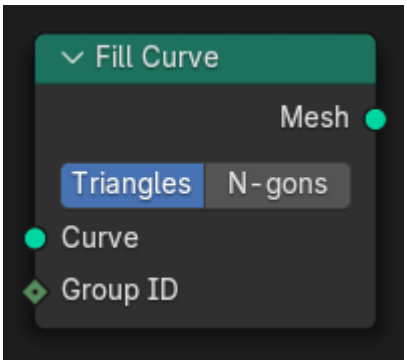
This node has no properties.

Outputs

Curves
: Standard curves output.

Fill Curve Node

The *Fill Curve* node generates a mesh using the constrained Delaunay triangulation algorithm with the curves as boundaries. The mesh is only generated flat with a local Z of 0.



Inputs

Curve
: Standard geometry input with a curve component.

Group ID
: Value used to group curves together. Curves with different Group ID are treated separately.

Properties

Mode
: The type of geometry the output consists of.

Triangles::	The output is made up of triangles.
N-gons::	The output is made up of n-gons.

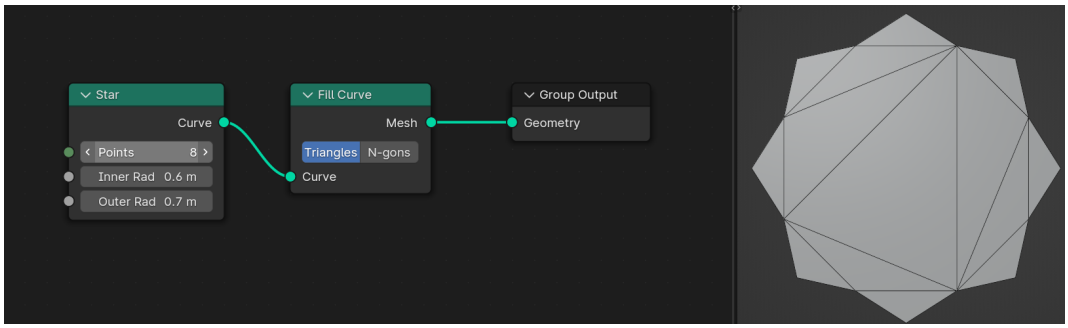
Outputs

Mesh
: The filled-in curves.

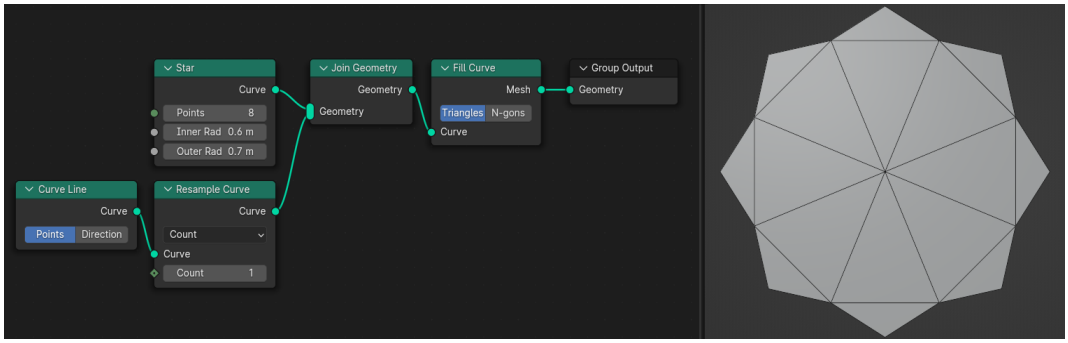
Examples

Customized triangulation

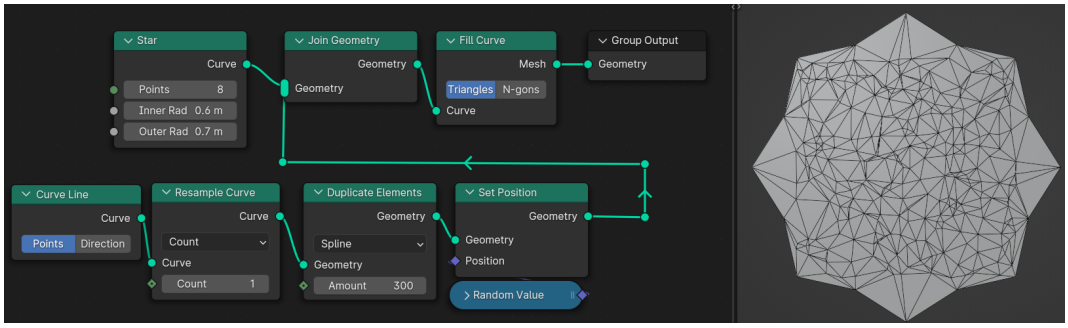
One or many “single point spline” can be used to customize the triangulation of the filled-in curves.



This is the default behavior of the *Fill Curve* node applied to the star primitive.



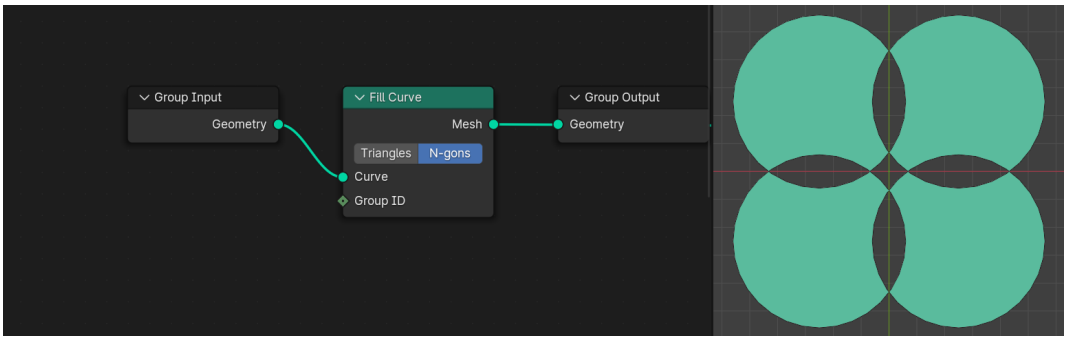
Here, a single curve point is joined to the star primitive to customize the triangulation.



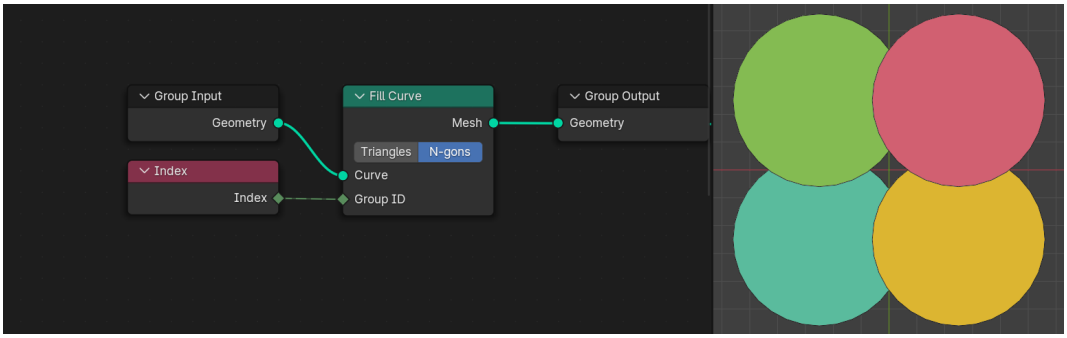
Here, 300 single curve point are joined to the star primitive to customize the triangulation.

Group ID

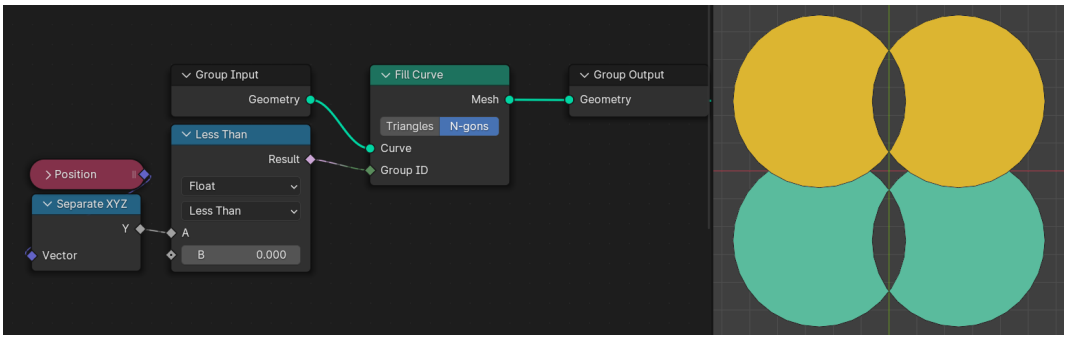
The following figures display diverse application of the Group ID.



Here, the 4 curves share the same Group ID, resulting in 1 mesh island (default behavior).



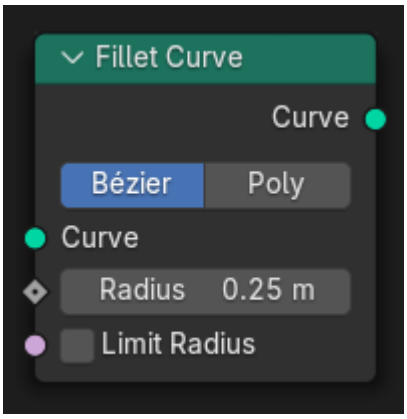
Here, the 4 curves have different Group ID, resulting in 4 mesh islands.



Here, the 4 curves are seperated into two groups based on their positions, resulting in 2 mesh islands.

Fillet Curve Node

The *Fillet Curve* rounds corners on curve control points, similar to the effect of the Bevel Modifier on a 2D mesh. However, a key difference is that the rounded portions created by the Fillet Curve node are always portions of a circle.



Inputs

Curve
: Standard geometry input with a curve component.

Radius
: The radius of the circle portion generated at each fillet.

Count
: Only in *Poly* mode, the number of control points to add for each fillet.

Limit Radius
: Whether to limit the maximum value of the radius in order to avoid overlapping fillets.

Properties

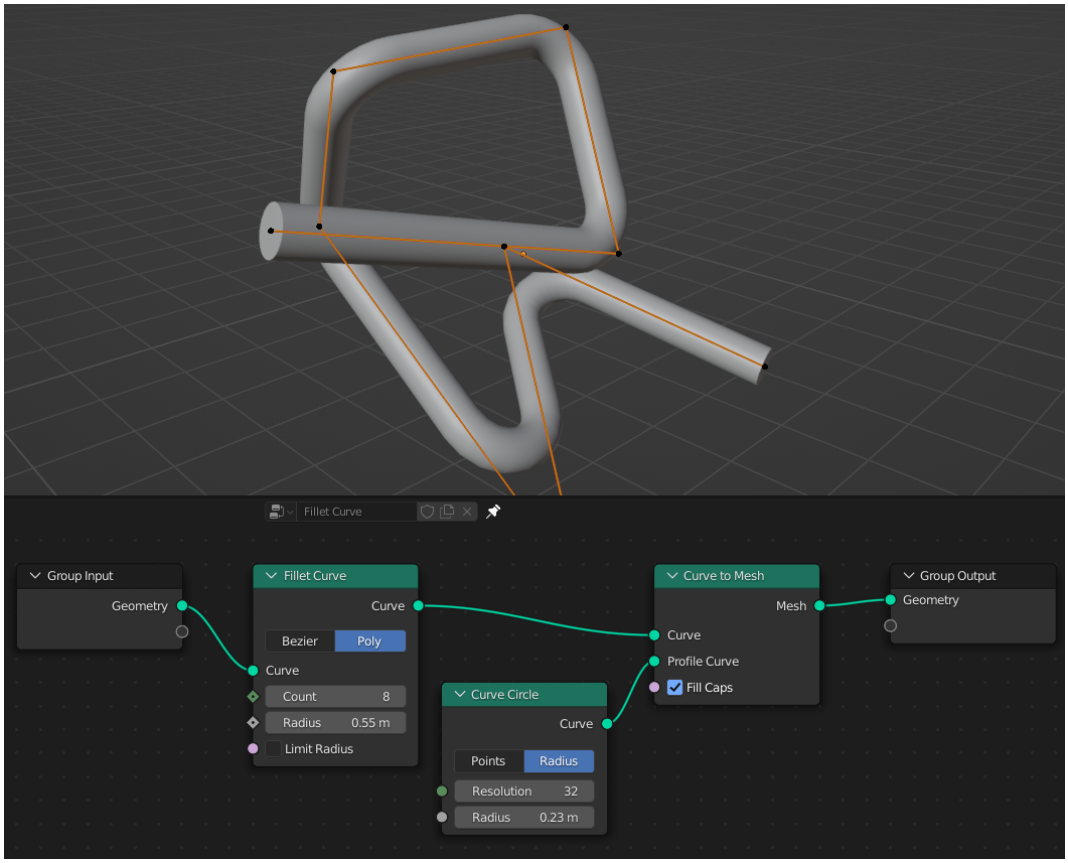
Method

Bézier::	Only two control points will be generated for every filleted control point. The shape generated by the aligned handles on the generated control points on either side of the fillet is used to create the circle portion shape, meaning that the number of segments in the fillet shape depends on the spline's resolution value.
Poly::	The number of control points generated for each field input is controlled directly with an integer field input. This mode works better for poly and NURBS splines.

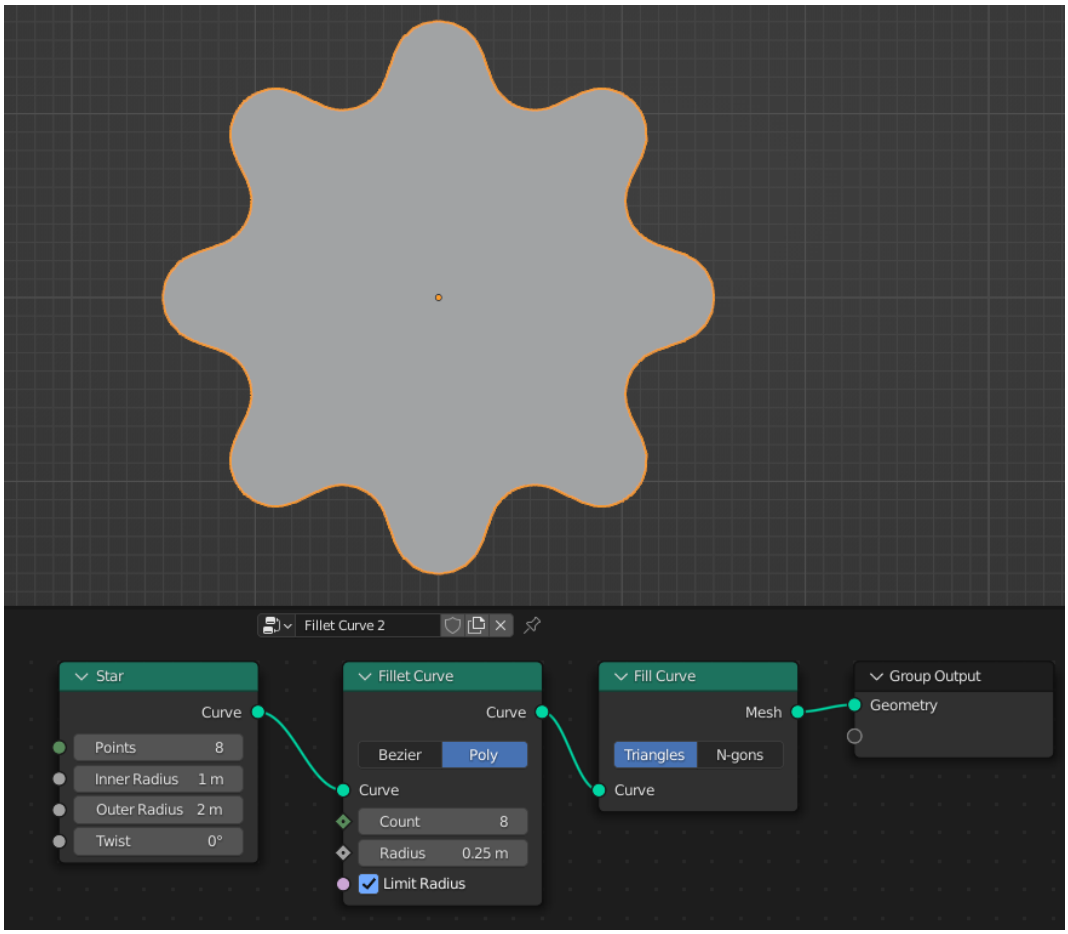
Outputs

Curve
: Standard geometry input with a curve component.

Examples



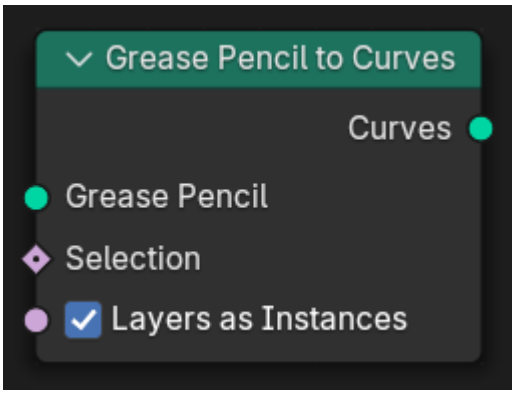
The node can be used to round the corners of simple 3D poly splines.



The node can be combined with the curve primitive nodes to make more interesting shapes.

Grease Pencil to Curves Node

The *Grease Pencil to Curves* node converts each Grease Pencil layer into an instance that contains curves.



Inputs

Grease Pencil
: Standard Grease Pencil geometry.

Selection
: Selects the layers to convert.

Instances as Layers
: Create a separate curve instance for every layer.

> **Tip**
>
> This is equivalent to a Realize Instances operation after creating the curve geometry instances. This can simplify some logic later on, but typically keeping this operation off will be much better for performance.

Properties

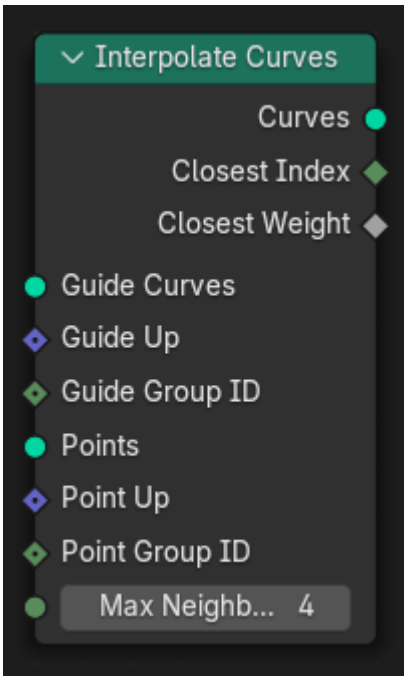
This node has no properties.

Outputs

Curves
: Standard curves geometry.

Interpolate Curves Node

Generate new curves on points by interpolating between existing curves. This is useful to have a smaller set of original curves to make editing easier and faster while still generating high-density curves for the viewport or a final render.



Inputs

Guide Curves

Base curves that new curves are interpolated between.

Guide Up

Optional up vector that is typically a surface normal. Supplying an up vector will improve the quality of the interpolation, making it aware of the surface shape which it otherwise wouldn't have a way to know about.

Tip

In a typical "child hair" generation setup, this up direction is retrieved with a combination of the Sample UV Surface Node using the same geometry that the points were distributed on, and the Normal Node.

Guide Group ID
: Splits guides into separate groups. New curves interpolate existing curves from a single group. This input can be useful for generating hair parts by stopping curves in different sections (with different group IDs) from affecting each other.

Points
: The positions of the first root control points of the newly generated interpolated curves.

Points Up
: Optional up vector that is typically a surface normal.

Point Group ID
: The curve group to interpolate in.

Max Neighbors
: Maximum amount of close guide curves that are taken into account for interpolation.

Properties

This node has no properties.

Outputs

Curves
: The guide curves with interpolated curves.

Closest Index
: Index of the closest guide curve for each generated curve.

> **Note**
>
> Internally this node mixes the data from multiple guide curves, with the maximum number of sources depending on the *Max Neighbors*

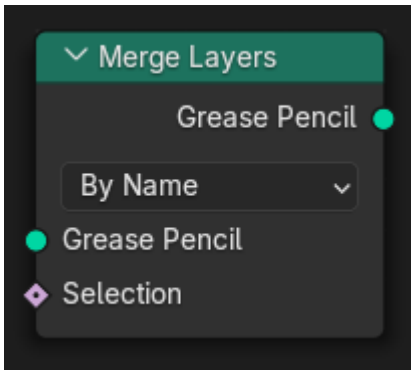
input. This output is only the index of the curve with the largest weight.

Closest Weight

Weight of the closest guide curve for each generated curve.

Merge Layers Node

Combines multiple Grease Pencil Layers into a single layer.



See also

Merge Layers Operator

Inputs

Grease Pencil
: The grease pencil object with multiple layers to combine.

Selection
: Operate of a subset of layers by setting the field layer index value to *true*. By default, all layers are selected.

Group ID
: The index numbers of the layer to be merged. This input is only visible when using *By Group ID* Mode.

Properties

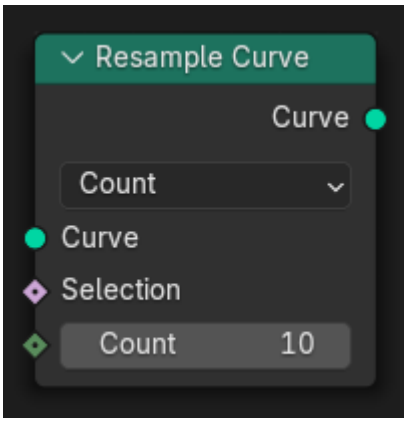
Mode
: Determines how to choose which layers are merged.

By Name::	Combine all layers which have the same name.
By Group ID::	Provide a custom group ID for each layer and all layers with the same ID will be merged into one.

Outputs

Grease Pencil
: The grease pencil object with combined layers.

Resample Curve Node



The Resample Curve node creates a poly spline for each input spline. In the *Count* and *Length* modes, the control points of the new poly splines will have uniform spacing.

Tip

Use a field as an input to have a different count/length for each spline.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to resample each spline. True values mean spline will be resampled to a poly spline, false values mean the spline will be unaffected.

Count
: The number of control points on the new splines.

Length
: The approximate length between the control points of the new splines.

> **Tip**
> A Trim Curve Node can be used with a multiple of the input length to make the distance between each sampled point exact, even when the length of the spline changes.

Properties

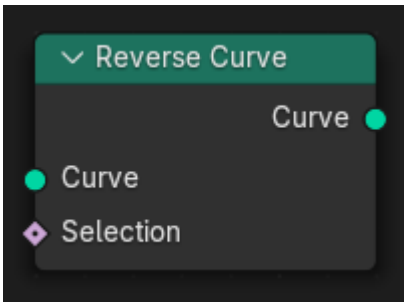
Mode
: How to specify the amount of samples.

Count::	Sample the specified number of points along each spline.
Length::	Calculate the number of samples by splitting each spline into segments with the specified length. The length will be rounded down so that a whole number of samples will fit in each input spline.
Evaluated::	Evaluate the spline's points based on the resolution attribute for NURBS and Bézier splines. Changes nothing for poly splines.

Outputs

Curve
: Standard geometry output.

Reverse Curve Node



The *Reverse Curve* node swaps the start and end of splines. The shape of the splines is not changed.

Tip

When used on the *Profile* input of the Curve to Mesh Node, this node fill flip the normals of the resulting mesh.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to change the direction of each spline. True values mean the direction will be changed, false values mean the spline will be unaffected.

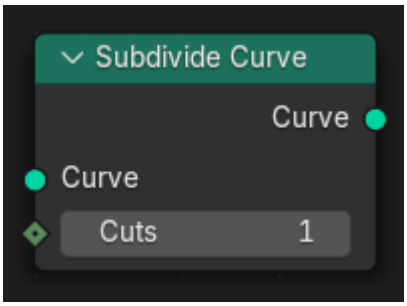
Properties

This node has no properties.

Outputs

Curve
: Standard geometry output.

Subdivide Curve Node



The *Subdivide Curve* node adds more control points in between existing control points on the curve input. For Bézier and poly splines, the shape of the spline will not be changed at all.

With Bézier curves, this can be used to increase the control on the shape of the curve while still having the higher-level provided by Bézier splines. Unlike the Resample Curve Node, where they are converted to poly splines.

Inputs

Curve
: Standard geometry input.

Cuts
: The number of control points to create on the segment following each point. When the input is a field, the number of cuts for a segment is determined by the value of the field when evaluated at the previous point.

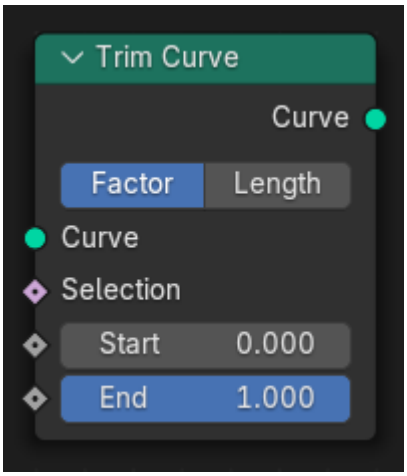
Properties

This node has no properties.

Outputs

Curve
: Standard geometry output.

Trim Curve Node



The *Trim Curve* node shortens each spline in the curve by removing sections at the start and end of each spline.

Bézier splines will still be Bézier splines in the output, with the first and last control point and its handles moved as necessary to preserve the shape. NURBS splines will be transformed into poly splines in order to be trimmed.

Warning

Currently the Trim Curve node does not support cyclic splines.

Note

Since curve normals are calculated the final curve, this node may change the resulting normals when the **Minimum** twist method is used, since the **Minimum** method considers the entire length of the curve to decide the final normals. In some cases the Capture Attribute Node could be used to avoid this, by saving the original normals to be used later.

Inputs

Curve
: Standard geometry input with a curve component.

Selection
: A boolean field input for each curve indicating whether it is trimmed or not.

Start
: The factor or length used to determine where to start each output spline.

> **Note**
>
> If the *Start* input is larger than the *End*, then the resulting spline will have a single point, located at the sample location of the *Start* value.

End
: The factor or length used to determine where to end each output spline.

Properties

Mode
: How to find endpoint positions for the trimmed spline.

Factor::	Find the endpoint positions using a factor of each spline's length. The input values should be between 0 or 1.
Length::	Find the endpoint positions using a length from the start of each spline. The input values should be between 0 and the length of the splines.

Outputs

Curve

Standard geometry output.

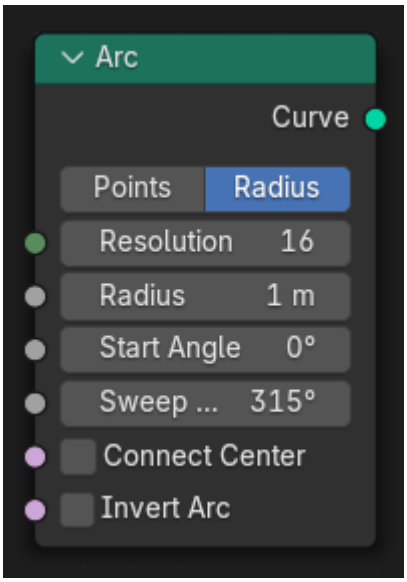
Curve Primitive Nodes

Nodes that create a primitive curve, e.g., a circle.

- Arc Node
- Bézier Segment Node
- Curve Circle Node
- Curve Line Node
- Curve Spiral Node
- Quadratic Bézier Node
- Quadrilateral Node
- Star Node

Arc Node

The *Arc* node generates a poly spline arc. The node has two modes, Radius and Points.



Inputs

Resolution
: Number of edges on the arc.

Radius
: The radius of the arc. *Radius* mode only.

Start Angle
: Starting angle of the arc. *Radius* mode only.

Sweep Angle
: Length of the arc. *Radius* mode only.

Connect Center
: Connect the arc at the center.

Invert Arc
: Invert and draw opposite arc.

Start, Middle, End
: The three points on the arc. *Points* mode only. The order of the points determines the direction (clockwise or counterclockwise) of the arc. The arc will always draw from Start to End via the Middle point. This can be changed by using the Invert Arc option.

Offset Angle
: Offset angle of the arc. *Points* mode only.

Note

Because of the finite resolution, the middle point does not necessarily lie on the generated arc.

Properties

Mode

Points::	The position and radius of the arc are determined by three points. The center of the arc, radius and normal are also given as outputs.
Radius::	The arc is determined by the radius, start angle and sweep angle.

Outputs

Curve
: Poly spline generated from the inputs.

Center
: The center of the arc described by the three points. *Points* mode only.

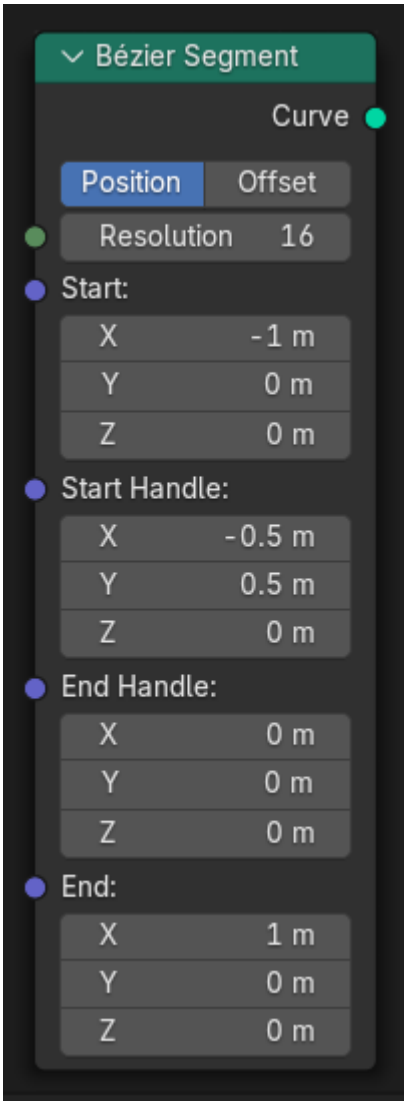
Normal
: The normal direction of the plane described by the three points, pointing towards the positive Z axis. *Points* mode only.

Radius

The radius of the arc described by the three points. *Points* mode only.

Bézier Segment Node

The *Bézier Segment* node generates a 2D Bézier spline from the given control points and handles.



Inputs

Resolution
: The number of edges on the curve.

Start, End
: Positions of the start and end control point of the curve.

Start Handle, End Handle
: Positions of the handles used to define the shape of the curve.

Properties

Mode

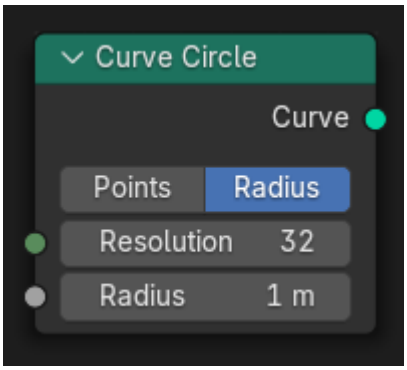
Position::	The handle inputs are the absolute positions of the handles in 3D space.
Offset::	The handle positions are relative to the control point on the curve. The handle inputs give the offset from the control points.

Outputs

Curve
: Bézier spline generated from the inputs.

Curve Circle Node

The *Curve Circle* node generates a poly spline circle.



Inputs

Resolution
: Number of edges on the circle.

Radius
: The radius of the circle.

Point 1, Point 2, Point 3
: The three points on the circle. The order of the points determines the direction (clockwise or counterclockwise) of the circle.

Note

Because of the finite resolution, the three points do not necessarily lie on the generated curve.

Properties

Mode

Points:: The position and radius of the circle are determined by three points. The center of the circle is also given as an output. If the three points lie on one line, no geometry is generated.

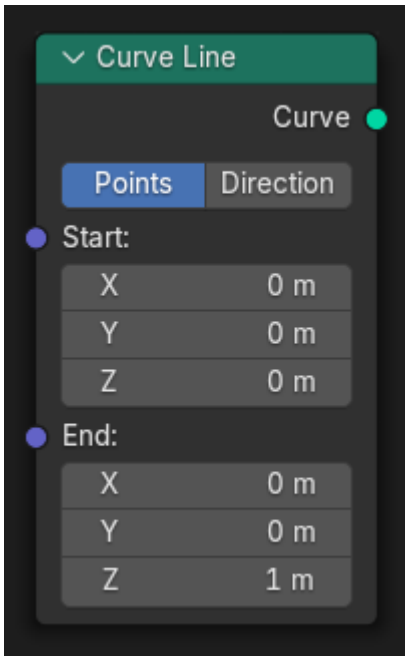
Radius:: The circle is determined by the radius.

Outputs

Curve
: Poly spline generated from the inputs.

Center
: The center of the circle defined by the three points.

Curve Line Node



The *Curve Line* node generates poly spline line.

Inputs

Start
: Position of the first control point.

End
: Position of the second control point. This is only available in the *Points* mode.

Direction
: Direction the line is going in. The length of this vector does not matter. This is only available in the *Direction* mode.

Length
: Length of the line. This is only available in the *Direction* mode.

Properties

Mode:

Points::	Define the spline with start and end points.
Direction::	Define the spline with a start, direction and length.

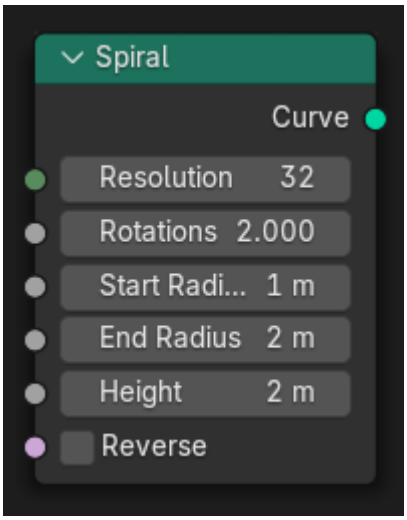
Outputs

Curve

Standard geometry output.

Curve Spiral Node

The *Curve Spiral* node generates a poly spline in a spiral shape. It can be used to create springs or other similar objects. By default the spiral twists in a clockwise fashion.



Inputs

Resolution
: Number of points in one rotation of the spiral.

Rotations
: Number of times the spiral makes a full rotation.

Start Radius, End Radius
: Radius of the start point and end point of the spiral. The radius of the spiral changes linearly between the two values over the whole spiral.

Height
: Height of the spiral.

Reverse
: Boolean value that changes the direction from clockwise to counterclockwise when it is enabled.

Properties

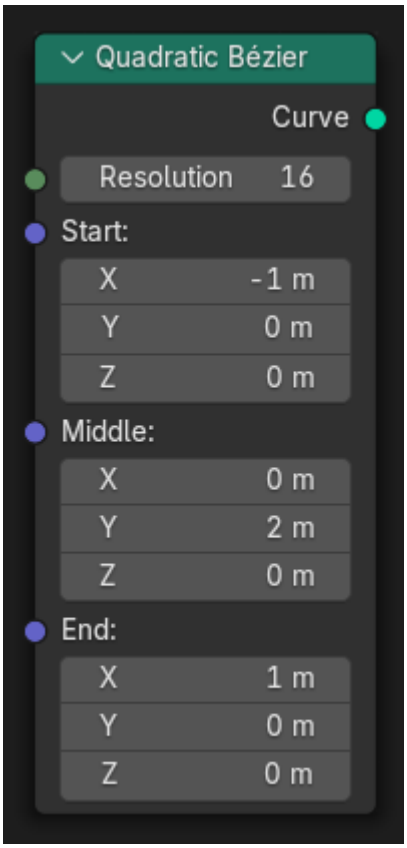
This node has no properties.

Outputs

Curve
: Poly spline generated from the inputs.

Quadratic Bézier Node

The *Quadratic Bézier* node generates a poly spline curve from the given control points. The generated shape is a parabola.



Inputs

Resolution
: The number of edges on the curve.

Start, Middle, End
: Positions of the three control points. The generated curve passes through the two end points, and is tangent to the lines between the middle point and the two end points.

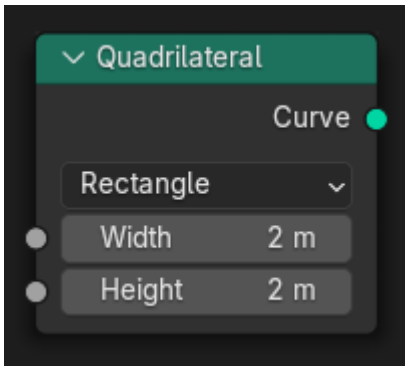
Properties

This node has no properties.

Outputs

Curve
: Poly spline generated from the inputs.

Quadrilateral Node



The *Quadrilateral* node generates a polygon with four points, with different modes.

Inputs

Width / Bottom Width / Top Width
: The X axis size of the shape.

Height
: The Y axis size of the shape.

Bottom Height / Top Height
: The distance between the bottom or top point and the X axis, in *Kite* mode.

Offset
: In *Parallelogram* mode, the relative X difference between the top and bottom edge. In *Trapezoid* mode, the amount to move the top edge in the positive X axis.

Point 1 - 4
: Input vectors for the *Points* mode.

Properties

Mode

Rectangle:: Generate a rectangle-shaped curve with a width and a height.

Parallelogram:: Generate a rectangle with an offset for the different X values of the top and bottom edges.

Trapezoid:: Generate a trapezoid-shaped curve with a height, a width for the top and bottom, and an offset.

Kite:: Generate a kite shape with a width, and the top and bottom points distance from the X axis.

Points:: Generate a four point cyclic poly spline by inputting the position vectors directly.

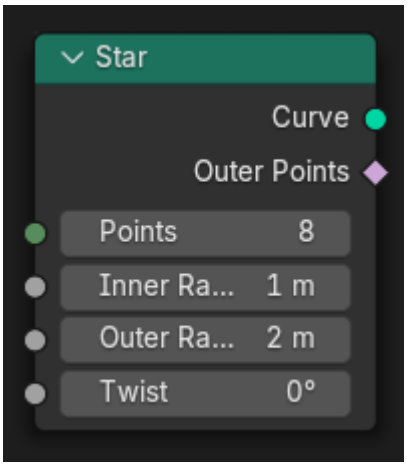
Outputs

Curve

Poly spline generated from the inputs.

Star Node

The *Star* node generates a poly spline in a star pattern by connecting alternating points of two circles. The points on the inner circle are offset by a rotation so that they lie in between the points on the outer circle. This offset can be changed with the twist input.



Inputs

Points
: Number of points on each of the circles.

Inner Radius, Outer Radius
: Radii of the two circles on which to place the control points. The inner radius can be larger than the outer radius.

Twist
: Angle offset of the inner circle. The twist value rotates the points on the circle corresponding with the inner radius counterclockwise by the given angle.

Properties

This node has no properties.

Outputs

Curve

Poly spline generated from the inputs.

Outer Points

A boolean attribute field with a selection of the points on the *Outer Radius*, which is every other point.

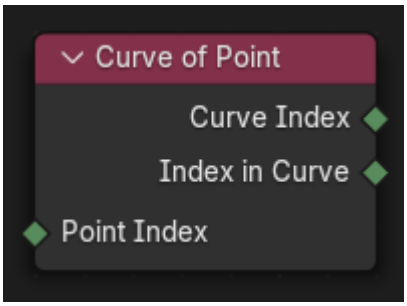
Curve Topology Nodes

Nodes that retrieve information about the connectivity between curves and control points.

- Curve of Point Node
- Offset Point in Curve Node
- Points of Curve Node

Curve of Point Node

The *Curve of Point* node retrieves the index of the curve a control point is part of. This node is conceptually similar to the Face of Corner Node.



Inputs

Point Index
: The index of the input control point.

> **Note**
>
> By default this uses the index from the field context, which makes it important that the node is evaluated on the point domain.

Properties

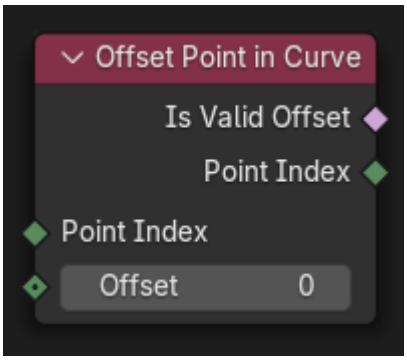
This node has no properties.

Outputs

Curve Index
: The index of the curve the control point is part of.

Index in Curve
: How far along the control point is along its curve, with a value of 0 for the first point in each curve.

Offset Point in Curve Node



The *Offset Point in Curve* node retrieves other points in the same curve as the input control point. This is like starting at a specific control point and walking along neighboring points toward the start or end of the curve.

Conceptually the operation is similar to the Offset Corner in Face Node, but the point index doesn't wrap around to the other end of the curve unless it is cyclic.

Inputs

Point Index
: The index of the input control point.

> **Note**
>
> By default this uses the index from the field context, which makes it important that the node is evaluated on the point domain.

Offset
: The number of points to move around the curve before finding the result. If the curve is cyclic and the offset goes past the start or end point of the curve, it will wrap around to the other side.

Properties

This node has no properties.

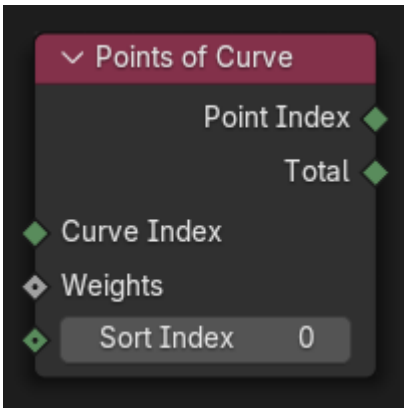
Outputs

Is Valid Offset
: Whether the input control point plus the offset is a valid index of the original curve. Any offset in a cyclic curve is always valid.

Point Index
: The index of the offset curve point.

Points of Curve Node

The *Points of Curve* node retrieves indices of specific control points in a curve.



Inputs

Curve Index
: The index of the input curve.

  Note

  By default this uses the index from the field context, which makes it important that the node is evaluated on the curve domain.

Weights
: Values used to sort the curve's control points. By default the control points are sorted by index, so the control points with the smallest indices come first.

Sort Index
: Which of the sorted control points to use for the *Point Index* output. If the value is larger than the total number of control points, it will wrap around to the beginning.

Properties

This node has no properties.

Outputs

Point Index
: The index of one of the curve's control points, chosen by the *Sort Index* input.

Total
: The number of control points in the curve.

Instances Nodes

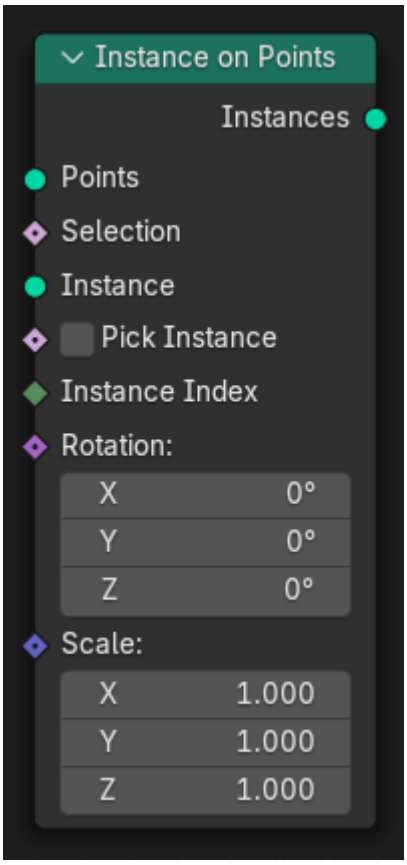
Nodes that only operate on instances.

- Instance on Points Node
- Instances to Points Node

- Realize Instances Node
- Rotate Instances Node
- Scale Instances Node
- Translate Instances Node
- Set Instance Transform Node

- Instance Transform Node
- Instance Rotation Node
- Instance Scale Node

Instance on Points Node



The *Instance on Points* node adds a reference to a geometry to each of the points present in the input geometry. Instances are a fast way to add the same geometry to a scene many times without duplicating the underlying data. The node works on any geometry type with a *Point* domain, including meshes, point clouds, and curve control points.

Any attributes on the points from the *Geometry* input will be available on the instance domain of the generated instances.

Tip

The Make Instances Real operator can be used to create objects from instances generated with this node.

Note

To instance object types that do not contain geometry, like a light object, the Object Info Node can be used. Other objects like Metaball objects are not supported for instancing.

Inputs

Points

Standard geometry input. The position of the points of this geometry affect the transforms of each instance output.

Note

If the input geometry contains instances, the node will create more instances on the points inside the instances, creating nested instancing. In this case, each new instance will have the transform created by the node from the *Rotation* and *Scale* inputs, but it will *also* be transformed based on the parent instances.

Selection

Whether to instance on each point. True values mean an instance will be generated on the point, false values mean the point will be skipped.

Instance

The geometry to instance on each selected point. This can contain real geometry, or multiple instances, which can be useful when combined with the *Pick Instance* option.

Pick Instances

If enabled, instead of adding the entire geometry from the *Instance* input on every point, choose an instance from the instance list of the geometry based on the *Instance Index* input. This option is intended to be used with the Collection Info Node.

Instance Index

The selection of index for every selected point, only used when *Pick Instances* is true. By default the point ID is used, or the index if that doesn't exist. Negative values or values that are too large are wrapped around to the other end of the instance list.

Rotation

The Euler rotation for every instance. This can use the rotation output of nodes like Distribute Points on Faces and Curve to Points. An Euler rotation can also be created from a direction vector like the normal with the Align Rotation to Vector Node.

Scale

The size of each generated instance.

Properties

This node has no properties.

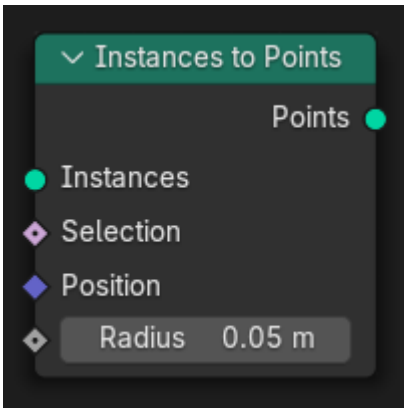
Outputs

Instances

Standard geometry output. If the `id` attribute exists on the input geometry, it will be copied to the result instances.

Instances to Points Node

The *Instances to Points* node generates points at the origins of top-level instances. Attributes on the instance domain are moved to the point cloud points.



Note

Top-level instances are those that are owned by the node's input geometry. Instances owned by other instances, i.e. nested instances, are not considered by this node.

Inputs

Instances
: Standard geometry input.

Selection
: The instances used to generate points. True values mean a point is created for the instance, false values mean the instance is skipped.

Position
: Overrides the default position of generated point.

Radius
: Controls the radius of the result points.

Properties

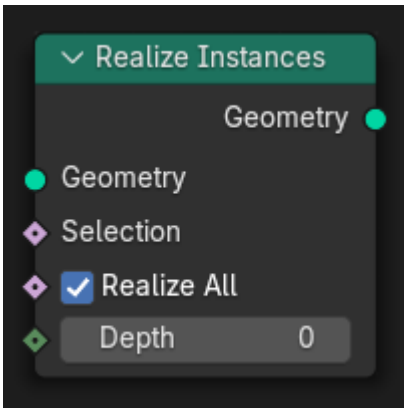
This node has no properties.

Outputs

Points
: Standard geometry output.

Realize Instances Node

The *Realize Instances* node makes any instances (efficient duplicates of the same geometry) into real geometry data. This makes it possible to affect each instance individually, whereas without this node, the exact same changes are applied to every instance of the same geometry. However, performance can become much worse when the input contains many instances of complex geometry, which is a fundamental limitation when procedurally processing geometry.



Note

If the input contains multiple volume instances, only the first volume component is moved to the output.

Attributes

When merging attributes from multiple geometry inputs, the highest complexity data type is chosen for the output attribute. In other words, if a `weight` attribute has a Boolean type on one geometry input and a vector data type on another geometry, the `weight` attribute on the output geometry will have a vector data type.

Named and anonymous attributes are propagated from the instance domain to the realized geometry. If the same attribute exists on the geometry and on an instance, the attribute values from the geometry has precedence over the values on the instances.

In order to avoid creating duplicate values, the `id` attribute has special handling. The `id` values or indices of each instance are combined with id values from the points on geometry data.

Warning

Like other geometry nodes, this node always outputs generic typed attributes. So instead of a Vertex Group attribute, it will create a "Float" attribute on the result, and it will create a generic 2D vector attribute instead of a special "UV Map" attribute. Some other areas of Blender don't properly handle generic attributes in version 3.0.

Custom face corner normals are also not transferred currently.

Inputs

Geometry
: Standard geometry input.

Selection
: Which top-level instances to realize.

Realize All
: Realize all levels of nested instances for each top-level instances (overrides the value of the *Depth* input).

Depth
: Number of levels of nested instances to realize for each top-level instance.

Properties

This node has no properties.

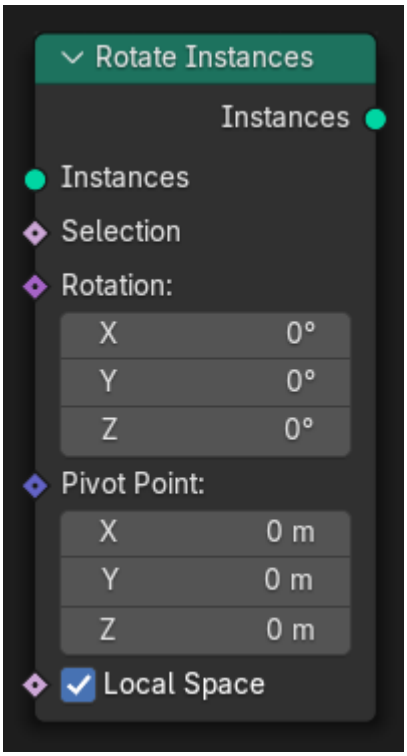
Outputs

Geometry
: Standard geometry output.

Rotate Instances Node

The *Rotate Instances* node rotates geometry instances in local or global space.

The Instances page contains more information about geometry instances.



Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be rotated.

Rotation
: The Euler rotation to rotate the instances by.

Pivot Point
: The position around which each instance is rotated. If the *Local Space* input is true, the location is relative to the initial transform of the instance.

Local Space
: If enabled, the instances are rotated in local space. In other words, they are rotated around the axes described by the initial transform of each instance. When the input is disabled, the pivot point and rotation are specified in the local space of the modifier object.

Properties

This node has no properties.

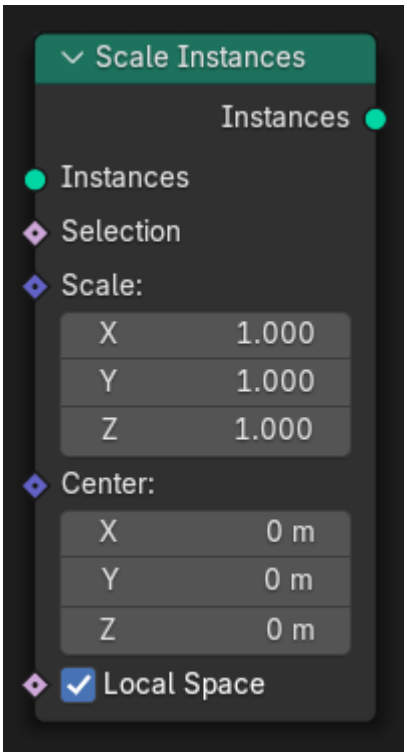
Outputs

Instances
: Standard geometry output.

Scale Instances Node

The *Scale Instances* node scales geometry instances in local or global space.

The Instances page contains more information about geometry instances.



Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be scaled.

Scale
: The scale factor to apply to the instance's transform on each axis.

Center
: The position from which the instance origins are scaled. Each instance will move away from this location. When the *Local Space* input is enabled, this location is relative to the initial transform of each instance.

Local Space
: If enabled, the instances are scaled in local space. In other words, they are scaled in the directions the described by the initial transform of each instance. When the input is disabled, the *Center* and *Scale* inputs are specified in the local space of the modifier object.

Properties

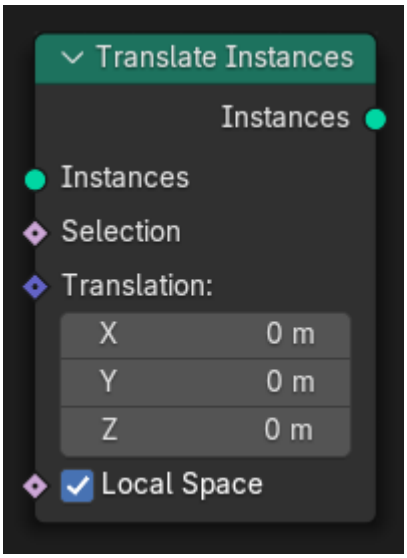
This node has no properties.

Outputs

Instances
: Standard geometry output.

Translate Instances Node

The *Translate Instances* node moves top-level geometry instances in local or global space.



The Instances page contains more information about geometry instances.

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be translated.

Translation
: The vector to translate the instances by.

Local Space
: If enabled, the instances are translated relative to their initial rotation. Otherwise they are translated in the local space of the modifier object.

Properties

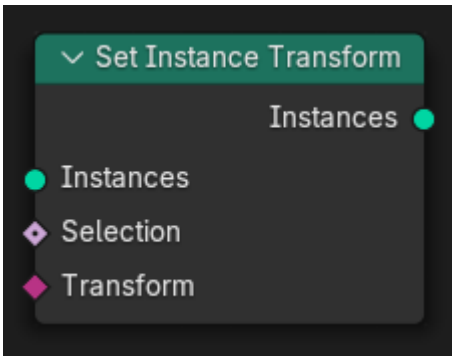
This node has no properties.

Outputs

Instances
: Standard geometry output.

Set Instance Transform Node

The *Set Instance Transform* node Transforms geometry instances using a Transformation Matrix.



The Instances page contains more information about geometry instances.

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be rotated.

Transform
: The transformation matrix to translate, rotate, and scale individual instances.

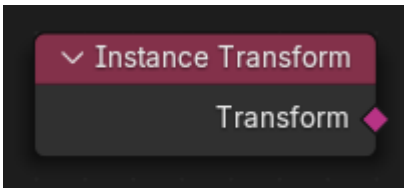
Properties

This node has no properties.

Outputs

Instances
: Standard geometry output.

Instance Transform Node



The *Instance Transform* outputs the Transformation Matrix of each top-level instance in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Inputs

This node has no inputs.

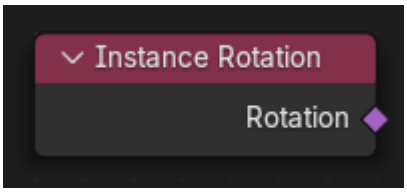
Properties

This node has no properties.

Outputs

Transformation
: Matrix that indicates the Transformation of each top-level instance.

Instance Rotation Node



The *Instance Rotation* outputs the XYZ Euler rotation of each top-level instance in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Note

Though rotations are often displayed in units of degrees in the spreadsheet or node editor, they are stored internally in radians, so this node outputs radians.

Inputs

This node has no inputs.

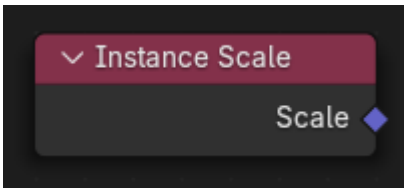
Properties

This node has no properties.

Outputs

Rotation
: Vector that indicates the rotation of each top-level instance in radians.

Instance Scale Node



The *Instance Scale* outputs the size of top-level instances on each axis in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Scale
: Vector that indicates the scale of each top-level instance.

Mesh Nodes

Nodes that only operate on meshes.

- Read
  - Edge Angle Node
  - Edge Neighbors Node
  - Edge Vertices Node
  - Edges to Face Groups Node
  - Face Area Node
  - Face Group Boundaries Node
  - Face Neighbors Node
  - Face Set Node
  - Is Face Planar Node
  - Is Edge Smooth Node
  - Is Face Smooth Node
  - Mesh Island Node
  - Shortest Edge Paths Node
  - Vertex Neighbors Node
- Sample
  - Sample Nearest Surface Node
  - Sample UV Surface Node
- Write
  - Set Face Set Node
  - Set Shade Smooth Node

- Operations
  - Dual Mesh Node
  - Edge Paths to Curves Node
  - Edge Paths to Selection Node
  - Extrude Mesh Node
  - Flip Faces Node
  - Mesh Boolean Node
  - Mesh to Curve Node

    - Mesh to Points Node
    - Mesh to Volume Node
    - Scale Elements Node
    - Split Edges Node
    - Subdivide Mesh Node
    - Subdivision Surface Node
    - Triangulate Node
- Primitives
    - Cone Node
    - Cube Node
    - Cylinder Node
    - Grid Node
    - Icosphere Node
    - Mesh Circle Node
    - Mesh Line Node
    - UV Sphere Node
- Topology
    - Corners of Edge Node
    - Corners of Face Node
    - Corners of Vertex Node
    - Edges of Corner Node
    - Edges of Vertex Node
    - Face of Corner Node
    - Offset Corner in Face Node
    - Vertex of Corner Node
- UV
    - Pack UV Islands Node
    - UV Unwrap Node

Read Mesh Nodes

- Edge Angle Node
- Edge Neighbors Node
- Edge Vertices Node
- Edges to Face Groups Node
- Face Area Node
- Face Group Boundaries Node
- Face Neighbors Node
- Face Set Node
- Is Face Planar Node
- Is Edge Smooth Node
- Is Face Smooth Node
- Mesh Island Node
- Shortest Edge Paths Node
- Vertex Neighbors Node

Edge Angle Node

The *Edge Angle* node calculates the angle in radians between two faces that meet at an edge. For the Face, Face Corner, and Point domains, the node uses simple domain interpolation to move values from the mesh's edges.



Note

The output of this node depends on the density of the mesh. If there are more edges closer together and the curvature of the mesh stays the same, the edge angle will be different

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Unsigned Angle
: The shortest angle in radians between two faces where they meet at an edge. The range of the data is from zero to PI. Flat edges and Non-manifold edges have an angle of zero. An edge between two faces completely folded back on each other has an angle of PI, or 180 degrees.

Tip

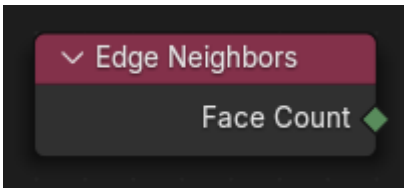
Computing this value is slightly faster than the signed angle, so if there is no need to distinguish between convex and concave angles, using this value can provide a performance improvement.

Signed Angle

The signed angle in radians between two faces where they meet at an edge. Flat edges and Non-manifold edges have an angle of zero. Concave angles are positive and convex angles are negative.

Edge Neighbors Node

The *Edge Neighbors* node outputs topology information relating to each edge of a mesh.



Inputs

This node has no inputs.

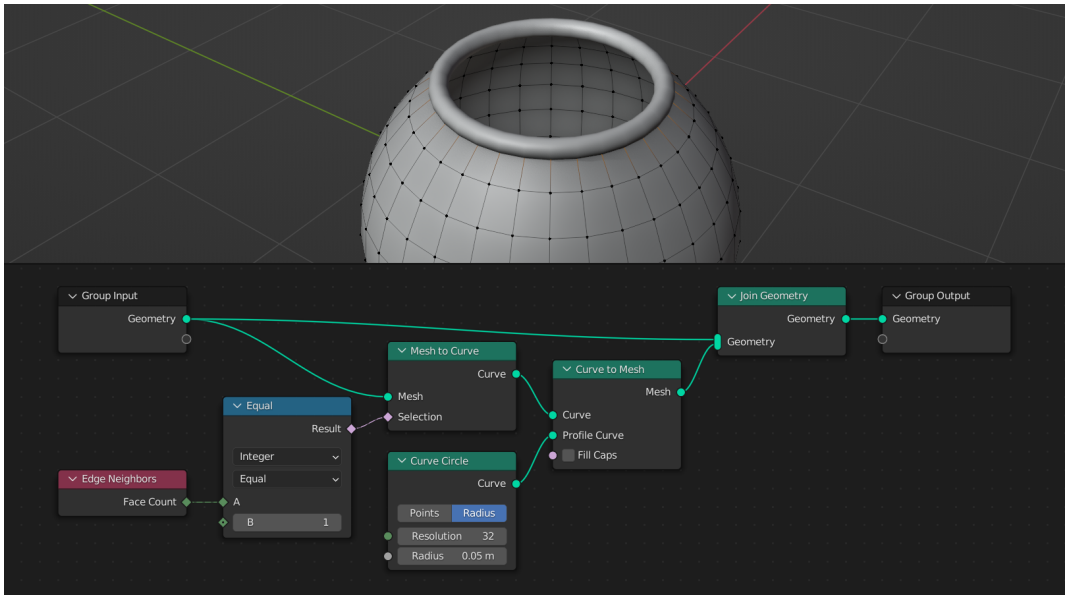
Properties

This node has no properties.

Outputs

Face Count
: The number of faces that use the edge as one of their sides. When the value is one, the edge is a non-manifold boundary edge. Alternatively, when the value is zero, the edge is a loose edge, not used by any faces.

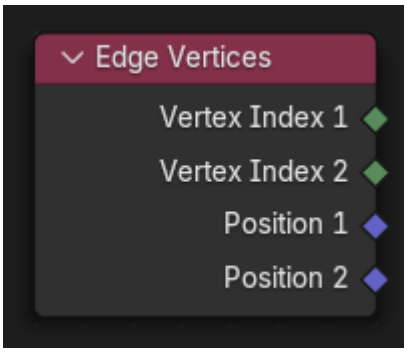
Examples



Using the *Face Count* output to create a curve on a mesh's boundary edges.

Edge Vertices Node

The *Edge Vertices* node outputs the position and index of the two vertices of each of a mesh's edges.



Note

The order of the two vertices of an edge is arbitrary. In some cases it may be predictable based on the internals of the algorithm that created the mesh, but in general the order should not be relied upon.

Inputs

This node has no inputs.

Properties

This node has no properties.

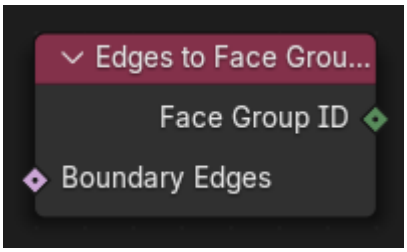
Outputs

Vertex Index 1/2
: The index of the two vertices of the edge.

Position 1/2
: The position of each of the edge's vertices. This output is for convenience, it is the same as using index output to retrieve the position from the Evaluate at Index Node.

Edges to Face Groups Node

The *Edges to Face Groups* node group faces into regions surrounded by the selected boundary edges.



Inputs

Boundary Edges
: Edges used to split faces into separate groups.

Properties

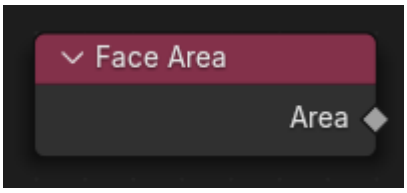
This node has no properties.

Outputs

Face Group ID
: Index of the face group inside each boundary edge region.

Face Area Node

The *Face Area* node outputs the surface area of a mesh's faces. The units are in Blender units no matter the unit system, equivalent to meters-squared at the default unit scale.



Note

For quads and N-gons, when the face's vertices are not planar, the output is not necessarily the same as the sum of every one of the face's triangles visible in the viewport. In this case it should only be used an approximation. In some cases, the Triangulate Node can be used to get an exact value.

Inputs

This node has no inputs.

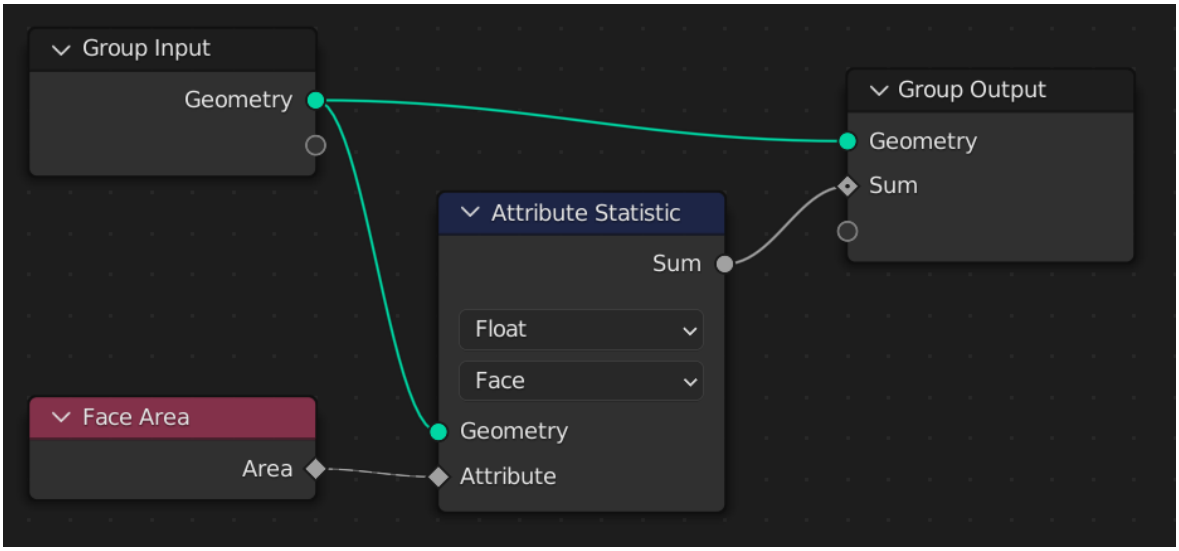
Properties

This node has no properties.

Outputs

Area
: The surface area of each of the mesh's faces.

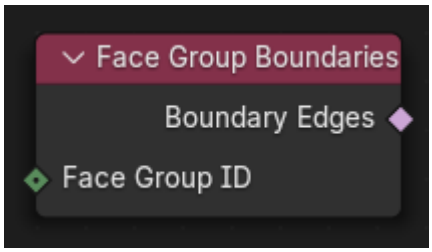
Examples



Combined with the Attribute Statistic Node, this node can be used to calculate the total surface area of a mesh.

Face Group Boundaries Node

The *Face Group Boundaries Node* finds the edges which lie on the boundaries of specified regions. These edges could be used to mark seams for UV unwrapping, for example.



Inputs

Face Group ID
: Identifier for which group of faces this face belongs to. All contiguous faces with the same value are in the same region.

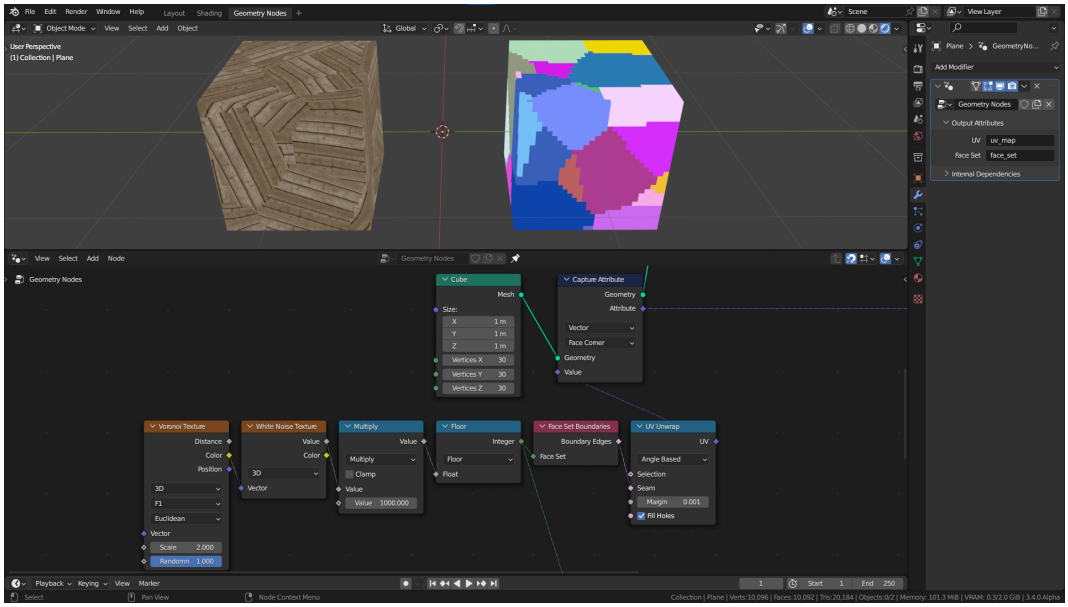
Properties

This node has no properties.

Output

Boundary Edges
: Selection of the boundary edges of the different face sets. An edge is considered to be at the boundary if it lies on at least two faces with different identifiers.

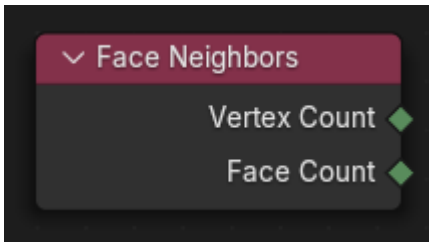
Examples



Combined with the [UV Unwrap Node], this node is used to turn the face sets (right cube) into a UV map for a texture (left cube).

Face Neighbors Node

The *Face Neighbors* node outputs topology information relating to each face of a mesh.



Inputs

This node has no inputs.

Properties

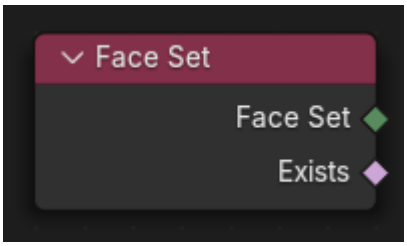
This node has no properties.

Outputs

Vertex Count
: This output is simply the number of sides of each face, or how many corners each face has.

Neighboring Face Count
: The number of faces that connect to this face with at least one edge. On a regular manifold mesh with only quads and triangles, this will be the same as the vertex count, otherwise it might be completely different.

Face Set Node



The *Face Set Node* outputs which face set a face is in, and whether or not face sets exist in the mesh at all.

The corresponding data flow node is the Set Face Set Node.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

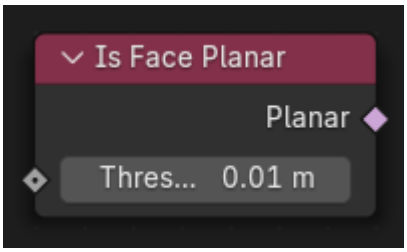
This node has no properties.

Output

Face Set
: Integer indicating which face set a face is in, or 0 when the mesh does not have face sets. When evaluated in the edge or point domain, outputs an interpolated value based on the connected faces.

Exists
: Boolean value that indicates whether the element's mesh has face sets.

Is Face Planar Node



The *Is Face Planar* node outputs whether every triangle of a quads or N-gons is on the same plane as all of the others, in other words, if they have the same normal.

For example, a non-planar face can be created by moving a single vertex in a face but not the others. Triangles will always be planar.

Inputs

Threshold
: The distance a point can be from the surface before the face is no longer considered planar.

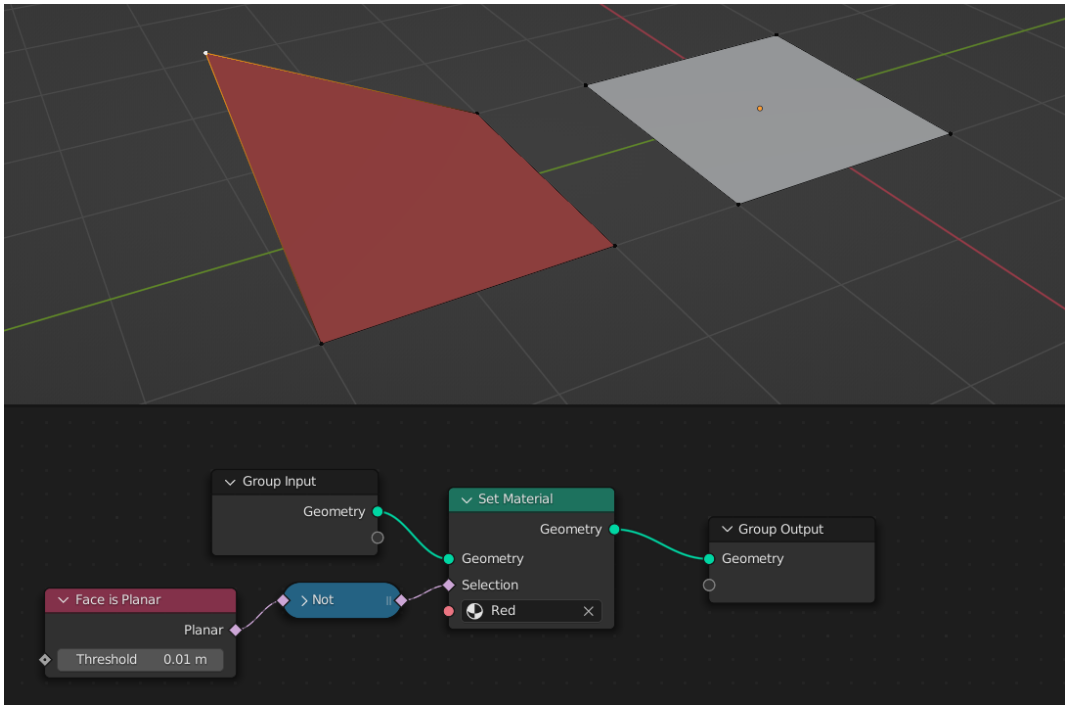
Properties

This node has no properties.

Outputs

Planar
: Whether each mesh face is planar.

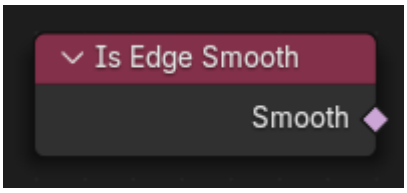
Examples



Combined with the Set Material Node, this node is used to visualize all non-planar faces in a mesh.

Is Edge Smooth Node

The *Is Edge Smooth* node outputs true for each edge of the mesh that is *not* marked as sharp. Otherwise, if the edge *is* marked as sharp, then the node outputs false.



See also

Mark Sharp & Clear Sharp

Inputs

This node has no inputs.

Properties

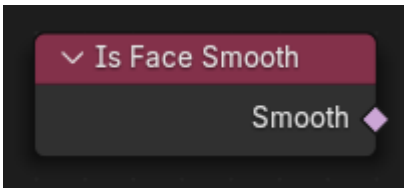
This node has no properties.

Outputs

Smooth
: Boolean value that indicates whether the edges of the mesh are *not* marked as sharp.

Is Face Smooth Node

The *Is Face Smooth* node outputs true for each face of the mesh if that face is marked to render smooth shaded. Otherwise, if the face is marked to render as flat shaded, then the node outputs false.



Inputs

This node has no inputs.

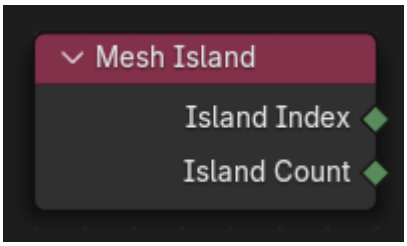
Properties

This node has no properties.

Outputs

Smooth
: Boolean value that indicates whether the normals of each face corner on the final mesh are smoothed with normal of all adjacent faces or not.

Mesh Island Node



The *Mesh Island* node outputs information about separate connected regions, or "islands" of a mesh. Whenever two vertices are connected together by an edge, they are considered as part of the same island, and will have the same *Island Index* output.

This node's behavior is similar to the Select Linked operator in edit mode, or the *Random per Island* output of the Geometry shader node.

Inputs

This node has no inputs.

Properties

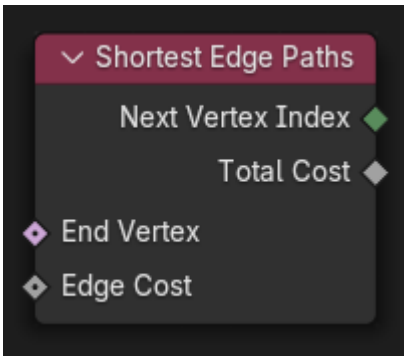
This node has no properties.

Outputs

Island Index
: The index of each vertex's island. Indices are decided based on the lowest vertex index contained in each island.

Island Count
: The total number of mesh islands. This is a single value, and does not vary per element.

Shortest Edge Paths Node

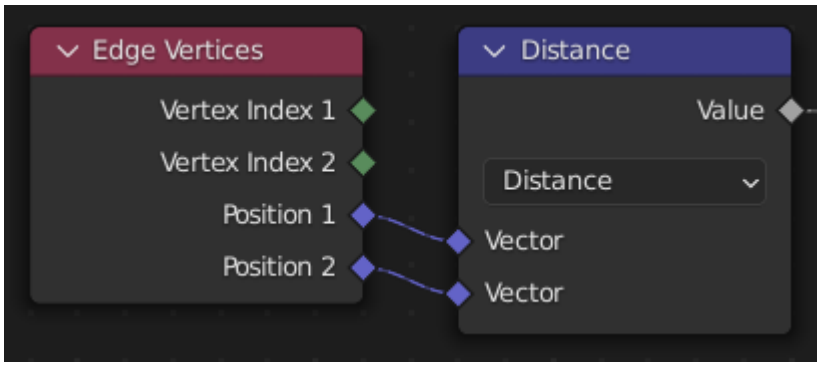


The *Shortest Edge Paths* node finds paths along mesh edges to a selection of end vertices. The cost used to define "shortest" can be set to anything. By default there is a constant cost for every edge, but a typical input would be the length of each edge.

The output is encoded with vertex indices, and is meant to be used on the vertex domain. For each vertex, the *Next Vertex Index* output gives the index of the following vertex in the path to the "closest" endpoint.

The node is implemented with Dijkstra's algorithm.

Tip



The edge length is a natural input to the *Edge Cost*. It can be implemented with the Edge Vertices Node and the Vector Math Node set to the *Distance* operation.

See also

This node can be used with the Edge Paths to Selection Node or the Edge Paths to Curves Node to generate new geometry based on the paths.

Inputs

End Vertex
: A selection of the goal vertices that terminate the edge paths.

Edge Cost
: The weight for each edge, used to determine the meaning of “shortest.”

Properties

This node has no properties.

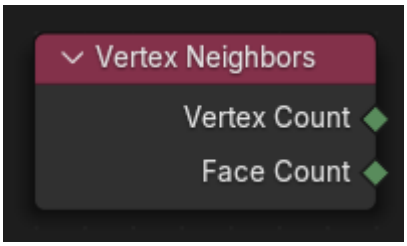
Outputs

Next Vertex Index
: The following vertex on the shortest path from every vertex to the closest endpoint (as defined by the cost input).

Total Cost
: The remaining cost before an end vertex is reached.

Vertex Neighbors Node

The *Vertex Neighbors* node outputs topology information relating to each vertex of a mesh.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

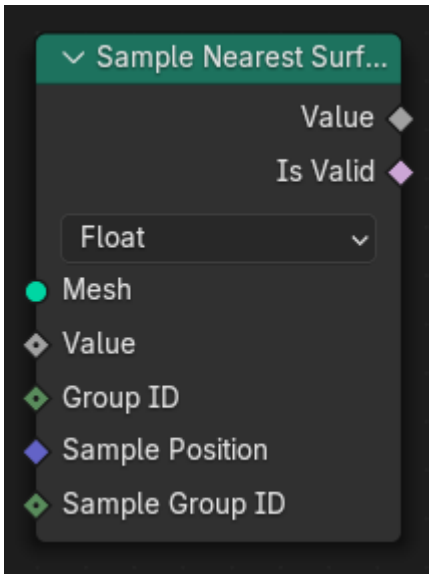
Vertex Count
: The number of vertices connected to this vertex with an edge, equal to the number of connected edges.

Face Count
: The number of faces that contain the vertex.

Mesh Sample Nodes

- Sample Nearest Surface Node
- Sample UV Surface Node

Sample Nearest Surface Node



The *Sample Nearest Surface* node finds values at the closest points on the surface of a source mesh geometry. Non-face attributes are interpolated across the surface.

This node is similar to the Geometry Proximity Node, but it gives the value of any attribute at the closest surface point, not just its position.

Warning

Because the node samples the *surface* of a mesh rather than its edges or vertices, values from loose points and edges are ignored.

Inputs

Mesh
: The geometry to retrieve the attribute from.

Value

A field to evaluate on the *Source* geometry for use with the transfer method.

Group ID
: Is evaluated on the face domain and splits the input mesh into multiple parts, each with its own id.

Sample Position
: The position to start from when finding the closest location on the target mesh. By default, this is the same as if the Position Node was connected.

Sample Group ID
: Determines in which group the closest nearest surface is detected.

Properties

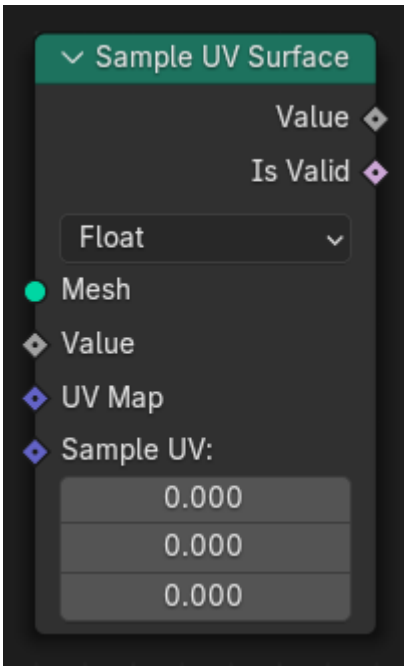
Data Type
: The data type to use for the retrieved values.

Outputs

Value
: The data retrieved and interpolated from the *Source* geometry, mapped based on the node's settings and inputs.

Is Valid
: Whether the sampling was successful. It can fail when the sampled group is empty.

Sample UV Surface Node



The *Sample UV Surface* node finds values on a mesh's surface at specific UV locations. Internally the process is a "reverse UV lookup" from a location in 2D space. The node then finds the face that corresponds to each UV coordinate, and the location within that face.

Warning

Because of the node's method of computation, the UV map should not have any overlapping faces. If the UV map is sampled at a location with no faces or overlapping faces, the node will output the default value for the data type, which is zeros for most types.

Inputs

Mesh

A geometry containing the mesh with a UV map for sampling.

Value
: A field to evaluate on the target *Mesh* geometry for later sampling at the surface positions.

UV Map
: The mesh UV map to sample, evaluated on the *Mesh* input. Should not have overlapping faces.

Sample UV
: The coordinates to sample within the UV map.

Properties

Data Type
: The data type to use for the retrieved values.

Outputs

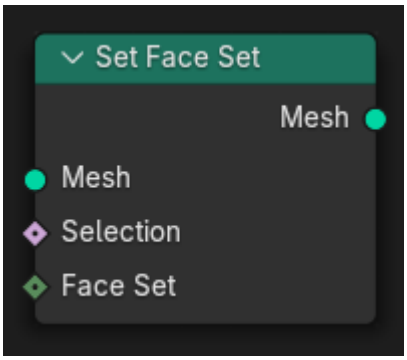
Value
: The data retrieved and interpolated from the *Mesh* geometry, mapped based on the node's settings and inputs.

Is Valid
: Whether the node could find a single face to sample at the UV coordinate.

Write Mesh Nodes

- Set Face Set Node
- Set Shade Smooth Node

Set Face Set Node



The *Set Face Set* node controls which face set that faces are in.

The input node for this data is the Face Set Node.

Note

This node can only be used in the Tool context.

Inputs

Mesh
: Standard geometry input.

Selection
: Boolean field that controls which faces will have the Face Set value applied.

Face Set
: Integer field for specifying which face set each selected face should be moved to. Ignored for faces where the value of Selection is false.

Properties

This node has no properties.

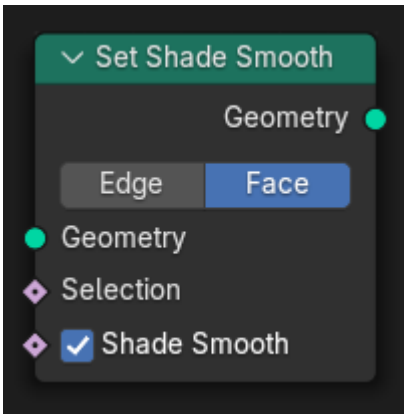
Outputs

Mesh

Standard geometry output.

Set Shade Smooth Node

The *Set Shade Smooth* node controls whether the mesh's faces look smooth in the viewport and renders. The smooth status of both edges and faces can be controlled, corresponding to the `sharp_edge` and `sharp_face` attributes. The input node for this data is the Is Face Smooth Node.



Inputs

Mesh
: Standard geometry input.

Shade Smooth
: When true, the selected faces will be marked to render smooth shaded. Otherwise the faces will be rendered flat shaded.

Selection
: Boolean input for selecting which faces will have the Shade Smooth value applied.

Properties

Domain
: Whether to write smoothness of mesh faces or edges.

Outputs

Mesh

Standard geometry output.

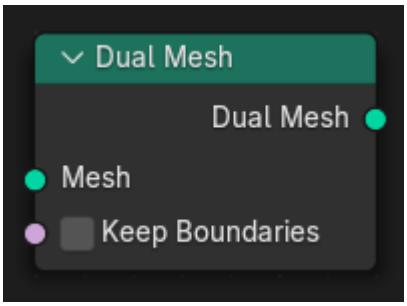
Mesh Operation Nodes

Nodes that only operate on meshes.

- Dual Mesh Node
- Edge Paths to Curves Node
- Edge Paths to Selection Node
- Extrude Mesh Node
- Flip Faces Node
- Mesh Boolean Node
- Mesh to Curve Node
- Mesh to Points Node
- Mesh to Volume Node
- Scale Elements Node
- Split Edges Node
- Subdivide Mesh Node
- Subdivision Surface Node
- Triangulate Node

Dual Mesh Node

The *Dual Mesh Node* converts a mesh into its dual, i.e. faces are turned into vertices and vertices are turned into faces. This also means that attributes which were on the face domain are transferred to the point domain in the dual mesh.



Warning

The Dual Mesh node only works on manifold geometry. To work with non-manifold geometry it's best to remesh the geometry first.

Inputs

Mesh
: Standard geometry input.

Keep Boundaries
: Keeps the non-manifold boundaries of the input mesh in place, by creating extra geometry, and avoiding the dual mesh transformation there.

Properties

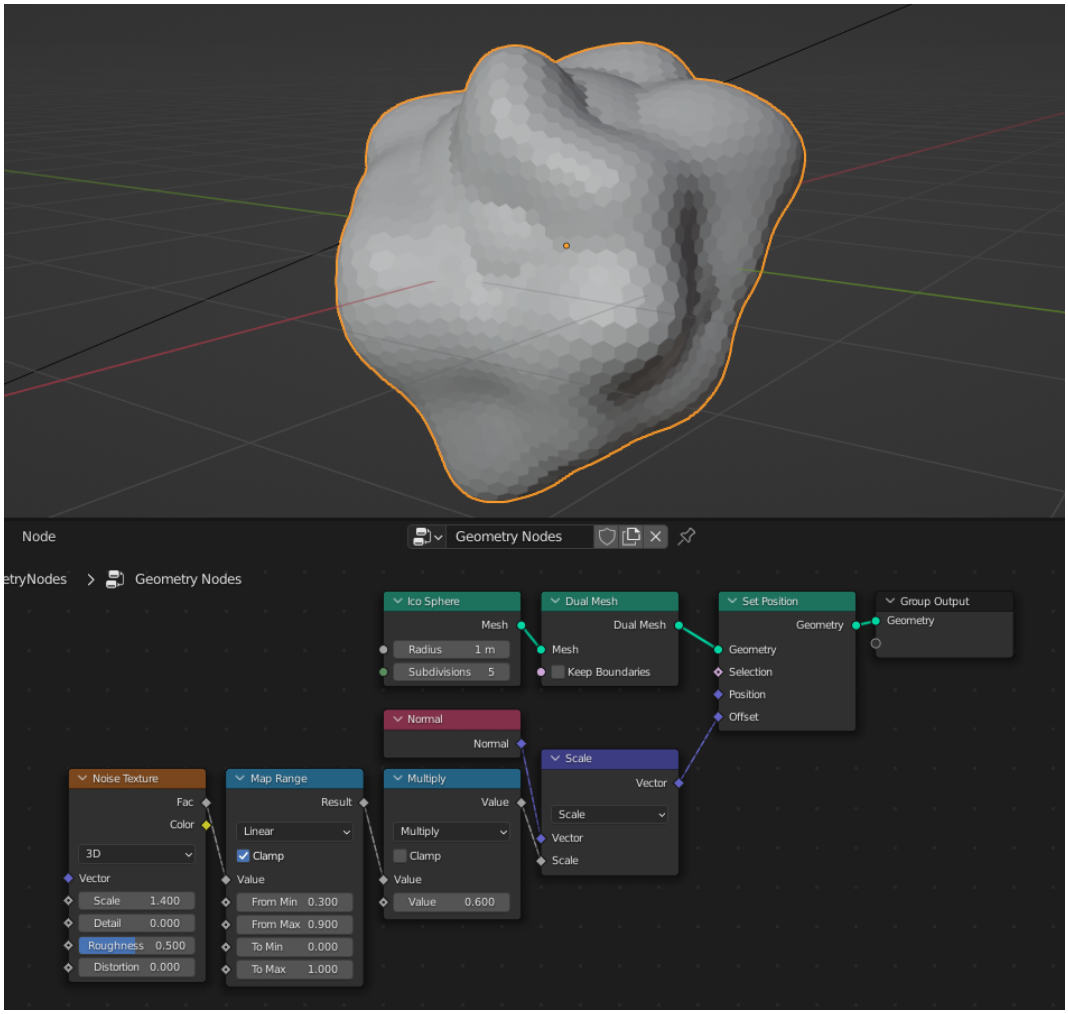
This node has no properties.

Output

Dual Mesh
: Standard geometry output.

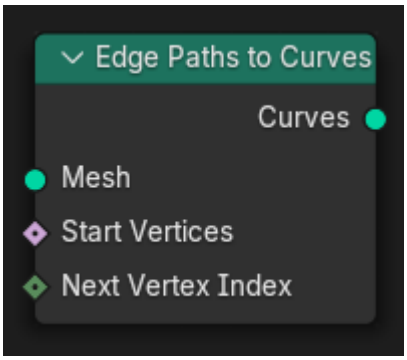
Examples

The *Dual Mesh Node* combines nicely with triangulated meshes. In this case an Ico Sphere is used, which is made up of nice and evenly spaced triangles.



Edge Paths to Curves Node

The *Edge Paths to Curves* node output curves that follow paths across mesh edges.



See also

This node is meant to use the output of the Shortest Edge Paths Node. It is similar to the Edge Paths to Selection Node, but it creates a curve that follow each path, rather than a selection of every visited edge.

Inputs

Mesh
: Standard mesh input.

Start Vertices
: A selection of the vertices to start at when traveling along the next vertex indices.

Next Vertex Index
: Describes the path to follow at every vertex.

Properties

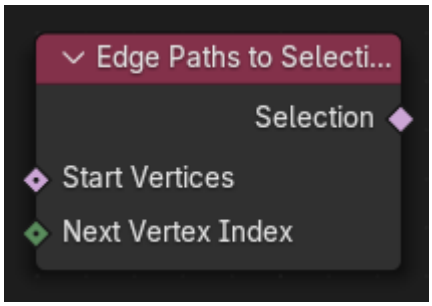
This node has no properties.

Outputs

Mesh

Standard curves output.

Edge Paths to Selection Node



The *Edge Paths to Selection* node follows paths across mesh edges and outputs a selection of every visited edge.

See also

This node is meant to use the output of the Shortest Edge Paths Node. It can be combined with the Separate Geometry Node to remove any unused edges.

Inputs

Start Vertices
: A selection of the vertices to start at when traveling along the next vertex indices.

Next Vertex Index
: Describes the path to follow at every vertex.

Properties

This node has no properties.

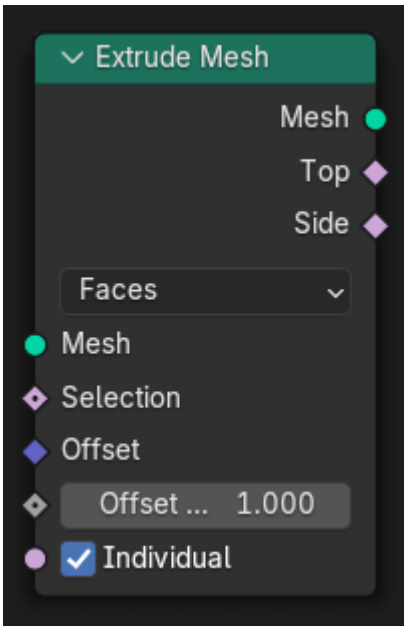
Outputs

Selection
: A boolean field indicating all edges visited when traversing the mesh.

Extrude Mesh Node

The *Extrude Mesh Node* generates new edges or faces on the selected geometry elements and moves them by a certain offset.

The operations are similar to the extrude tools in mesh edit mode, though there are some differences. Most importantly, the node never keeps the back-faces of the extrusion in place, they are always removed. Attribute propagation rules may also be different.



Inputs

Mesh
: Standard geometry input.

Selection
: A boolean field indicating which elements should be extruded.

Offset
: The translation vector for each extruded element. By default, this is the element's normal.

> **Tip**
>
> If all the elements are extruded in the same direction, you may be able to improve performance by connecting a Vector Node to this input, thereby skipping the normal calculation.

Offset Scale

Scaling factor for the above translation vector.

Individual *Face Mode Only*

Whether to extrude each face individually rather than extruding connected groups of faces together.

Properties

Mode

Vertices:: Attaches a new freestanding edge to each selected vertex.

Edges:: Attaches a new quad face to each selected edges. Vertices shared by the original selected edges are also shared in the duplicated edges.

> **Note**
> Depending on the situation, the normals of the new faces may be arbitrary. If the selected edges each have only one connected face, then the node can pick a consistent orientation for the new faces, but if there is more than one connected face, or no connected faces, the normals may have to be adjusted afterwards.

Faces:: Extrudes contiguous regions of selected faces, or each selected face individually, depending on the *Individual* boolean input.
When the *Individual* input is false, the node will find regions of connected faces and generate new "side" faces on the boundaries of those regions. Any vertices, edges or faces on the *inside* of the regions simply are moved, not duplicated. If the whole mesh is selected and it is already a Manifold shape, then the result will just be that the whole mesh gets resized.

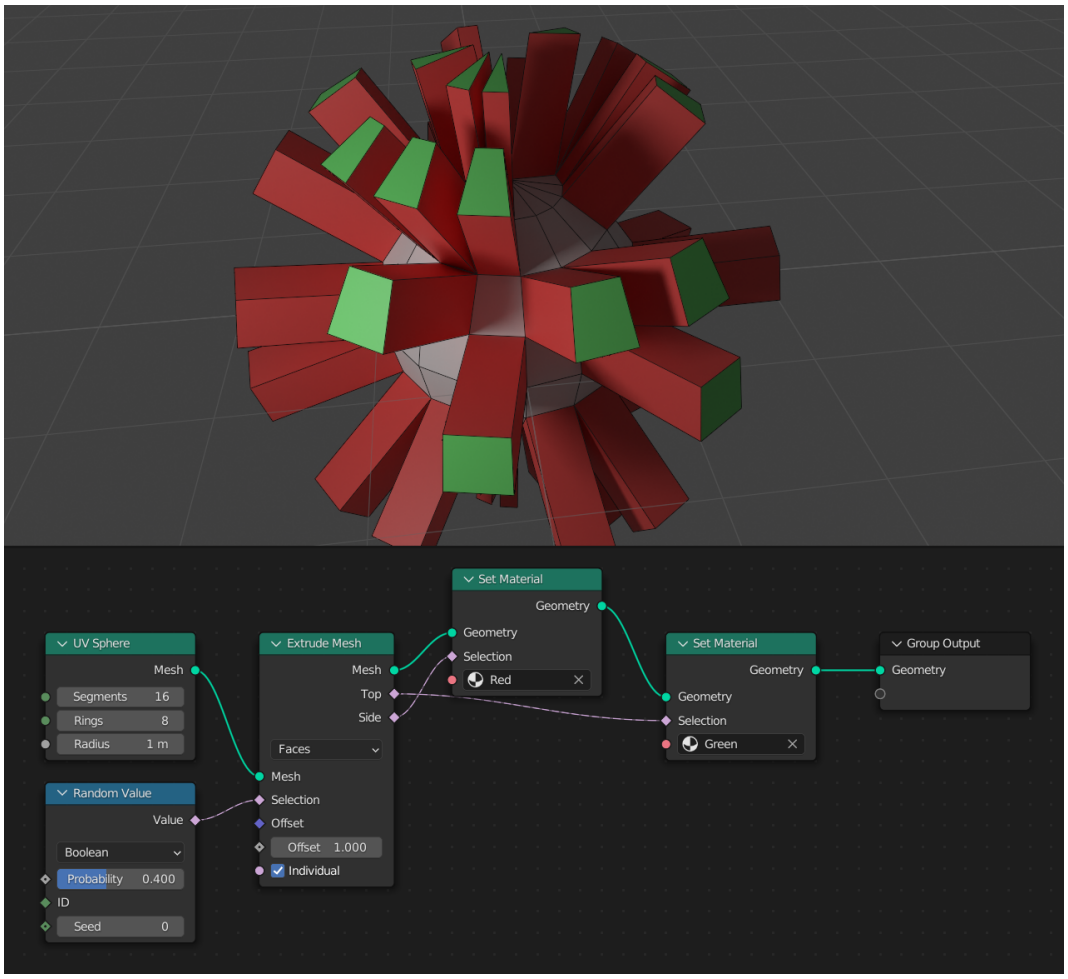
Output

Mesh
: Standard geometry output.

Top
: A boolean field indicating the “top” elements in the extrusion. In *Vertex* mode, these are the new vertices; in *Edge* mode, the new edges; and in *Face* mode, the moved faces.

Side
: A boolean field indicating the “side” elements in the extrusion. In *Vertex* mode, these are the new edges; in *Edge* mode, the new faces; and in *Face* mode, too, the newly generated faces (as opposed to the moved ones).

Examples



Here, the selection outputs are used to set materials on certain faces of the mesh. A Random Value Node is used to limit the extrusion to a random set of faces.

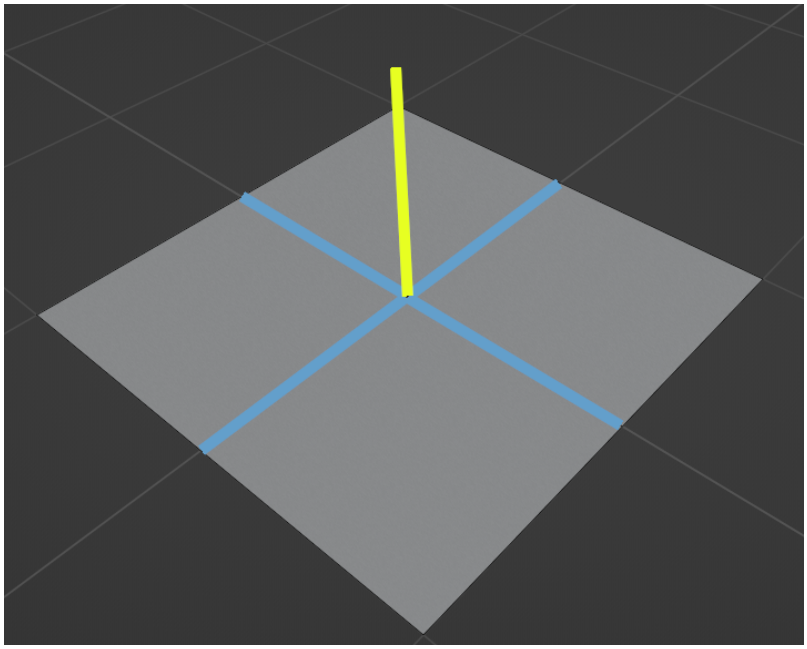
Attribute Propagation

Attributes are transferred to the new elements with specific rules. An attribute will never change domains on the resulting mesh. The `id` attribute does not have any special handling.

Generally boolean attributes are propagated with “or”, meaning any connected “true” value that is mixed in for other types will cause the new value to be “true” as well.

The following sections describe:

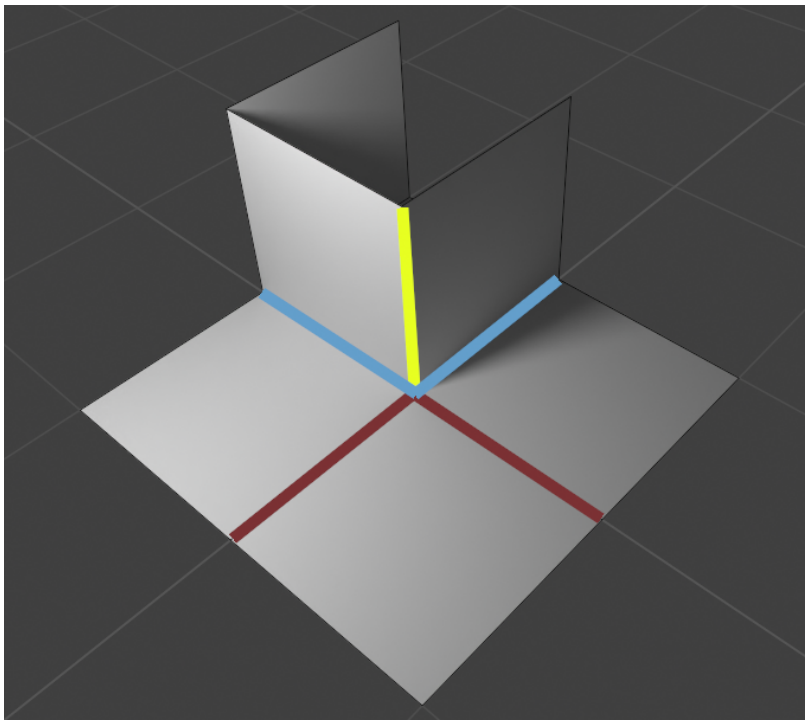
Vertex Mode



The new edges created in vertex mode use the average value of all connected edges.

- New **vertices** have copied values from their original vertices.
- New **edges** have the average value of any connected original edges. For boolean attributes, edges are selected if any connected edges were selected.

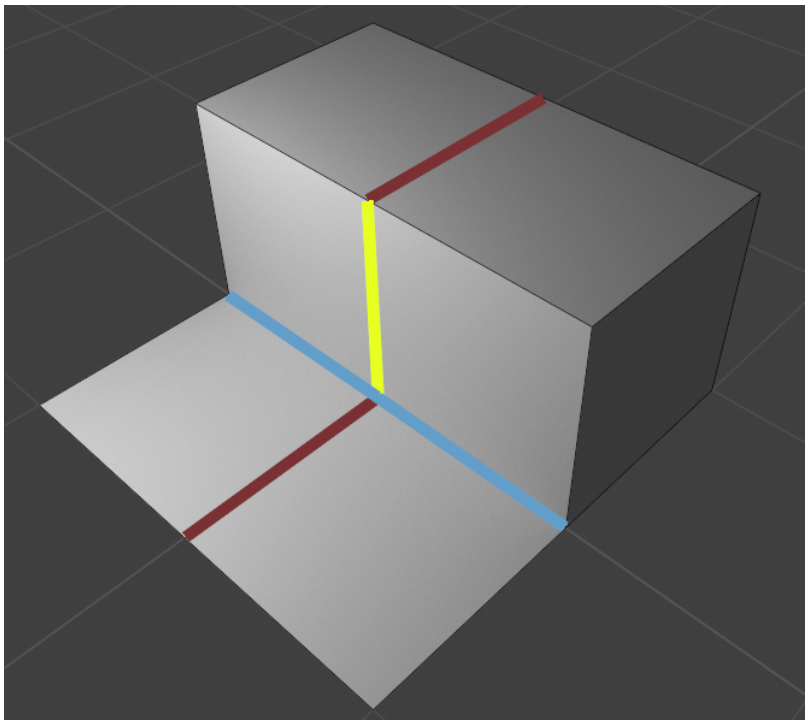
Edge Mode



Attribute propagation for new connecting edges (the vertical yellow edge). The final value is a mix of the values from the two middle blue edges. The darker maroon edges lower on the image are not used.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value from any connected extruded edges. For booleans, the edges are selected if any connected extruded edges were selected. (*Propagation rules are shown in the figure above*.)
- Horizontal duplicate **edges** have copied values from their original edges.
- New **faces** get the average values of all faces connected to the selected edge. For booleans, faces are selected if any connected original faces were selected.
- New **face corners** get the averaged value of corresponding corners in all faces connected to selected edges. For booleans, corners are selected if one of those corners are selected.

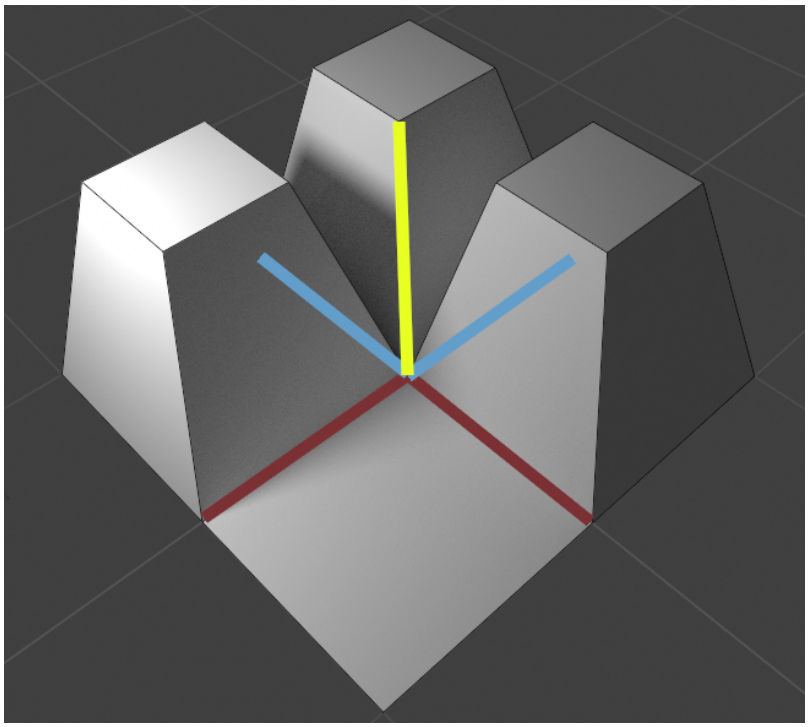
Face Mode



Attribute propagation for new connecting edges (the vertical yellow edge). The final value is a mix of the values from the two middle blue edges. The values from the darker maroon edges between unselected faces and on top of the extruded region are not used.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value from any connected extruded edges, not including the edges "on top" of extruded regions. For booleans, the edges are selected if any of those connected edges were selected. (*Propagation rules are shown in the figure above*.)
- Horizontal duplicate **edges** have copied values from their original edges.
- New **faces** have copied values from the corresponding extruded faces.
- New **face corners** have copied values from the corresponding corresponding corners of extruded faces.

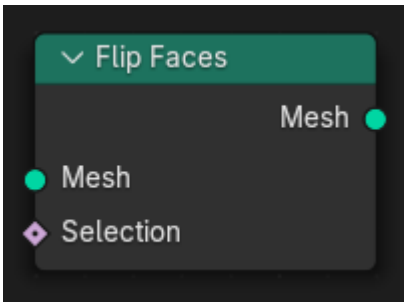
Individual Face Mode



Attribute propagation for new connecting edge. Each edge uses the average values of the two neighboring edges on its extruded face.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value of the two neighboring edges on each extruded face. For booleans, the edges are selected when at least one neighbor on the extruded face was selected.
- Horizontal duplicate **edges** have copied values from their original edges.
- New side **faces** have copied values from their corresponding selected face.
- New **face corners** have copied values from the corresponding corners of selected faces.

Flip Faces Node



The *Flip Faces Node* reverses the order of the vertices and edges of each selected face. The most common use of this node is to flip the normals of a face. Any face corner domain attributes of selected faces are also reversed.

Though this node is usually used to affect normals, it is not called “Flip Normals” for an important reason. The node does not actually interact with normals directly. Normals are defined by the right hande rule, so if a face’s vertex list is reversed, then its normal will point in the opposite direction.

Inputs

Mesh
: Standard geometry input.

Selection
: Whether to flip the direction of each face. True values mean the face will be flipped, false means the face will be unaffected.

Properties

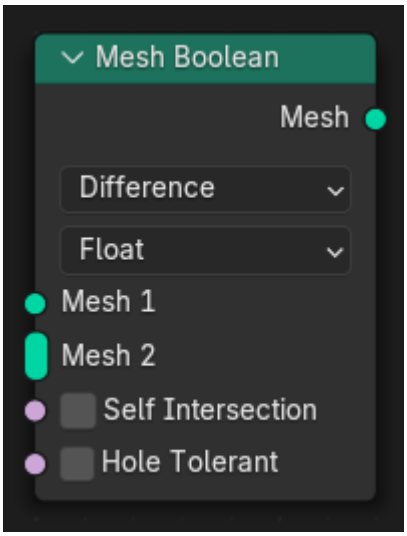
This node has no properties.

Output

Mesh
: Standard geometry output.

Mesh Boolean Node

The *Mesh Boolean Node* allows you to cut, subtract, and join the geometry of two inputs. This node offers the same operations as the Boolean modifier.



Inputs

Mesh 1/2
: Standard geometry input.

Self Intersection
: Correctly calculates cases when one or both operands have self-intersections. This involves more calculations making the node slower.

Hole Tolerant
: Optimizes the Boolean output for Non-manifold geometry at the cost of increased computational time. Because of the performance impact, this option should only be enabled when the solver demonstrates errors with non-manifold geometry.

Properties

Operation

Intersect:: Produce a new geometry containing only the volume inside of both geometry 1 and geometry 2.

Union:: The two input meshes are joined, then any interior elements are removed.

Difference:: Geometry 2 is subtracted from geometry 1 (everything outside of geometry 2 is kept).

Solver

Algorithm used to calculate the Boolean intersections.

Float:: Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.

Exact:: Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Float Solver*.

Output

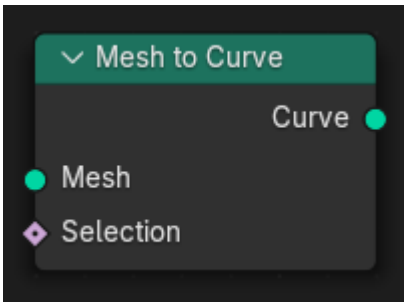
Mesh

Standard geometry output.

Intersecting Edges *Exact Solver*

A boolean attribute field with a selection of the edges that were created where the two inputs meet.

Mesh to Curve Node



The *Mesh to Curve* node turns each string of connected mesh edges into a poly spline. Whenever two or more strings cross each other, the splines will be split.

Loose vertices are ignored – they will not be turned into single-point splines.

Attributes, both named and unnamed ones, are transferred to the resulting splines. If there is a `radius` attribute, it will be applied as such, although you may find it more convenient to use the Set Curve Radius Node for this.

Inputs

Mesh
: Standard mesh input.

Selection
: A field input evaluated on the edge domain to determine whether each edge will be included in the result.

  Tip

  Using this input is more efficient than deleting parts of the geometry before or after the conversion.

Properties

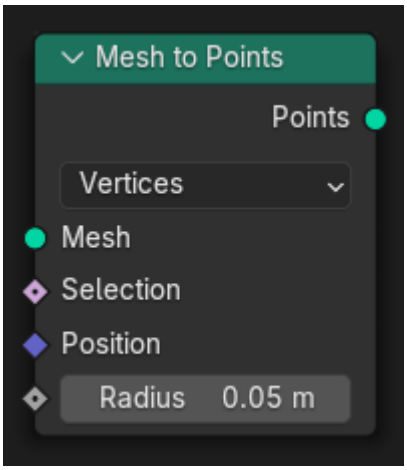
This node has no properties.

Outputs

Curve
: Generated curve.

Mesh to Points Node

The *Mesh to Points* node generates a point cloud from a mesh.



Inputs

Mesh
: Standard Mesh input.

Selection
: The meshes used to generate a point cloud.

Position
: Positions of generated points. By default, this input is the same as if the Position Node was connected.

Radius
: Radii of generated points.

Properties

Mode

Vertices:: Points are generated for each vertex.

Edges:: Points are generated for each edge, at the middle of each edge, by default.

Faces:: Points are generated for each face, at the average of all of each face's vertices, by default.

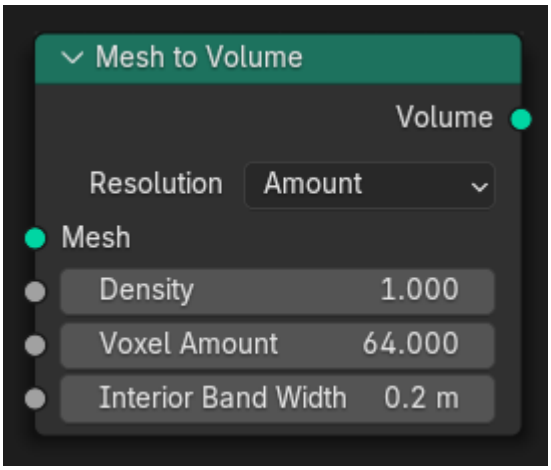
Corners:: Points are generated for each corner. The points are all placed at the location of each corners vertex, so they will overlap by default.

Outputs

Points
: Generated point cloud.

Mesh to Volume Node

The *Mesh to Volume* node creates a fog volumes based on the shape of a mesh. The volume is created with a grid of the name `"density"`.



Inputs

Mesh
: Standard Mesh input.

Density
: Value of voxels inside the generated fog volume.

Voxel Amount
: Specify the approximate number of voxels along the diagonal.

Voxel Size
: Specify the voxel side length.

Interior Band Width
: The maximum distance of the included voxels to the surface on the inside of the mesh.

Properties

Resolution

How the voxel size is specified.

Amount:: Specify the approximate number of voxels along the diagonal.

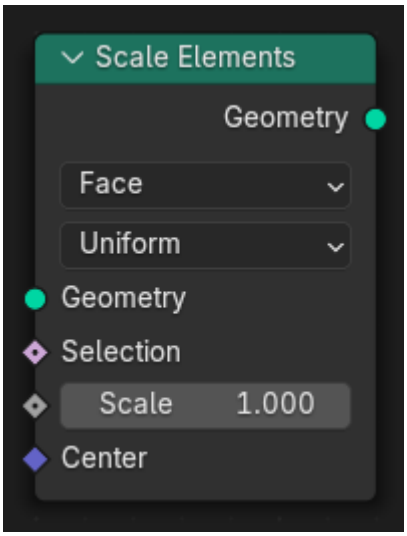
Size:: Specify the voxel side length. It is recommended to be careful when tweaking this value, because small changes can have a large effect on the processing time.

Outputs

Volume
: The generated volume grid.

Scale Elements Node

Scales the selected faces or edges, letting you specify a scaling factor and pivot point for each one. Connected faces/edges are scaled together using their average factor and pivot point.



Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field indicating which elements to scale.

Scale
: The scaling factor for each element.

Center
: The pivot point for each element.

Axis *Single Axis Mode Only*
: Axis along which to scale each element. This vector is normalized internally, so the length does not matter.

Properties

Domain
The element type to transform.

Face::	Scale faces.
Edge::	Scale edges.

Scale Mode

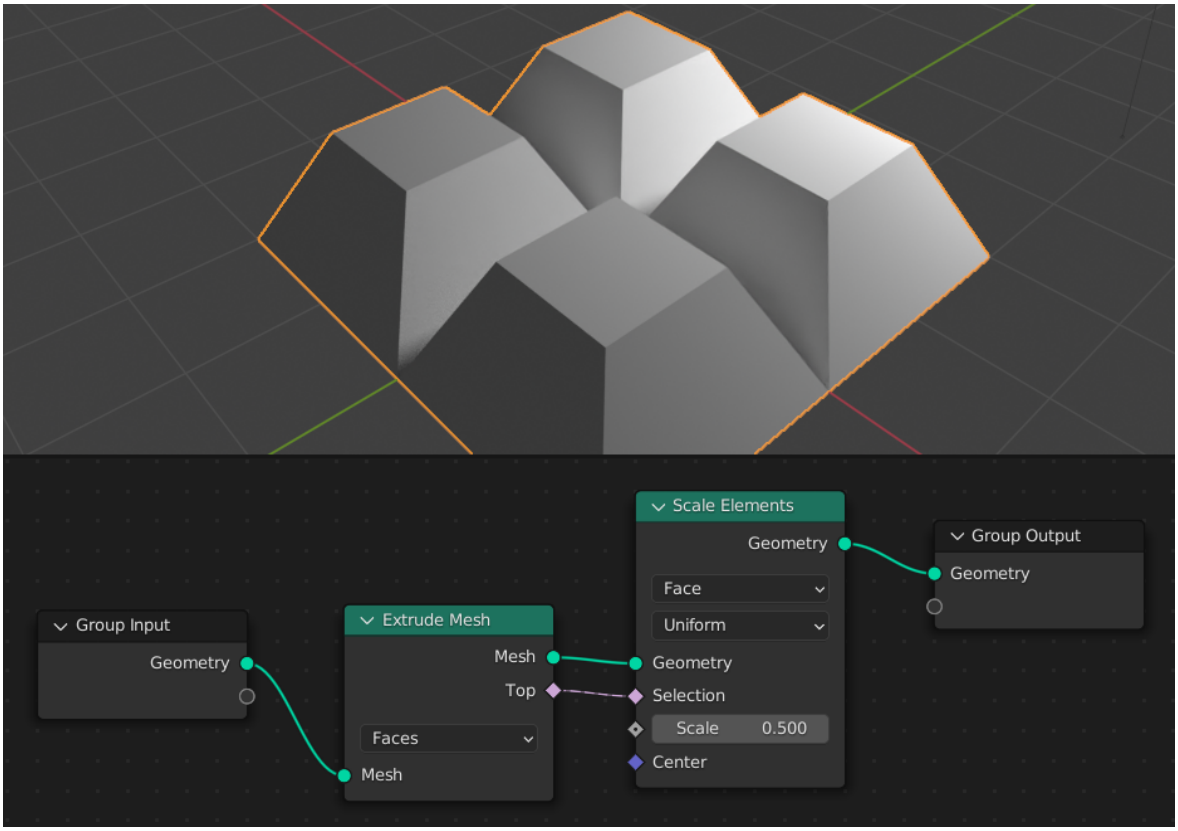
Uniform::	Scale elements by the same factor in every direction.
Single Axis::	Scale elements in a single direction defined by the *Axis* input.

Output

Geometry
Standard geometry output.

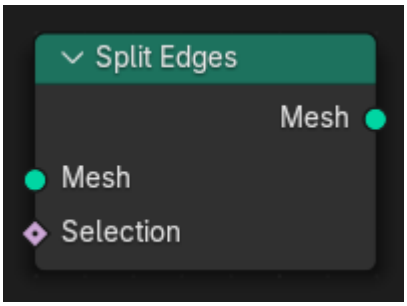
Examples

The node is useful when combined with the Extrude Mesh Node, especially in *Individual* mode where connected faces aren't extruded together.


Group Input
Geometry
Extrude Mesh
Mesh
Top
Faces
Mesh
Scale Elements
Geometry
Face
Uniform
Geometry
Selection
Scale 0.500
Center
Group Output
Geometry

Split Edges Node

Like the Edge Split Modifier, the *Split Edges* node splits and duplicates edges within a mesh, breaking 'links' between faces around those split edges.



Inputs

Mesh
: Standard geometry input.

Selection
: A standard Boolean selection input to determine which edges will be split.

Note

Because of mesh topology requirements, sometimes more or fewer edges than are selected will be split.

Properties

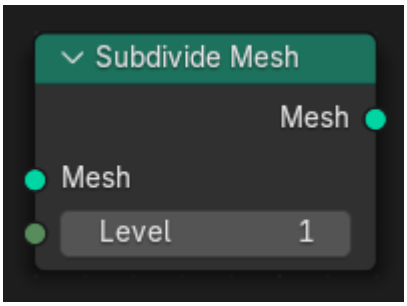
This node has no properties.

Outputs

Mesh
: Standard geometry output.

Subdivide Mesh Node

The *Subdivide Mesh* node adds new faces to mesh geometry using a simple interpolation for deformation.



Inputs

Mesh
: Standard geometry input.

Level
: The number of subdivisions to apply to the input geometry.

Properties

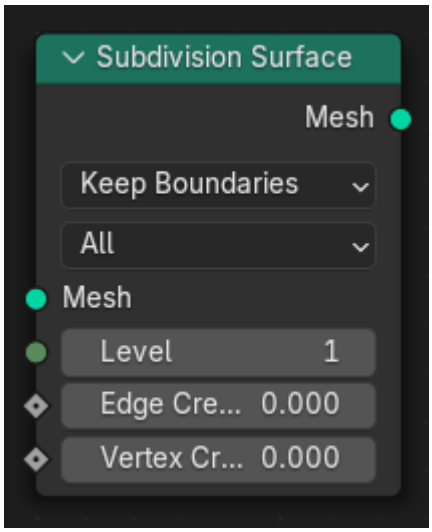
This node has no properties.

Outputs

Mesh
: Standard geometry output.

Subdivision Surface Node

The *Subdivision Surface* node adds new faces to mesh geometry using a Catmull-Clark subdivision method.



Inputs

Mesh
: Standard geometry input.

Level
: The number of subdivisions to apply to the input geometry.

Edge Crease
: Controls how smooth edges should be with Weighted Edge Creases.

Vertex Crease
: Controls how much the subdivision surface should be pulled towards the vertex. Similar to edge creases, but allows individual vertices to be creased.

Properties

UV Smooth

Controls how subdivision smoothing is applied to UVs.

None:: UVs remain unchanged.
Keep Corners:: UV islands are smoothed, but their boundary remain unchanged.
Keep Corners, Junctions:: UVs are smoothed, corners on discontinuous boundary and junctions of three or more regions are kept sharp.
Keep Corners, Junctions, Concave:: UVs are smoothed, corners on discontinuous boundary, junctions of three or more regions and darts and concave corners are kept sharp.
Keep Boundaries:: UVs are smoothed, boundaries are kept sharp.
All:: UVs and their boundaries are smoothed.

Boundary Smooth

Controls how open boundaries (and corners) are smoothed.

All:: Smooth boundaries, including corners.
Keep Corners:: Smooth boundaries, but corners are kept sharp.

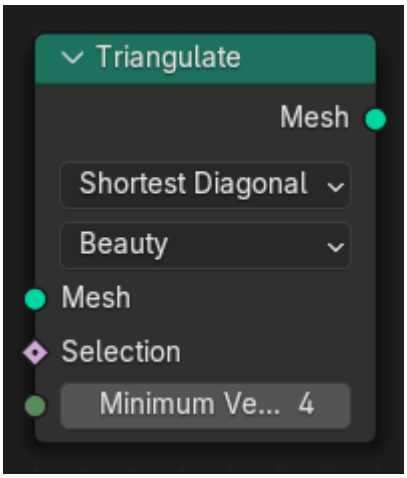
Outputs

Mesh

Standard geometry output.

Triangulate Node

The *Triangulate* node converts all faces in a mesh (quads and n-gons) to triangular faces. It functions the same as the Triangulate tool in Edit Mode.



Inputs

Mesh
: Standard geometry input.

Selection
: A standard Boolean selection input to determine which faces will be triangulated.

Minimum Vertices
: Minimum number of vertices a face must have to be triangulated. For example, setting this value to 5, will prevent triangulation of Quads and only triangulate N-gons.

Properties

Quad Method

Beauty:: Split the quads in nice triangles, slower method.

Fixed:: Split the quads on their 1st and 3rd vertices.

Fixed Alternate:: Split the quads on their 2nd and 4th vertices.

Shortest Diagonal:: Split the quads along their shortest diagonal.
Longest Diagonal:: Split the quads along their longest diagonal. This is the preferred mode for cloth simulations.

N-gon Method

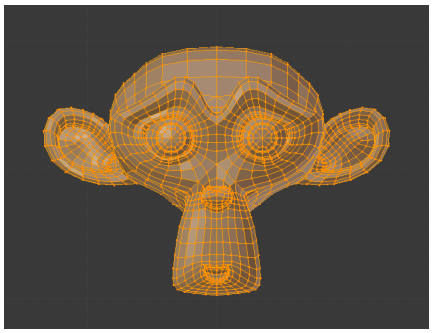
Beauty:: Arrange the new triangles nicely, slower method.
Clip:: Split n-gons using an ear-clipping algorithm (the same method of tessellation used for viewport display).

Outputs

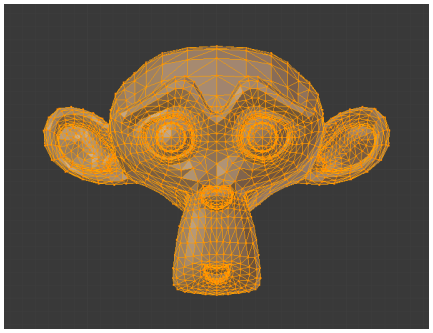
Mesh

Standard geometry output.

Example



Mesh before triangulation.



Mesh after triangulation.

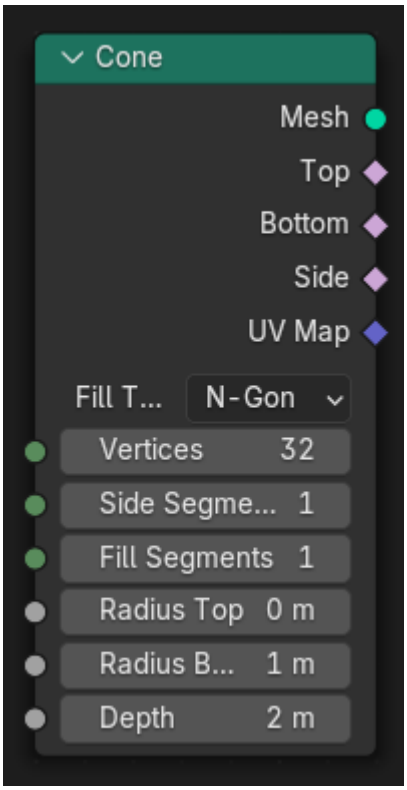
Mesh Primitive Nodes

Nodes that create a primitive mesh, e.g. a cube.

- Cone Node
- Cube Node
- Cylinder Node
- Grid Node
- Icosphere Node
- Mesh Circle Node
- Mesh Line Node
- UV Sphere Node

Cone Node

Generates a cone mesh that is optionally truncated.



Inputs

Vertices
: Number of vertices in the top and/or bottom circle of the cone. No geometry is generated if the number is below three.

Side Segments
: Number of vertically stacked face loops that make up the cone's sides. Increasing this will add horizontal cuts. No geometry is generated if the number is below one.

Fill Segments
: Number of concentric rings in the top and/or bottom. No geometry is generated if the number is below one.

Radius Top
: The radius of the cone's top circle. If this is zero, the circle is reduced to a single vertex.

Radius Bottom
: Same as *Radius Top* but for the bottom circle.

Depth
: Height of the generated cone.

Note

If the top and bottom radii are both zero, this node will output a single line.

Properties

Fill Type
: How the circles at the top and bottom are filled with faces when their radius is larger than zero.

None::	Do not fill the circles.
N-Gon::	Fill the innermost circles with a single face.
Triangles::	Fill the innermost circles with triangles connected to a vertex in the center.

Outputs

Mesh
: Standard geometry output.

Top
: A boolean field with a selection of the faces on the top of the cone. If the *Fill Type* is set to *None*, this will be a selection of the top edges instead. If *Radius Top* is zero, this will be a selection of the top vertex.

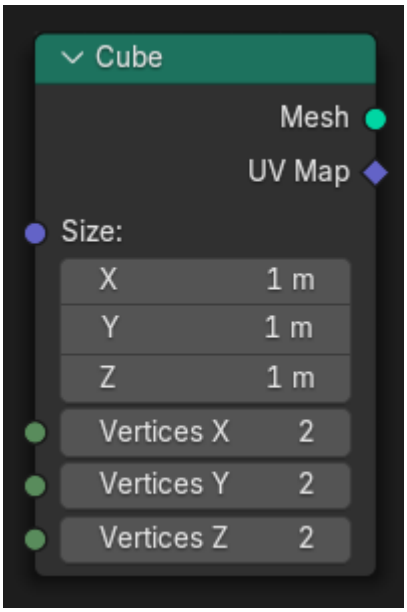
Side
: A boolean field with a selection of the faces on the side of the cone.

Bottom
: A boolean field with a selection of the faces on the bottom of the cone. If the *Fill Type* is set to *None*, this will be a selection of the bottom edges instead. If *Radius Bottom* is zero, this will be a selection of the bottom vertex.

UV Map
: The default UV coordinate of each face corner. This can be connected to the Store Named Attribute Node for populating a UV Map.

Cube Node

The *Cube* node generates a cuboid mesh with variable side lengths and subdivisions. The inside of the mesh is still hollow like a normal cube.



Inputs

Size
: Side lengths along each of the main axes.

Vertices X, Y, Z
: Number of vertices for each side of the cube. The number of vertices should be at least 1.

Properties

This node has no properties.

Outputs

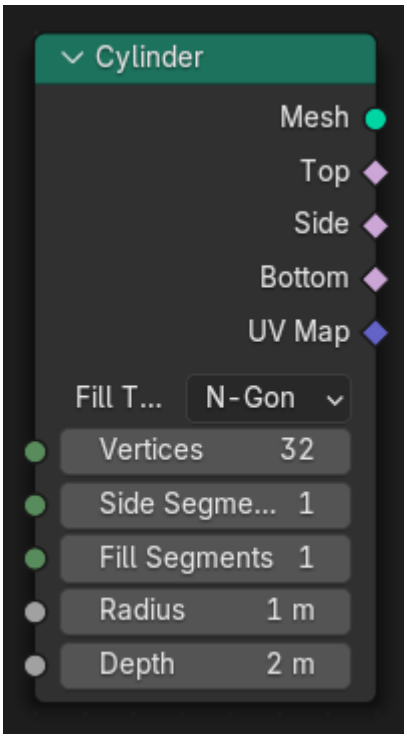
Mesh
: Standard geometry output.

UV Map

A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Cylinder Node

The *Cylinder* node generates a cylinder mesh. It is similar to the Cone node but always uses the same radius for the circles at the top and bottom.



Inputs

Vertices
: Number of vertices on the circle at the top and bottom. No geometry is generated if the number is below three.

Side Segments
: Number of edges running vertically along the side of the cone. No geometry is generated if the number is below one.

Fill Segments
: Number of concentric rings used to fill the round faces at the top and bottom. No geometry is generated if the number is below one.

Radius

Distance of the vertices from the Z axis. If this is zero, the output will be a single line.

Depth
: Height of the cylinder.

Properties

Fill Type
: How the circles at the top and bottom are filled with faces when their radius is larger than zero.

None::	Do not fill the circles.
N-Gon::	Fill the innermost segment of the circles with a single face.
Triangles::	Fill the innermost segment of the circles with triangles connected to a new vertex on the Z axis.

Outputs

Mesh
: Standard geometry output.

Top
: A boolean attribute field with a selection of the faces on the top of the cylinder. If the *Fill Type* property is *None*, then this will be a selection of the top edges instead. If the *Radius* is zero, this will be a selection of the top point.

Side
: A boolean attribute field with a selection of the faces on the side of the cylinder.

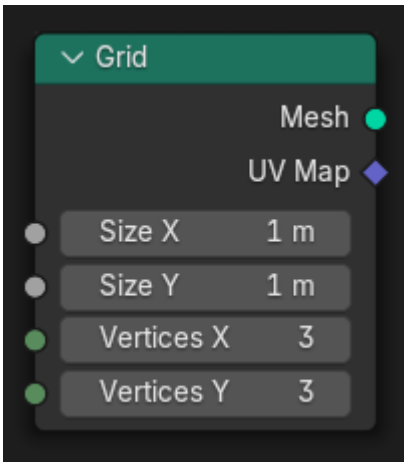
Bottom
: This is the same as the *Top* selection output, but on the bottom side of the geometry instead.

UV Map

A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Grid Node

The *Grid* node generates a planar mesh on the XY plane.



Inputs

Size X
: Side length of the plane in the X direction.

Size Y
: Side length of the plane in the Y direction.

Vertices X
: Number of vertices in the X direction. If this is smaller than two, no mesh is generated.

Vertices Y
: Number of vertices in the Y direction. If this is smaller than two, no mesh is generated.

Properties

This node has no properties.

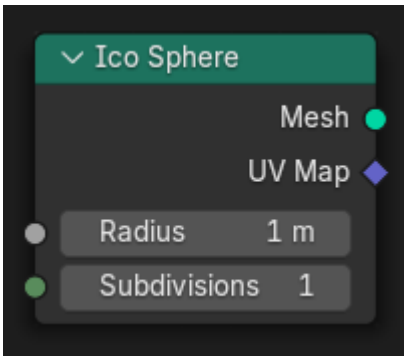
Outputs

Mesh
: Standard geometry output.

UV Map
: A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Icosphere Node

The *Icosphere* node generates a spherical mesh that consists of equally sized triangles.



Inputs

Radius
: Distance of the vertices from the origin.

Subdivisions
: Number of subdivisions on top of the most basic icosphere. The number of faces quadruple with every subdivision.

Properties

This node has no properties.

Outputs

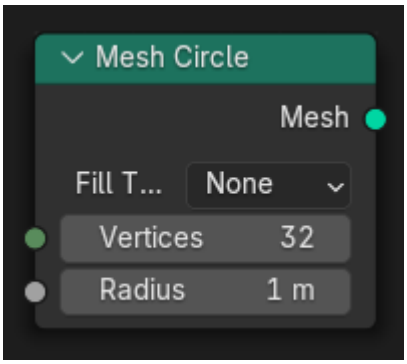
Mesh
: Standard geometry output.

UV Map
: A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get

applied. The UV map must be stored on the face corner in order to be accessed.

Mesh Circle Node

The *Mesh Circle* node generates a circular ring of edges that is optionally filled with faces.



Inputs

Vertices
: Number of vertices on the circle. No geometry is generated when the number is below three.

Radius
: Distance of the vertices from the origin.

Properties

Fill Type
: How the circle is filled with faces.

None::	Output just the edge ring without any faces.
N-Gon::	Fill the circle with a single face.
Triangles::	Fill the circle with triangles connected to a new vertex at the origin.

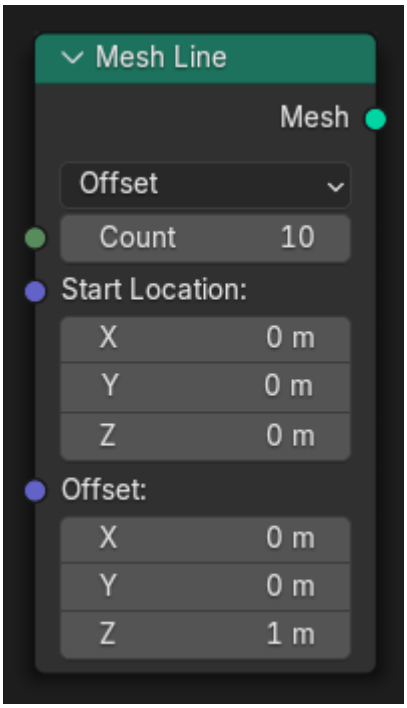
Outputs

Mesh

Standard geometry output.

Mesh Line Node

The *Mesh Line* node generates vertices in a line and connects them with edges.



Inputs

Count
: Number of vertices on the line.

Resolution
: Length of individual edges. The node tries to fit as many vertices as possible between the start and end point. The exact end point might not be hit. This is only available when the mode is set to *End Points* and the count mode is set to *Resolution*.

Start Location
: Position of the first vertex.

Offset

Controls the direction of the line and distance between the vertices. This is only available when the mode is set to *Offset*.

End Location
: Position of the last vertex. This is only available when the mode is set to *End Points*.

Properties

Mode
: Inputs to use to control the line.

Offset::	Specify the offset from one vertex to the next.
End Points::	Specify the start and end point of the line.

Count Mode
: Determines how the number of vertices is chosen. This is only available when the mode is set to *End Points*.

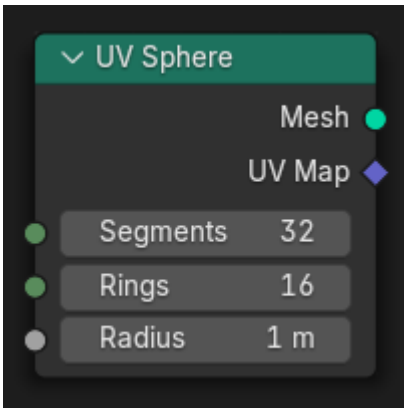
Count::	Specify the total number of vertices.
Resolution::	Specify the distance between vertices.

Outputs

Mesh
: Standard geometry output.

UV Sphere Node

The *UV Sphere* node generates a spherical mesh mostly out of quads except for triangles at the top and bottom.



Inputs

Segments
: Horizontal resolution of the sphere. If this is smaller than three, no mesh is generated.

Rings
: Vertical resolution of the sphere. If this is smaller than two, no mesh is generated.

Radius
: Distance of vertices to the origin.

Properties

This node has no properties.

Outputs

Mesh
: Standard geometry output.

UV Map

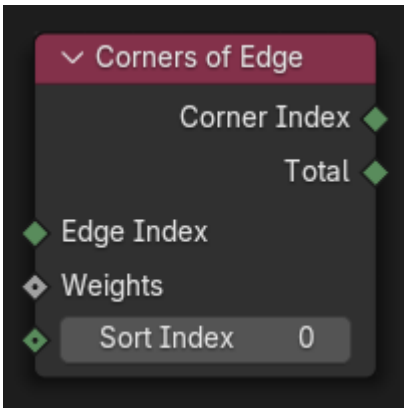
A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Mesh Topology Nodes

Nodes that retrieve information about the connectivity between mesh elements.

- Corners of Edge Node
- Corners of Face Node
- Corners of Vertex Node
- Edges of Corner Node
- Edges of Vertex Node
- Face of Corner Node
- Offset Corner in Face Node
- Vertex of Corner Node

Corners of Edge Node



Selects a neighboring face corner of an edge and outputs its index.

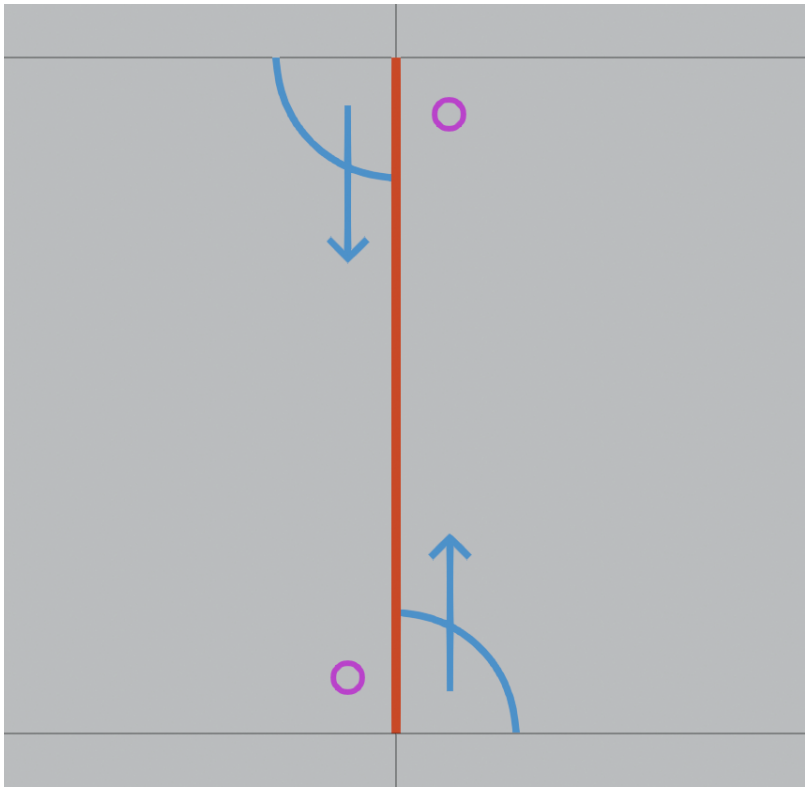
This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick an edge from the geometry based on the *Edge Index*.
- Find *some* (not all) face corners connected to this edge – see below.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Warning

As illustrated below, the node only looks at one corner per connected face. Even though the edge has four neighboring corners, *Corner Index* can only return the indexes of two of them, and *Total* will similarly return 2.

You can use the Offset Corner in Face Node to retrieve the indexes of the other corners.



A graphic for which corners are returned for a given edge

- Red: selected edge.
- Blue: the corners whose index can be retrieved using this node.
- Purple: the corners that can be retrieved by offsetting the blue corner indices using the Offset Corner in Face Node.

Inputs

Edge Index

The index of the edge for which to find connected corners.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Edge domain.

Weights
: The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Face Corner domain.

  The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index
: The 0-based index of the corner to select from the edge's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Corner Index
: The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

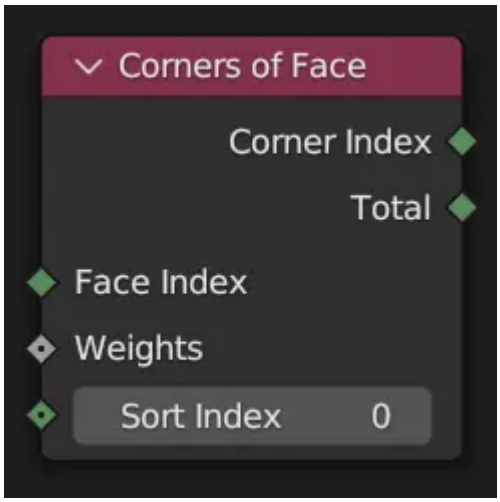
  If the edge has no connected corners, *Corner Index* will be zero.

Total
: The number of faces (not face corners!) connected to the edge.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Corners of Face Node



Selects a corner of a face and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick a face from the geometry based on the *Face Index*.
- Find the corners of this face.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Inputs

Face Index

The index of the face for which to find the corners.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face domain.

Weights

The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Face Corner domain.

The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index
: The 0-based index of the corner to select from the face's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

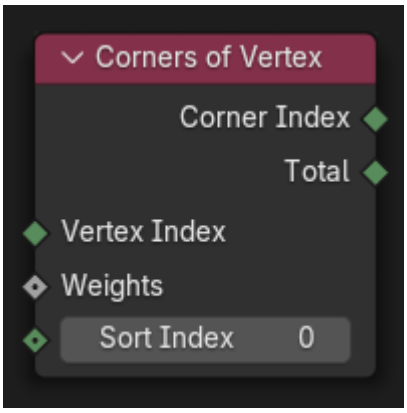
Corner Index
: The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

Total
: The number of corners in the face, which is also its number of edges.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Corners of Vertex Node



Selects a neighboring face corner of a vertex and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick a vertex from the geometry based on the *Vertex Index*.
- Find the face corners adjacent to this vertex.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Inputs

Vertex Index
: The index of the vertex for which to find the corners.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Point domain.

Weights
: The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the

Face Corner domain.

The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index
: The 0-based index of the corner to select from the vertex's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Corner Index
: The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

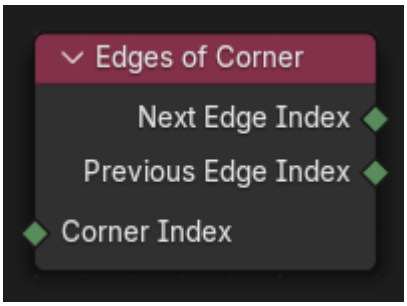
Total
: The number of adjacent corners, which is also the number of faces.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Edges of Corner Node

The *Edges of Corner* node retrieves the edges on both sides of a face corner.



Inputs

Corner Index
: The index of the input face corner.

> **Note**
>
> By default this uses the index from the field context, which makes it important that the node is evaluated on the face corner domain.

Properties

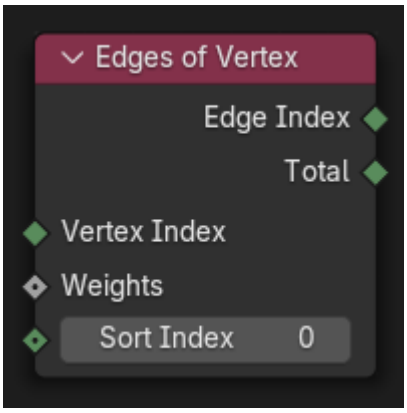
This node has no properties.

Outputs

Next Edge Index
: The index of the neighboring edge in the face, in the direction of increasing face corner indices.

Previous Edge Index
: The index of the neighboring edge in the face, in the direction of decreasing face corner indices.

Edges of Vertex Node



Selects a neighboring edge of a vertex and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each edge in the geometry. Then, for each item in the context domain, it will:

- Pick a vertex from the geometry based on the *Vertex Index*.
- Find the edges connected to this vertex.
- Sort these edges by their associated weight.
- Pick an edge from the above sorted list based on the *Sort Index*, where 0 means the edge with the lowest weight, 1 means the edge with the second-lowest weight and so on.
- Output the geometry-wide index of this edge.

Inputs

Vertex Index
: The index of the vertex for which to find the edges.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Point domain.

Weights
: The weights of the edges in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Edge domain.

The edges are sorted by their associated weight in ascending order. Edges with the same weight are sorted by their index.

Sort Index

The 0-based index of the edge to select from the vertex's sorted edges. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Edge Index

The geometry-wide index of the selected edge. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Edge) to retrieve details about the edge.

If the vertex has no connected edges, *Edge Index* will be zero.

Total

The number of edges connected to the selected vertex.

Example

The example below creates a cone at each vertex of a "cube," aligned to the neighboring edge that's the most vertical.

First, we calculate a "verticality score" for each of the cube's edges. To do this, we subtract the positions of its vertices to get its direction vector, which we normalize and use to calculate the dot product with the Z axis. The absolute value of that gives us a number between 0 and 1, where 0 means fully horizontal and 1 means fully vertical.

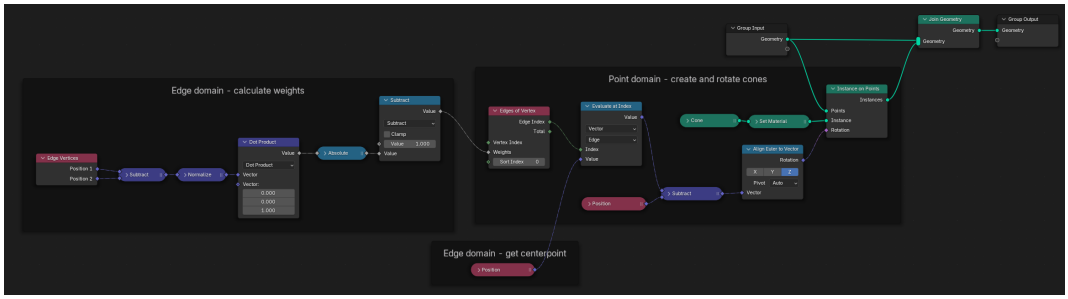
Because the edges will be sorted by ascending weight, we set weight = 1 - verticality. This way, the most vertical connected edge of each vertex will

have the lowest weight and come first in the list.

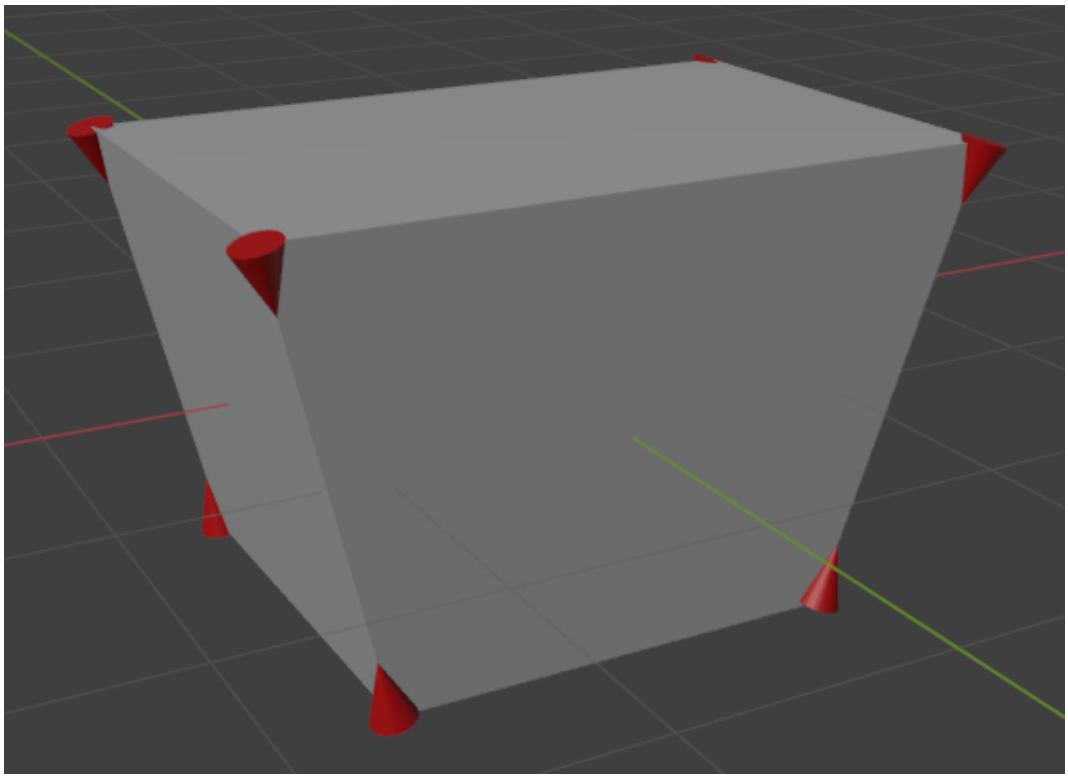
Next, in the point domain, we need to calculate the rotation of each cone. By using the Align Rotation to Vector Node, the problem gets simplified and we only need to calculate a direction vector.

The direction vector of each cone is the centerpoint of the most vertical neighboring edge minus the position of the vertex. Finding that most vertical neighboring edge is where the *Edges of Vertex* node comes in: for each vertex, it sorts the connected edges by their weight and pick the first one (because the Sort Index is 0). Once we have the edge's index, we use the Evaluate at Index Node to retrieve its centerpoint.

With the rotations of the cones calculated, we use the Instance on Points Node to create them.



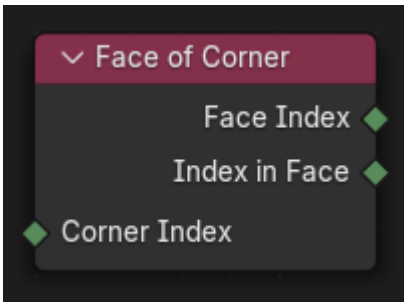
Example node setup. (Rightclick and choose "Open image in new tab" to see a larger version.)



The resulting geometry.

Face of Corner Node

Retrieves the face that a face corner is part of.



Inputs

Corner Index
: The geometry-wide index of the corner.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face Corner domain.

Properties

This node has no properties.

Outputs

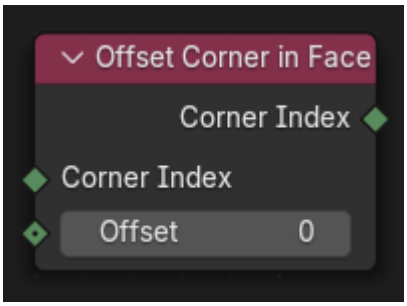
Face Index
: The geometry-wide index of the face which the corner belongs to.

Index in Face
: The face-local index of the corner. This is 0 for the first corner of the face, 1 for the next corner, and so on up to (number of corners - 1) for

the last corner.

Offset Corner in Face Node

Retrieves another corner in the same face as the input corner. This is like “rotating” the input corner around in its face.



Conceptually the operation is similar to the Offset Point in Curve Node.

Inputs

Corner Index
: The index of the input face corner.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it’s important that the node is evaluated in the Face Corner domain.

Offset
: The number of corners to move around the face before finding the result, circling back to the first corner if necessary.

Properties

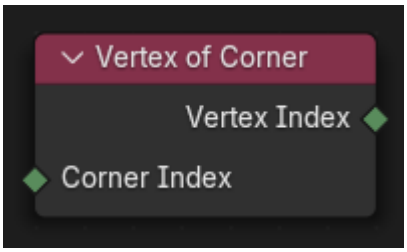
This node has no properties.

Outputs

Corner Index
: The index of the offset face corner.

Vertex of Corner Node

Outputs the index of the vertex that a face corner is attached to.



Inputs

Corner Index
: The index of the face corner.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face Corner domain.

Properties

This node has no properties.

Outputs

Vertex Index
: The index of the vertex that the face corner is attached to.

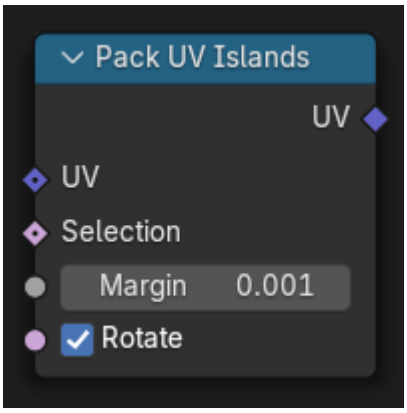
UV Nodes

Nodes for creating or modifying UV Maps.

- Pack UV Islands Node
- UV Unwrap Node

Pack UV Islands Node

The *Pack UV Islands Node* scales islands of a UV map and moves them so they fill the UV space as much as possible.



See also

The Pack Islands operator performs a similar operation in the UV editor.

Inputs

UV
: The UV map to modify.

Selection
: Faces to consider when packing islands. UVs that are part of any other face will not be affected.

Margin
: The distance to leave between UV islands.

Rotate
: Allow Rotating islands for best fit.

Properties

This node has no properties.

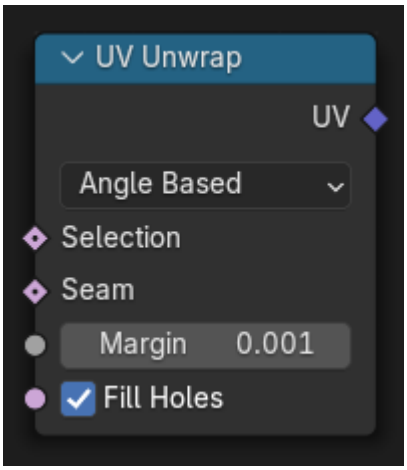
Output

UV

The modified UVs.

UV Unwrap Node

The *UV Unwrap Node* generates a UV map islands based on a selection of seam edges. The node implicitly performs a Pack Islands operation upon completion, because the results may not be generally useful otherwise.



See also

The Unwrap operator performs a similar operation in the UV editor. Unlike the Unwrap operator, the node doesn't perform aspect ratio correction, because it is trivial to implement with a Vector Math Node.

Inputs

Selection
: Faces to participate in the unwrap operation. UVs that are part of any other face will not be affected.

Seam
: Edges to mark where the mesh is "cut" for the purposes of unwrapping.

Margin
: The distance to leave between UV islands.

Fill Holes
: Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry.

Properties

Method

Angle Based:: This method gives a good 2D representation of a mesh

Conformal:: Uses LSCM (Least Squares Conformal Mapping). This usually gives a less accurate UV mapping than Angle Based, but works better for simpler objects

Output

UV

The generated UV coordinates between 0 and 1 for each face corner in the selected faces.

Note

In order for Blender to recognize the created attribute as a UV map, it must be created with the Store Named Attribute Node on the Face Corner domain with the 2D Vector data type. This is necessary because there is no 2D Vector socket type.

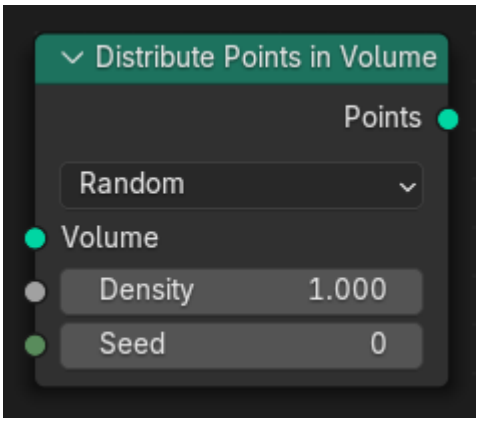
Point Nodes

Nodes generate or modify point clouds.

- Distribute Points in Volume
- Distribute Points on Faces
- Points Node
- Points to Curves Node
- Points to Vertices Node
- Points to Volume Node

- Set Point Radius Node

Distribute Points in Volume



The *Distribute Points in Volume* node creates points inside of volume grids. The node has two basic modes of operation: distributing points randomly, or in a regular grid. Both methods operate on all of the float grids in the volume.

Inputs

Volume
: Standard volume geometry input.

Density
: Number of points to sample per unit volume.

Spacing
: Spacing between grid points.

Threshold
: Minimum value of a volume cell to contain a grid point

Properties

Distribution Method

Random:: Distribute points randomly inside of the volume. The local point count is implicitly defined as a product of the global from the *Density* input and the local voxel value. This method creates a distribution that is not stable as the input volume deforms.

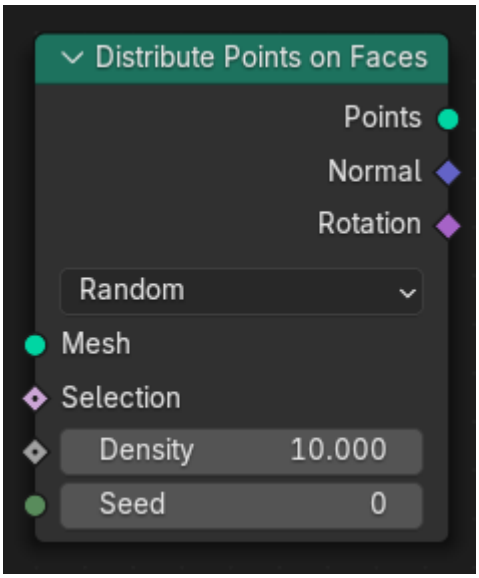
Grid:: Distribute the points in a grid pattern inside of the volume. At each grid point, the voxel value is used to determine whether to add a point.

Outputs

Points

Standard point cloud geometry output.

Distribute Points on Faces



The *Distribute Points on Faces* node places points on the surface of the input geometry object. Point, corner, and polygon attributes of the input geometry are transferred to the generated points. That includes vertex weights and UV maps. Additionally, the node has *Normal* and *Rotation* outputs.

The node also generates a stable ID, stored in the built-in `id` attribute, used as a stable identifier for each point. When the mesh is deformed or the density changes the values will be consistent for each remaining point. This attribute is used in the Random Value and Instance on Points nodes.

Inputs

Mesh
: Standard geometry input.

  Note

  The input geometry must contain a mesh with faces.

Selection
: The selection of which face corners should be considered for point distribution.

Distance Min
: The minimal distance points can have to each other. This option is only available for the *Poisson Disk* distribution method. At its default value

of zero, the node's behavior is the same as it is in *Random* mode, because none of the internally generated points are removed.

Density Max

The point density for the point distribution. The unit is in number of points per square meter. This value is multiplied by the values from the *Density* input. Only available in *Poisson Disk* mode.

Note

This will be capped on distributions by the *Distance Min* option. If the density is greater than what the minimal distance allows, no new points will be added after this threshold has been passed.

Density

The number of points to distribute per square meter on each mesh face. This value is multiplied by the values from the *Density Attribute*.

In *Poisson Disk* mode, this value is multiplied by the *Density Max* input for the final density.

Seed

The random Seed to use when generating points.

Properties

Distribution Method

Random::	Distribute points randomly on the surface. This is the fastest distribution method.
Poisson Disk::	Distribute points randomly on the surface while taking a minimum distance into account.

Legacy Normal

By default, the node uses smooth and custom normals for the *Normal* and *Rotation* values . An earlier version of this node only uses "true" normals, this option brings back this behavior of only using "true" normals.

This option can only be available in the Sidebar.

Outputs

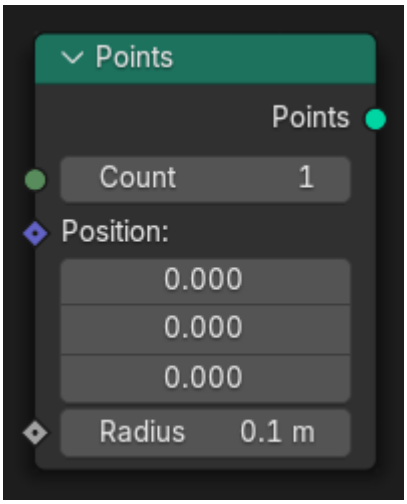
Points
: Generated points. Named attributes are copied to the result mesh, along with the data in the other attribute field outputs.

Normal
: The Normal of the triangle on which each point is scattered.

Rotation
: An XYZ Euler rotation built from the normal attribute for convenience. Such a value can also be built from the normal with the Euler to Rotation Node. Keep in mind that the Z axis of the result rotation will be arbitrary, since the mesh normal used to create the rotation does not have enough information to set all three rotation axes.

Points Node

The *Points* node generate a point cloud with positions and radii defined by fields.



Inputs

Count
: The number of points to create.

Position
: The position of each generated point.

Radius
: The radius of each point.

Note

Since the point cloud is created from scratch, the *Position* and *Radius* inputs can only depend on the index node. Regular input nodes like the position won't work.

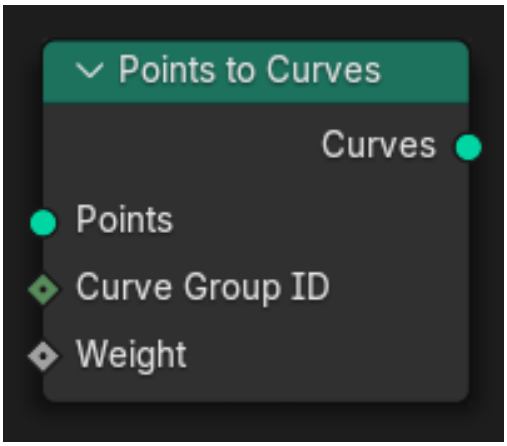
Properties

This node has no properties.

Outputs

Points
: Standard geometry output.

Points to Curves Node



The *Points to Curves* node generates a Curves geometry by taking all points and inserting them to new curves. All Attributes from points are propagated to Curve Points. Built-in curves attributes stored in points will be ignored.

Tip

To simplify thinking about points, attributes and their positions in each curve, The weight of each point in curve can be associated with a point attributes value. The sorting and grouping will be reflected on the attributes as like on the Weight and Group ID.

Inputs

Points
: The Point Cloud geometry component.

Curve Group ID
: All points with the same Group ID value will be joined in the same curve. The value of Group ID can be any value (negative, zero, or infinity, etc.). All created curves must have at least a single point. The order of curves depends both on Group ID value and on the order of Group ID values in the Point Cloud.

Weight
: If the curve contains more than one Point, the Weight of each Point is used to define the order of all points in curve via sorting. The goal of

sorting is to have points with the minimal Weight value at the start of curve and the maximum Weight at the end of curve.

Note

If points of curve have the same Weight value, the order will be the same as its original relative location. Without any Weight and Group ID inputs, each point will have the same indices in the curve.

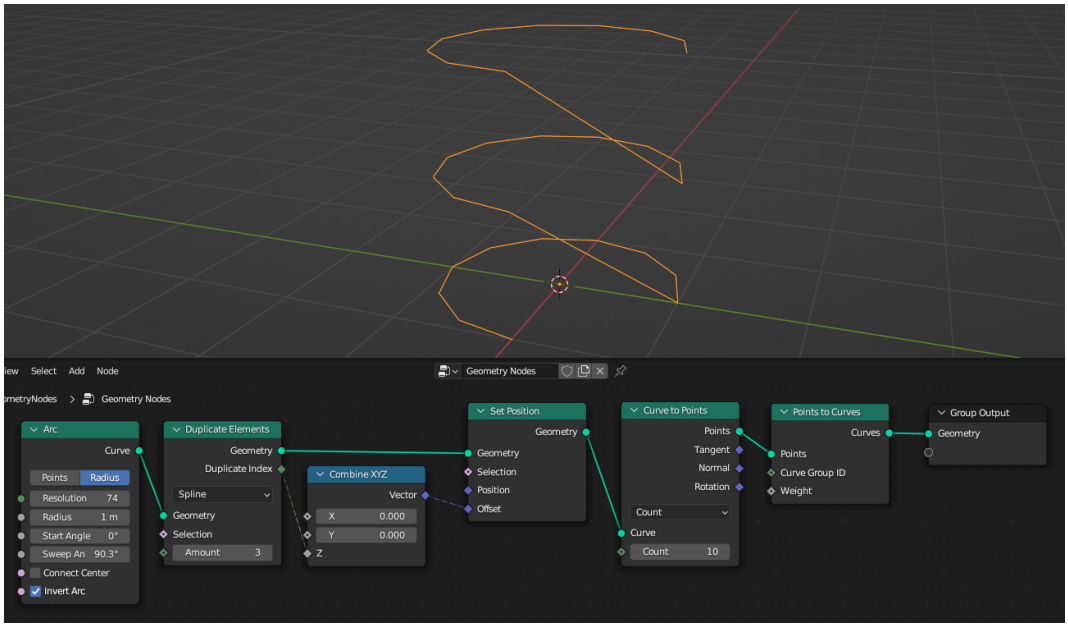
Properties

This node has no properties.

Outputs

Curves
: The curves with all copied points from the Point Cloud, but joined in curves. All other components aren't saved. The resulting curves are always non-cyclic.

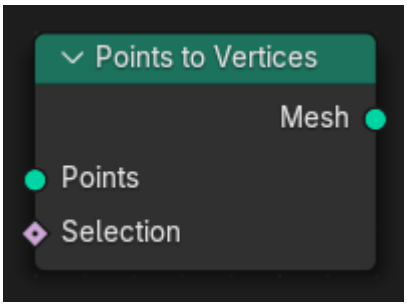
Examples



The above example creates a curve Array with connections between curves. This is created by duplicating the Arc primitive curve with the Duplicate Elements Node. Each curve is shifted in a top direction based on its index value. All the curves are converted to the Point Cloud by the Curve to Points Node. Finally, the points are converted to curves by the Points to Curves node.

All the Points of the resulting Curves geometry have the same attributes as points on the initial Arc primitive.

Points to Vertices Node



The *Points to Vertices* node generate a mesh vertex in the output geometry for each point cloud point in the input geometry.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field used to determine if each point will be converted to a vertex.

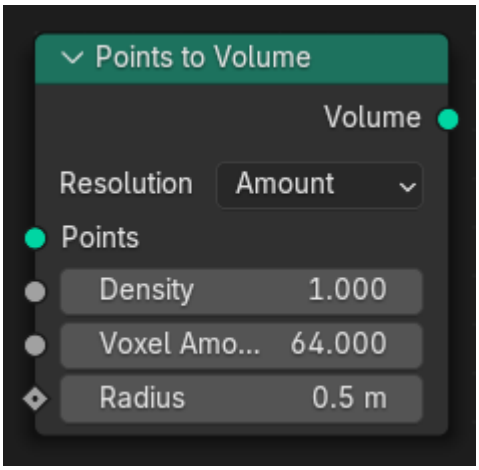
Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Points to Volume Node



The *Points to Volume* node generates a fog volume sphere around every point in the input geometry. The new volume grid is named “density”.

It usually makes sense to combine this node with the Volume to Mesh Node.

Warning

This node expects that point positions are not extremely large. For position values of many billions, the behavior isn’t guaranteed, and it may be unstable.

Inputs

Points
: Standard geometry input.

Density
: Value of voxels inside the generated fog volume.

Voxel Amount
: Specify the approximate number of voxels along the diagonal.

Voxel Size
: Specify the voxel side length.

Radius
: Specify the radius of the sphere generated at each point.

Properties

Resolution
: How the voxel size is specified.

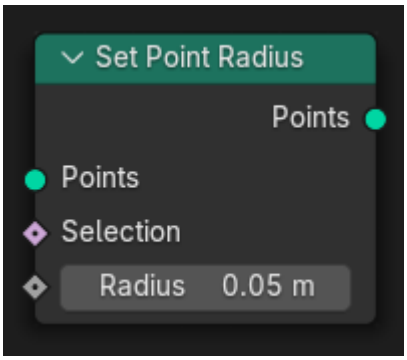
Amount::	Specify the approximate number of voxels along the diagonal.
Size::	Specify the voxel side length. It is recommended to be careful when tweaking this value, because small changes can have a large effect on the processing time.

Outputs

Volume
: Standard geometry output.

Set Point Radius Node

The *Set Point Radius* node controls the size each selected point cloud point should display with in the viewport.



The input node for this data is the Radius Node.

Inputs

Geometry
: Standard geometry input.

Radius
: Float value indicating the radius of the point geometry at each point.

Selection
: Boolean input for selecting which points will have the radius value applied.

Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Volume Nodes

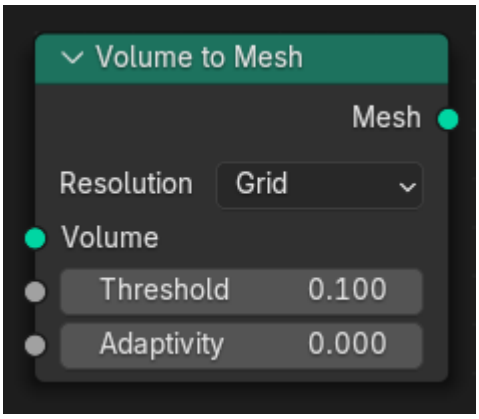
Nodes for creating or working with volumes.

- Operations
- Primitives

Volume Operation Nodes

- Volume to Mesh Node

Volume to Mesh Node



The *Volume to Mesh* node generates a mesh on the “surface” of a volume. The surface is defined by a threshold value. All voxels with a larger value than the threshold are considered to be inside.

Inputs

Volume
: Standard geometry input.

Voxel Amount
: Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.

Voxel Size
: Use a fixed resolution that does not change when the volume changes.

Threshold
: Voxels with a larger value are considered to be inside the mesh. The mesh will be generated on the boundary of inside and outside voxels. This is also called “iso value”.

Adaptivity

Reduces the final face count by simplifying geometry where detail is not needed. This is similar to decimating the final to reduce resolution where it is not needed.

Properties

Resolution Mode

Mode for how the resolution of the final mesh is controlled.

Grid::	This makes the resolution dependent on the resolution of the grid that is converted. Higher resolution grids result in a higher resolution mesh. In many cases, that is the most efficient mode.
Voxel Amount::	Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.
Voxel Size::	Use a fixed resolution that does not change when the volume changes.

Note

This option applies individually for every grid in the input geometry.

Outputs

Mesh

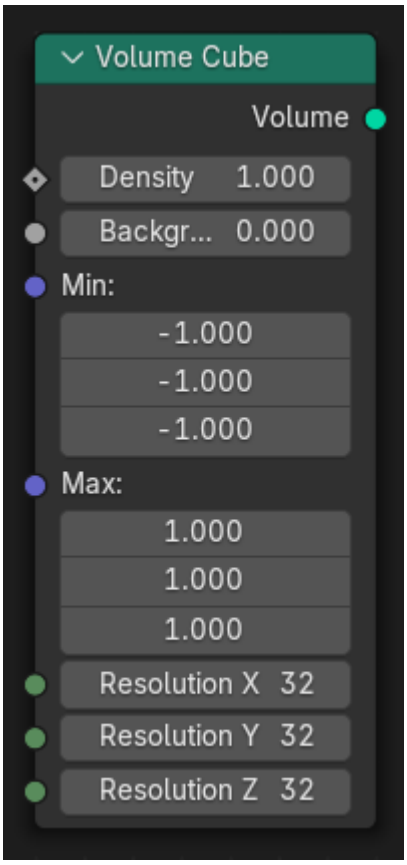
Standard geometry output.

Volume Primitive Nodes

- Volume Cube Node

Volume Cube Node

The *Volume Cube* generates a volume from scratch by evaluating an input field on every single voxel in a rectangular prism. The *Density* field defines the output volume grid's value at every voxel. The field can only depend on the Position Node.



Inputs

Density
: The value for the new grid at each voxel.

Background
: The value of the grid outside the rectangular prism controlled by the *Min* and *Max* inputs. The node can generate a more memory-efficient volume when the values of the *Density* input are the same as the background value.

Min

One corner of the rectangular prism in which to fill evaluate the field.

Max
: The other corner of the rectangular prism in which to fill evaluate the field.

Resolution X,Y,Z
: The number of voxels to evaluate the field in on each axis.

> **Note**
>
> Changing these values can have a significant impact on performance. For example, the default values of 32 mean the input field will be evaluated about 33 thousand times. Increasing the values to 100 will give 1 million evaluations, and 1000 would give 1 billion.

Properties

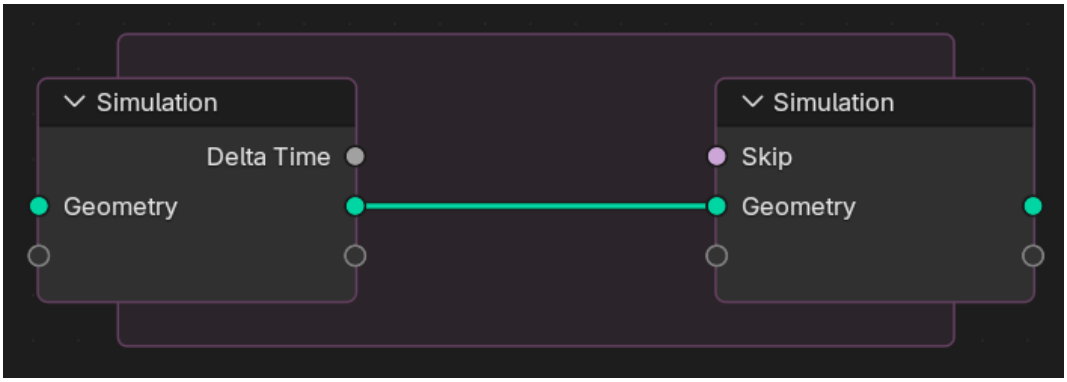
This node has no properties.

Outputs

Volume
: Geometry containing the generated volume.

Simulation Zone

Simulation zones allow the result of one frame to influence the next one. That way even a set of simple rules can lead to complex results, with the passing of time. The most common type of them is physics simulation, with specific solvers for physical phenomena.



Initial simulation nodes and simulation zone.

When adding a simulation, two nodes are added, defining between them a “Simulation Zone”.

The inputs that are connected to the Simulation Input node are evaluated only once, at the beginning of the simulation, passed to the next simulation state and eventually outputted. Other nodes can be linked inside the simulation region from the outside. Those are re-evaluated every step based on their value at the given frame.

It is not possible to have any link going towards outside. The result of the simulation can only be accessed via the Simulation Output node. This also allows sub-frame interpolation for motion blur.

Note

This node cannot be used in the Tool context—only in the *Modifier* context.

Clock

The simulation is tied to the animation system, with support for sub-steps. It will only be evaluated while the animation frame changes, and is cached like the existing physics simulations in Blender.

Properties

In the Node Editor the inputs can be renamed, shuffled and removed. This is also the place where sub-steps can be defined for a simulation.

Inputs

Geometry
: Standard geometry input, which is available by default to input geometry into the simulation zone. More bake items can be added by dragging sockets into the blank socket or in the *Simulation State* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

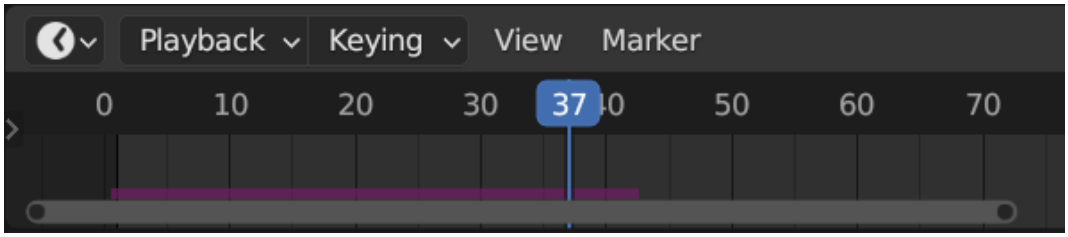
Delta Time
: The time in seconds between frames. Essentially this the inverse of the render Frame Rate.

  This delta is used to drive the simulation by connecting it node setups that depend on a rate. This will keep the simulation playback consistent when the frame rate changes.

Skip
: Forward the output of the simulation input node directly to the output node and ignore the nodes in the simulation zone.

Baking

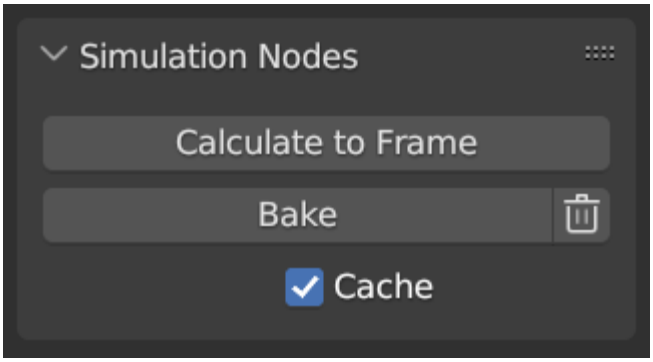
The simulation is automatically cached during playback. The valid cache can be seen as a strong yellow line in the timeline editor. This allows for animators to quickly inspect all the previous frames of a simulation.



Cached frames in the Timeline.

For the cases where the current frame is the only one relevant, users can opt-out of “Cache” to save memory.

When the result is ready to be sent to a render-farm, it can be baked to disk. This allows for the simulation to be rendered in a non-sequential order.



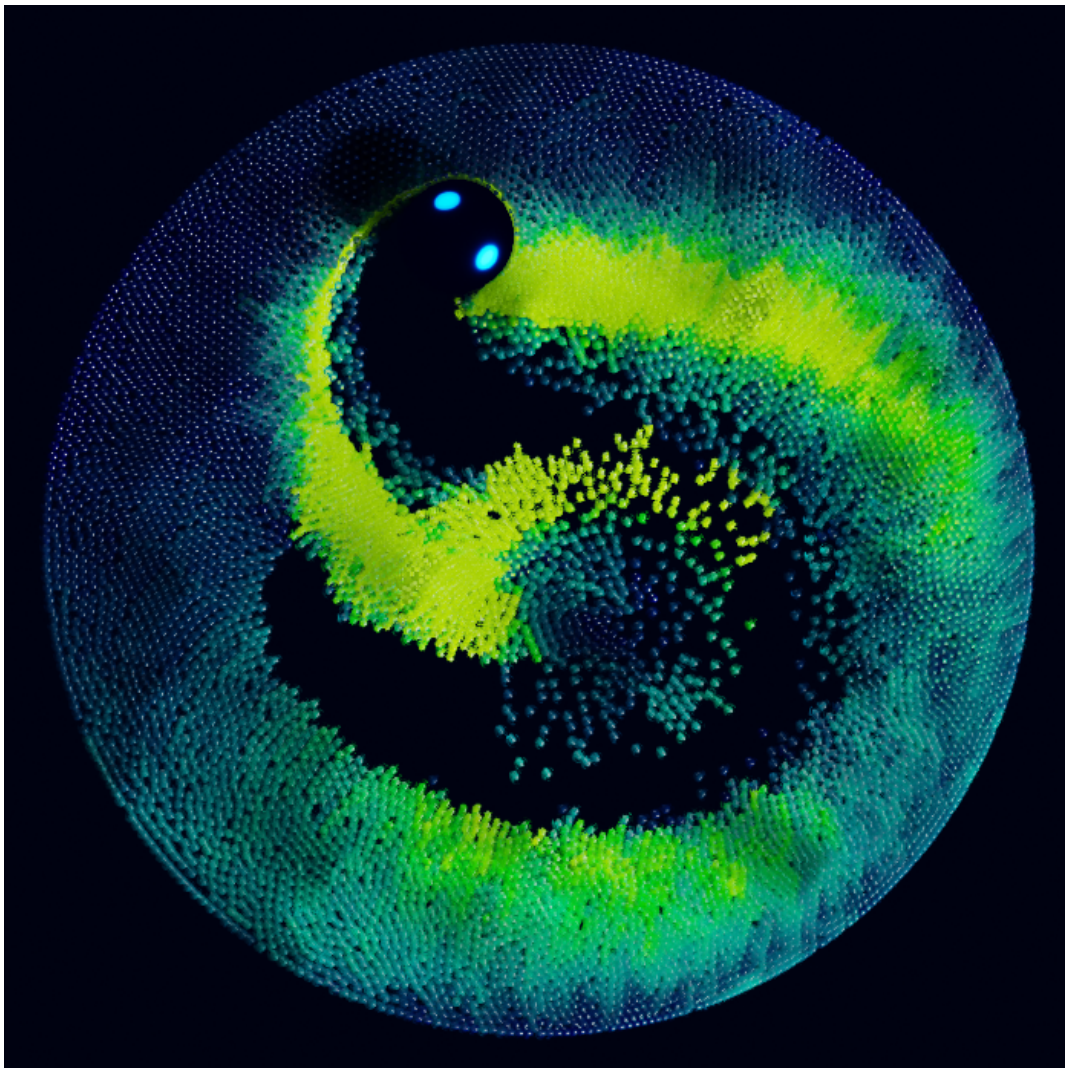
Simulation and Physics, Simulation Nodes user interface.

Note

Baking the simulation will bake all the simulations in all modifiers for the selected objects.

Examples

Combined with the Index of Nearest, this can be used for a number of sphere-based simulations.



Index of Nearest sample file CC-BY Sean Christofferson.

Material Nodes

Nodes that work with materials.

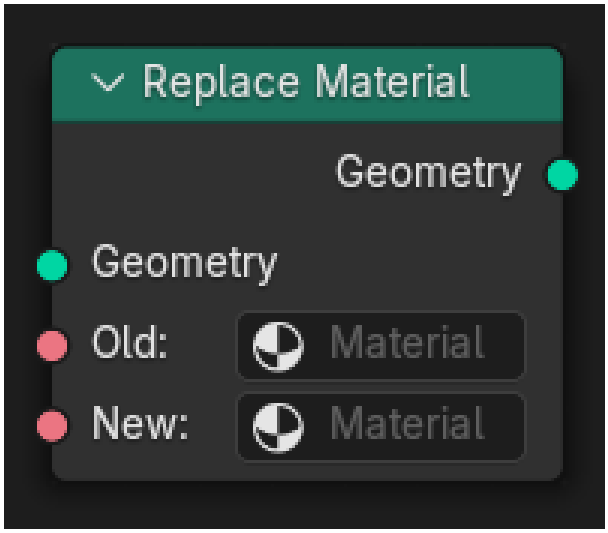
- Replace Material Node

- Material Index Node
- Material Selection Node

- Set Material Node
- Set Material Index Node

Replace Material Node

The *Replace Material* node swaps one material with another. Replacing a material with this node is more efficient than creating a selection of all faces with the old material with the Material Selection Node and then using the Set Material Node.



Note

Currently this node only adjusts mesh data.

Inputs

Geometry
: Standard geometry input.

Old
: Material that is going to be replaced.

New
: Material that is replacing the old material.

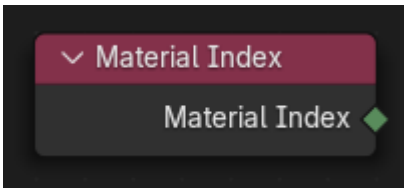
Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Material Index Node



The *Material Index* node outputs which material in the list of materials of the geometry each element corresponds to. Currently the node supports mesh data, where `material_index` is a built-in attribute on faces.

The node to set this data is the Set Material Index node.

Inputs

This node has no inputs.

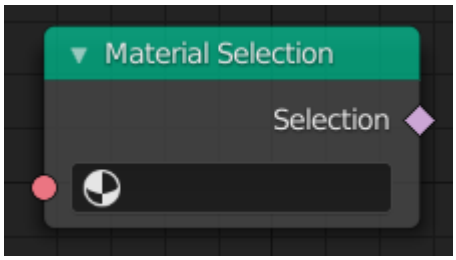
Properties

This node has no properties.

Outputs

Material Index
: Standard integer value, with a minimum value of zero.

Material Selection Node



The *Material Selection* node provides a selection for meshes that use this material. Since the `material_index` is stored on each face, the output will be implicitly interpolated to a different domain when necessary. For example, every vertex connected to a selected face will be selected.

Inputs

Material
: Standard material input.

Properties

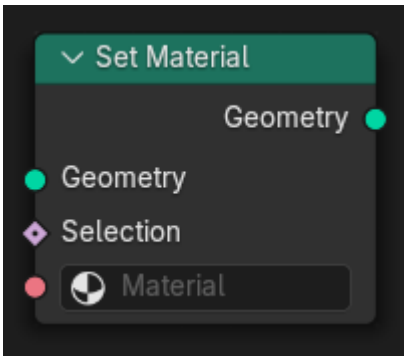
This node has no properties.

Outputs

Selection
: Selection of faces that use the input material.

Set Material Node

The *Set Material* changes the material assignment in the specified selection, by adjusting the `material_index` attribute. If the material is already used on the geometry, the existing material index will be reused.



Note

This node adjusts mesh, point clouds, and volume data; other data types do not support materials.

Inputs

Geometry
: Standard geometry input containing a mesh.

Material
: The material to apply to the geometry.

Selection
: Whether to change the material of each face. True values mean the material will be changed, false values mean it will remain the same.

  Note, volumes and point clouds only support a single material, in these cases a field input will be ignored.

Properties

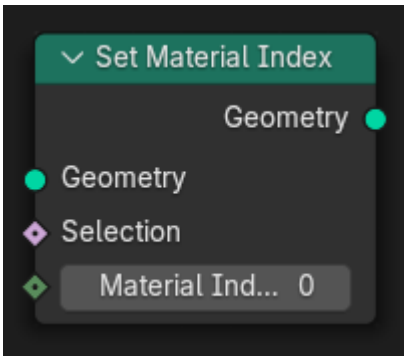
This node has no properties.

Outputs

Geometry
: Standard geometry output.

Set Material Index Node

The *Set Material Index* node sets the material index for a geometry.



The node to get this data is the Material Index node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether to change the material index for each face. True values mean the material index will be changed, false values mean it will remain the same.

Material Index
: The new material index.

Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Texture Nodes

Nodes to add textures.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This is more evident with textures that produce abrupt changes such as brick and checker. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

- Brick Texture Node
- Checker Texture Node
- Gabor Texture Node
- Gradient Texture Node
- Image Texture Node
- Magic Texture Node
- Musgrave Texture Node
- Noise Texture Node
- Voronoi Texture Node
- Wave Texture Node
- White Noise Texture Node

Brick Texture Node

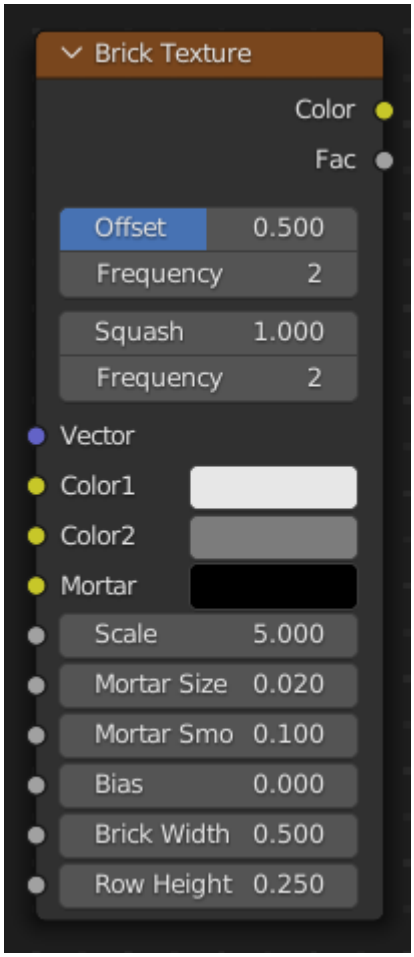
Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

The *Brick Texture* is used to add a procedural texture producing bricks.



Inputs

Color 1/2
: Color of the bricks.

Mortar
: The color of the area between bricks.

Scale
: Overall texture scale.

Mortar Size
: The size of the filling between the bricks known as “mortar”; 0 means no mortar.

Mortar Smooth
: Blurs/softens the edge between the mortar and the bricks. This can be useful with a texture and displacement textures.

Bias
: The color variation between *Color 1/2*. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width
: The width of the bricks.

Row Height
: The height of the brick rows.

Properties

Offset
: Determines the brick offset of the various rows.

Frequency
: Determines the offset frequency. A value of 2 gives an even/uneven pattern of rows.

Squash
: Amount of brick squashing.

Frequency
: Brick squashing frequency.

Outputs

Color
: Texture color output.

Factor
: Mortar mask (1 = mortar).

Examples



Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Checker Texture Node

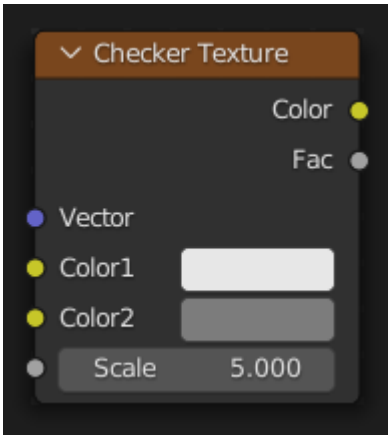
Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

The *Checker Texture* is used to add a checkerboard texture.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Warning

This node can have precision issues with some vector inputs. See the notes for the White Noise Texture for ways to mitigate this issue.

Color1, Color 2
: Color of the checkers.

Scale
: Overall texture scale. The scale is a factor of the bounding box of the face divided by the scale. For example, a scale of 15 will result in 15 alternate patterns over the overall UV bounding box. Different patterns could be achieved using other nodes to give different input patterns to this socket. For example, using the Math node.

Properties

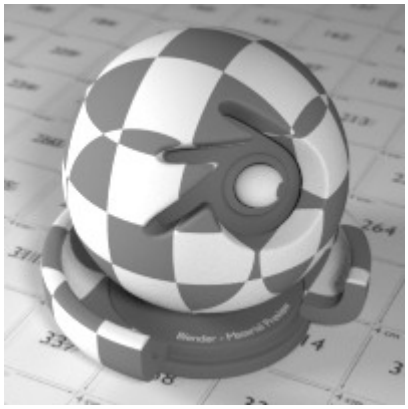
This node has no properties.

Outputs

Color
: Texture color output.

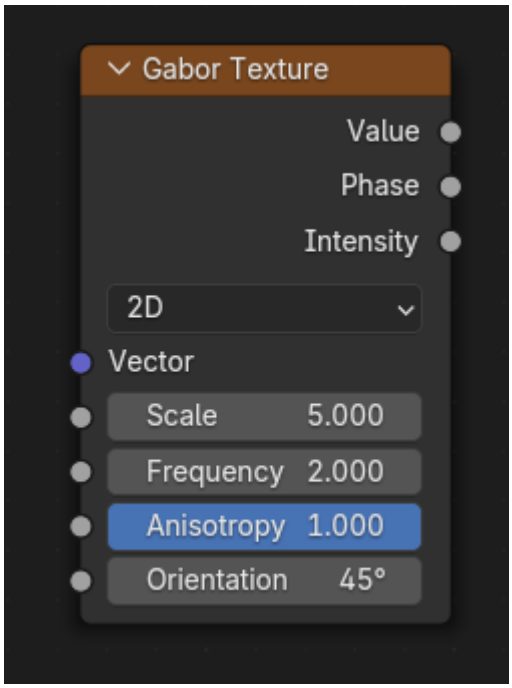
Factor
: Checker 1 mask (1 = Checker 1).

Examples



Default Checker texture.

Gabor Texture Node



The *Gabor Texture* node evaluates a Gabor noise at the input texture coordinates. Gabor noise is visually characterized by random interleaved bands whose direction and width can be controlled. Additionally, it can be used to create omnidirectional noise like the standard Noise Texture node, but since it is more expensive to compute, using the Noise Texture node is probably the better option in those cases. See the examples for more information.

Note

This node is ported from shader nodes.
The manual and images are referencing the shader version of the node.
This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Inputs

Vector
: The coordinates at which Gabor noise will be evaluated. The Z component is ignored in the 2D case. Defaults to *Generated* texture coordinates if the socket is left unconnected.

Scale
: Scale of the Gabor noise.

Frequency

The rate at which the Gabor noise changes across space. This is different from the Scale input in that it only scales perpendicular to the Gabor noise direction.

Anisotropy
: The directionality of Gabor noise. 1 means the noise is completely directional, while 0 means the noise is omnidirectional.

Orientation
: The direction of anisotropic Gabor noise. This is an angle for the 2D case, while it is a unit direction vector for the 3D case.

Properties

Type
: Type of Gabor noise texture.

2D::	Evaluates the noise in 2D space. The Z component of the input vector is ignored.
3D::	Evaluates the noise in 3D space.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Outputs

Value
: The Gabor noise value with both random intensity and phase. This is equal to sine the phase multiplied by the intensity.

Phase
: The phase of the Gabor noise, which has no random intensity.

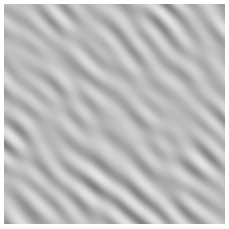
Intensity

The intensity of the Gabor noise, which has no random phase.

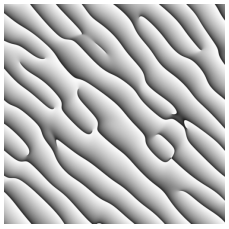
Examples

The following table demonstrates different outputs of the node with different parameters. As can be seen, the noise is visually characterized by interleaved bands that are generally oriented in a specific direction. But the *Anisotropy* parameter can be decreased below 1 to make the bands more random in directions. The *Frequency* parameter determines the number of bands perpendicular to the direction of the noise. However, the *Scale* parameter can also be used to globally increase the number of bands, so consider increasing the scale first since high frequency noise can suffer from low contrast and limited interleaving of bands.

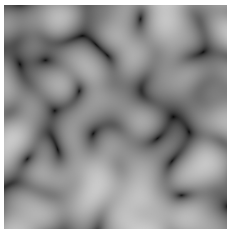
Different outputs with different parameters.



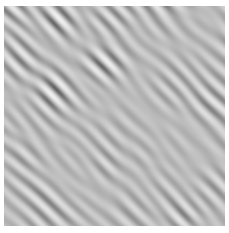
Value output. Frequency = 2. Anisotropy = 1.



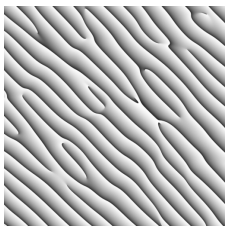
Phase output. Frequency = 2. Anisotropy = 1.



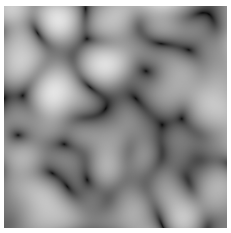
Intensity output. Frequency = 2. Anisotropy = 1.



Value output. Frequency =



Phase output. Frequency =

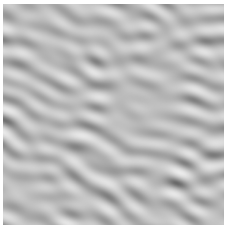


Intensity output.

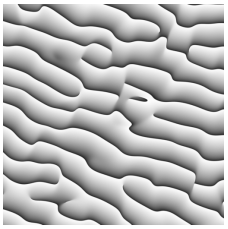
3. Anisotropy = 1.

3. Anisotropy = 1.

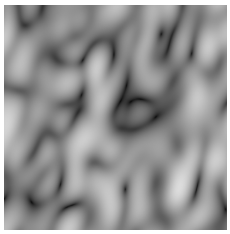
Frequency = 3. Anisotropy = 1.



Value output. Frequency = 2. Anisotropy = 0.7.

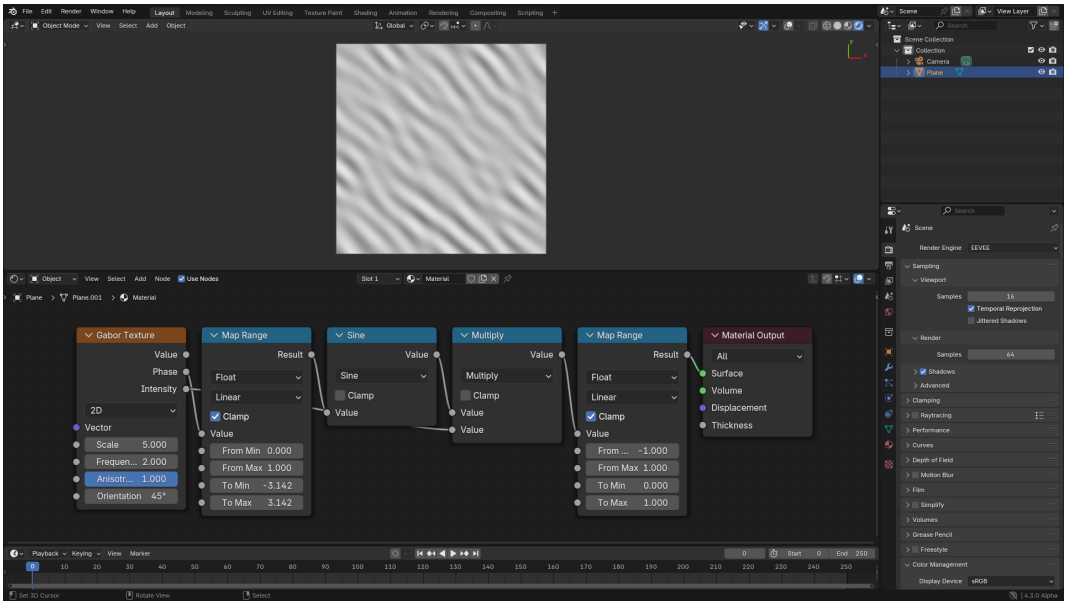


Phase output. Frequency = 2. Anisotropy = 0.7.



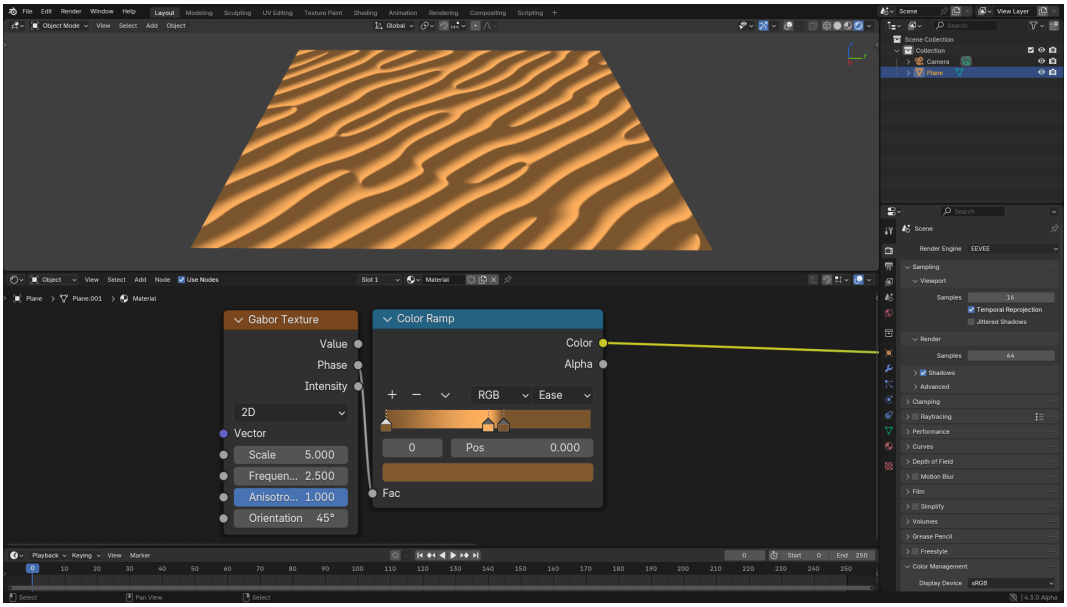
Intensity output. Frequency = 2. Anisotropy = 0.7.

Gabor noise is decomposed into a *Phase* and an *Intensity* components, where the Gabor value is computed as sine the phase multiplied by the intensity, noting that the phase output is normalized to the [0, 1] range.



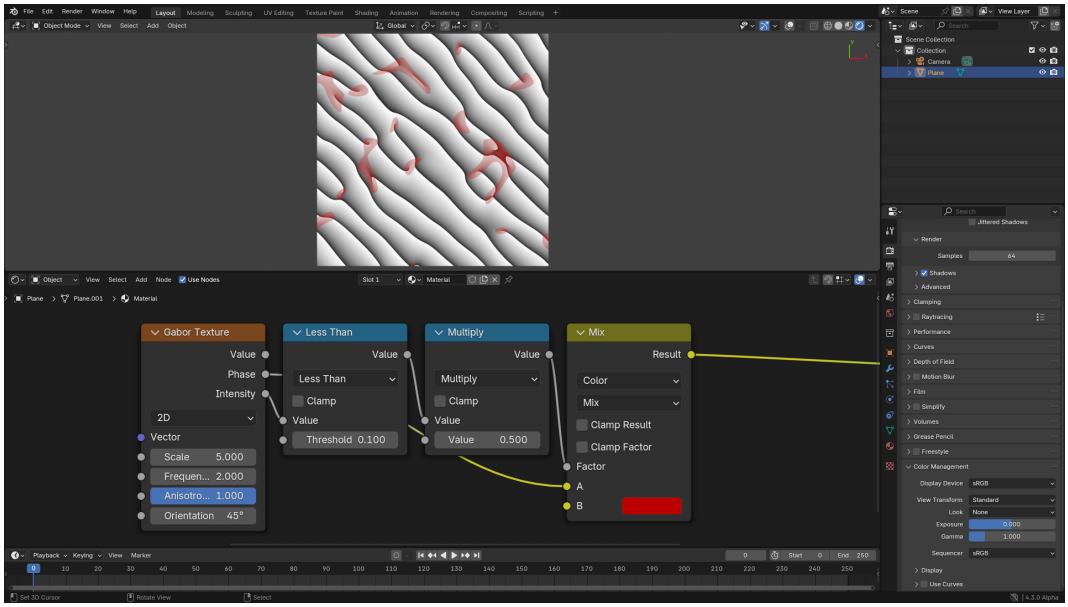
Compute the value output from the phase and intensity outputs.

The advantage of the *Phase* output is that it has no random intensities and no low contrast areas as in the value output, so it can be used as a base for textures that are more structured in nature, like sand dunes.



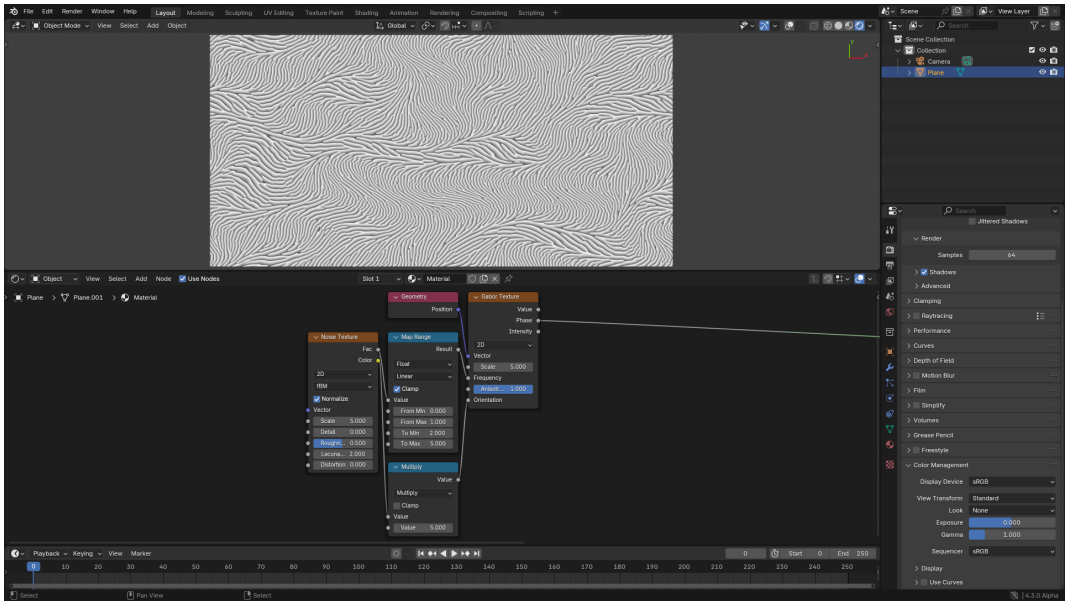
Sand dune-like structures creates using the phase output.

The main advantage and use of the *Intensity* output is that it provides information about the location of *singularities* in the *Phase* output. Singularities are those areas in the phase where the bands meet, which are shown in red in the following figure. Those areas will be close to zero in the *Intensity* output. So if those areas are undesirable, they can be hidden by multiplying by a variant of the *Intensity* output.



Visualization of the areas where singularities happen.

Inputs can be varies across space to get more interesting patterns.



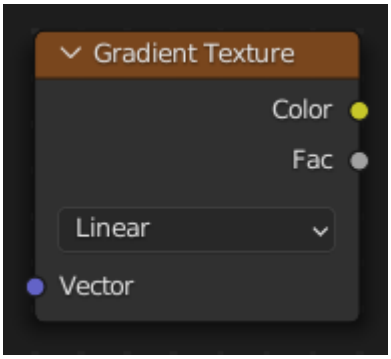
Varying the frequency and orientation across space.

Gradient Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Gradient Texture* node generates interpolated color and intensity values based on the input vector.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Type
: Controls the type of gradient generated.

Linear:: Directly outputs the input X coordinate.
Quadratic:: Interpolates the input X coordinate quadratically.
Easing::

	Uses a combination of quadratic and linear interpolation to generate a smooth gradient from the input X coordinate.
Diagonal::	Averages the input X and Y coordinates.
Spherical::	Creates an inverse gradient using the length of the input vector; the maximum value is at (0, 0, 0).
Quadratic Sphere::	The same as Spherical, except interpolated quadratically.
Radial::	Outputs a value based on the angle of the input around the Z axis.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

Examples



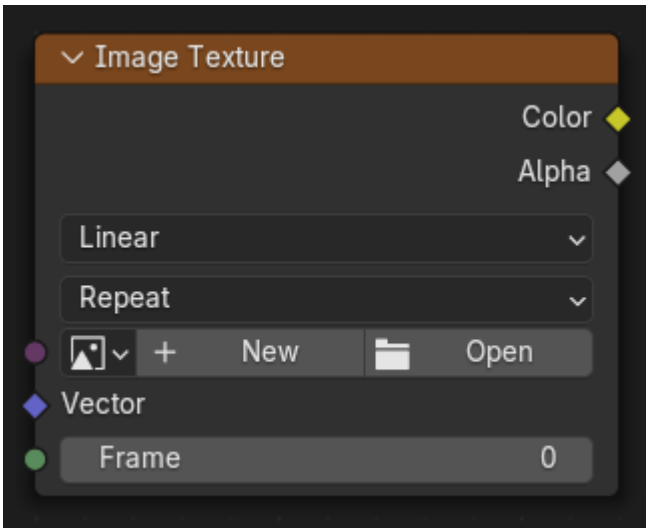
Gradient texture using object coordinates.

Image Texture Node

Note

Unlike the other texture nodes, this node operates differently in geometry nodes compared to the equivalent shader node. When not connected the Vector input has an implicit `position` attribute value.

The *Image Texture* node is used to add an image file as a texture. The image data is sampled with the input Vector and outputs a Color and Alpha value.



Inputs

Image
: The image socket can be used to connect to the Group Input node. If this is not connected the image can be opened or selected from the node.

Vector
: Texture coordinate for texture look-up. If this socket is left unconnected, the Position attribute is used.

Frame
: If the Image supports animation, the frame can be set here. This can be keyframed so that the image changes between frames.

Properties

Interpolation
: Method to scale images up or down for sampling.

Linear::	Regular quality interpolation.
Cubic::	Smoother, better quality interpolation. For bump maps this should be used to get best results.
Closest::	No interpolation, use only closest pixel for rendering pixel art.

Extension
: Extension defines how the image is extrapolated past the original bounds:

Repeat::	Will repeat the image horizontally and vertically giving tiled-looking result.
Extend::	Will extend the image by repeating pixels on its edges.
Clip::	Clip to the original image size and set all the exterior pixels values to transparent black.
Mirror::	Repeatedly flip the image horizontally and vertically.

Outputs

Color
: RGBA color from the image.

Alpha
: Alpha channel from image.

Examples

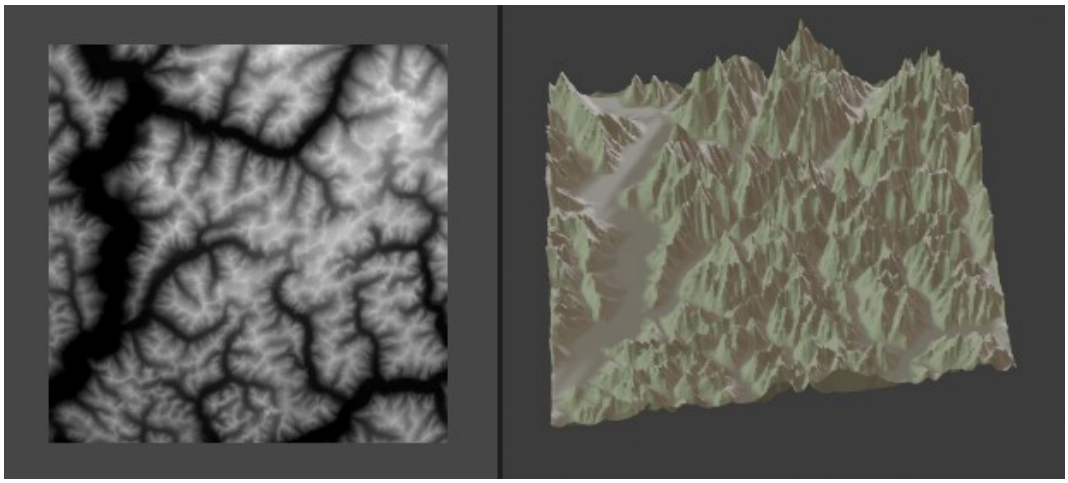


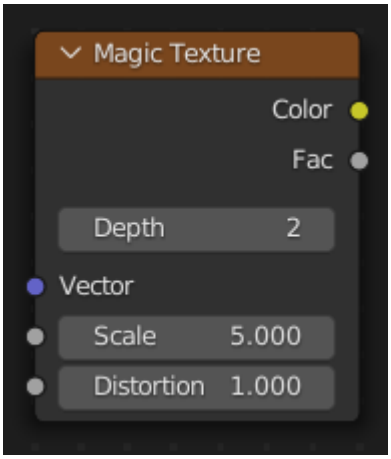
Image Texture displacing a plane.

Magic Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Magic Texture* node is used to add a psychedelic color texture.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Scale of the texture.

Distortion
: Amount of distortion.

Properties

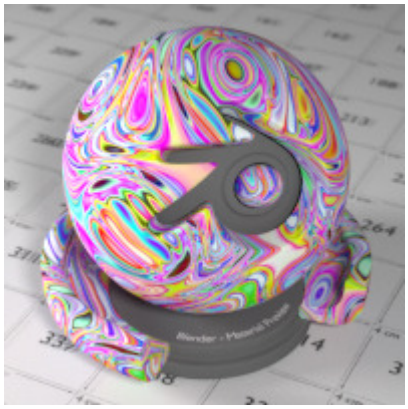
Depth
: Number of iterations.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

Examples



Magic texture: Depth 10, Distortion 2.0.

Musgrave Texture Node

The Musgrave texture node was replaced by the Noise Texture node, which includes all the same functionality.

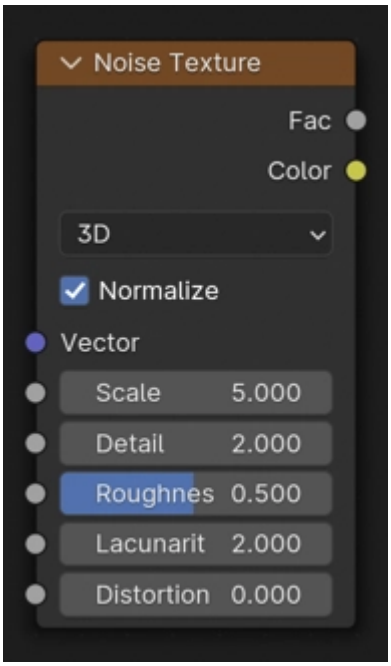
- The Dimension input was replaced by a Roughness input, where \(Roughness = Lacunarity^{-Dimension}\).
- The Detail input value must be subtracted by 1 compared to the old Musgrave Texture node.

Noise Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Noise Texture* node evaluates a fractal Perlin noise at the input texture coordinates. It can be used for a single Perlin noise evaluation, or for combining multiple octaves (layers) with increasingly finer detail.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the base noise octave.

Detail
: Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher render time.

Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

Lacunarity
: The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

Offset
: An added offset to each octave, determines the level where the highest octave will appear.

Gain
: An extra multiplier to tune the magnitude of octaves.

Distortion
: Amount of distortion.

Properties

Dimensions
: The dimensions of the space to evaluate the noise in.

1D::	Evaluate the noise in 1D space at the input *W*.
2D::	Evaluate the noise in 2D space at the input *Vector*. The Z component is ignored.

3D:: Evaluate the noise in 3D space at the input *Vector*.

4D:: Evaluate the noise in 4D space at the input *Vector* and the input *W* as the fourth dimension.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Normalize

If enabled, ensures that the output values stay in the range 0.0 to 1.0. When disabled, output values are in the range -1.0 to 1.0.

Type

Type of Noise texture, with different ways to combine octaves.

FBM:: Fractal Brownian motion, produces a homogeneous and isotropic result. Values from octaves are added together.

Multifractal:: More uneven, varying by location similar to real terrain. Values from octaves are multiplied together.

Hybrid Multifractal:: Creates peaks and valleys with different roughness values, like real mountains rise out of flat plains. Combines octaves using both addition and multiplication.

Ridged Multifractal:: Creates sharp peaks. Calculates the absolute value of the noise, creating “canyons”, and then flips the surface upside down.

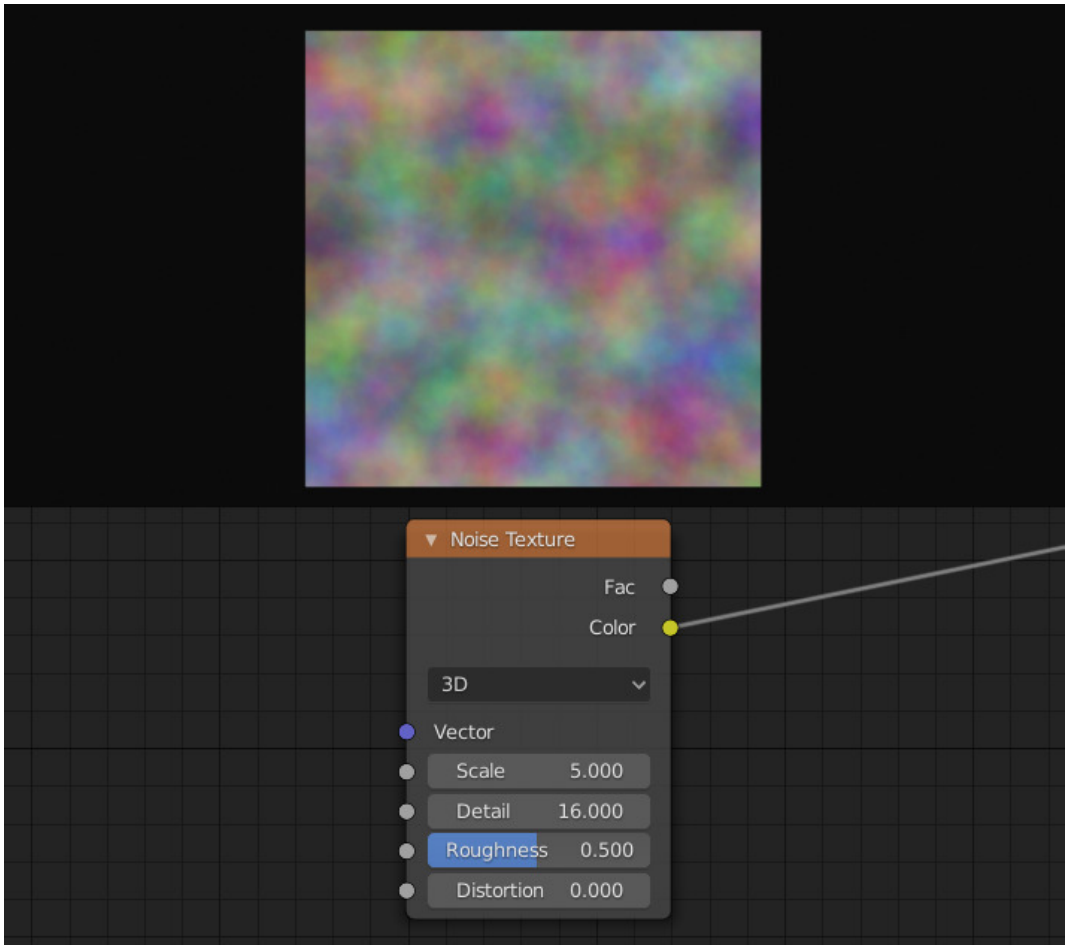
Hetero Terrain:: Similar to *Hybrid Multifractal* creates a heterogeneous terrain, but with the likeness of river channels.

Outputs

Factor
: Value of fractal noise.

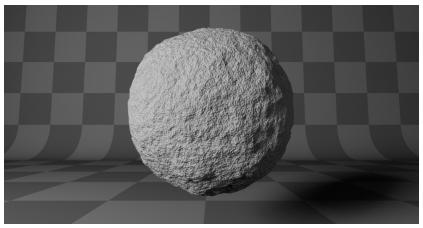
Color
: Color with different fractal noise in each component.

Examples

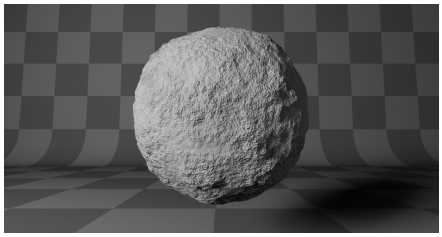


Noise Texture with high detail.

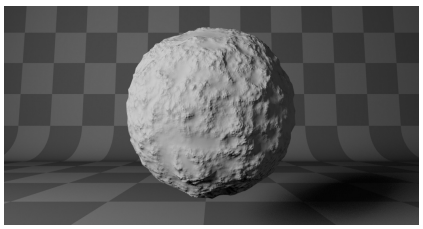
Different Noise types with the same parameters.



fBM (fractal Brownian Motion).



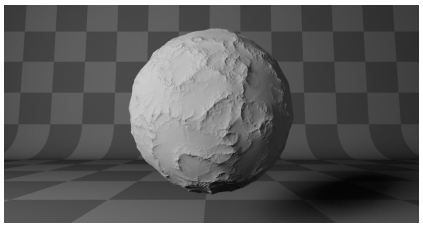
Multifractal.



Hybrid Multifractal.



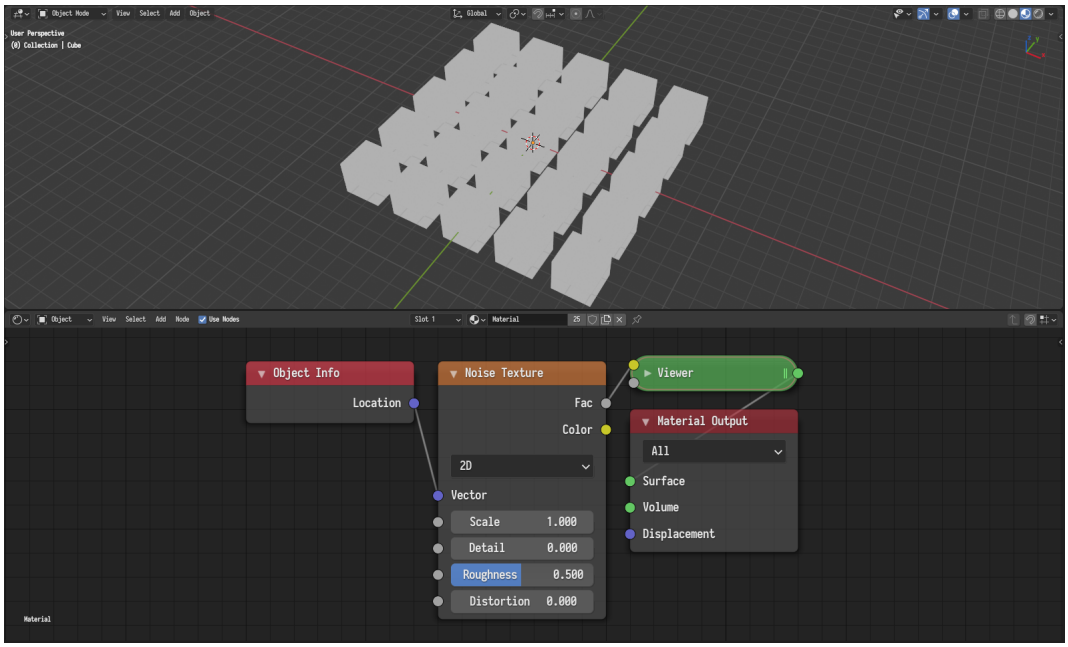
Heterogeneous Terrain.



Ridged Multifractal.

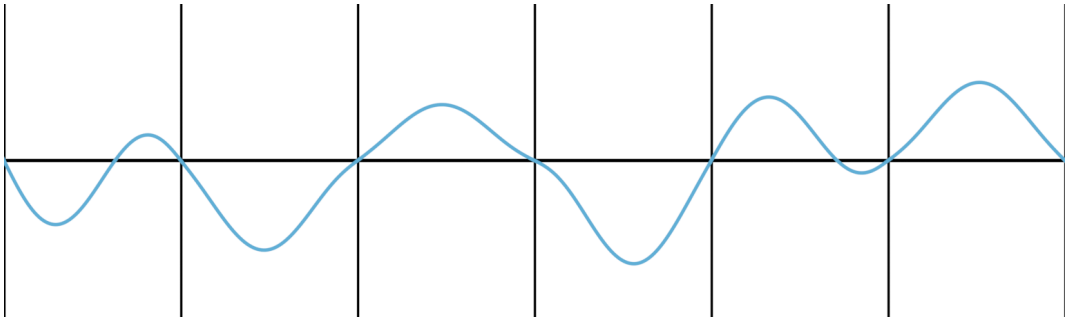
Notes

While the noise is random in nature, it follows a certain pattern that might not evaluate to random values in some configurations. For instance, consider the following configuration where a grid of objects have a material that evaluates a noise texture at their locations. One might expect the objects to have random values since they have different locations, but this is not the case.



An example configuration where the noise evaluates to a constant value.

It seems all objects have a value of 0.5. To understand why this happens, let us look at the following plot of a 1D noise texture.



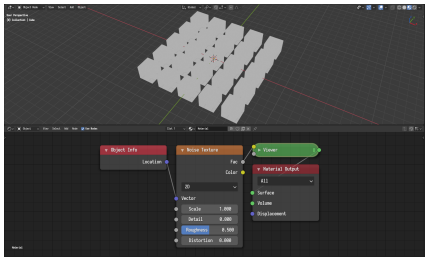
A plot of a 1D noise with zero details and zero distortion.

The horizontal line denotes a value of 0.5 and the vertical lines denotes whole numbers assuming a noise scale of 1. As can be seen, the noise always intersects the 0.5 line at whole numbers. Since the aforementioned objects were distributed on a grid and have whole number locations, they all evaluate to 0.5. Which explains the issue at hand.

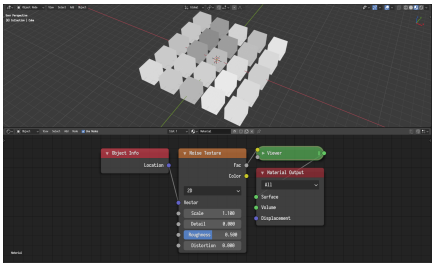
Generally, any discrete evaluation of noise at integer multiples of the reciprocal of the noise scale will always evaluate to 0.5. It also follows that evaluations closer to that will have values close to 0.5. In such cases, it is almost always preferred to use the White Noise Texture.

Regardless, one can mitigate this issue in a number of ways:

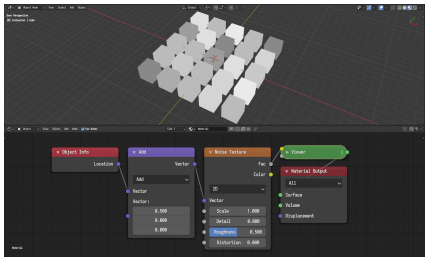
- Adjust the scale of the noise to avoid aligning the noise with the evaluation domain.
- Add an arbitrary offset to the texture coordinates to break the alignment with the evaluation domain.
- Evaluate the noise at a higher dimension and adjust the extra dimension until a satisfactory result is achieved.



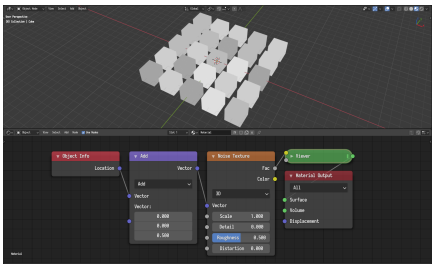
Constant value issue.



Mitigating the issue by adjusting the scale.

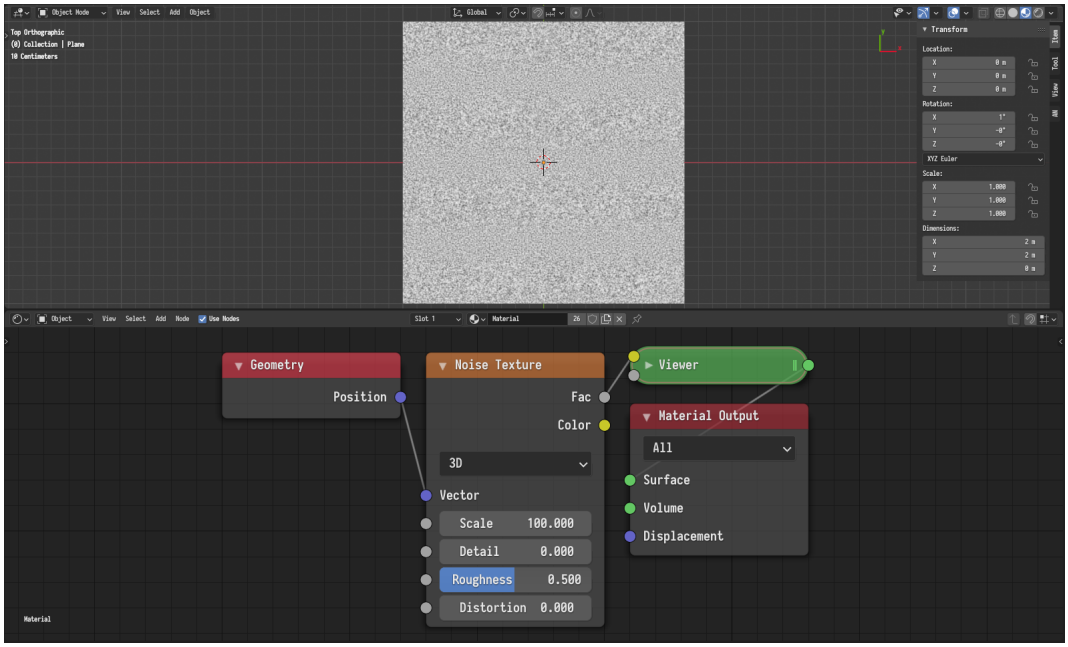


Mitigating the issue by adding an arbitrary offset.



Mitigating the issue by evaluating at a higher dimension.

Similarly, in other configurations, one might experience some banding patterns in the noise, where there are bands of high contrast areas followed by banding of low contrast areas. For instance, planar surfaces that are slightly tilted along one of the axis will have such a banding pattern.



An example configuration where the noise have a banding pattern.

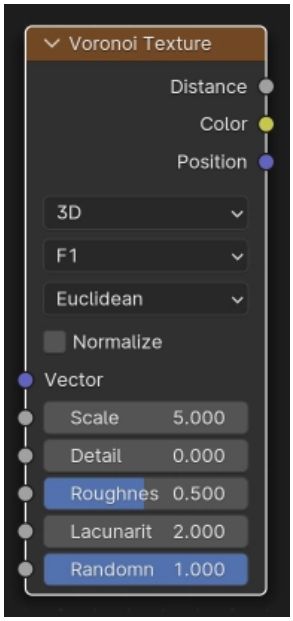
This happens because the slight tilt along one of the axis causes values along the perpendicular axis to change very slowly making the grid structure of the noise more apparent. The easiest way to mitigate this issue to rotate the coordinates by an arbitrary amount.

Voronoi Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Voronoi Texture* node evaluates a Worley Noise at the input texture coordinates.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the noise.

Detail
: Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher evaluation time.

Roughness

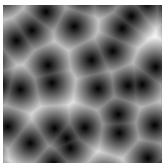
Blend between a smoother noise pattern, and rougher with sharper peaks.

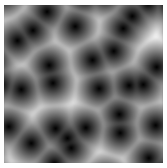
Lacunarity

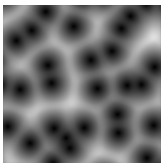
The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

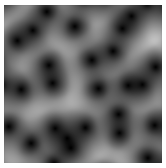
Smoothness

The smoothness of the noise.


Smoothness: 0.0.


Smoothness: 0.25.


Smoothness: 0.5.


Smoothness: 1.0.


Smoothness: 0.0.

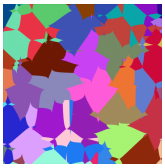

Smoothness: 0.25.


Smoothness: 0.5.


Smoothness: 1.0.

Exponent

Exponent of the Minkowski distance metric.


Exponent: 0.5.


Exponent: 1.0.

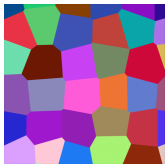

Exponent: 2.0.

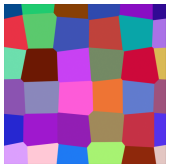

Exponent: 32.0.

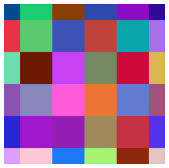
Randomness

The randomness of the noise.









Randomness: 1.0.	Randomness: 0.5.	Randomness: 0.25.	Randomness: 0.0.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

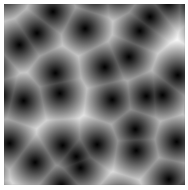
1D:: Evaluate the noise in 1D space at the input W.
2D:: Evaluate the noise in 2D space at the input Vector. The Z component is ignored.
3D:: Evaluate the noise in 3D space at the input Vector.
4D:: Evaluate the noise in 4D space at the input Vector and the input W as the fourth dimension.

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Feature

The Voronoi feature that the node will compute.

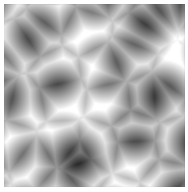
F1:: The distance to the closest feature point as well as its position and color.

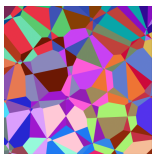

Distance.

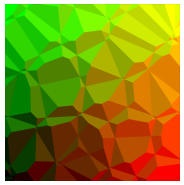

Color.


Position.

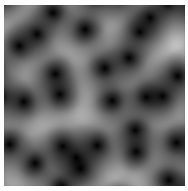
F2:: The distance to the second closest feature point as well as its position and color.


Distance.


Color.


Position.

Smooth F1:: A smooth version of F1.

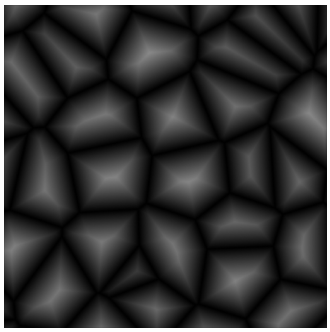



Color.

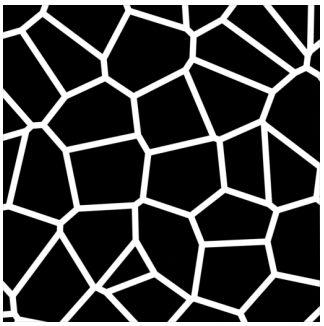


Distance. Position.

Distance to Edge:: The distance to the edges of the Voronoi cells.

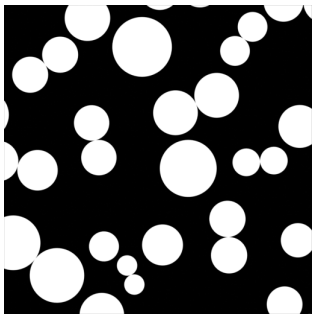


Distance.

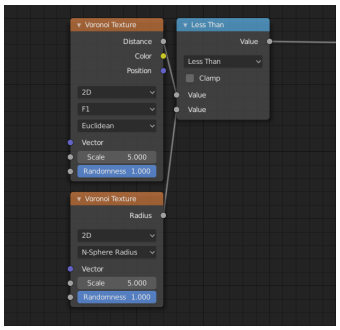


Distance smaller than 0.05.

N-Sphere Radius:: The radius of the n-sphere inscribed in the Voronoi cells. In other words, it is half the distance between the closest feature point and the feature point closest to it.



The n-sphere radius can be used to create tightly packed n-spheres.



Node tree for the shader to the left.

Distance Metric

The distance metric used to compute the texture.

Euclidean:: Use the Euclidean distance metric.

Manhattan:: Use the Manhattan distance metric.

Chebychev:: Use the Chebychev distance metric.

Minkowski:: Use the Minkowski distance metric. The Minkowski distance is a generalization of the aforementioned metrics with an *Exponent* as a parameter. Minkowski with an exponent of one is equivalent to the *Manhattan* distance metric. Minkowski with an exponent of two is equivalent to the *Euclidean* distance metric. Minkowski with an infinite exponent is equivalent to the *Chebychev* distance metric.



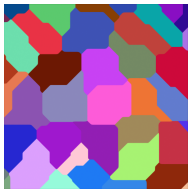
Minkowski Exponent: 0.5 (Minkowski 1/2).



Minkowski Exponent: 1.0 (Manhattan).



Minkowski Exponent: 2.0 (Euclidean).



Minkowski Exponent: 32.0 (approximation of Chebychev).

Normalize
: If enabled, ensures that the output values stay in the range 0.0 to 1.0. In rare cases, the output value may be outside that range when *Feature* is *F2*.

Outputs

Distance
: Distance.

Color
: Cell color. The color is arbitrary.

Position
: Position of feature point.

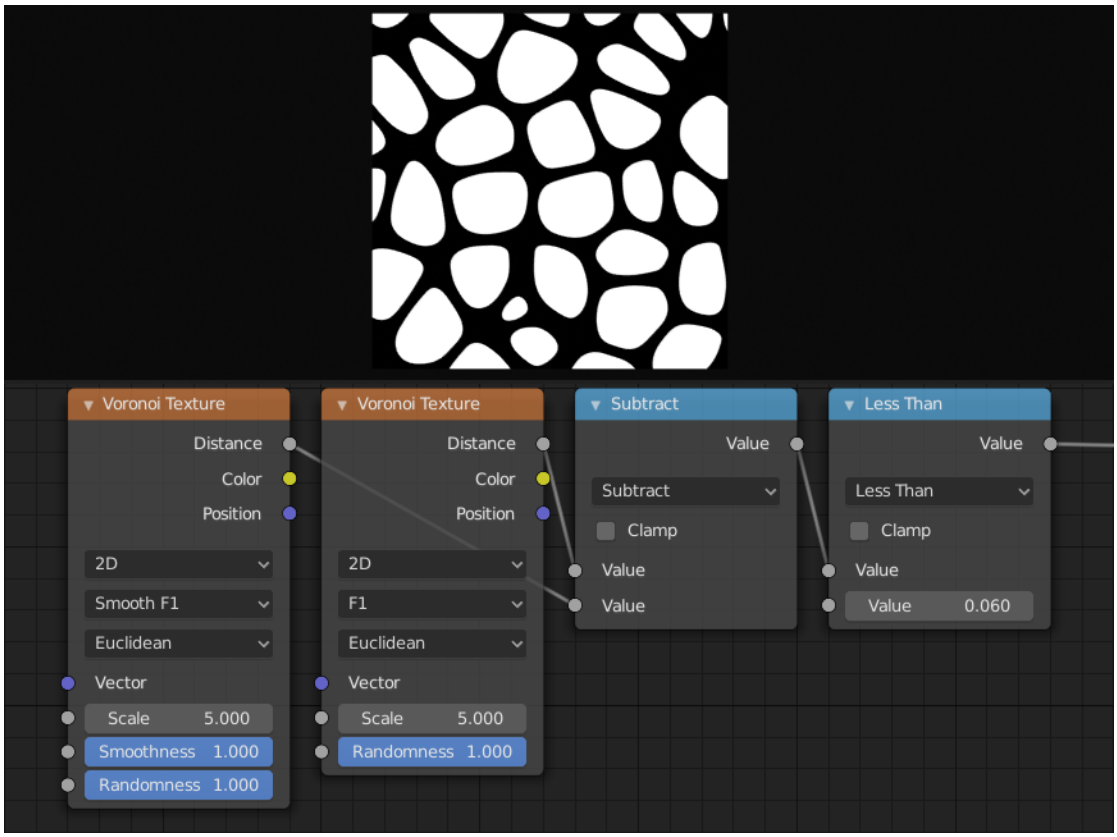
W
: Position of feature point.

Radius
: N-Sphere radius.

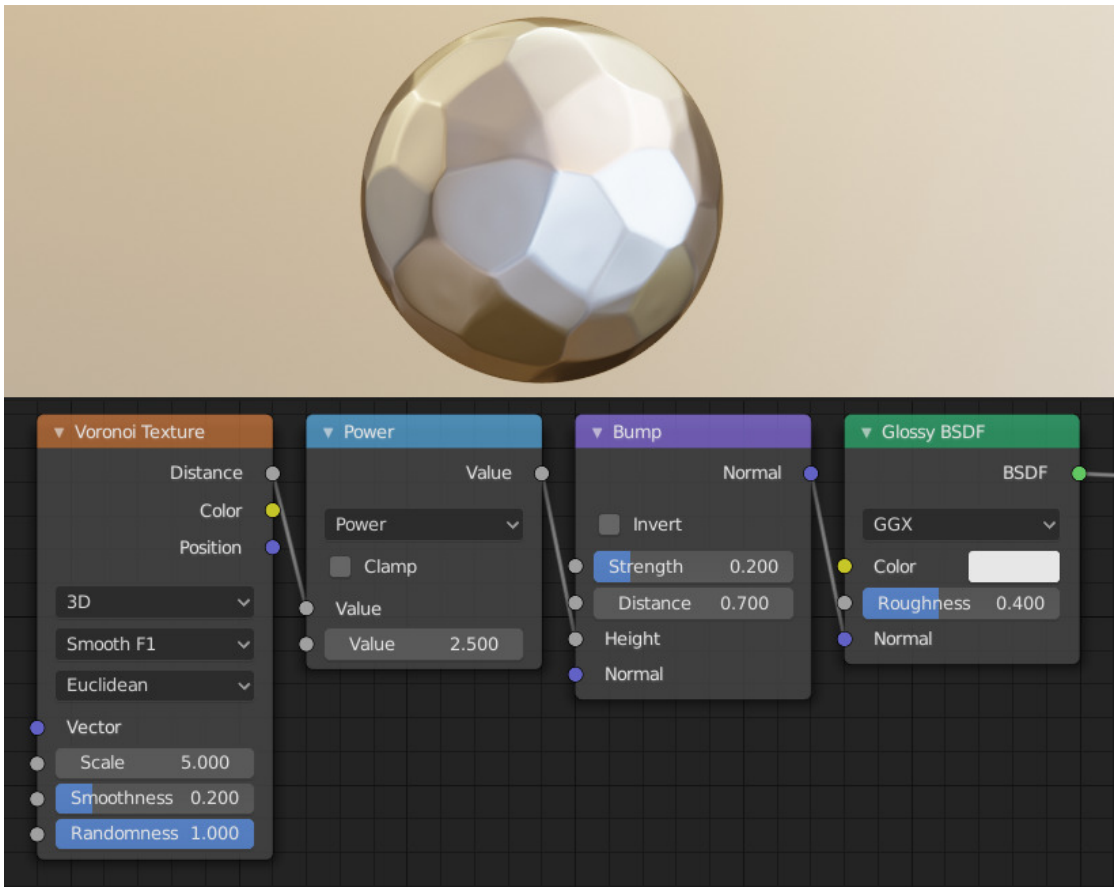
Notes

In some configurations of the node, especially for low values of *Randomness*, rendering artifacts may occur. This happens due to the same reasons described in the Notes section in the White Noise Texture page and can be fixed in a similar manner as described there.

Examples



The difference between *F1* and *Smooth F1* can be used to create beveled Voronoi cells.

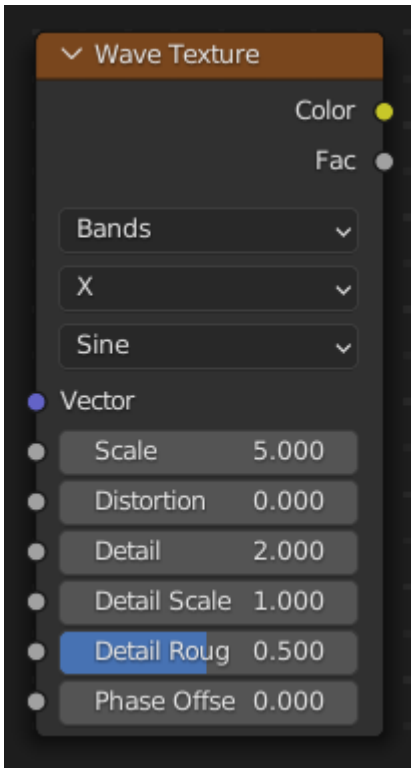


Creating a hammered metal shader using the *Voronoi Texture* node.

Wave Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.



The *Wave Texture* node adds procedural bands or rings with noise distortion.

Hint

In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.

To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion*/*Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale* and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Inputs

Vector

Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Overall texture scale.

Distortion
: Amount of distortion of the wave.

Hint

In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.

To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion*/*Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale*, and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Detail
: Amount of distortion noise detail.

Detail Scale
: Scale of distortion noise.

Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

Phase Offset
: Position of the wave along the *Bands Direction*. This can be used as an input for more control over the distortion.

Properties

Type

Bands or *Rings* shaped waves.

Bands/Rings Direction
: The axis the bands or rings propagate from i.e. which axis they are perpendicular to. When using *Bands* a *Diagonal* axis is an option and when using *Rings* the rings can propagate outwards from a single point by using *Spherical* direction.

Wave Profile
: Controls the look of the wave type.

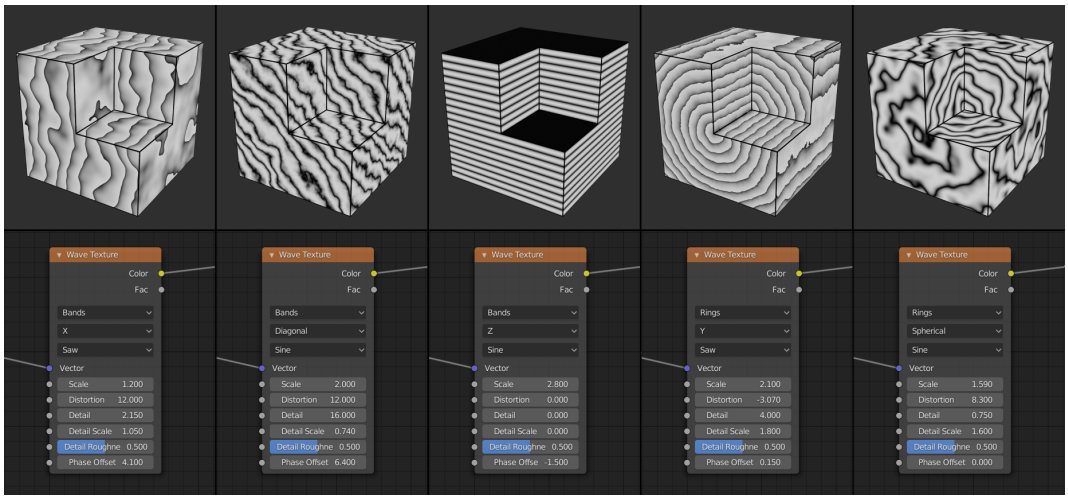
Saw:: Uses a sawtooth profile.
Sine:: Uses the standard sine profile.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

Examples



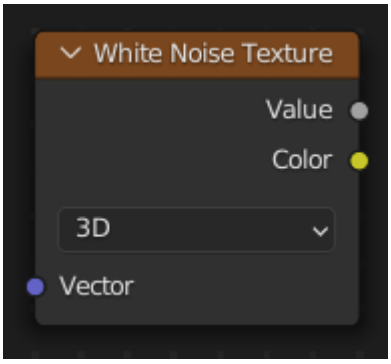
Wave Texture.

White Noise Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *White Noise Texture* node returns a random number based on an input Seed. The seed can be a number, a 2D vector, a 3D vector, or a 4D vector; depending on the *Dimensions* property. The output number ranges between zero and one.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Vector used as seed in 2D, 3D, and 4D dimensions.

W
: Value used as seed in 1D and 4D dimensions.

Properties

Dimensions
: The dimensions of the space to evaluate the noise in.

1D::	The *W* input is used as seed.
2D::	The X and Y components of the *Vector* input are used as seed.
3D::	The *Vector* input is used as seed.
4D::	Both the *Vector* input and the *W* input are used as seed.

Outputs

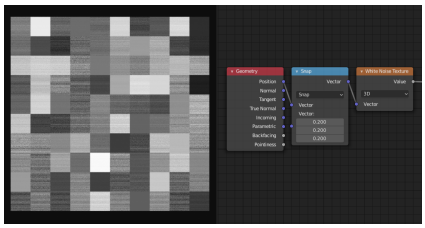
Value
: Output random value.

Color
: Output random color.

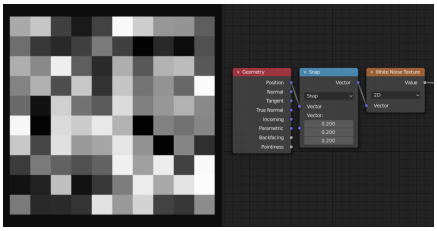
Notes

The slightest difference in seed values would result in completely different outputs. Consequently, bad precision may have significant impact on the output. Usually, we can mitigate this issue by:

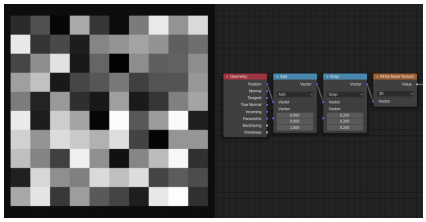
- Eliminating the problematic seed value. If the problematic seed value is constant, it should be eliminated by choosing a lower dimension or multiplying it by zero.
- Adding an arbitrary value to the seed. The issue might only happen at certain boundaries, like unit boundaries, so simply adding an arbitrary value might solve the issue.
- Taking the absolute value of the seed. In computing, zero may be positive or negative, so taking the absolute values unifies the zero into a single value.



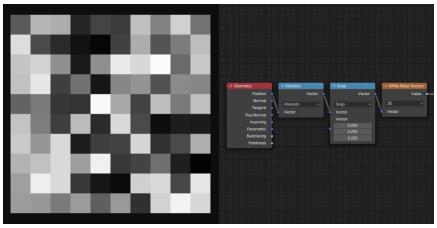
Precision issue due to signed zeros on the Z axis.



Mitigating the issue by eliminating the Z axis.

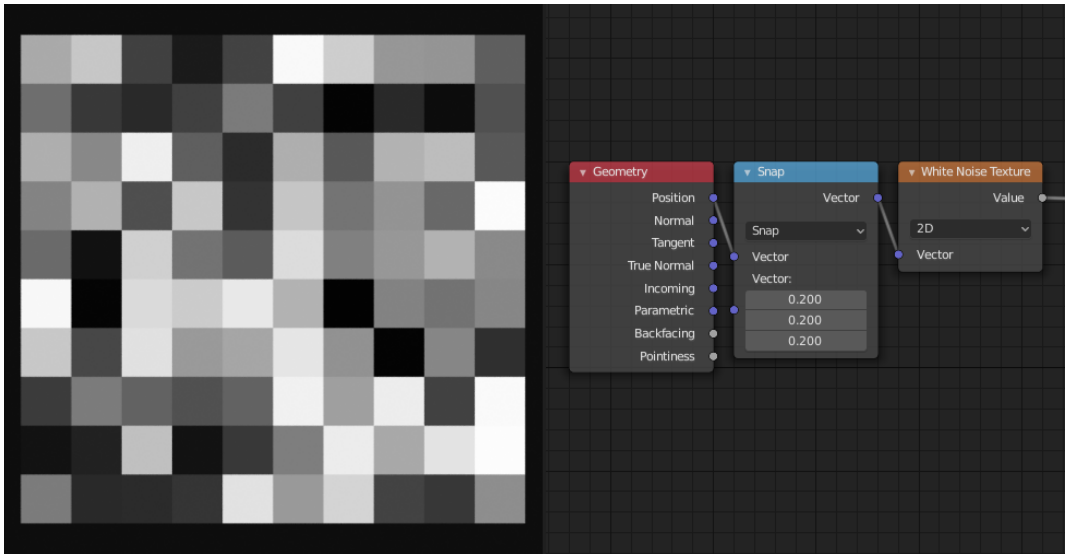


Mitigating the issue by adding an arbitrary value.



Mitigating the issue by taking the absolute value.

Examples



Generating cell noise using the *Snap* vector operation and the *White Noise* node.

Utilities Nodes

General purpose nodes for modifying data.

- Color
  - Blackbody Node
  - Color Ramp Node
  - Combine Color Node
  - Mix Color Node
  - RGB Curves Node
  - Separate Color Node
- Text
  - Join Strings Node
  - Replace String Node
  - Slice String Node
  - Special Characters Node
  - String Length Node
  - String to Curves Node
  - Value to String Node
- Vector
  - Vector Curves Node
  - Vector Math Node
  - Vector Rotate Node
  - Combine XYZ Node
  - Mix Vector Node
  - Separate XYZ Node

- Field
  - Accumulate Field Node
  - Evaluate at Index Node
  - Evaluate on Domain Node
- Math
  - Boolean Math Node
  - Clamp Node

    - Compare Node
    - Float Curve
    - Float To Integer Node
    - Hash Value Node
    - Integer Math Node
    - Map Range Node
    - Math Node
    - Mix Node
- Matrix
    - Combine Matrix Node
    - Combine Transform Node
    - Invert Matrix Node
    - Matrix Determinant Node
    - Multiply Matrices Node
    - Project Point Node
    - Separate Matrix Node
    - Separate Transform Node
    - Transform Direction Node
    - Transform Point Node
    - Transpose Matrix Node
- Rotation
    - Align Rotation to Vector Node
    - Axes to Rotation Node
    - Axis Angle to Rotation Node
    - Euler to Rotation Node
    - Invert Rotation Node
    - Rotate Rotation Node
    - Rotate Vector Node
    - Rotation to Euler Node
    - Rotation to Quaternion Node
    - Quaternion to Rotation Node
- Deprecated
    - Align Euler to Vector Node
    - Rotate Euler Node

- For Each Geometry Element Zone
- Index Switch Node

- Menu Switch Node
- Random Value Node
- Repeat Zone
- Switch Node

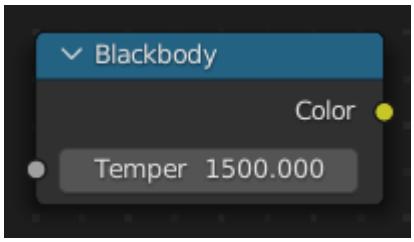
Color Utility Nodes

Nodes for modifying color data passed through color sockets.

- Blackbody Node
- Color Ramp Node
- Combine Color Node
- Mix Color Node
- RGB Curves Node
- Separate Color Node

Blackbody Node

The *Blackbody* node converts a blackbody temperature to RGB value. This can be useful for materials that emit light at natural occurring frequencies.



Inputs

Temperature
: The temperature in Kelvin.

Properties

This node has no properties.

Outputs

Color
: RGB color output.

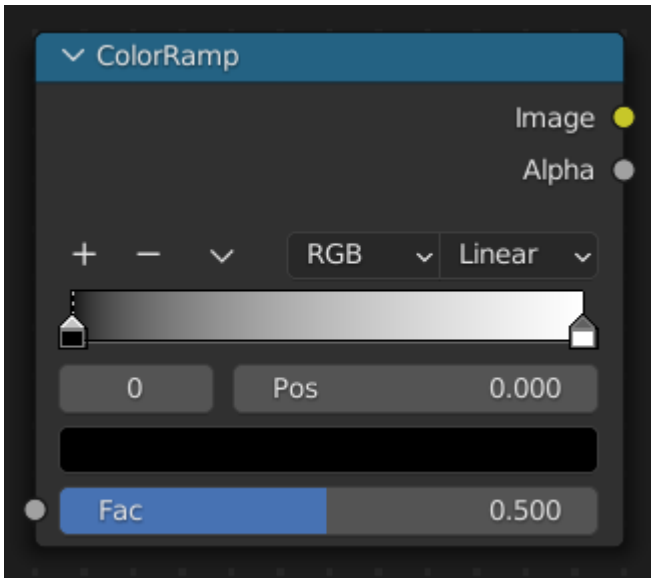
Examples



Example of the color ranges of the Blackbody node.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Inputs

Factor
: The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp
: See Color Ramp Widget.

Outputs

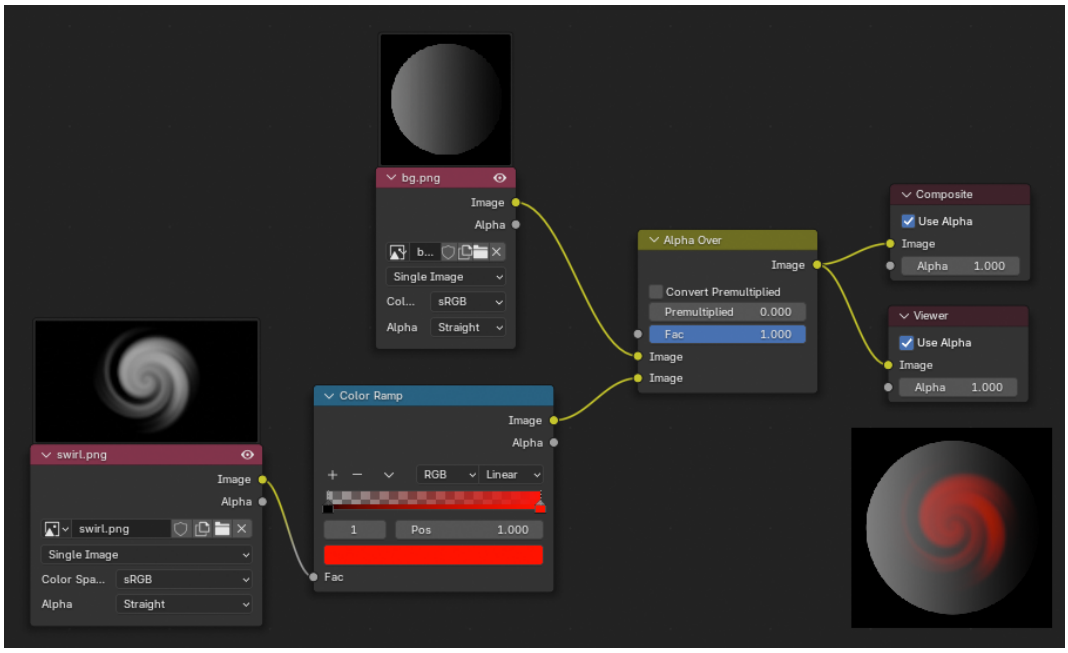
Image/Color
: Standard color output.

Alpha
: Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

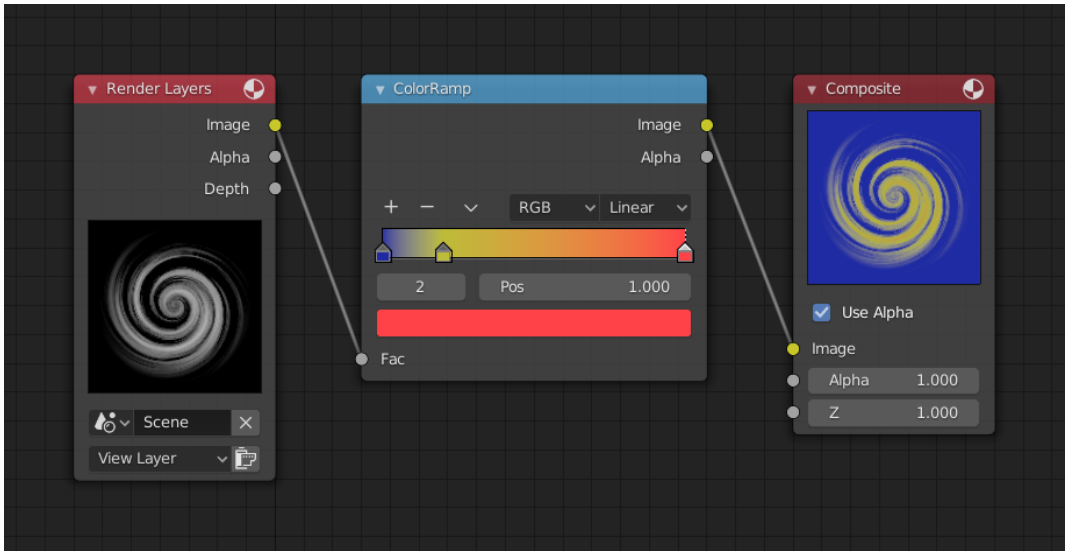


In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

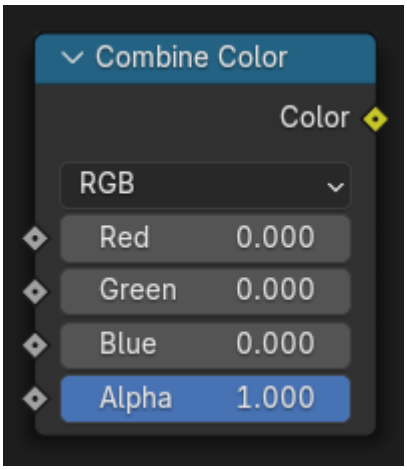
In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.



The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Combine Color Node

Combines four grayscale channels into one color image, based on a particular Color Model.



Inputs

The inputs of this node depend on the Mode property (see below).

Alpha
: The opacity of the output color.

Properties

Mode
: The color model to use.

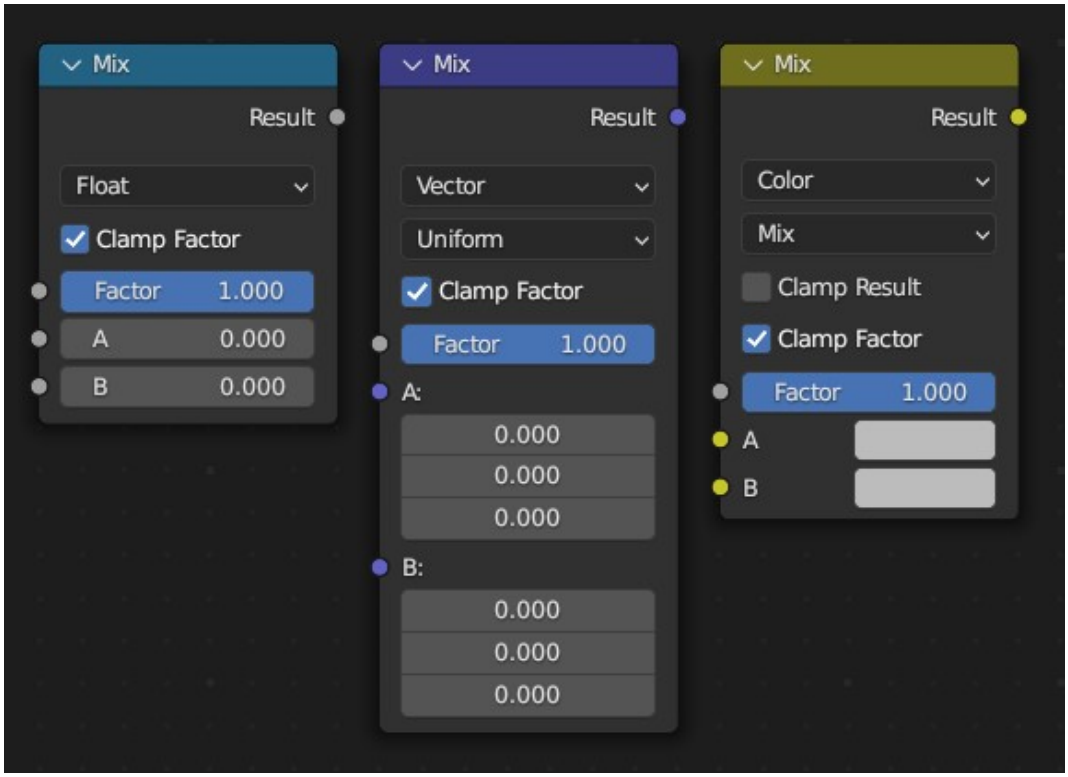
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Output

Color
: Standard color output.

Mix Color Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.



Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
: The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
: The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
: Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
: Limit the Result to the range between 0.0 and 1.0.

Outputs

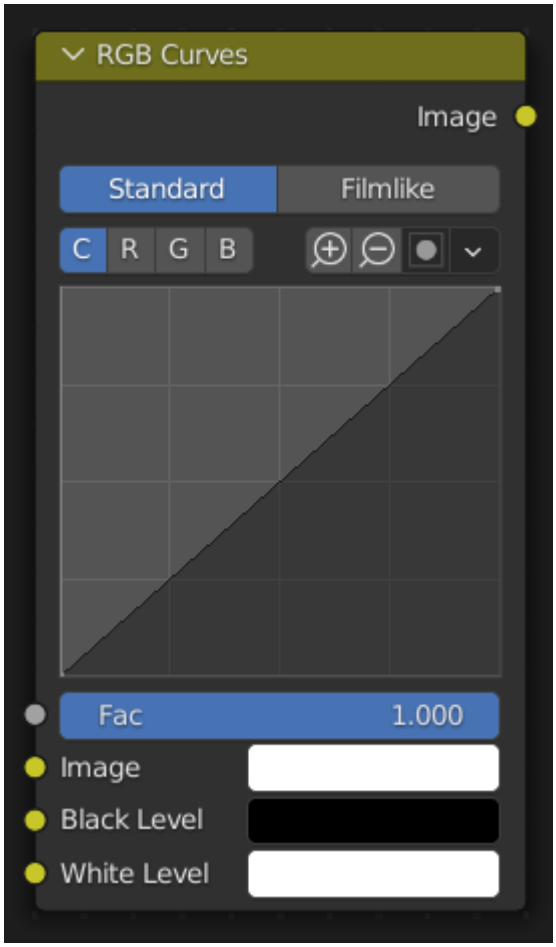
Result
: Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.



Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*
: Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel
: The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve
: A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

Outputs

Image/Color
: Standard color output.

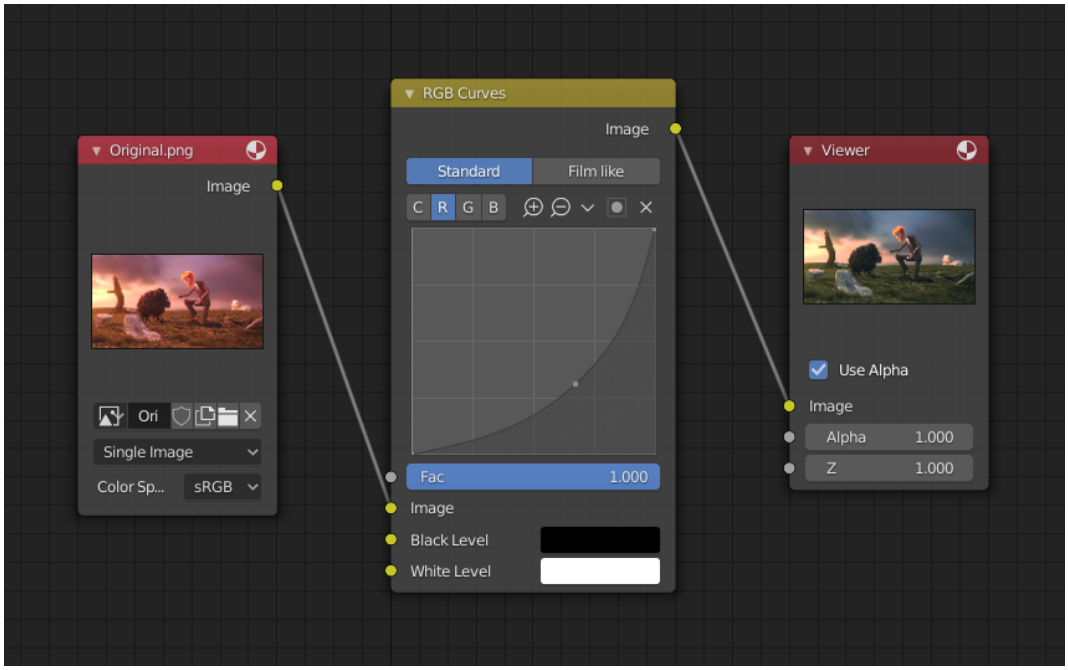
Examples

Below are some common curves you can use to achieve desired effects.



From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

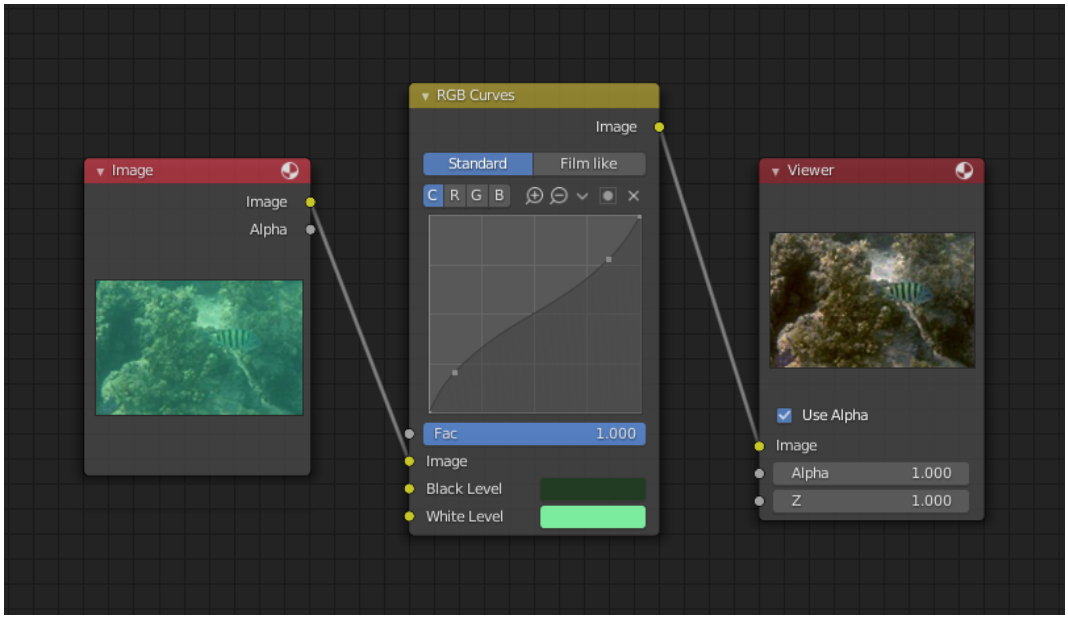


Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



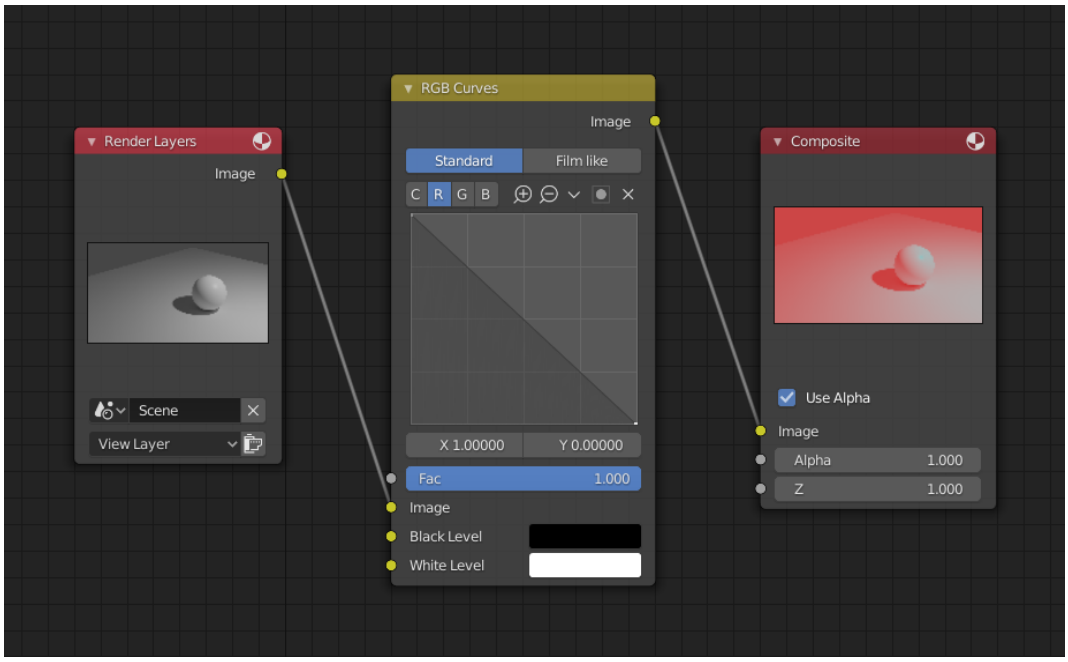
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

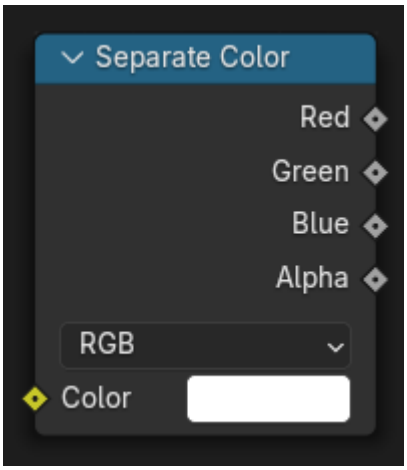
Effects



Changing colors by inverting the red channel.

Separate Color Node

Splits an image into its channels, based on a particular Color Model.



Inputs

Color
: Standard color input.

Properties

Mode
: The color model to output.

RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Outputs

The outputs of this node depend on the Mode property (see above).

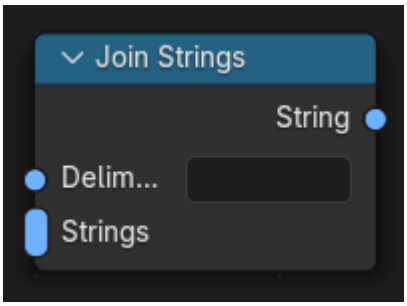
Alpha
: The opacity value.

Text Utility Nodes

Nodes to manipulate strings.

- Join Strings Node
- Replace String Node
- Slice String Node
- Special Characters Node
- String Length Node
- String to Curves Node
- Value to String Node

Join Strings Node



The *Join Strings* node combines any number of input strings into the output string. The order of the result depends on the vertical ordering of the inputs in the multi-input socket.

Tip

This node can be used to create a multi-line string for the String to Curves Node, when combined with the line break output from the Special Characters Node.

Inputs

Delimiter
: String value to place between each concatenated string.

Strings
: Multiple string values to be combined in connection order.

Properties

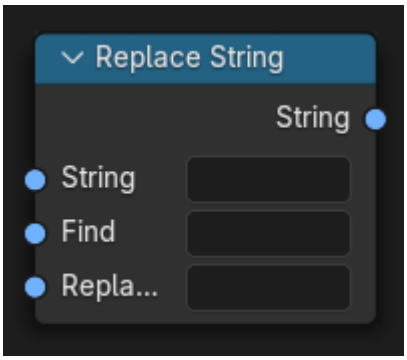
This node has no properties.

Outputs

String
: String result of the concatenation.

Replace String Node

The *Replace String* node replaces a string segment with another.



Inputs

String
: Standard string input.

Find
: The substring to find in *String* to be replaced.

Replace
: A string segment which replaces occurrences of the *Find* substring.

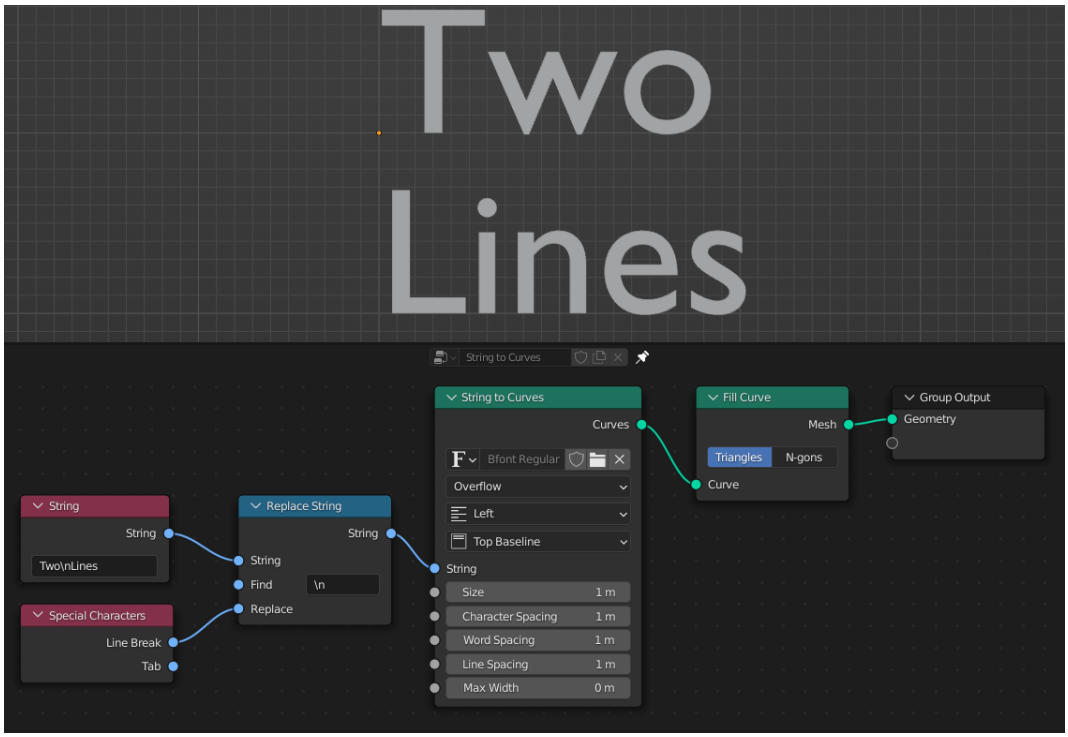
Properties

This node has no properties.

Outputs

String
: Standard string output.

Examples



Using the node to add the newline character to a string.

Slice String Node



The *Slice String* node extracts a string segment from a larger string.

Inputs

String
: String value to be sliced.

Position
: Integer value used to determine the starting point of the new string within the input string. The first letter of the string is at index 0.

Length
: Integer value used to determine how many characters are extracted from the input string.

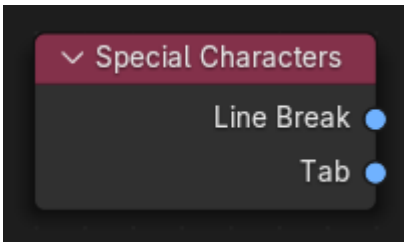
Properties

This node has no properties.

Outputs

String
: String value of the extracted substring.

Special Characters Node



The *Special Characters* node is used to output string characters that can't be typed directly with the keyboard.

Tip

This node can be used to create a multi-line string for the String to Curves Node, when combined with the Join Strings Node or the Replace String Node.

Inputs

This node has no inputs.

Properties

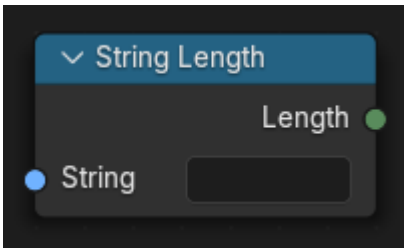
This node has no properties.

Outputs

Line Break
: A new line character (escape character `\n`).

Tab
: A tab character used to add an indentation in the output.

String Length Node



The *String Length* node outputs the number of characters in the input string.

Inputs

String
: String value to be evaluated.

Properties

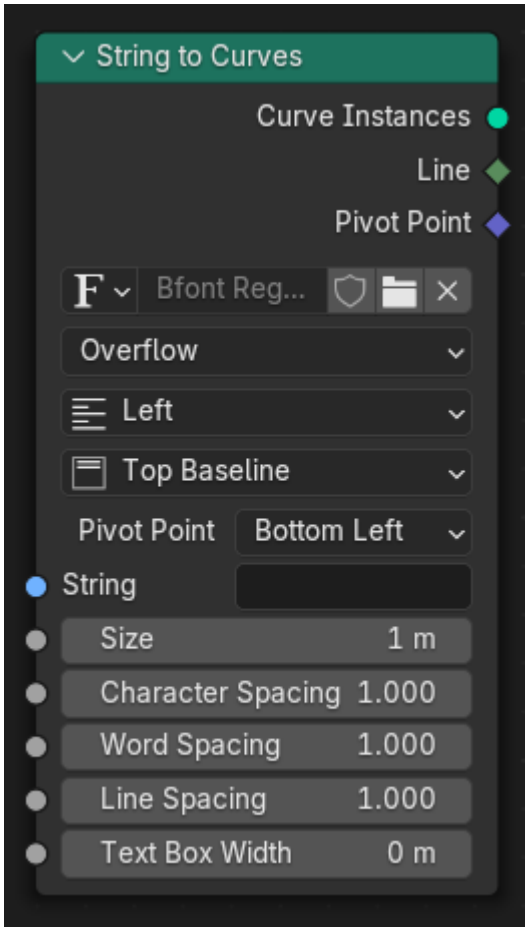
This node has no properties.

Outputs

Length
: Integer value representing the length of the input string.

String to Curves Node

The *String to Curves* converts a string to curve instances. Each unique character used in the string is converted to a curve once, and further uses of that character are more instances of the same geometry.



This makes processing the output geometry very efficient, because each unique character only has to be processed once. However, it means that the result will be the same for every instance of the same character. To process each character individually, the Realize Instances Node can be used.

Tip

Socket inspection can be used to see the value of the string input used when the node was evaluated, by holding the mouse over the socket.

Inputs

String
: Standard string input.

Size
: The size of each character. The values of the other inputs are scaled by this value.

Character Spacing
: A factor by which the space between each character (kerning) is scaled on the X axis.

Word Spacing
: A factor by which whitespace between words is scaled on the X axis.

Line Spacing
: The distance between separate lines in the output. Scaled by the *Size* input.

Text Box Width
: The maximum width of each line, though individual words will not be wrapped.

Text Box Height
: The maximum height for all the lines of the text.

Properties

Font
: Font glyph used to generate the curve.

Overflow

Overflow::	Wraps the text at the *Text Box Width*.
Scale To Fit::	Scales the text size to fit the *Text Box Width* and *Text Box Height*.
Truncate::	Only outputs text characters that fit within the width and height, based on the *Size* input. Any part of the string that did not fit is moved to the *Remainder* output.

Alignment

Left::	Aligns the text to the left.
Center::	Aligns the text to the center.
Right::	Aligns the text to the right.
Justify::	Aligns the text to the left and right.
Flush::	

Aligns the text to the left and right with equal character spacing.

Align Y

Top:: Aligns the text to the top.

Top Baseline:: Aligns the text to the top baseline.

Middle:: Aligns the text to the middle.

Bottom Baseline:: Aligns the text to the bottom baseline.

Bottom:: Aligns the text to the bottom.

Pivot Point

Controls where on each character the output *Pivot Point* is placed.

Midpoint:: Place the pivot points at the center of each character's bounds.

Top Left:: Place the pivot points at the top left of each character's bounds.

Top Center:: Place the pivot points at the middle of the top of each character's bounds.

Top Right:: Place the pivot points at the top right of each character's bounds.

Bottom Left:: Place the pivot points at the bottom left of each character's bounds.

Bottom Center:: Place the pivot points at the middle of bottom of each character's bounds.

Bottom Right:: Place the pivot points at the bottom right of each character's bounds.

Outputs

Curve Instances

Curve instances geometry.

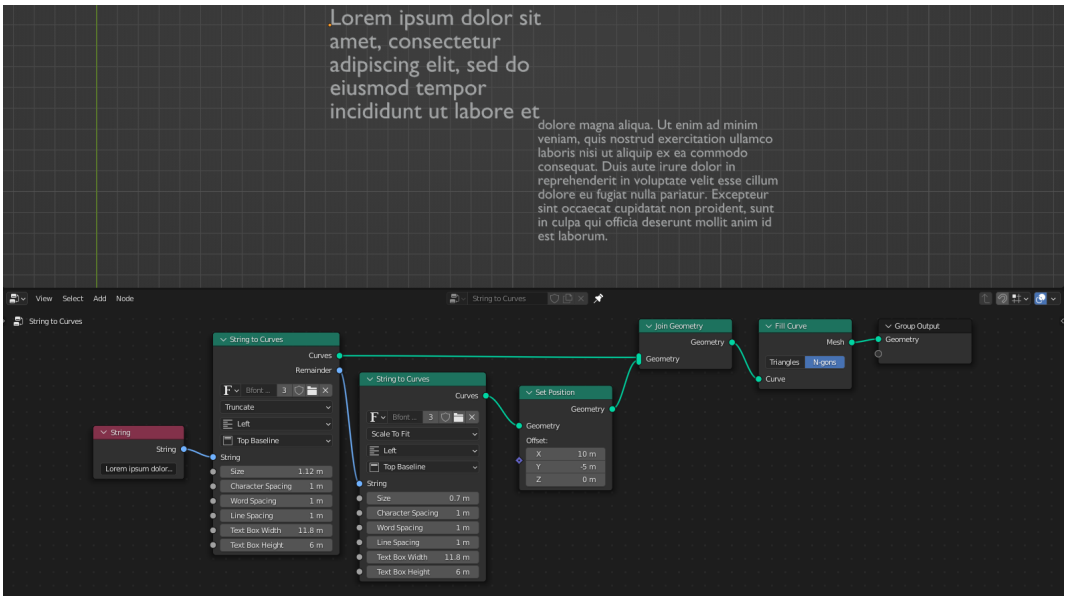
Remainder

The part of the text that did not fit in the box described by the *Text Box Height* and *Text Box Width* inputs. Only used in the *Truncate* overflow mode.

Line
: An attribute field containing the line index of each character (on the instance domain).

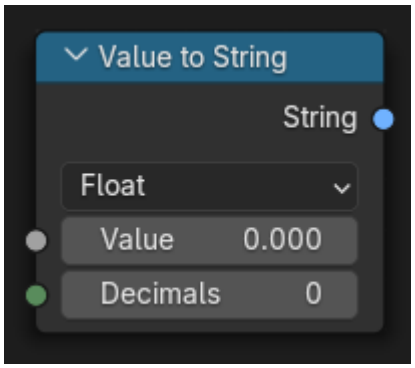
Pivot Point
: Outputs the position described by the *Pivot Point* drop-down in the local space of each instance.

Examples



The node can be used to make overflowing text boxes. Here, the text that does not fit into the first node's fix-sized text box is passed to a separate *String to Curves* node. And finally added with a *Scale to Fit* node.

Value to String Node



The *Value to String* node generates string representation of the input value.

Inputs

Value
: Floating-point value to be converted.

Decimals *Float Data Type*
: Integer value used to determine the precision of the output value.

Properties

Data Type
: The type of numerical value to convert to a string.

Float::	Convert a floating-point value to a string.
Integer::	Convert a 32-bit integer to a string.

Outputs

String
: String value representation of the input.

Vector Utility Nodes

Nodes for modifying vector quantities.

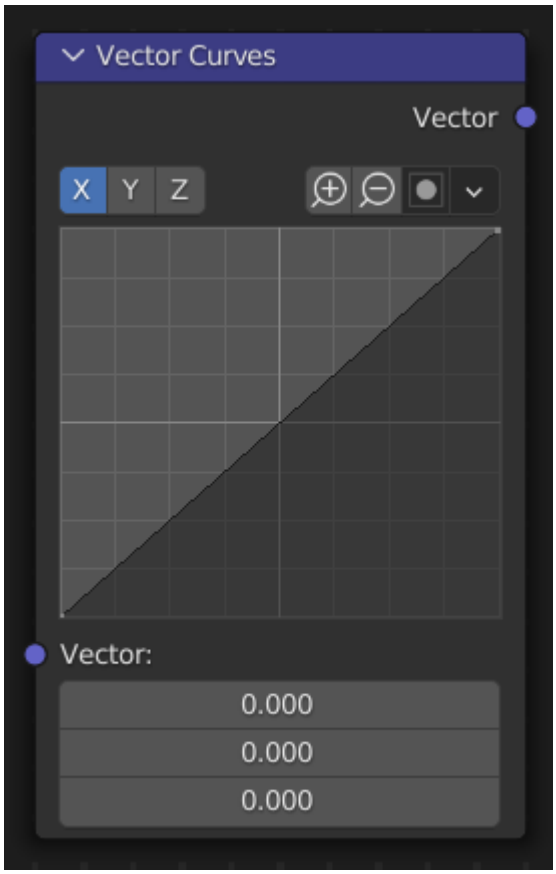
- Vector Curves Node
- Vector Math Node
- Vector Rotate Node

- Combine XYZ Node
- Mix Vector Node
- Separate XYZ Node

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.



Inputs

In the shader context the node also has an additional Factor property.

Factor
: Controls the amount of influence the node exerts on the output vector.

Vector
: Standard vector input.

Properties

Channel
: X, Y, Z

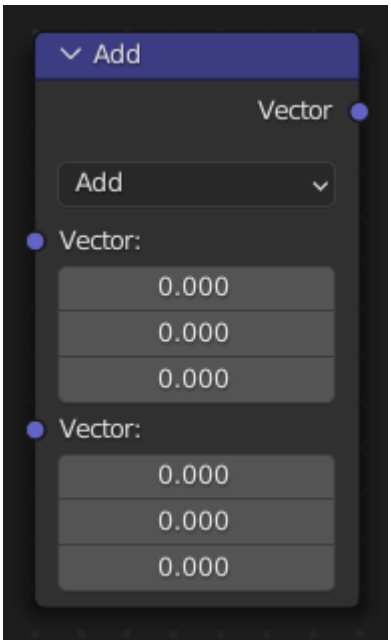
Curve
: For the curve controls see: Curve widget.

Outputs

Vector
: Standard vector output.

Vector Math Node

The *Vector Math* node performs the selected math operation on the input vectors.



Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Scale* input is only available in the *Scale* operator.

Vector
: Input vector $A = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$.

Vector
: Input vector $B = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$.

Scale
: Input Scale s.

Properties

Operation

The vector math operator to be applied on the input vectors.

Add:: The sum of A and B. $\begin{pmatrix} A_x + B_x \\ A_y + B_y \\ A_z + B_z \end{pmatrix}$

Subtract:: The difference between A and B. $\begin{pmatrix} A_x - B_x \\ A_y - B_y \\ A_z - B_z \end{pmatrix}$

Multiply:: The entrywise product of A and B. $\begin{pmatrix} A_x \cdot B_x \\ A_y \cdot B_y \\ A_z \cdot B_z \end{pmatrix}$

Divide:: The entrywise division of A by B. Division by zero results in zero. $\begin{pmatrix} A_x / B_x \\ A_y / B_y \\ A_z / B_z \end{pmatrix}$

Multiply Add:: The entrywise combination of the multiply and addition operations. $A \times B + C$

Cross Product:: The cross product of A and B. $\begin{pmatrix} A_y \cdot B_z - A_z \cdot B_y \\ A_z \cdot B_x - A_x \cdot B_z \\ A_x \cdot B_y - A_y \cdot B_x \end{pmatrix}$

Project:: The projection of A onto B.

Reflect:: The reflection of A around the normal B. B need not be normalized.

Refract:: For a given incident vector A, surface normal B and ratio of indices of refraction (IOR), refract outputs the refraction vector R.

Faceforward:: Orients a vector A to point away from a surface B as defined by its normal C. Computes $((dot(B, C) < 0) ? A : -A)$.

Dot Product:: The dot product of A and B. $A_x \cdot B_x + A_y \cdot B_y + A_z \cdot B_z$

Distance:: The distance between A and B.

Length:: The length of A. $\sqrt{A_x^2 + A_y^2 + A_z^2}$

Scale:: The result of multiplying A by the scalar input *Scale*. $\begin{pmatrix} s \cdot A_x \\ s \cdot A_y \\ s \cdot A_z \end{pmatrix}$

Normalize::	The result of normalizing A. The result vector points to the same direction as A and has a length of 1. If A is (0, 0, 0), the result is (0, 0, 0) as well.
Wrap::	The entrywise output of a value between Min and Max based on the absolute difference between the input value and the nearest integer multiple of Max less than the value.
Snap::	The result of rounding A to the largest integer multiple of B less than or equal A.
Floor::	Rounds the input value entrywise down to the nearest integer.
Ceil::	Rounds the input value entrywise up to the nearest integer.
Modulo::	The entrywise modulo of A by B.
Fraction::	Returns the fractional part of the *value* entrywise.
Absolute::	The entrywise absolute value of A.
Minimum::	The entrywise minimum value from A and B.
Maximum::	The entrywise maximum value from A and B.
Sine::	The entrywise Sine of A.
Cosine::	The entrywise Cosine of A.
Tangent::	The entrywise Tangent of A.

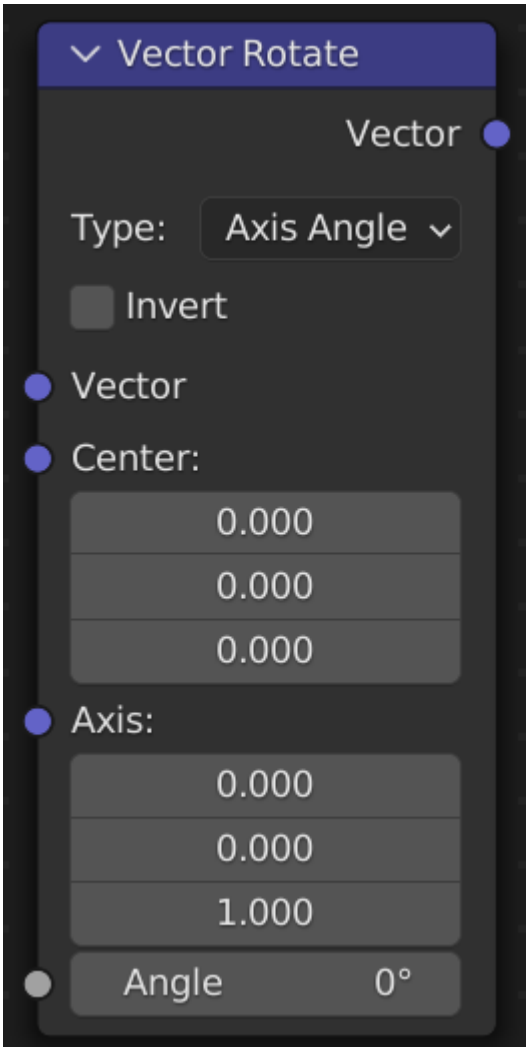
Outputs

The output of the node is dynamic. It is either a vector or a scalar depending on the operator. For instance, the *Length* operator has a scalar output while the *Add* operator has a vector output.

Vector
: Output vector.

Value
: Output value.

Vector Rotate Node



The *Vector Rotate Node* provides the ability to rotate a vector around a pivot point (*Center*).

Inputs

Vector
: Vector to be rotated.

Center

Point to rotate around.

Axis
: Axis to rotate around.

Angle
: Angle to rotate the input vector by.

Rotation
: When *Type* is set to *Euler*, rotate the input vector by these angles around the X, Y, then Z axes in that order.

Properties

Type
: The type of angle input.

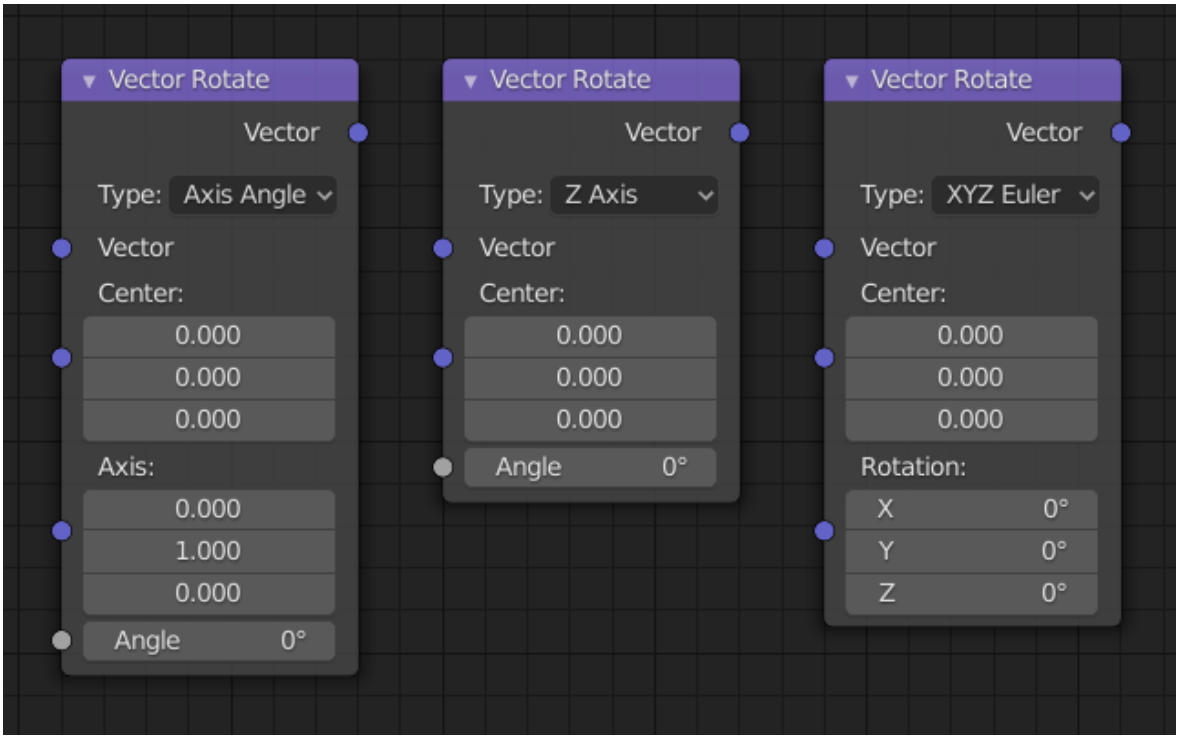
X/Y/Z Axis::	Rotates the vector around the defined axis and the amount of rotation is defined by the *Angle* input.
Axis Angle::	Rotates the vector around an arbitrary axis defined by the *Axis* input vector. The amount of rotation is defined by the *Angle* input.
Euler::	Rotates the vector about a center point defined by the *Center* input vector. The amount of rotation on each axis is defined by the *Rotation* input vector.

Invert
: Inverts the rotation angle.

Outputs

Vector
: The rotated vector.

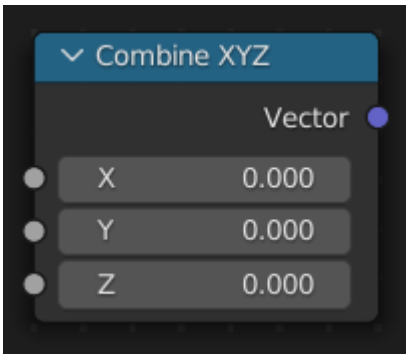
Examples



Vector Rotate node example.

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.



Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

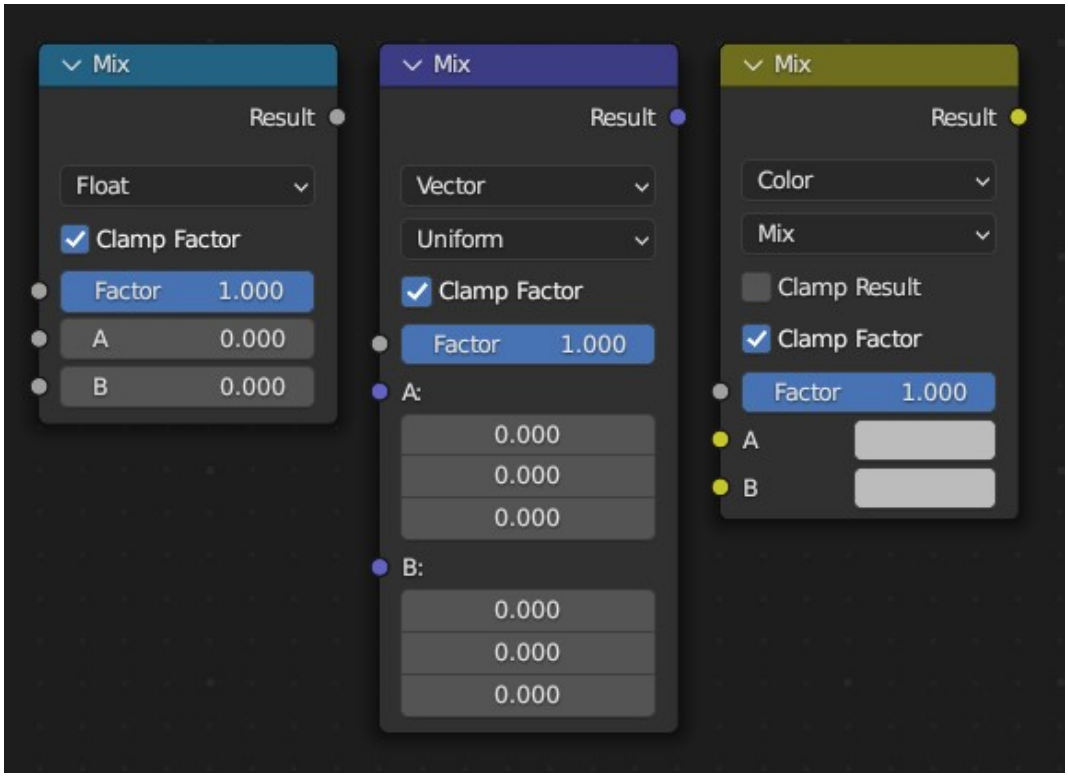
Vector
: Standard vector output.

Note

The vector is not normalized.

Mix Vector Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.



Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
: The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
: The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
: Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
: Limit the Result to the range between 0.0 and 1.0.

Outputs

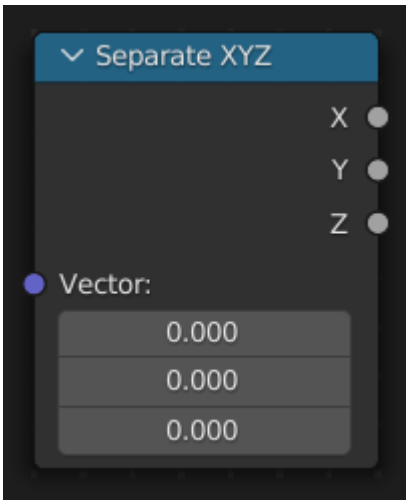
Result
: Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

Separate XYZ Node

The *Separate XYZ Node* splits a vector into its individual components.



Input

Vector
: Standard vector input.

Properties

This node has no properties.

Outputs

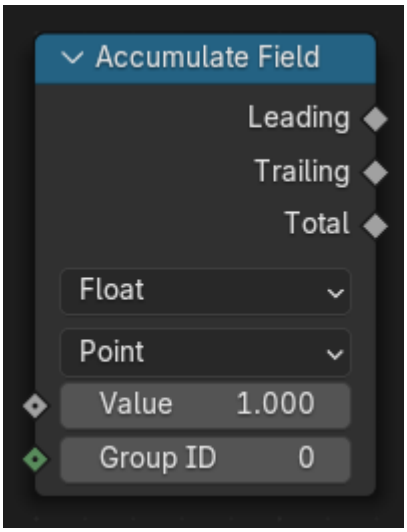
- X
- Y
- Z

Field Utility Nodes

- Accumulate Field Node
- Evaluate at Index Node
- Evaluate on Domain Node

Accumulate Field Node

The *Accumulate Field* node counts a running total of its input values, in the order defined by the geometry's indices. The node's essential operation is just addition, but instead of only outputting the final total, it outputs the current value at every element.



Inputs

Value
: The values to be accumulated.

> **Warning**
>
> When accumulating integer values, be careful to make sure that there are not too many large values. The maximum integer that Blender stores internally is around 2 billion. After that, values may wrap around and become negative. See wikipedia for more information.

Group Index
: An index used to group values together for multiple separate accumulations. This can be thought of as a choice of the "bin" in which to place each value. This input has no effect when it is only a single value.

Properties

Data Type

Float:: The node will accumulate a *Float* field.
Integer:: The node will accumulate an *Integer* field.
Vector:: The node will accumulate a *Vector* field.
Transform:: The node will accumulate a *Matrix* field.

Domain

The attribute domain used for accumulation and for evaluation of the *Value* input.

Output

Leading

The running total of values in the corresponding group, starting at the first value.

Trailing

The running total of values in the corresponding group, starting at zero.

Total

The total of all the values in the corresponding group.

Examples

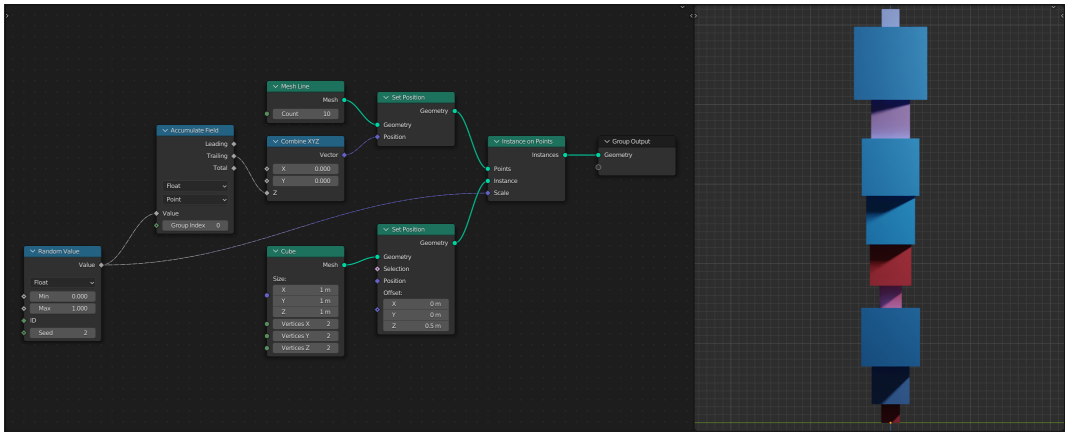
Table

Value	Group Index	Leading	Trailing	Total
1	7	1	0	6
3	7	4	1	6
2	7	6	4	6

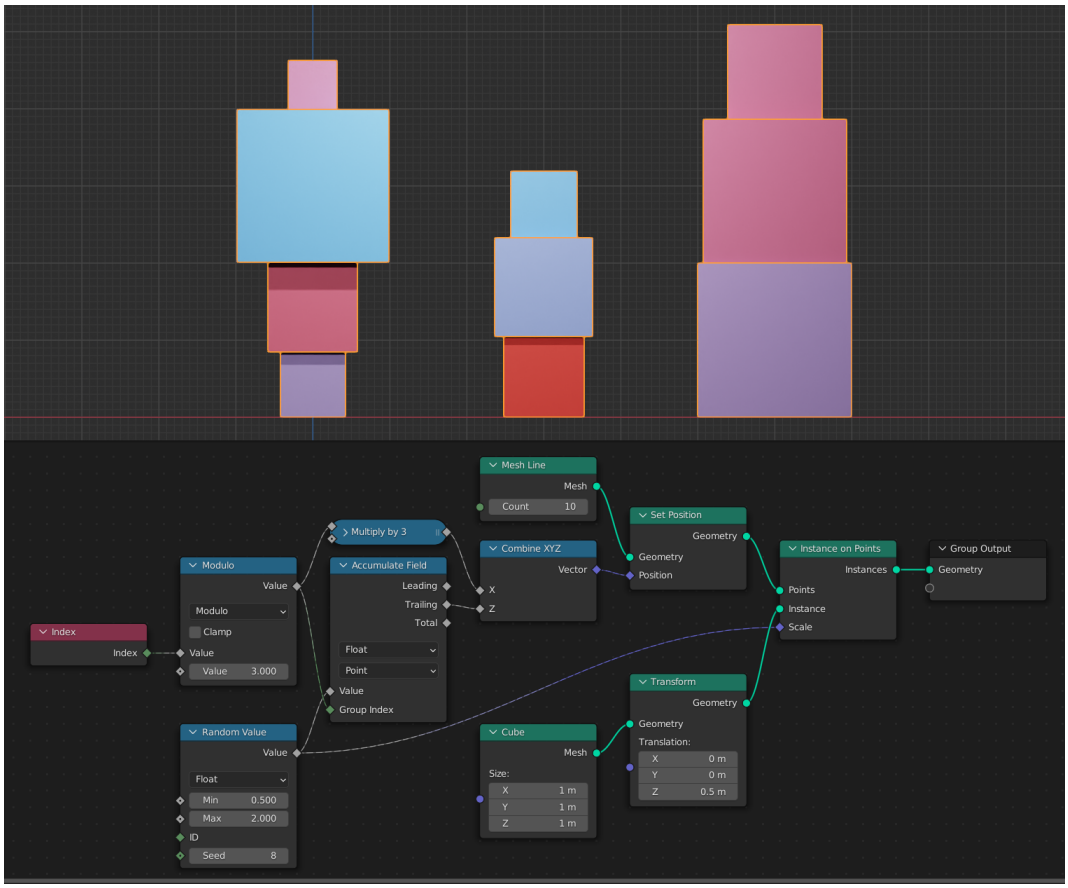
Value	Group Index	Leading	Trailing	Total
1	3	1	0	3
0	3	1	1	3
2	3	3	1	3

A few examples of input values and the node's results. One important take-away from this table is that the specific values for the *Group Input* do not matter; it only matters that the values are shared between elements.

Stacking Boxes

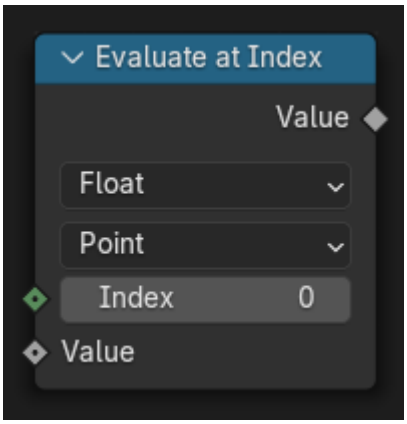


Here, the node is used in combination with the Random Value Node to create a stack of randomly scaled boxes. The *Group Index* input is not used, because all boxes are meant to be in the same stack.



A slightly more complicated version of the previous example, using the *Group Index* input to create three separate stacks.

Evaluate at Index Node



The *Evaluate at Index* node allows accessing data of other elements in the context geometry. It is similar to the Sample Index Node. The main difference is that this node does not require a geometry input, because the geometry from the field context is used.

This node is also similar to the Evaluate on Domain Node node, except that the value to retrieve from the specified domain is specified by an index rather than an automatic domain interpolation.

Inputs

Index
: The index of the element in the selected domain to retrieve data from, i.e. “the fourth face”, or “the first control point”.

Value
: The field to retrieve data from.

Properties

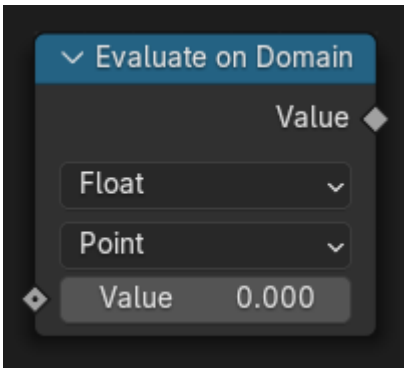
Domain
: The attribute domain used for evaluation of the *Value* input. This is useful because it can be a different domain than the domain from the field context, i.e. to choose a vertex for each face.

Output

Value

The value of the input field at the given index.

Evaluate on Domain Node



The *Evaluate on Domain* allows evaluating a field for a different attribute domain than the domain from the field context. For example, the face index could be used instead of the face corner index, when setting the values of a UV Map

Note

This node is not necessary to retrieve data from other attribute domains; that is done automatically. Its utility comes from the fact that it's possible to control *when* the domain interpolation happens. Normally, input nodes interpolate their data to the current context's domain as soon as they create their output.

Tip

It may be preferable to use this node over the Capture Attribute Node, since it allows using a specific attribute domain without requiring a geometry socket input, which allows creating more reusable node groups.

See also

The method of retrieving data from another domain is somewhat similar to the Evaluate at Index Node.

Inputs

Value

The field to evaluate on the chosen attribute domain.

Properties

Domain
: The attribute domain used for evaluation of the *Value* input.

Output

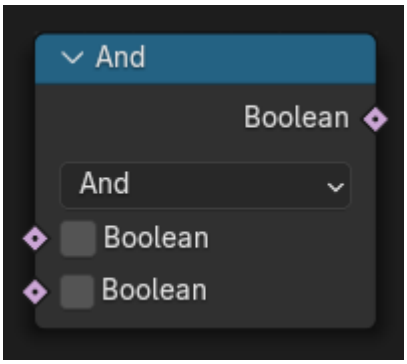
Value
: The values from the input, evaluated on the chosen domain, then interpolated to the domain from the field context.

Math Utility Nodes

- Boolean Math Node
- Clamp Node
- Compare Node
- Float Curve
- Float To Integer Node
- Hash Value Node
- Integer Math Node
- Map Range Node
- Math Node
- Mix Node

Boolean Math Node

The *Boolean Math* node performs a basic logical operation on its inputs.



Inputs

Boolean
: Two standard Boolean inputs.

Properties

Mode

And:: True when both inputs are true. (AND)

Or:: True when at least one input is true. (OR)

Not:: Opposite of the input. (NOT)

Not And:: (True when at least one input is false. NAND)

Nor:: True when both inputs are false. (NOR)

Equal:: True when both inputs are equal. Also known as “exclusive nor”. (XNOR)

Not Equal:: (XOR) True when both inputs are different. Also known as “exclusive or”.

Imply:: True unless the first input is true and the second is false. (IMPLY)

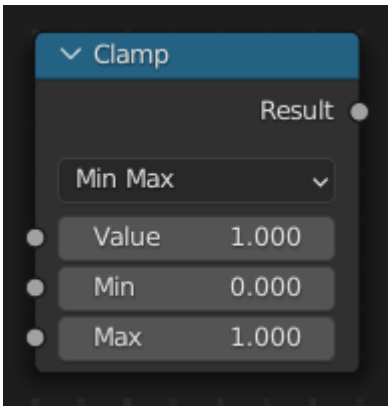
Subtract:: True when the first input is true and the second is false. Also known as “”not imply”. (NIMPLY)

Output

Boolean
: Standard Boolean output.

Clamp Node

The *Clamp* node clamps a value between a minimum and a maximum.



Inputs

Value
: The input value to be clamped.

Min
: The minimum value.

Max
: The maximum value.

Properties

Clamp Type
: Method to clamp.

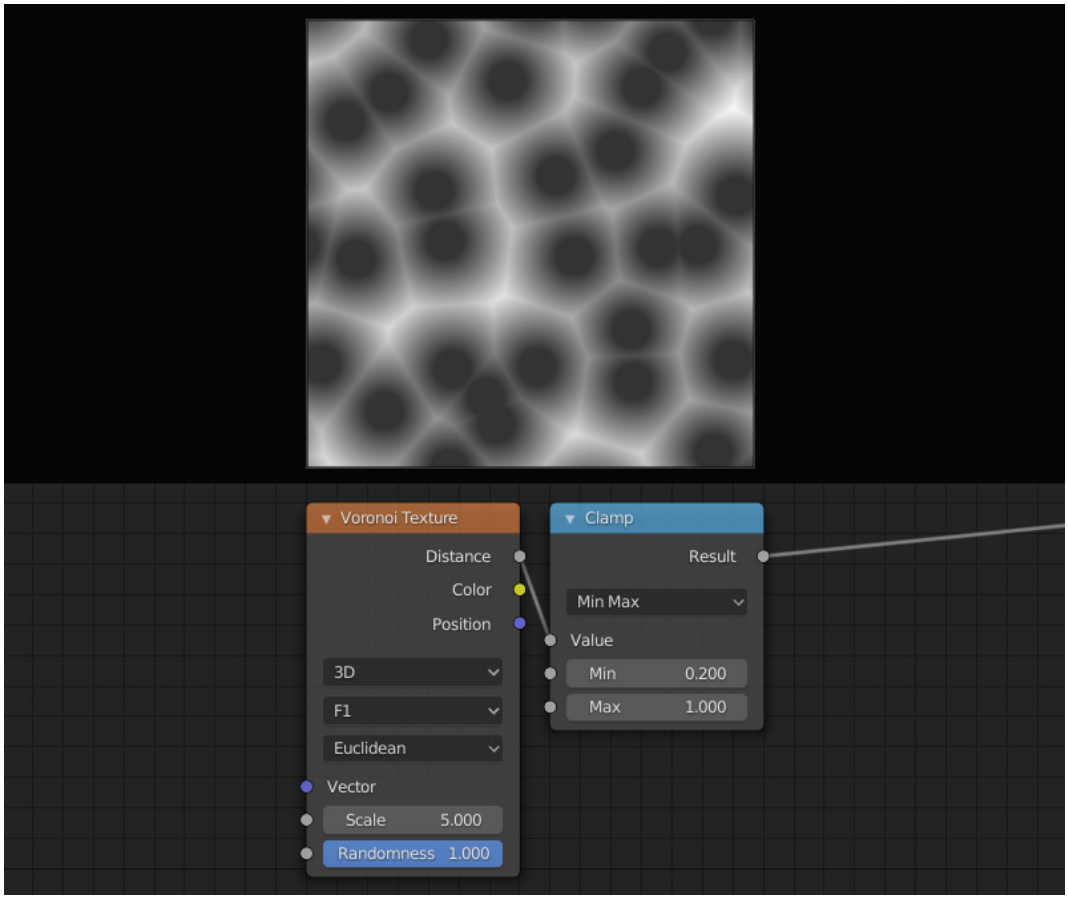
Min Max::	Constrain values between Min and Max.
Range::	Constrain values between Min and Max. When Min is greater than Max, constrain between Max and Min instead.

Outputs

Result
: The input value after clamping.

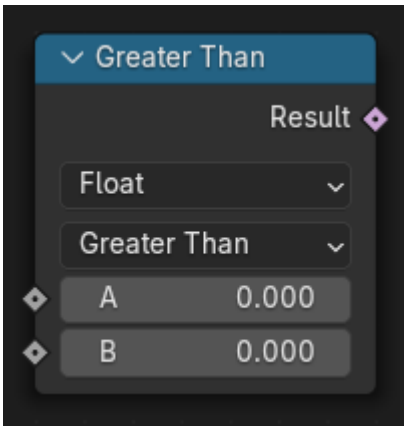
Examples

The *Voronoi Texture* node outputs a value whose minimum is zero. We can use the *Clamp* node to clamp this value such that the minimum is 0.2.



Example of Clamp node.

Compare Node



The *Compare* node takes two inputs and does an operation to determine whether they are similar. The node can work on all generic data types, and has modes for vectors that contain more complex comparisons, which can help to reduce the number of necessary nodes, and make a node tree more readable.

Inputs

A, B
: Standard value inputs of the selected type.

C
: Compared against the dot product of two input vectors in when the *Mode* property is set to *Dot Product*.

Epsilon
: This value is used as a threshold for still considering the two inputs as equal for the Equal and Not Equal operations.

Properties

Mode

Element-Wise:: Compare each axis of the input vectors separately, and output true only when the result is true for each axis.

Length:: Compare the length of the two input vectors.

Average:: Compare the average of the elements of the input vectors. This is the same as the implicit conversion used when setting the node's data type to *Float*.

Dot Product:: Compare the dot product of the two vectors with the separate *C* input, using the selected operation. The dot product outputs a single value that says how much the two vectors "agree".

Direction:: Compare the angle between the two vectors with the separate *Angle* input, using the selected operation. The vectors are normalized, so their length does not matter.

Operation

Less Than:: True when the first input is smaller than second input.

Less Than or Equal:: True when the first input is smaller than the second input or equal.

Greater Than:: True when the first input is greater than the second input.

Greater Than or Equal:: True when the first input is greater than the second input or equal.

Equal:: True when both the difference between the two inputs is smaller than the *Epsilon* input.

Not Equal:: True when both the difference between the two inputs is larger than the *Epsilon* input.

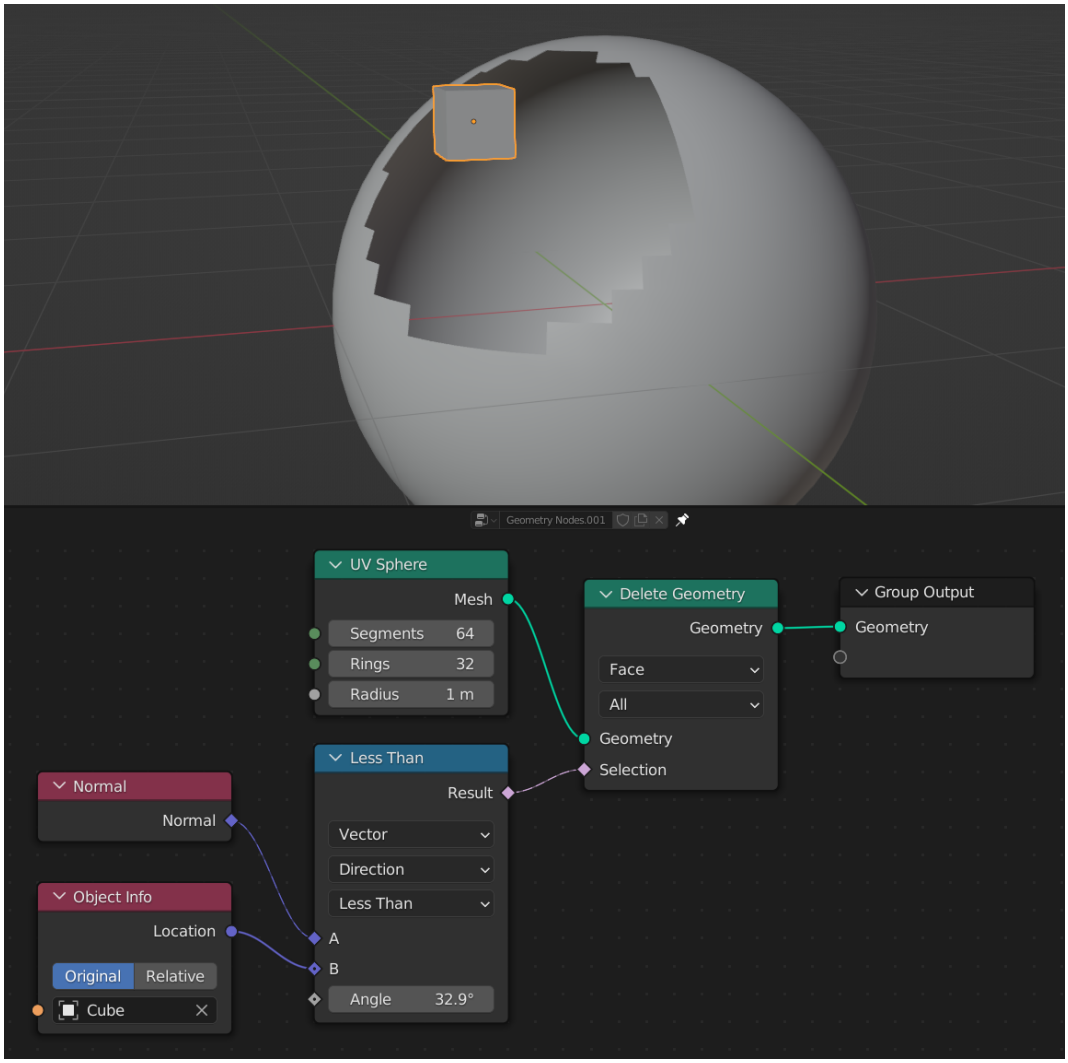
Brighter:: True when the first color input is brighter than the second.

Darker:: True when the first color input is darker than the second.

Output

Result
: Standard Boolean output.

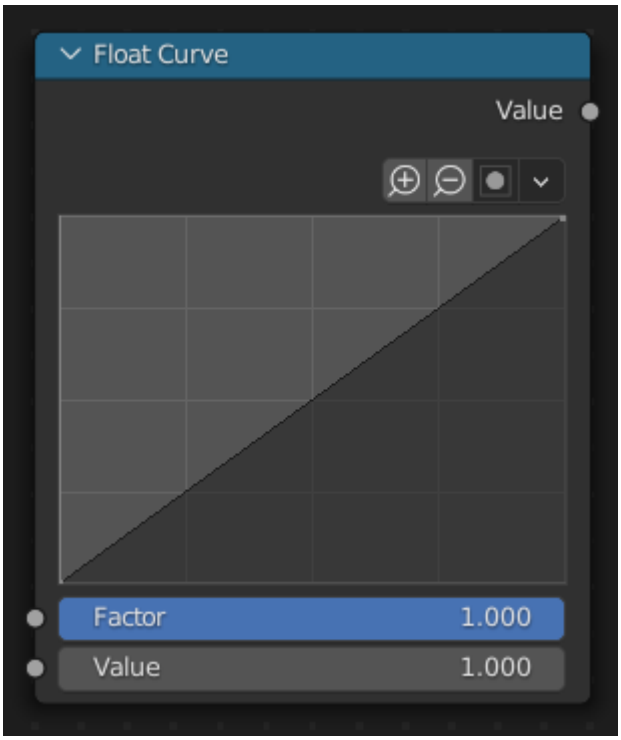
Examples



Here, the compare node is used with the *Direction* mode to compare the direction of the sphere's face normals to the "direction" of the cube object's location. Anywhere that the directions are less than 32.9 degrees apart, the faces will be selected, and deleted.

Float Curve

The Float Curve node maps an input float to a curve and outputs a float value.



Inputs

Factor
: Controls the amount of influence the node exerts on the output value.

Value
: Standard float input.

Properties

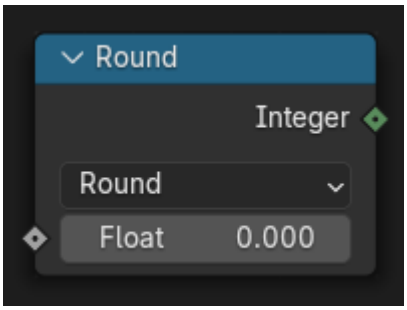
Curve
: For the curve controls see: Curve widget.

Outputs

Float
: Standard float output.

Float To Integer Node

The *Float To Integer* node takes a single floating point number input and converts it to an integer with a choice of methods.



Inputs

Float
: Standard float value input.

Properties

Rounding Mode

Round:: Outputs the closest integer to Float, rounding either up or down based on the value.

Floor:: Outputs the closest integer less than Float, always rounding down.

Ceiling:: Outputs the closest integer greater than Float, always rounding up.

Truncate:: Outputs the closest integer between Float and zero. For positive numbers, acts like Floor. For negative numbers, acts as Ceiling.

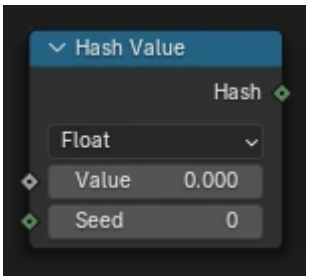
Output

Result
: Standard integer output.

Examples

Input Value	Round	Floor	Ceiling	Truncate
-69.6574	-70	-70	-69	-69
-3.14159	-3	-4	-3	-3
-1.5	-2	-2	-1	-1
1.5	2	1	2	1
3.14159	3	3	4	3
69.6574	70	69	70	69

Hash Value Node



The *Hash Value* node takes a value input and hashes this to an integer.

Important

Hashes cannot be relied upon to be used as unique identifiers because they are not guaranteed to be unique. It can be used to generate somewhat stable randomness especially in cases where *White Noise* does not offer enough flexibility.

Inputs

Value
: Value input determined by *Data Type* property.

Seed
: Integer input used to generate different Hashes .

Properties

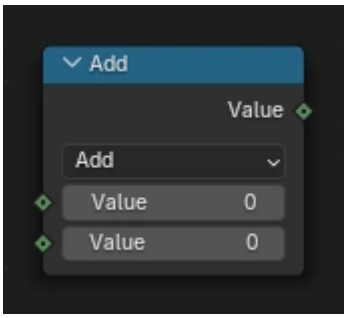
Data Type
: The data type that is used for the *Value* input. The node supports float, integer, vector, color, boolean, rotation, matrix and string data types.

Output

Hash
: Standard integer output.

Integer Math Node

The *Integer Math* node performs math operations.



Inputs

Value
: Standard integer value input. Depending on the operation there will be more than one input.

Properties

Operation
: The mathematical operator to be applied to the input values:

  Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Absolute::	The input value is read without regard to its sign. This turns negative values into positive values.
Negate::	Changes the sign of the input value.
Power::	The *Base* raised to the power of *Exponent*.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Sign:: Extracts the sign of the input value. All positive numbers will output 1. All negative numbers will output -1. And 0 will output 0.

Rounding

Divide Round:: Divide the values and round the result toward zero.

Divide Floor:: Divide the values and floor the result down to the nearest integer.

Divide Ceiling:: Divide the values and ceil the result up to the nearest integer .

Floored Modulo:: Returns the positive remainder of a division operation.

Modulo:: Outputs the remainder once the first value is divided by the second value.

Greatest Common Divisor:: The largest positive integer that divides into each of the values.

Least Common Multiple:: The smallest positive integer that is divisible by both values.

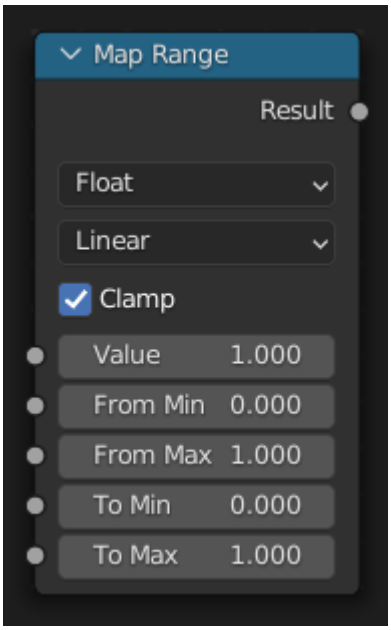
Output

Value

Standard integer output.

Map Range Node

The *Map Range* node remaps a value from a range to a target range.



Inputs

Value/Vector
: The input value or vector to be remapped.

From Min
: The lower bound of the range to remap from.

From Max
: The higher bound of the range to remap from.

To Min
: The lower bound of the target range.

To Max
: The higher bound of the target range.

Steps

The number of values allowed between *To Min* and *To Max* when using *Stepped Linear* interpolation. A higher value will give a smoother interpolation while lower values will progressively quantize the input.

Properties

Data Type
: Map Range supports both Float and Vector data types. Changing the data type will also update the sockets to reflect the data type chosen.

Interpolation Type
: The mathematical method used to transition between gaps in the numerical inputs.

Linear::	Linear interpolation between From Min and From Max values.
Stepped Linear::	Stepped linear interpolation between From Min and From Max values.
Smooth Step::	Smooth Hermite edge interpolation between From Min and From Max values.
Smoother Step::	Smoother Hermite edge interpolation between From Min and From Max values.

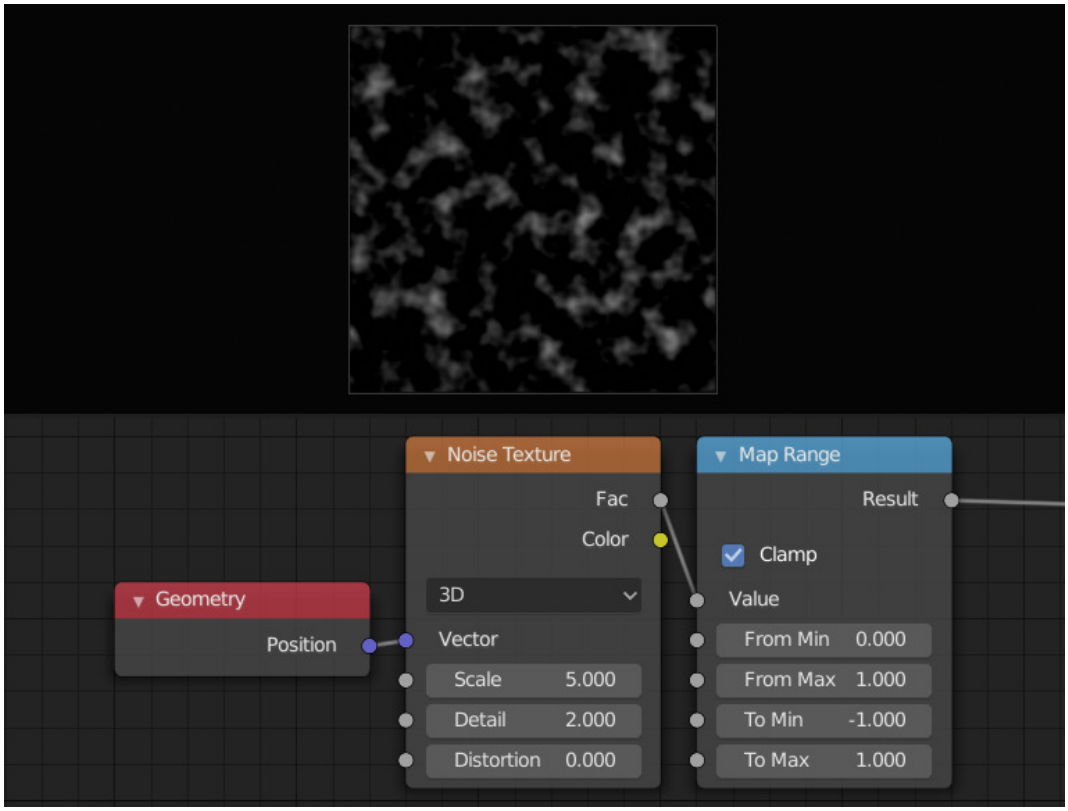
Clamp
: If enabled, the output is clamped to the target range.

Outputs

Result/Vector
: The input value after remapping.

Examples

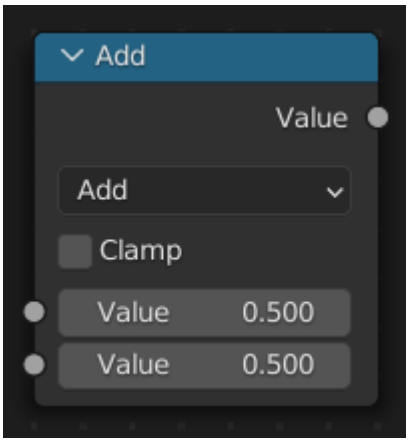
The *Noise Texture* node outputs a value in the range [0, 1]. We can use the *Map Range* node to remap this value into the range [-1, 1].



Example of Map Range node.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value
: Input Value. Trigonometric functions read this value as radians.

Addend
: Input Addend.

Base
: Input Base.

Exponent
: Input Exponent.

Epsilon
: Input Epsilon.

Distance

Input Distance.

Min
Input Minimum.

Max
Input Maximum.

Increment
Input Increment.

Scale
Input Scale.

Degrees
Input Degrees.

Radians
Input Radians.

Properties

Operation
The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

One divided by the square root of the value.

Absolute:: The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises Euler's number to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: Smooth Minimum.

Smooth Maximum:: Smooth Maximum.

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo:: Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.

Snap:: Rounds the input value down to the nearest integer multiple of *Increment*.

Ping-pong:: Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine:: The Sine of the input value.

Cosine:: The Cosine of the input value.

Tangent:: The Tangent of the input value.

Arcsine:: The Arcsine of the input value.

Arccosine:: The Arccosine of the input value.

Arctangent:: The Arctangent of the input value.

Arctan2:: Outputs the Inverse Tangent of the first value divided by the second value measured in radians.

Hyperbolic Sine:: The Hyperbolic Sine of the input value.

Hyperbolic Cosine:: The Hyperbolic Cosine of the input value.

Hyperbolic Tangent:: The Hyperbolic Tangent of the input value.

Conversion

To Radians:: Converts the input from degrees to radians.

To Degrees:: Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

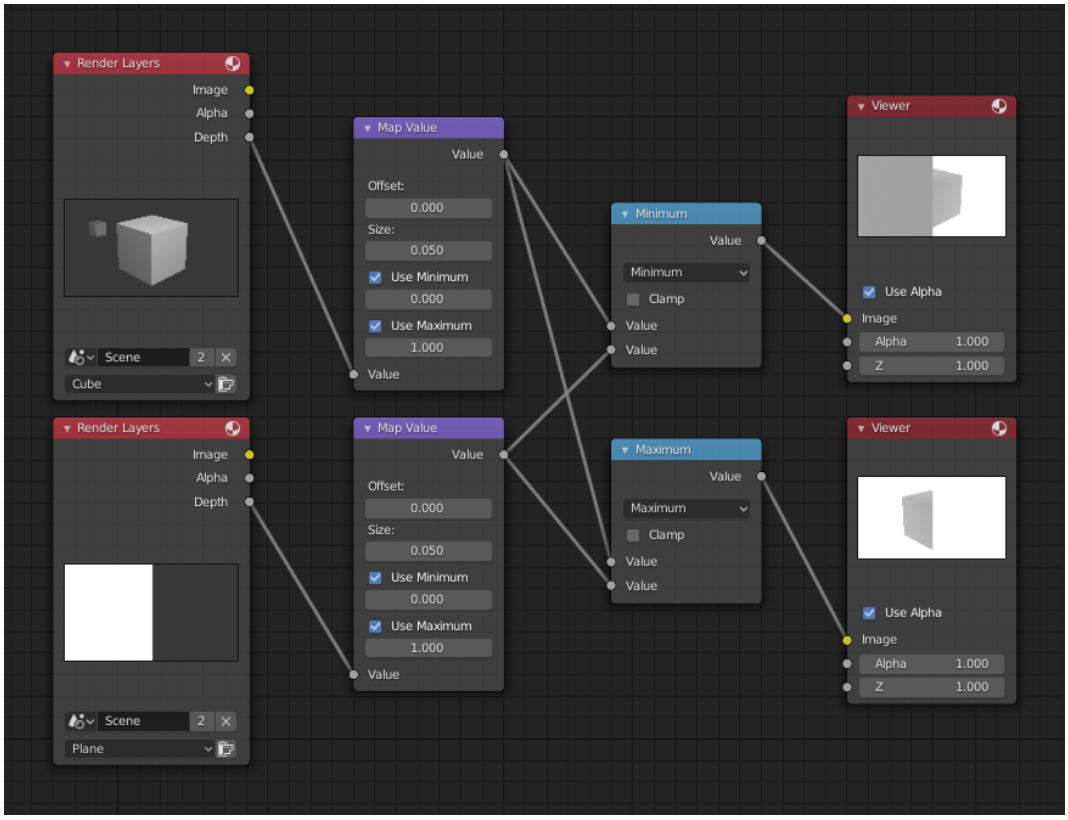
Outputs

Value

Numerical value output.

Examples

Manual Z-Mask



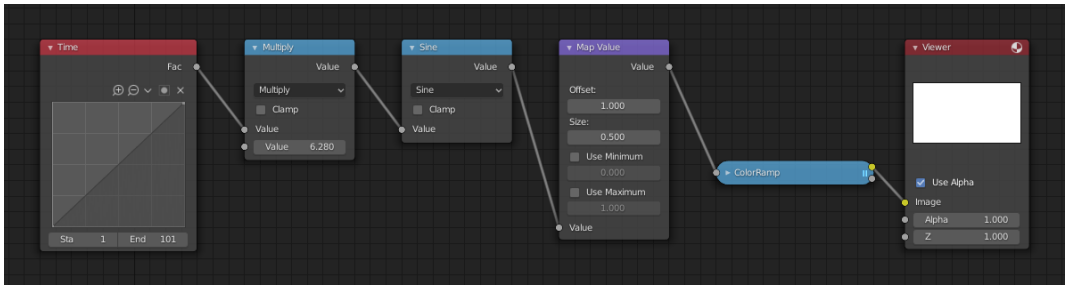
Minimum and maximum function example.

The top *Render Layers* node has a cube that is about 10 units from the camera. The bottom *Render Layers* node has a plane that covers the left half of the view and is 7 units from the camera. Both are fed through their respective *Map Value* nodes to multiply the depth value by 0.05 and clamp it to [0.0, 1.0], bringing it into a suitable range for displaying it as a color.

The Minimum node selects the smallest of the two depth values at each pixel. In the left half, it chooses the plane (because it's closer than the cube), and in the right half, it chooses the cube (because it's closer than the background, which is infinitely far away).

The Maximum node selects the largest of the two depth values at each pixel. In the left half, it chooses the cube (because it's farther away than the plane), and in the right half, it chooses the background (because it's farther away than the cube).

Using Sine Function to Pulsate



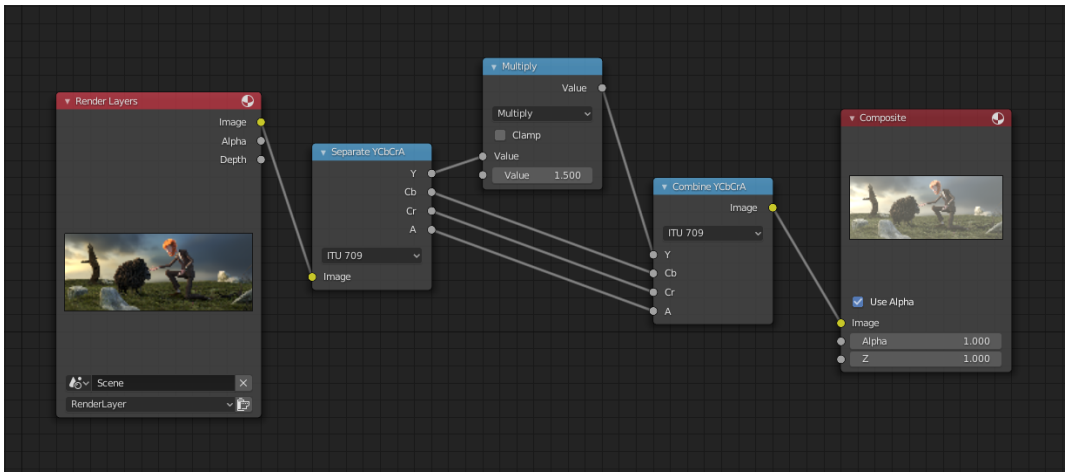
Using sine function example.

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. At frame 25, the output value is 0.25. That value is multiplied by 2 × pi (6.28) and converted to 1.0 by the Sine function, since \(sin(2 × pi/ 4) = sin(pi/ 2) = +1.0\).

Since the sine function can output values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by 0.5 (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a gray-scale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, \(sin(pi/ 2) = 1.0\). Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color “pulse”.

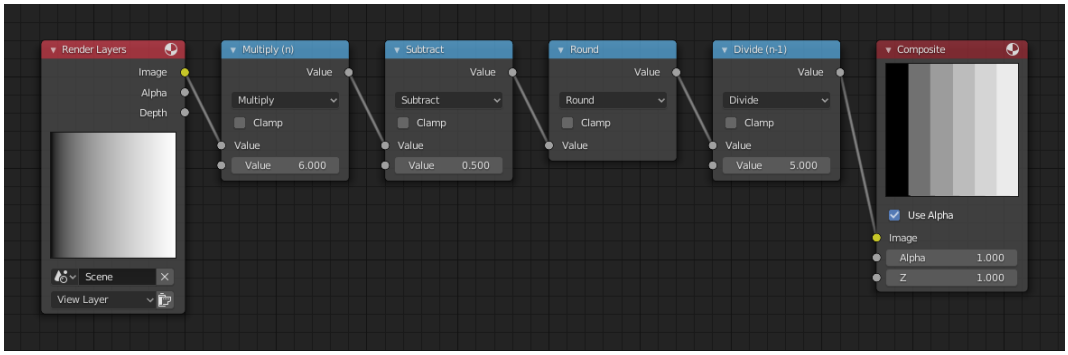
Brightening (Scaling) a Channel



Scaling a channel example.

This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high dynamic range image. For this particular example, there is also a *Brightness/Contrast node* that might give simpler control over brightness.

Restrict Color Selection (Posterization)



Posterization example.

In this example, we restrict the color values to be one of the six values: 0, 0.2, 0.4, 0.6, 0.8, 1.

To split up a continuous range of values between 0 and 1 to certain set of values, the following function is used: \(round(x × n - 0.5) / (n - 1)\), where

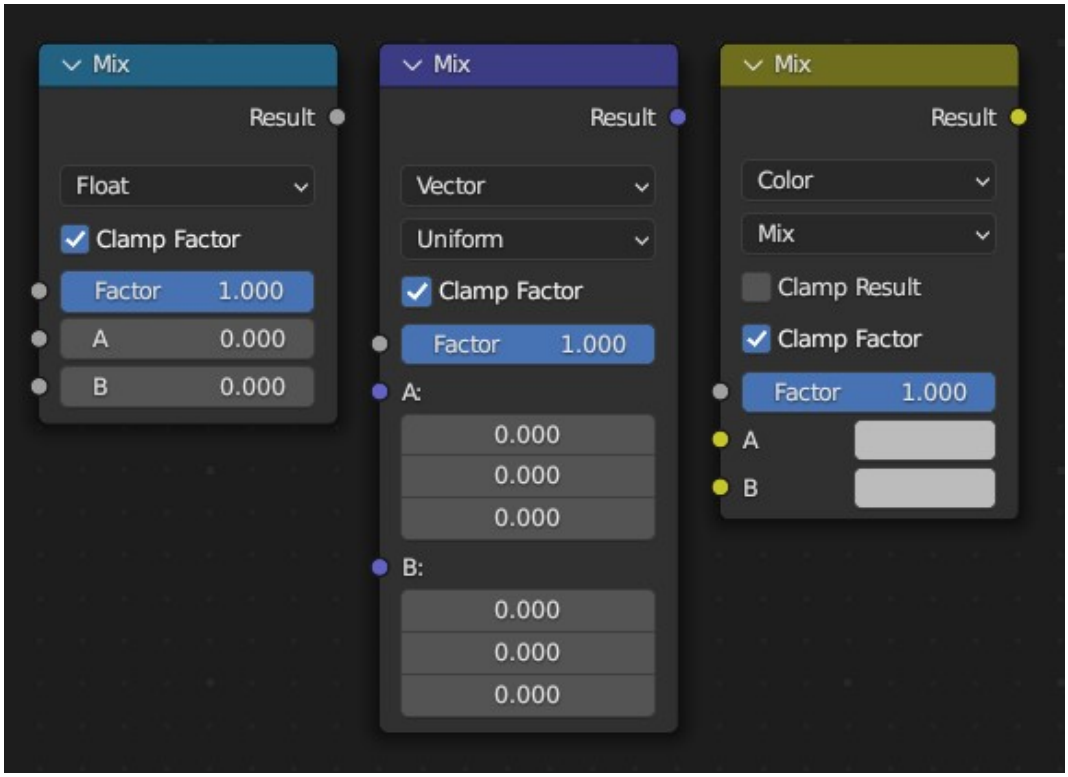
“n” is the number of possible output values, and “x” is the input pixel color. Read more about this function.

To implement this function in Blender, consider the node setup above. We string the Math nodes into a function that takes each color (values from 0 to 1), multiplies it up by six, the desired number of divisions (values become from 0 to 6), offsets it by 0.5 (-0.5 to 5.5), rounds the value to the nearest whole number (produces 0, 1, 2, 3, 4, 5), and then divides the image pixel color by five (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

In the case of a color image, you need to split it into separate RGB channels using *Separate/Combine Color* nodes and perform this operation on each channel independently.

Mix Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.



Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
: The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
: The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
: Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
: Limit the Result to the range between 0.0 and 1.0.

Outputs

Result
: Output the result of the mix using the data type selected.

Examples

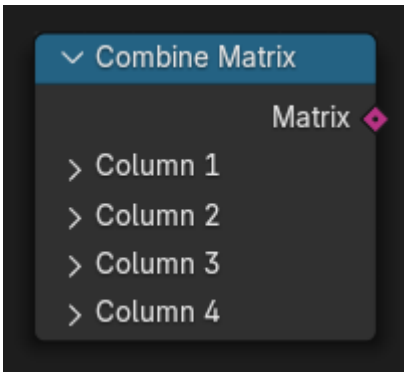
See the Color > Mix page for additional examples: Mix Color Node

Matrix Utility Nodes

- Combine Matrix Node
- Combine Transform Node
- Invert Matrix Node
- Matrix Determinant Node
- Multiply Matrices Node
- Project Point Node
- Separate Matrix Node
- Separate Transform Node
- Transform Direction Node
- Transform Point Node
- Transpose Matrix Node

Combine Matrix Node

The *Combine Matrix* node constructs a 4x4 matrix from its individual values.



Inputs

The inputs of this node are split into panels for each column of the matrix. Each panel, has four value inputs for the four rows of the matrix.

Properties

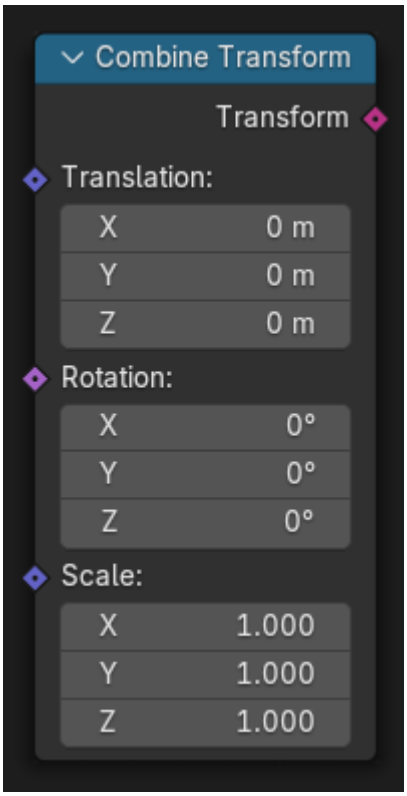
This node has no properties.

Outputs

Matrix
: The constructed matrix.

Combine Transform Node

The *Combine Transform* node combines a translation vector, a rotation vector, and a scale vector into a Transformation Matrix.



Inputs

Translation
: The translation vector.

Rotation
: The rotation vector.

Scale
: The scale vector.

Properties

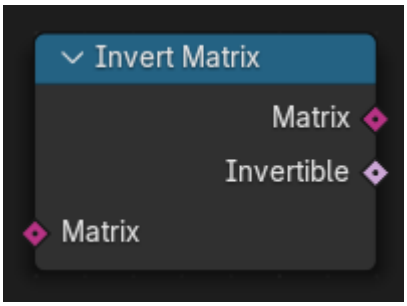
This node has no properties.

Outputs

Transform
: The combined transformation matrix.

Invert Matrix Node

Returns the inverse of the given matrix.



Inputs

Matrix
: The matrix to invert.

Properties

This node has no properties.

Outputs

Matrix
: The inverted matrix.

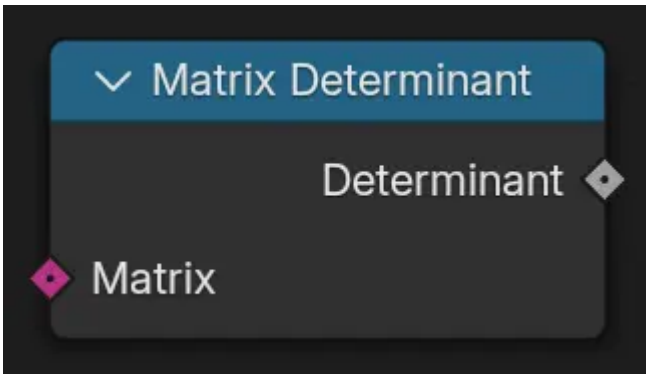
Invertible
: Returns whether the matrix could be inverted. This can be false when a transformation matrix has a scale of zero, for example. See Invertible matrix for more information.

Important

When a matrix is not invertible, the identity matrix is returned.

Matrix Determinant Node

The *Matrix Determinant* node computes the determinant of the passed in matrix.



Inputs

Matrix
: The matrix to compute the determinant of.

Properties

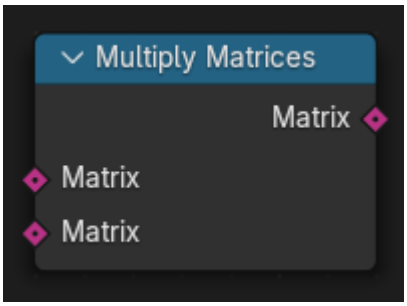
This node has no properties.

Outputs

Determinant
: The compute determinant value.

Multiply Matrices Node

The *Multiply Matrices* node performs a matrix multiplication on two input matrices.



Inputs

Matrix
: The first multiplication.

Matrix
: The second multiplication.

Properties

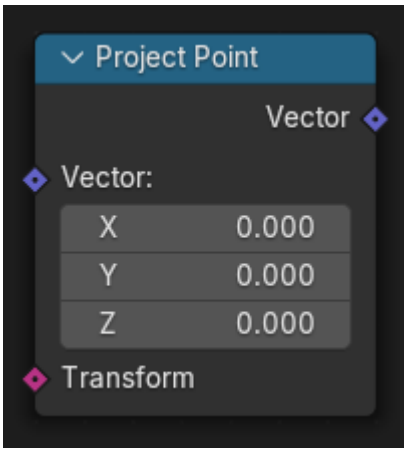
This node has no properties.

Outputs

Matrix
: The resulting matrix multiplication.

Project Point Node

Applies a projection matrix to a point. Specifically, this node turns the given Euclidean vector (X, Y, Z) into the homogeneous vector (X, Y, Z, 1), multiplies the given projection matrix by it, and turns the resulting homogeneous vector back into a Euclidean one by dividing it by the absolute value of its W component. This last step is also known as perspective division.



Inputs

Vector
: The position vector to project.

Transformation
: The projection matrix.

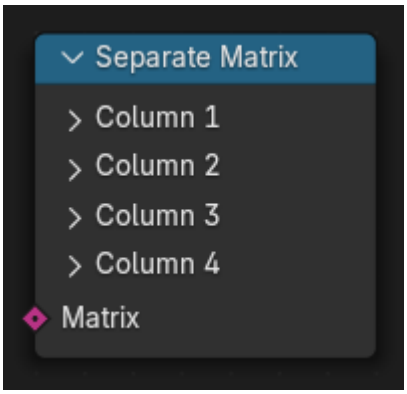
Properties

This node has no properties.

Outputs

Vector
: The projected position vector.

Separate Matrix Node



The *Separate Matrix* node splits a 4x4 matrix into its individual values.

Inputs

Matrix
: The matrix to split into individual values.

Properties

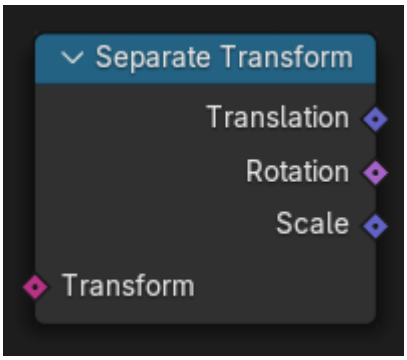
This node has no properties.

Outputs

The outputs of this node are split into panels for each column of the matrix. Each panel, has four value outputs for the four rows of the matrix.

Separate Transform Node

The *Separate Transform* node separates a Transformation Matrix into a translation vector, a rotation vector, and a scale vector.



Inputs

Transform
: The transformation matrix to separate.

Properties

This node has no properties.

Outputs

Translation
: The translation vector.

Rotation
: The rotation vector.

Scale
: The scale vector.

Transform Direction Node

The *Transform Direction* node multiplies a Transformation Matrix by a vector.



Inputs

Direction
: The vector.

Transformation
: The transformation matrix.

Properties

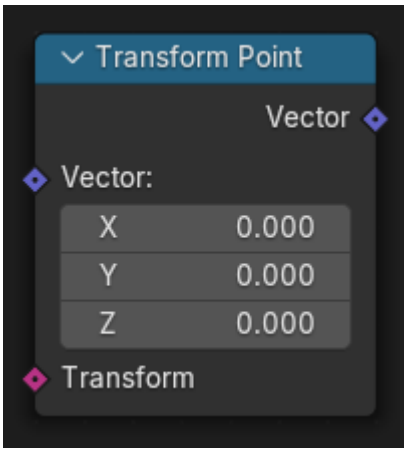
This node has no properties.

Outputs

Direction
: The vector.

Transform Point Node

The *Transform Point* node applies a Transformation Matrix to a position vector.



Inputs

Vector
: The position of a point to transform.

Transformation
: The transformation matrix.

Properties

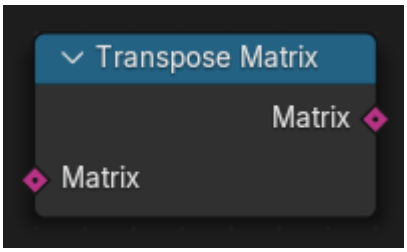
This node has no properties.

Outputs

Vector
: The transformed point.

Transpose Matrix Node

The *Transpose Matrix* node flips a matrix over its diagonal.



See also

Transpose on Wikipedia.

Inputs

Matrix
: The matrix to be transposed.

Properties

This node has no properties.

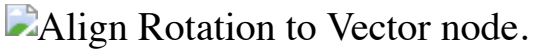
Outputs

Matrix
: The transposed matrix.

Rotation Utility Nodes

- Align Rotation to Vector Node
- Axes to Rotation Node
- Axis Angle to Rotation Node
- Euler to Rotation Node
- Invert Rotation Node
- Rotate Rotation Node
- Rotate Vector Node
- Rotation to Euler Node
- Rotation to Quaternion Node
- Quaternion to Rotation Node

Align Rotation to Vector Node



The *Align Rotation to Vector* node rotates an Euler rotation into the given direction.

Inputs

Rotation
: The Euler rotation to align.

  Important

  This input has to be a rotation input. Be careful not to connect a direction vector like the normal.

Factor
: Determines how much the points are rotated towards the vector. Zero effectively disables the node and one means that the points are aligned with the vector perfectly.

Vector
: The direction vector that points should be rotated to. The vector is in the local space of the object that is being modified. When it is all zeros for a point, it is not rotated at all.

Properties

Axis
: Local axis of the object that is to be rotated towards the vector input.

Pivot
: The local axis to rotate around.

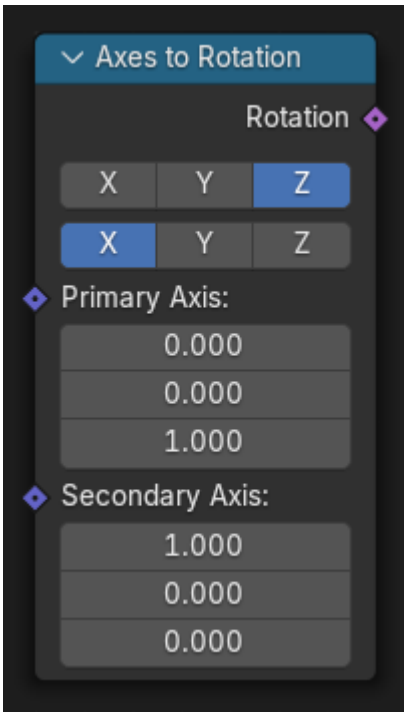
Auto::	The best rotation angle is computed automatically. This minimizes the angle of rotation.
X, Y, Z::	Rotate around a specific local axis.

Outputs

Rotation
: The rotated Euler rotation.

Axes to Rotation Node

Creates a rotation based on two axis directions.



Tip

In many cases, these directions are a normal and tangent on a mesh or curve.

Inputs

Primary Axis
: The desired direction of the primary axis.

Secondary Axis
: The desired direction of the secondary axis. Ideally, this is orthogonal to the primary direction.

Properties

Primary Axis
: The axis (X, Y or Z) that should be aligned exactly to the primary direction.

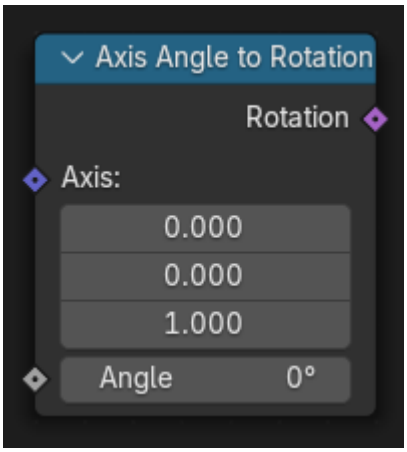
Secondary Axis
: The axis that should be aligned as closely as possible to the secondary direction.

Outputs

Rotation
: The rotation that results in the given axes being aligned to the given directions.

Axis Angle to Rotation Node

The *Axis Angle to Rotation* node converts a axis angle rotation to a standard rotation value.



Inputs

Axis
: Unit vector representing the axis to rotate around.

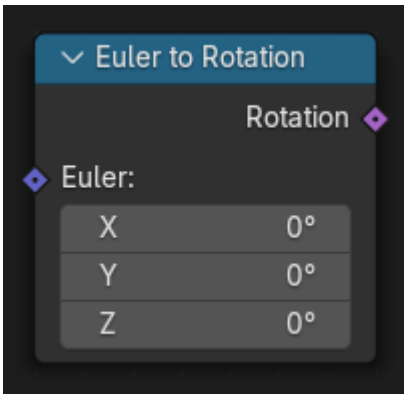
Angle
: The rotation angle around the axis.

Outputs

Rotation
: Standard rotation value.

Euler to Rotation Node

The *Euler to Rotation* node creates a rotation value from an Euler rotation.



Inputs

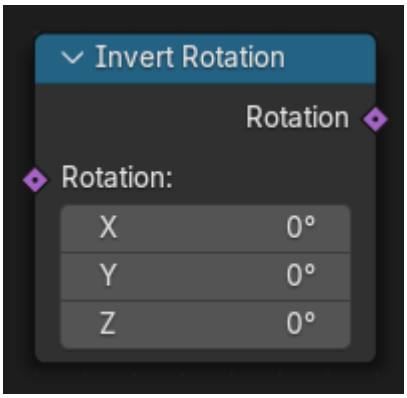
Euler
: The Euler rotation.

Outputs

Rotation
: Standard rotation value.

Invert Rotation Node

The *Invert Rotation* node inverts a rotation.



Inputs

Rotation
: Standard rotation value.

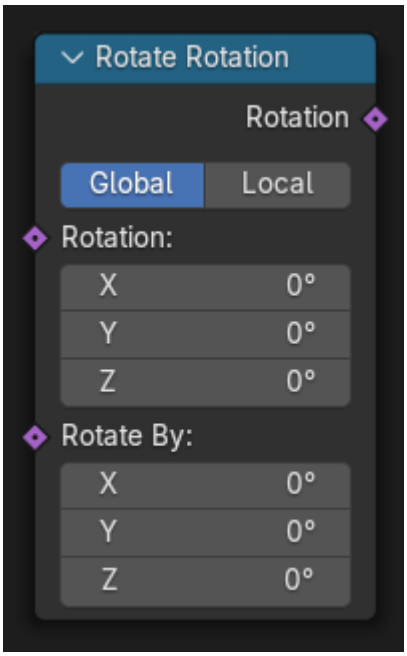
Outputs

Rotation
: The inverted rotation.

Rotate Rotation Node

The *Rotate Rotation* node applies an additional rotation to a given one.

To rotate an Euler Rotation, first use the Euler to Rotation Node.



Inputs

Rotation
: The starting rotation.

Rotate By
: The additional rotation.

Properties

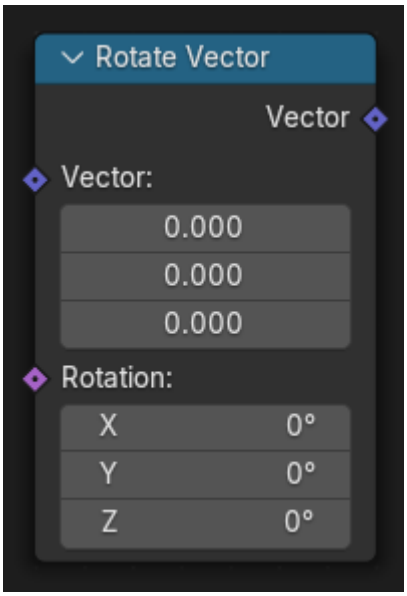
Space
: **Global::** Rotate in Global Space.
: **Local::** Rotate in Local Space.

Outputs

Rotation
: The resulting rotation.

Rotate Vector Node

The *Rotate Vector* node rotates a vector by a given rotation value.



Inputs

Vector
: The vector to rotate.

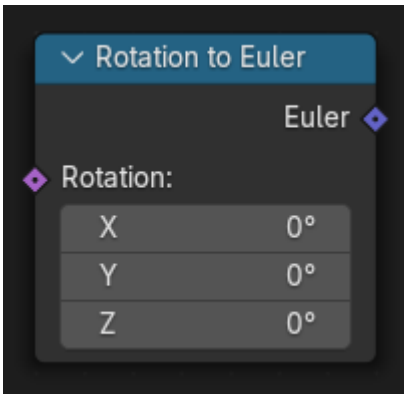
Rotation
: Standard rotation value.

Outputs

Vector
: The rotated vector.

Rotation to Euler Node

The *Rotation to Euler* node converts a standard rotation socket value to an Euler rotation.



Inputs

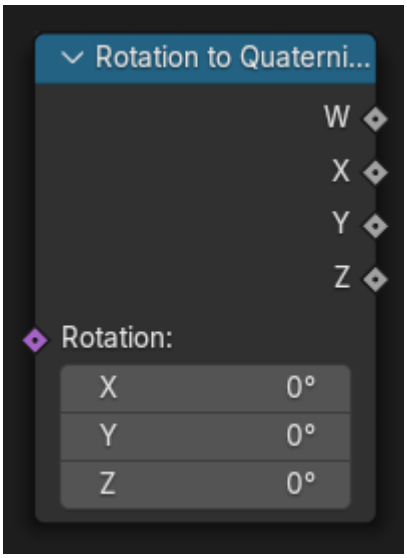
Rotation
: Standard rotation socket value.

Outputs

Euler
: The Euler rotation.

Rotation to Quaternion Node

The *Rotation to Quaternion* node converts a standard rotation value to a quaternion rotation.



Inputs

Rotation
: Standard rotation value.

Outputs

W
: The W value of the quaternion.

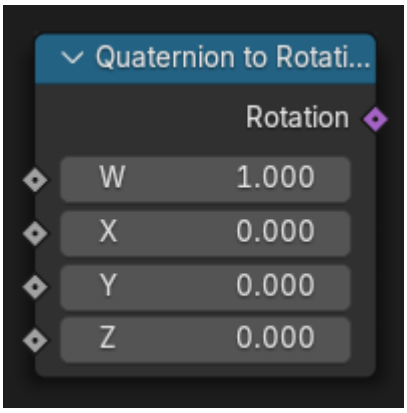
X
: The X value of the quaternion.

Y
: The Y value of the quaternion.

Z
: The Z value of the quaternion.

Quaternion to Rotation Node

The *Quaternion to Rotation* node converts a quaternion rotation to a standard rotation.



Inputs

W
: The W value of the quaternion.

X
: The X value of the quaternion.

Y
: The Y value of the quaternion.

Z
: The Z value of the quaternion.

Outputs

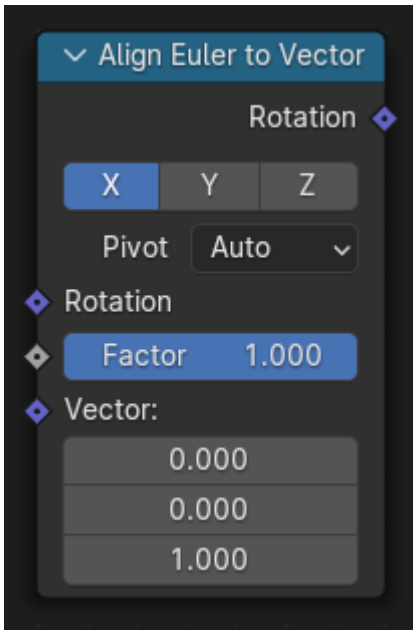
Rotation
: Standard rotation value.

Deprecated Nodes

These nodes have been made deprecated, meaning they will be removed in a future version and **should not be used**.

- Align Euler to Vector Node
- Rotate Euler Node

Align Euler to Vector Node



The *Align Euler to Vector* node rotates an Euler rotation into the given direction.

Important

This node is deprecated, use the Align Rotation to Vector Node instead.

Inputs

Rotation
: The Euler rotation to align.

  Important

  This input has to be a rotation input. Be careful not to connect a direction vector like the normal.

Factor
: Determines how much the points are rotated towards the vector. Zero effectively disables the node and one means that the points are aligned with the vector perfectly.

Vector
: The direction vector that points should be rotated to. The vector is in the local space of the object that is being modified. When it is all zeros for a point, it is not rotated at all.

Properties

Axis
: Local axis of the object that is to be rotated towards the vector input.

Pivot
: The local axis to rotate around.

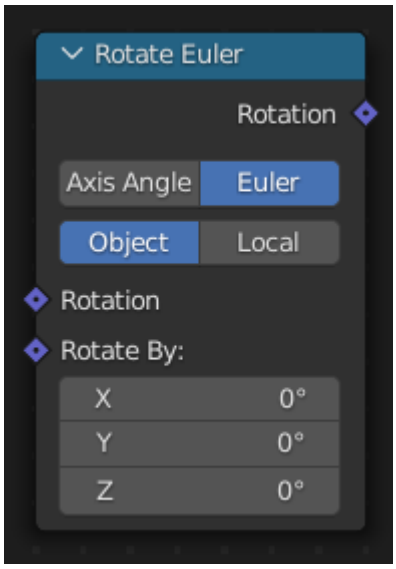
Auto::	The best rotation angle is computed automatically. This minimizes the angle of rotation.
X, Y, Z::	Rotate around a specific local axis.

Outputs

Rotation
: The rotated Euler rotation.

Rotate Euler Node

The *Rotate Euler* node rotates an Euler rotation.



Important

This node is deprecated, use the Rotate Rotation Node instead.

Inputs

Rotation
: The Euler rotation to rotate.

Rotate By
: Specifies how much an Euler rotation is rotated. This input is only available when the rotation type is set to *Euler*.

Axis
: The axis to rotate around. This input is only available when the rotation type is set to *Axis Angle*.

Angle
: The angle to rotate by around the specified axis. This input is only available when the rotation type is set to *Axis Angle*.

Properties

Rotation Type

Axis Angle:: Use separate axis and angle inputs to control the rotation.

Euler:: Use an Euler input to control the rotation.

Space

Object:: Rotate an Euler rotation in the evaluated object's space.

Local:: Rotate an Euler rotation in local space.

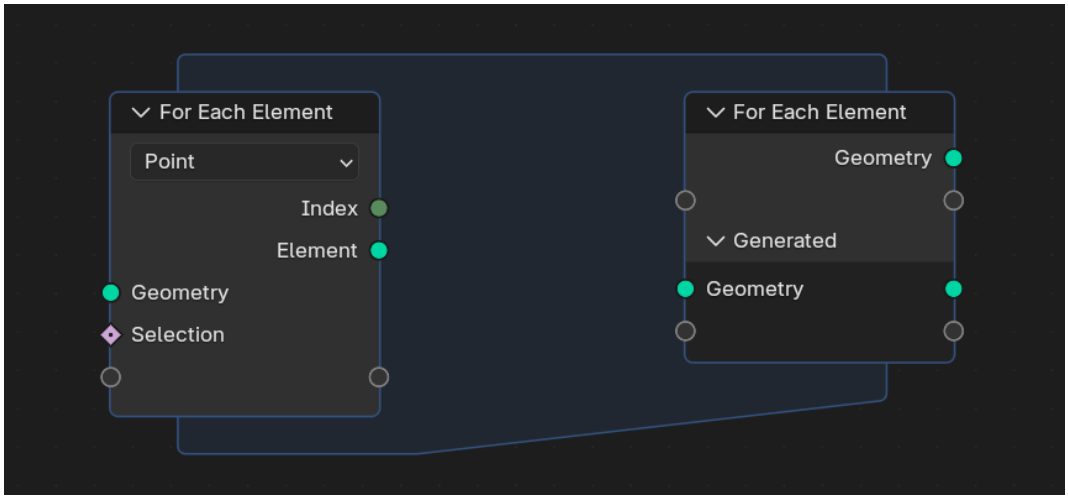
Outputs

Rotation

The rotated Euler rotation.

For Each Geometry Element Zone

This zone type allows executing nodes for each element of a geometry. For example, the nodes can process every face of a mesh, or every instance.



The *For Each Element* zone.

The zone is ideal for tasks that generate large or complex geometry for every element of an input geometry. For example, generating a unique tree for every input curve, or a unique building on every input face.

The zone makes less sense for processing small amounts of geometry. In that case (for example each of a character's hairs separately) it will likely **always be slower** than working on fewer larger geoemtries. The additional flexibility from processing each element separately comes at the cost that Blender can't optimize the operation as well. For node groups that need to handle lots of geometry elements, it's recommended to design the node setup so that iteration over tiny sub-geometries is not required.

Inputs

Geometry
: Geometry whose elements are iterated over.

Selection
: Which subset of the chosen *Domain* to process.

Index
: Index of the element in the source geometry. Note that the same index can occure more than once when iterating over multiple geometry component types at once.

Element
: The input geometry is split up into a separate geometry for each element. This is the single element geometry for the current iteration. This is not available for the *Face Corner* domain, since face corners cannot exist without their face.

> **Note**
>
> It can be quite inefficient to split up large geometries into many small elements. Because this output isn't computed if it's not used in the node graph, not using it will typically improve performance.

Properties

Domain
: Which attribute domain to process.

Inspection Index
: Geometry element index that is used by inspection features like the Viewer Node or socket inspection.

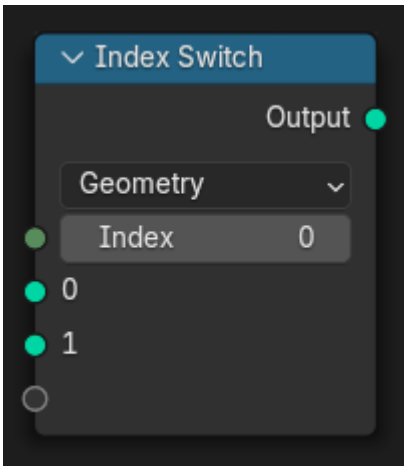
Outputs

The *Main Geometry* outputs create attributes on the "main" output geometry (the first output). Every single value on the inside of the zone becomes a value of the attribute at the current index.

The outputs in the *Generated* panel, including the default *Geometry* output are joined together from the geometry generated from each element. Any non-geometry type below a specific geometry in this list will output as an anonymous attribute on that joined geometry (and not the others). Attributes from the zone's input geometry are also propagated to these geometry outputs.

Index Switch Node

The *Index Switch* node outputs one of its inputs depending on an index value. Only the input that is passed through the node is computed.



See also

The Menu Switch Node is similar but it exposes the choices as a menu.

Inputs

Index
: Determines which of the input options below will be passed through.

Item Inputs
: One input is created for every menu entry. The input is used when the matching option is selected.

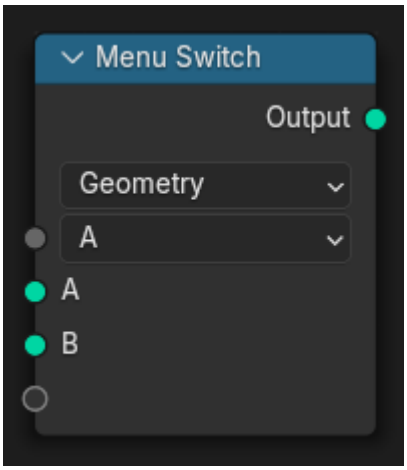
Properties

Type
: Determines the type of the data that is handled by the node.

Outputs

Output
: One of the inputs without any modifications.

Menu Switch Node

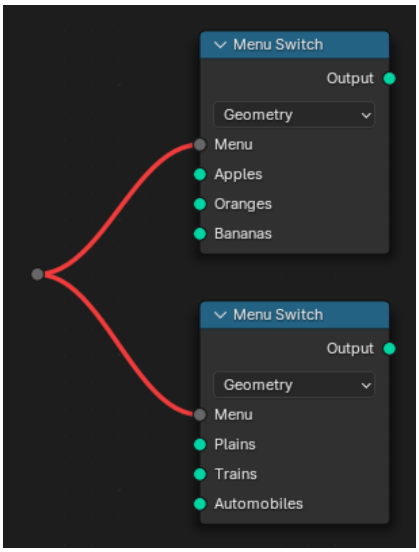


The *Menu Switch* node outputs one of its inputs depending on a menu selection. Only the input that is passed through the node is computed.

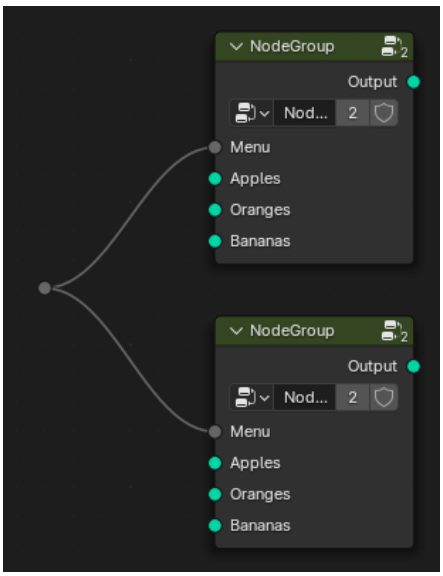
The available menu entries are defined by the user. Menu items can be added and removed, as well as renamed and reordered in the editor side bar. Renaming a menu entry keeps existing links of the matching input socket.

The menu can be used in node groups and the nodes modifier UI. Connecting the menu input with a *Group Input* node will expose the menu as a group input. A menu socket in a node group, reroute node, or other pass-through nodes needs to be connected to a *Menu Switch* node in order to work. An unconnected menu socket will show an empty menu by default.

Connecting multiple *Menu Switch* nodes to the same output socket creates a conflict (even when the menu entries are the same). To avoid this a menu switch can be wrapped in a node group. Multiple node groups of the same type can be connected to the same menu, since they contain the same menu switch node.



Conflict caused by connecting different menus.



Same node group can be connected without conflict.

See also

The Index Switch Node is similar but it exposes the choices as an integer index.

Inputs

Menu
: Determines which of the input options below will be passed through.

Item Inputs
: One input is created for every menu entry. The input is used when the matching option is selected. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

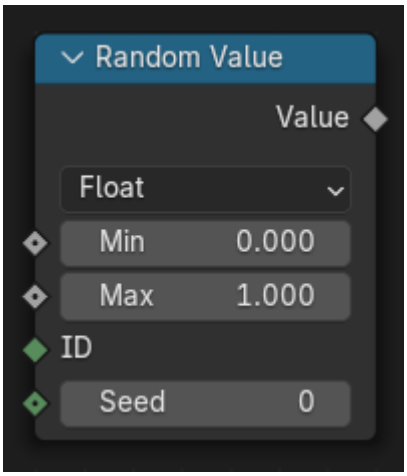
Type
: Determines the type of the data that is handled by the node.

Outputs

Output
: One of the inputs without any modifications.

Random Value Node

The *Random Value* node outputs a white noise like value as a *Float*, *Integer*, *Vector*, or *Boolean* field.



Inputs

Min
: The minimum value of the range where random values are sampled from. This input is only available for *Float*, *Integer*, and *Vector* types.

Max
: The maximum value of the range where random values are sampled from. This input is only available for *Float*, *Integer*, and *Vector* types.

Probability
: The probability ratio for the random *Boolean* output to be *True*. This input is only available for *Boolean* types.

ID
: An ID to drive the random number generator seed. By default, this input uses the same value as of the ID Node, which is the `id` attribute of the context geometry if it exists, and otherwise the index.

Tip

Single Random Value

By default, the random value node generates a value for each unique index. If a single random value is desired, connect a single value (such as an Integer Node) to the ID input.

Seed

A field to Seed the random number generator. This can be used to generate a different set of random values, even for two nodes with the same *ID* input.

Properties

Data Type

Float::	The output will be a *Float* field.
Integer::	The output will be an *Integer* field.
Vector::	The output will be a *Vector* field.
Boolean::	The output will be a *Boolean* field.

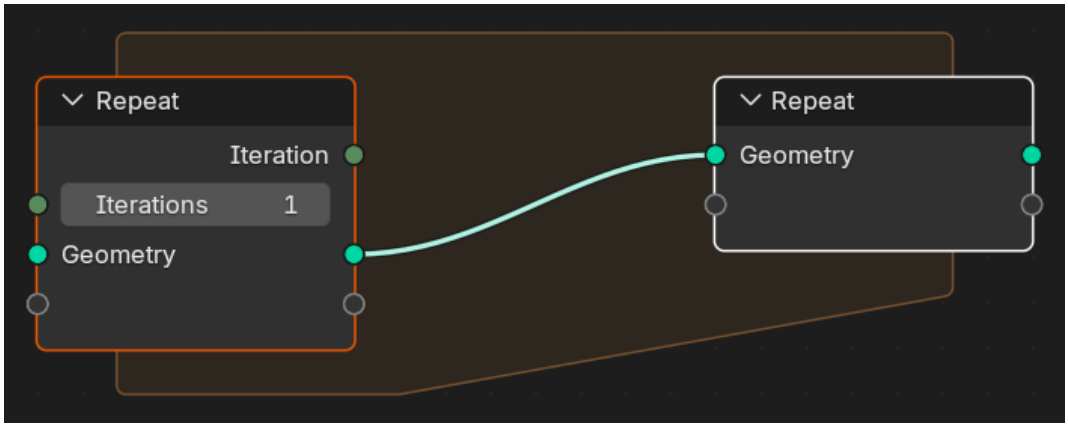
Outputs

Value

Random values as a field.

Repeat Zone

Repeat zones allow running nodes many times in a loop. Compared with simply duplicating a node, they support executing a node an arbitrary number of times, possibly determined when the node group is evaluated. For example, the nodes could be repeated based on the number of stories in a building generator.



Repeat zone used to repeat a node group a few times

When adding a repeat zone, two nodes are added, with the “zone” defined between them. The inputs connected to the *Repeat Input* node are read at the beginning, before starting the repetitions. Then they are passed to the inside of the zone where they can be changed, and passed to the next iteration. This process is repeated the specified number of times.

Other nodes can be connected as inputs to the inside of the repeat zone from the outside. Those are constant throughout every iteration based on their value at the current frame. However, outputs of the zone must be connected through the *Repeat Output* node.

Inputs

Iterations

Number of times to repeat the execution of the zone. The current iteration is available with the *Iteration* input on the inside of the zone.

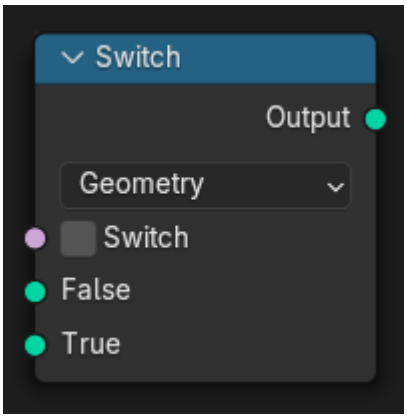
Geometry
: Standard geometry input, which is available by default to input geometry into the repeat zone. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

Inspection Index
: Iteration number that is used by inspection features like the Viewer Node or socket inspection.

Switch Node

The *Switch* node outputs one of two inputs depending on a condition. Only the input that is passed through the node is computed.



See also

The Menu Switch Node and Index Switch Node can be used to switch between an arbitrary amount of inputs.

Inputs

Switch
: Determines which of the two inputs below will be passed through.

False
: Is passed through when the switch is set to false.

True
: Is passed through when the switch is set to true.

Properties

Type
: Determines the type of the data that is handled by the node.

Outputs

Output
: One of the two inputs without any modifications.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Hair Nodes

Nodes focussed on generating or editing curves, typically used for hair.

- Deformation
  - Blend Hair Curves
  - Displace Hair Curves
  - Frizz Hair Curves
  - Hair Curves Noise
  - Roll Hair Curves
  - Rotate Hair Curves
  - Shrinkwrap Hair Curves
  - Smooth Hair Curves
  - Straighten Hair Curves
  - Trim Hair Curves
- Generation
  - Duplicate Hair Curves
  - Generate Hair Curves
  - Interpolate Hair Curves
- Guides
  - Braid Hair Curves
  - Clump Hair Curves
  - Create Guide Index Map
  - Curl Hair Curves
- Read
  - Curve Info
  - Curve Root
  - Curve Segment
  - Curve Tip
  - Hair Attachment Info
- Utility
  - Attach Hair Curves to Surface
  - Redistribute Curve Points
  - Restore Curve Segment Length
- Write

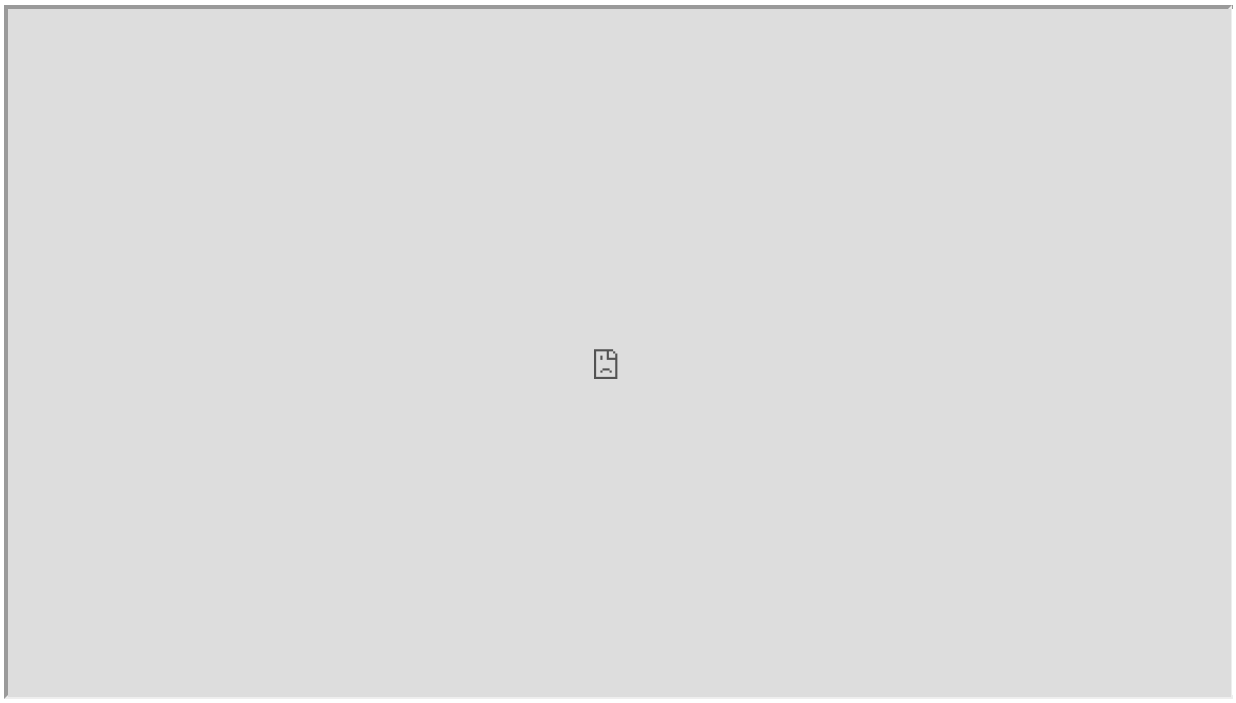
- Set Hair Curve Profile

Hair Deformation Nodes

- Blend Hair Curves
- Displace Hair Curves
- Frizz Hair Curves
- Hair Curves Noise
- Roll Hair Curves
- Rotate Hair Curves
- Shrinkwrap Hair Curves
- Smooth Hair Curves
- Straighten Hair Curves
- Trim Hair Curves

Blend Hair Curves

Blends shape between multiple hair curves in a certain radius.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Factor
: Factor to blend overall effect.

Blend Radius
: Radius to select neighbors for blending.

Blend Neighbors
: Amount of neighbors used for blending.

Preserve Length
: Preserve each curve's length during deformation.

Properties

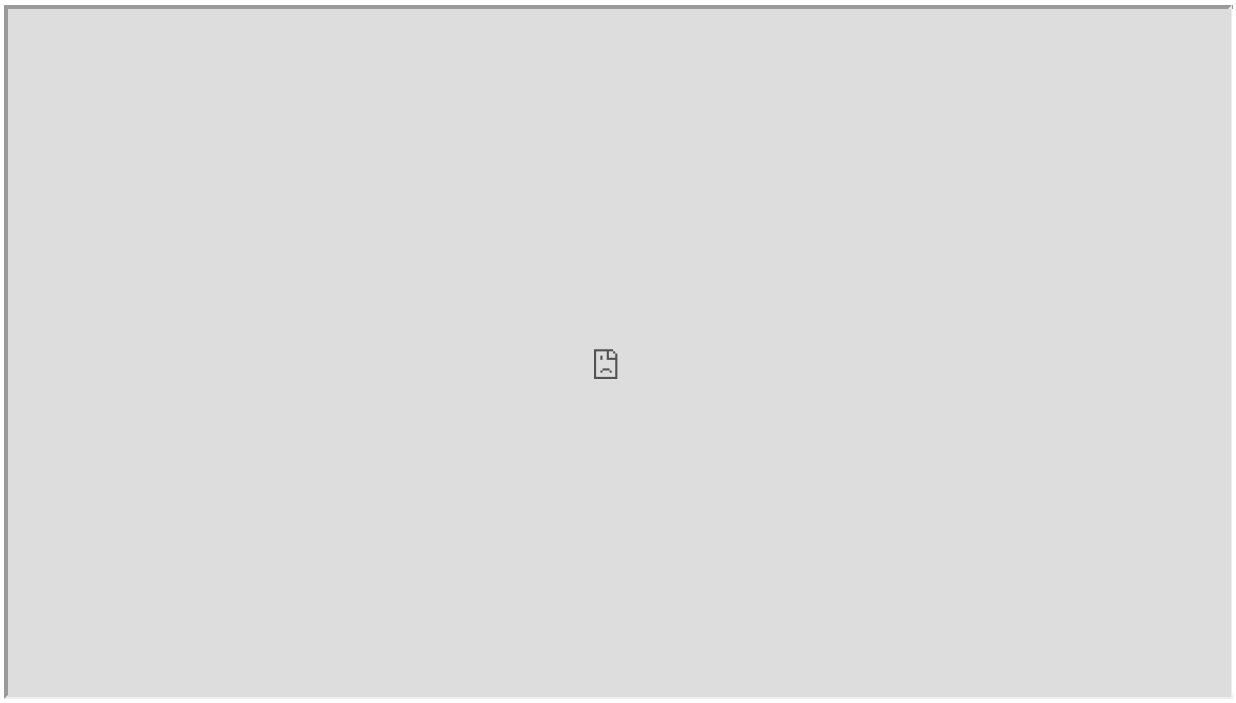
This node has no properties.

Outputs

Geometry

Displace Hair Curves

Displaces hair curves by a vector based on various options.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Factor
: Factor to scale overall displacement.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Object Space
: Object used to define the displacement space.

Displace Vector
: Vector for displacement.

Surface
: Surface geometry used to sample the normal for displacement. This input takes priority over the corresponding object input, if used.

Surface
: Surface object used to sample the normal for displacement.

Surface UV Map
: Surface UV map used to sample the normal for displacement.

Surface Normal Displacement
: Amount of displacement along the surface normal.

Properties

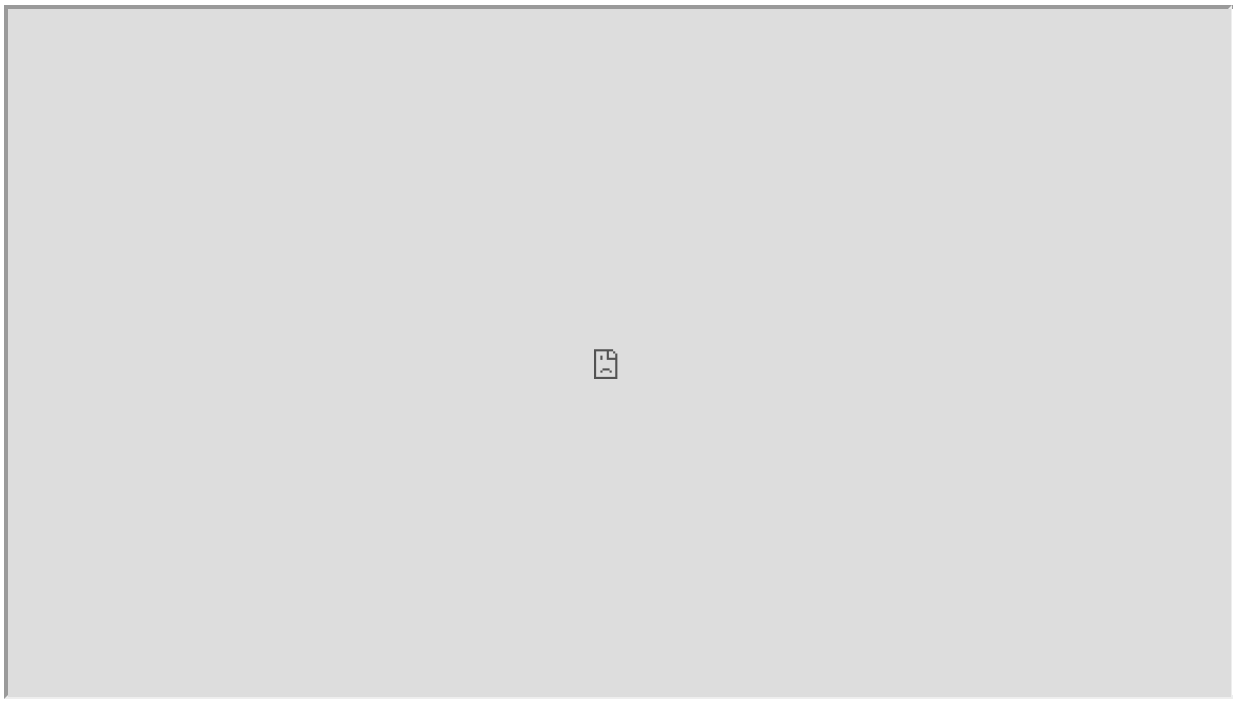
This node has no properties.

Outputs

Geometry

Frizz Hair Curves

Deforms hair curves using a random vector per point to frizz them.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Cumulative Offset
: Apply offset cumulatively (previous points affect points after).

Factor
: Factor to blend overall effect.

Distance
: Overall distance factor for the deformation.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Seed
: Random Seed for the operation.

Preserve Length
: Preserve each curve's length during deformation.

Properties

This node has no properties.

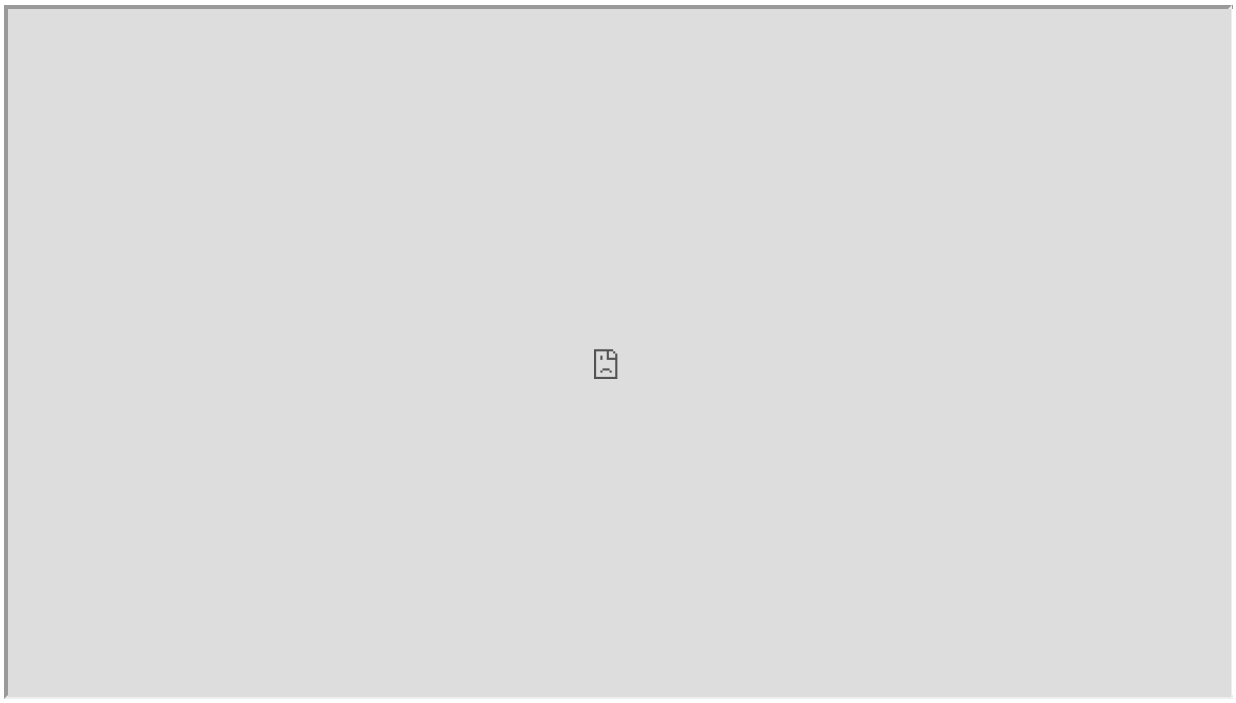
Outputs

Geometry

Offset Vector
: Vector by which each point was offset during deformation.

Hair Curves Noise

Deforms hair curves using a noise texture.



Inputs

Geometry

Cumulative Offset
: Apply offset cumulatively (previous points affect points after).

Factor
: Overall factor for the deformation.

Distance
: Overall distance factor for the deformation.

Shape

Shape of amount along each curve (0=constant, 0.5=linear).

Scale
: Scale of the noise texture by root position.

Scale along Curve
: Scale of noise texture along each Curve.

Offset per Curve
: Random offset of noise texture for each Curve.

Seed
: Seed value for randomization.

Preserve Length
: Preserve the length of the Curves on a segment basis.

Properties

This node has no properties.

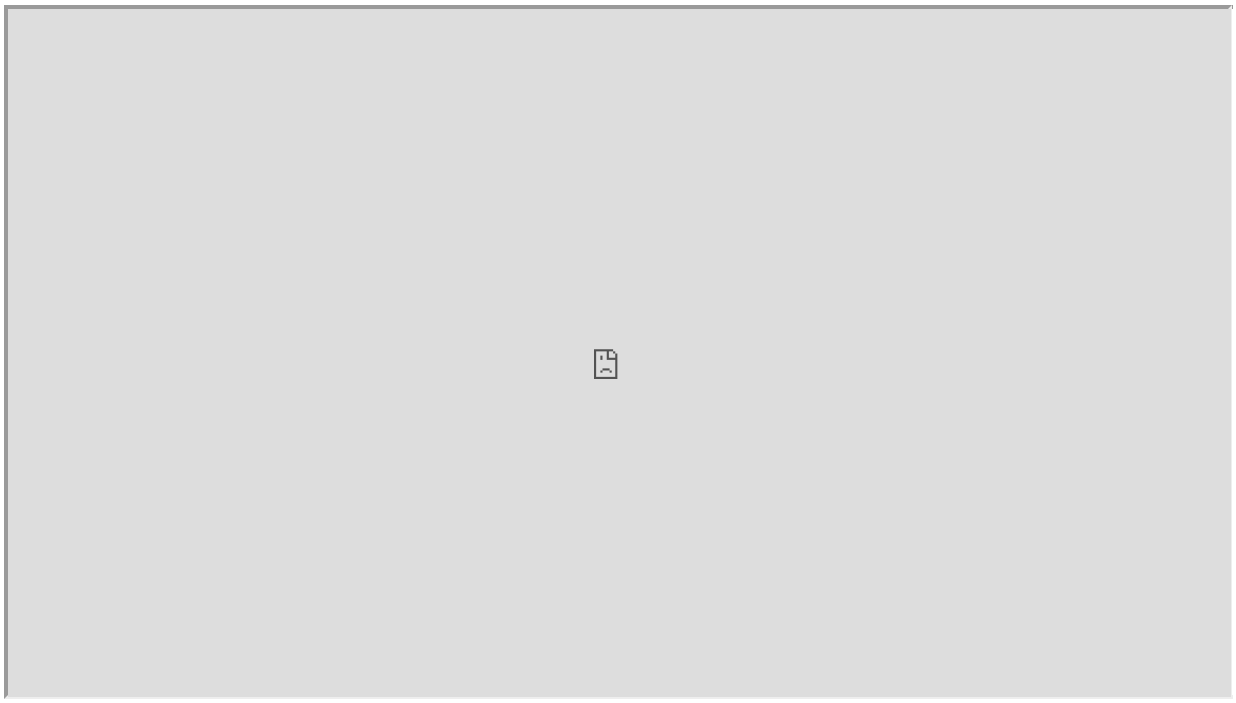
Outputs

Geometry

Offset Vector

Roll Hair Curves

Rolls up hair curves starting from their tips.



Inputs

Geometry

Factor
: Factor to blend overall effect.

Subdivision
: Subdivision level applied before deformation.

Variation Level
: Level of smoothing on the roll path to include shape variation.

Roll Length

Length of each curve to be rolled.

Roll Radius
: Radius of the rolls.

Roll Depth
: Depth offset of the roll.

Roll Taper
: Taper of the roll.

Retain Overall Shape
: Offset the roll along the original curve to retain shape.

Roll Direction
: Axis around which each curve is rolled.

Random Orientation
: Amount of randomization of the direction of the roll.

Seed
: Random Seed for the operation.

Preserve Length
: Preserve each curve's length during deformation.

Properties

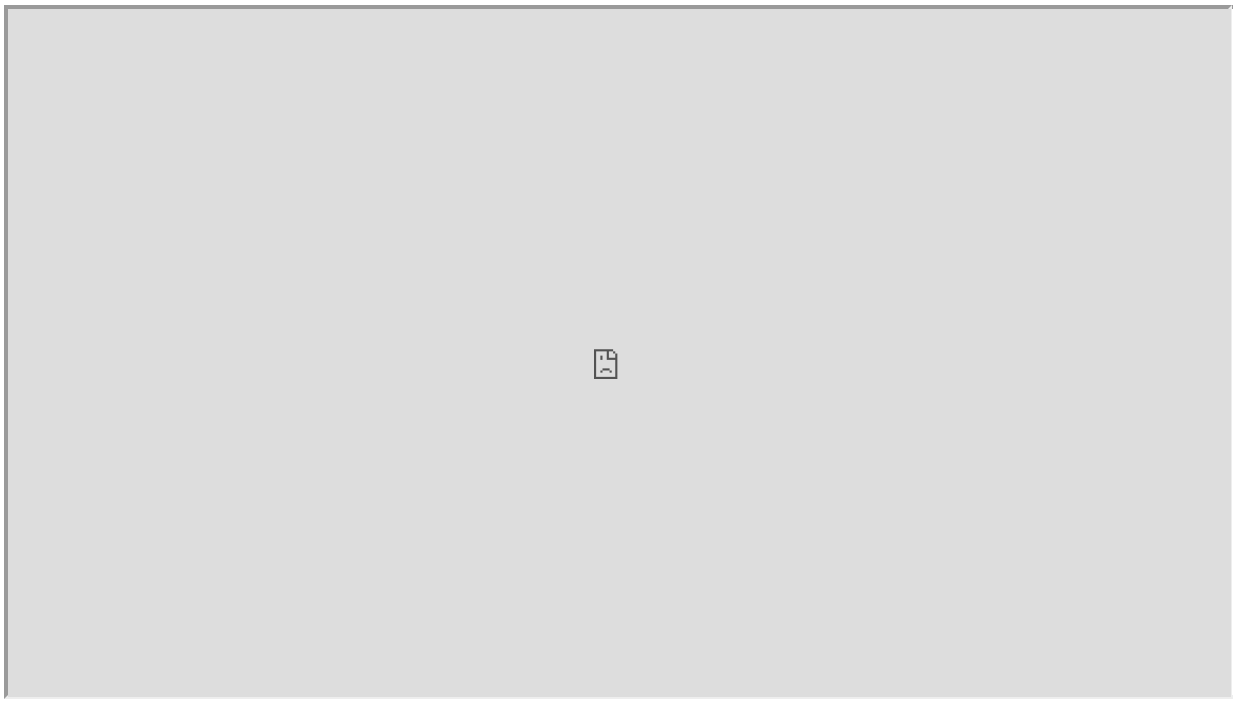
This node has no properties.

Outputs

Geometry

Rotate Hair Curves

Rotates each hair curve around an axis.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Factor
: Factor to influence the rotation angle.

Axis
: Rotation Axis (Default: Tangent at root).

Angle
: Angle of rotation.

Random Offset
: Random offset to the rotation angle per Curve.

Lock Ends
: Lock rotation to the axis between the curve ends.

Seed
: Random Seed for the operation.

Properties

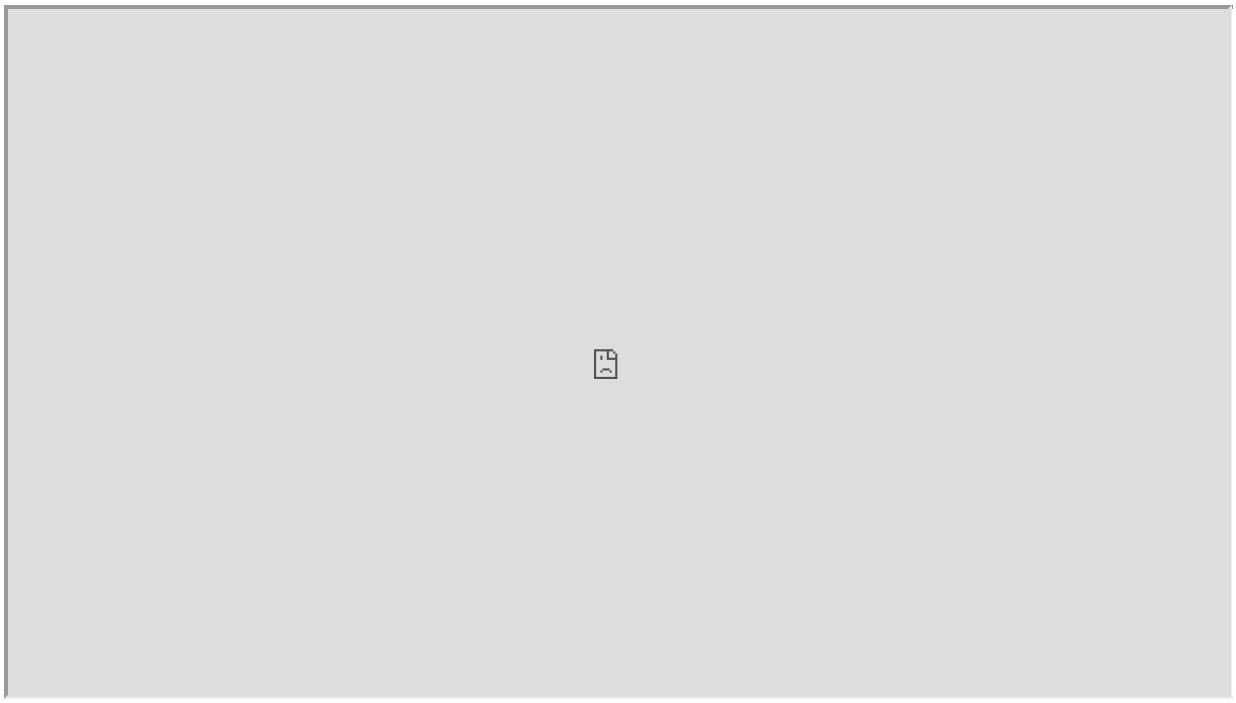
This node has no properties.

Outputs

Geometry

Shrinkwrap Hair Curves

Shrinkwraps hair curves to a mesh surface from below and optionally from above.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Surface
: Surface geometry used for shrinkwrap. This input takes priority over the corresponding object input, if used.

Surface
: Surface object used for shrinkwrap.

Factor

Offset Distance
: Distance from the surface used for shrinkwrap.

Above Surface
: Blend shrinkwrap for points above the surface.

Smoothing Steps
: Amount of steps of smoothing applied after shrinkwrap.

Lock Roots
: Lock the position of root points.

Properties

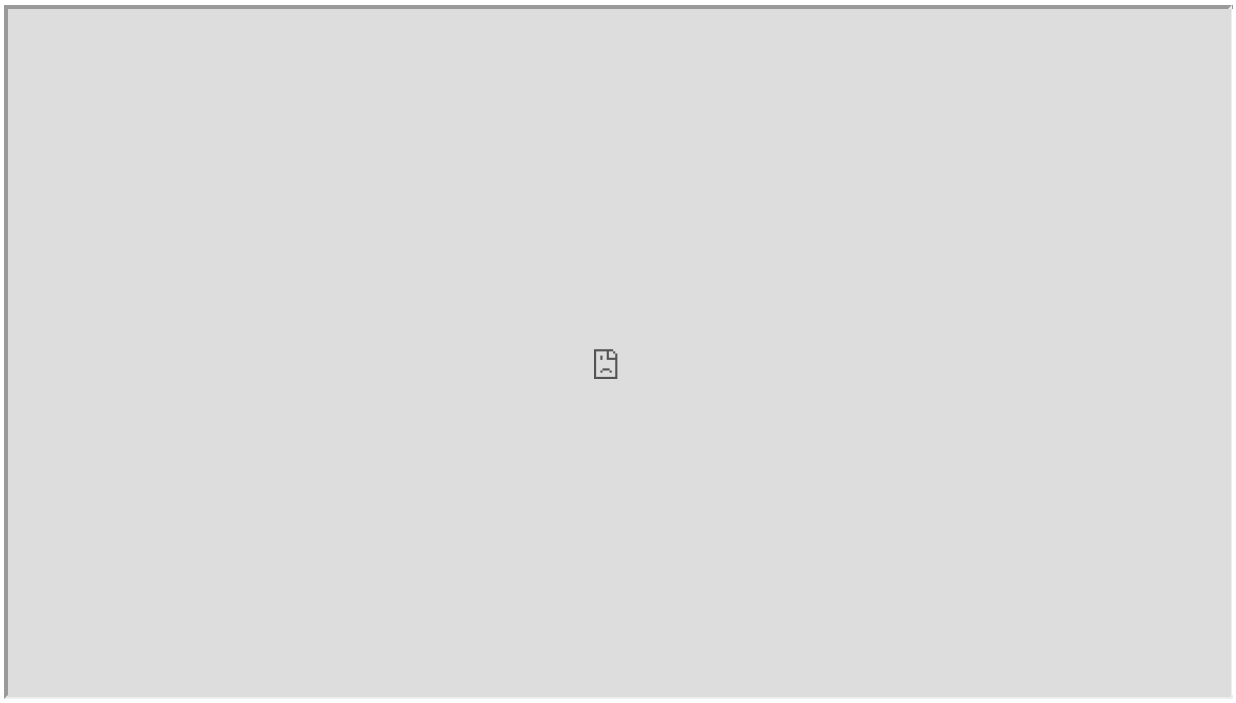
This node has no properties.

Outputs

Geometry

Smooth Hair Curves

Smooths the shape of hair curves.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Amount
: Amount of smoothing. Negative values will result in crumpling the curves.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Iterations
: Amount of smoothing steps.

Weight
: Weight used for smoothing.

Lock Tips
: Lock tip position when smoothing.

Preserve Length
: Preserve each curve's length during deformation.

Properties

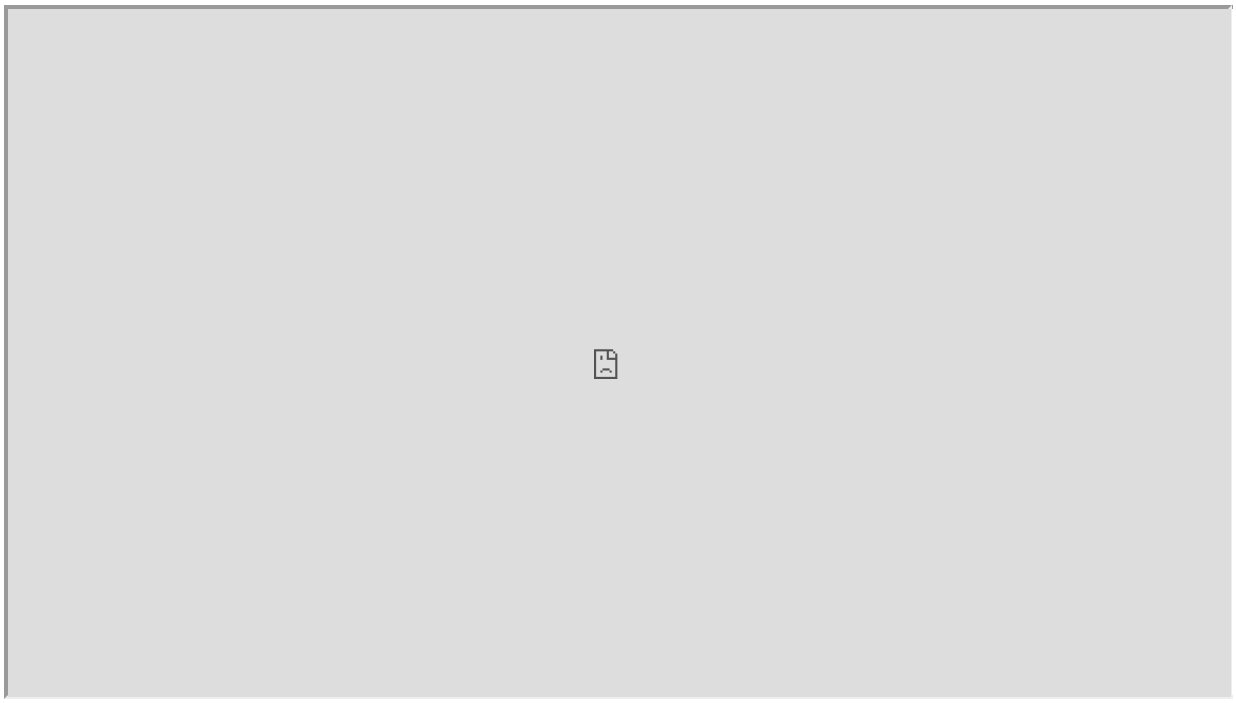
This node has no properties.

Outputs

Geometry

Straighten Hair Curves

Straightens hair curves between root and tip.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Amount
: Amount of straightening. Negative values will result in crumpling the curves.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Preserve Length
: Preserve each curve's length during deformation.

Properties

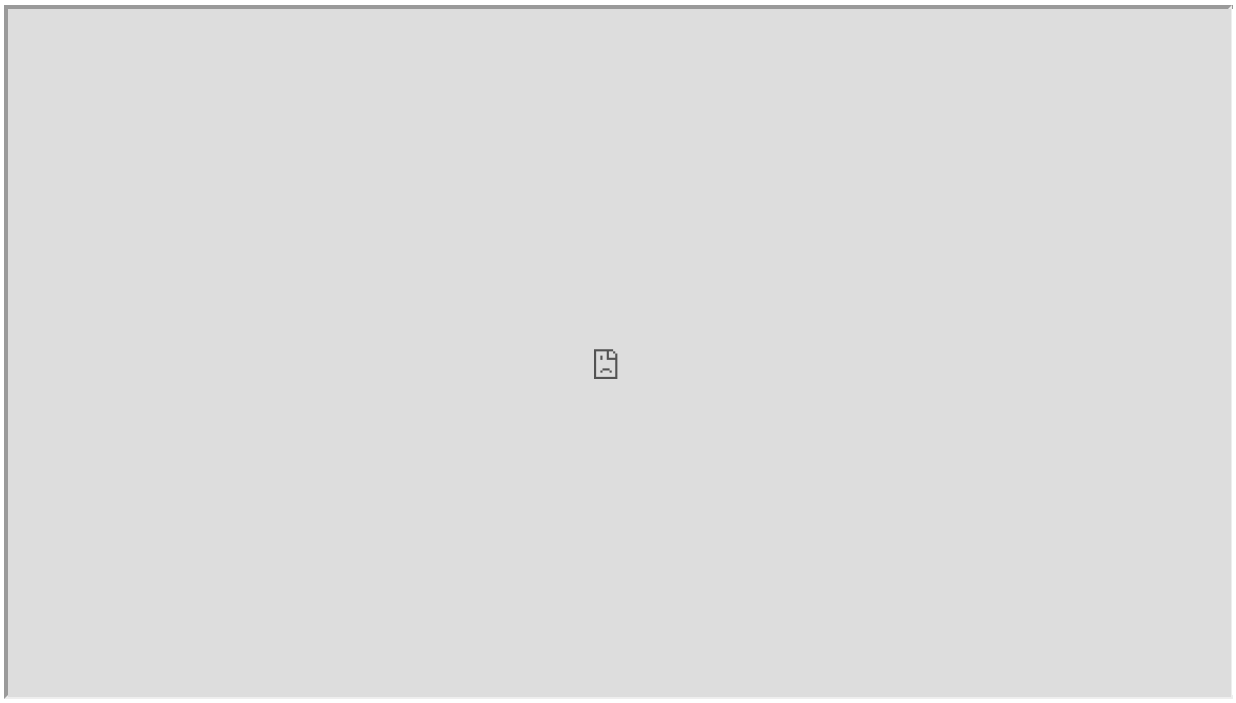
This node has no properties.

Outputs

Geometry

Trim Hair Curves

Trims or scales hair curves to a certain length.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Scale Uniform
: Scale each curve uniformly to reach the target length.

Length Factor
: Multiply the original length by a factor.

Replace Length
: Use the length input to fully replace the original length.

Length
: Target length for the operation.

Mask
: Mask to blend overall effect.

Random Offset
: Trim hair curves randomly up to a certain amount.

Pin at Parameter
: Pin each curve at a certain point for the operation.

Seed
: Random Seed for the operation.

Properties

This node has no properties.

Outputs

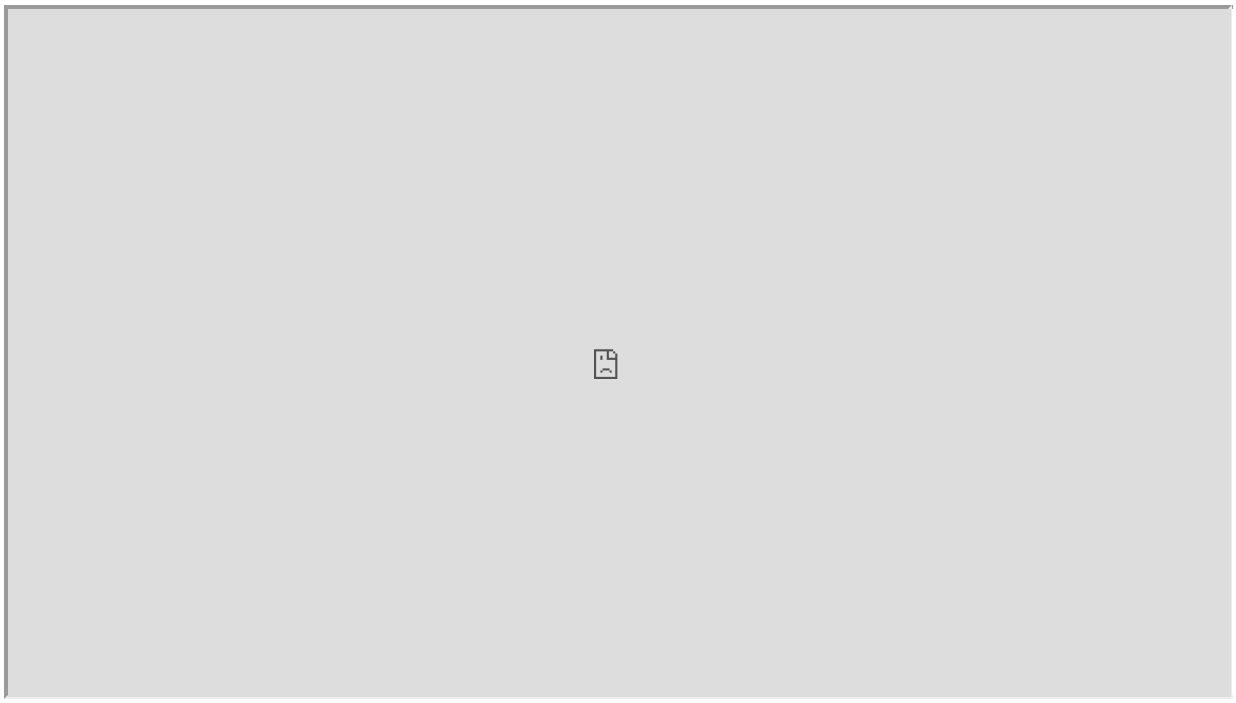
Geometry

Hair Generation Nodes

- Duplicate Hair Curves
- Generate Hair Curves
- Interpolate Hair Curves

Duplicate Hair Curves

Duplicates hair curves a certain number of times within a radius.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Amount
: Amount of duplicates per curve.

Viewport Amount
: Percentage of amount used for the viewport.

Radius
: Radius in which the duplicate curves are offset from the guides.

Distribution Shape
: Shape of distribution from center to the edge around the guide.

Tip Roundness
: Offset of the curves to round the tip.

Even Thickness
: Keep an even thickness of the distribution of duplicates.

Seed
: Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

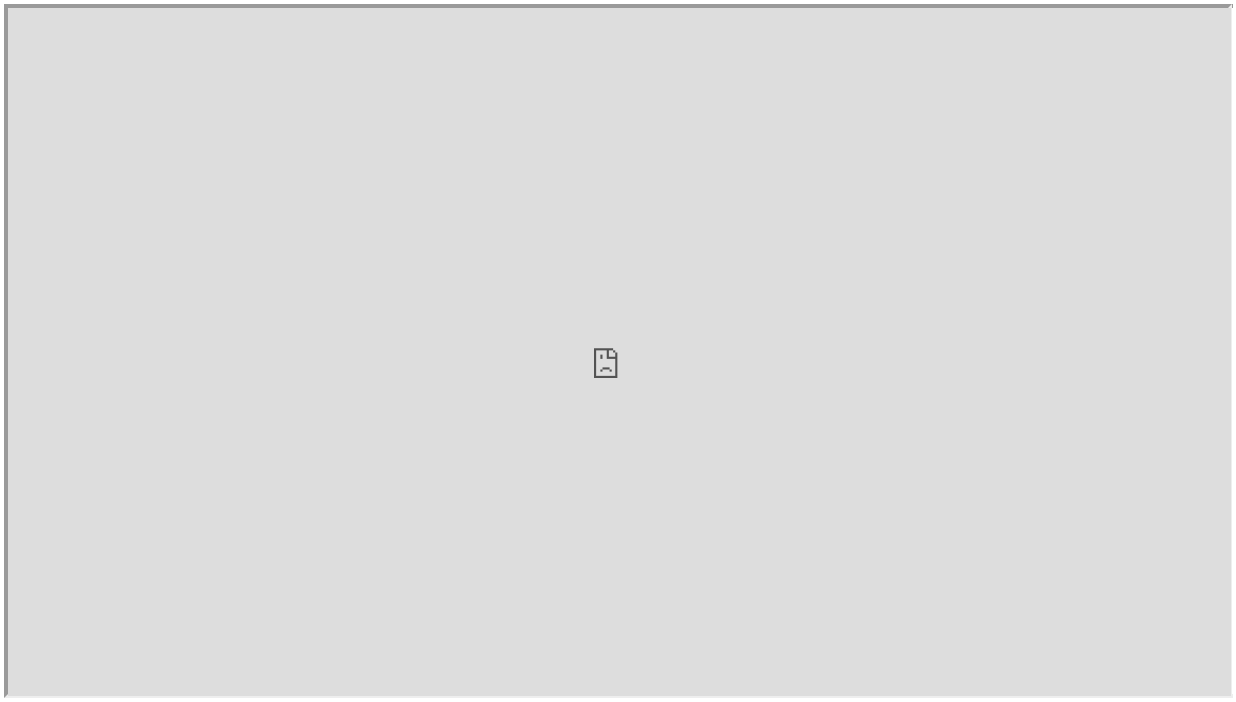
Guide Index
: Guide index map that was used for the operation.

Generate Hair Curves

Generates new hair curves on a surface mesh. The curves are generated from scratch at point locations; if creating curves that depend on existing curves is desired, the Interpolate Hair Curves is a better choice.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.



Inputs

Surface

Surface geometry for generation. This input takes priority over the corresponding object input if both are provided.

Surface
: Surface object for generation (The transforms of this object must match the modifier object).

Surface UV Map
: Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position
: Set the surface mesh into its rest position before attachment.

> **Tip**
>
> In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Hair Length
: Length of the generated hair curves.

Hair Material
: Material of the generated hair curves.

Control Points
: Amount of control points of the generated hair curves.

Poisson Disk Distribution
: Use poisson disk distribution method to keep a minimum distance. See the Distribute Points on Faces for more information.

Density
: Surface density of generated hair curves.

Density Mask
: Factor applied on the density for curve distribution.

Mask Texture
: Discard points based on an mask texture after distribution. The image is sampled with the *Surface UV Map* input.

> **Tip**
>
> The accuracy of sampling the image doesn't depend on the density of the surface mesh's vertices because it is sampled after the curve root points are generated, the accuracy . However, using the *Density Mask* input instead can give better performance. Using them in combination can give the benefits of both methods.

Viewport Amount
: Factor applied on the density for the viewport.

Seed
: Random seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Curves

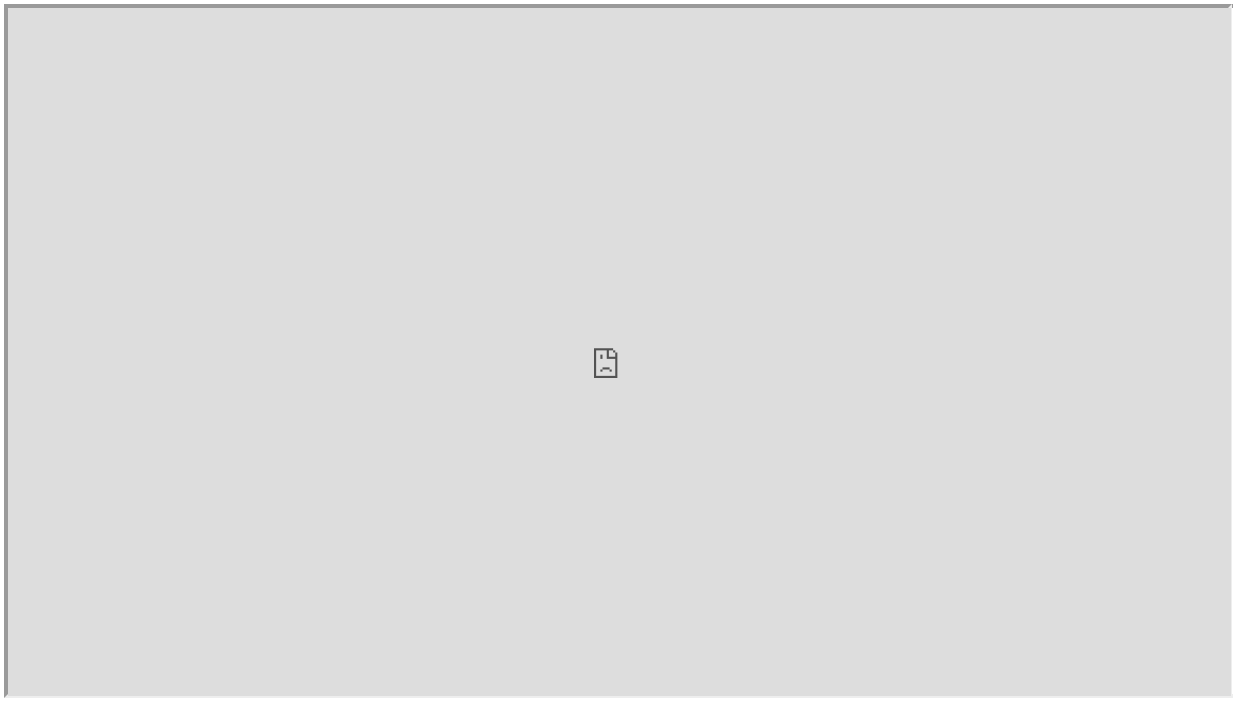
Surface Normal
: Normal direction of the surface mesh at the attachment point.

Interpolate Hair Curves

Interpolates existing guide curves on a surface mesh. The Duplicate Hair Curves is a similar option with simpler behavior that may offer better performance.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Surface

Surface geometry for generation. This input takes priority over the corresponding object input if both are provided.

Surface
: Surface object for generation (Needs matching transforms).

Surface UV Map
: Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position
: Set the surface mesh into its rest position before attachment.

> **Tip**
>
> In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Follow Surface Normal
: Align the interpolated curves to the surface normal.

Part by Mesh Islands
: Use mesh islands of the surface geometry for parting.

Interpolation Guides
: Amount of guides to be used for interpolation per curve.

Distance to Guides
: Distance around each guide to spawn interpolated curves.

Poisson Disk Distribution
: Use poisson disk distribution method to keep a minimum distance.

Density

Surface density of generated hair curves.

Density Mask
: Factor applied on the density for curve distribution.

Mask Texture
: Discard points based on an mask texture after distribution. The image is sampled with the *Surface UV Map* input.

> **Tip**
>
> The accuracy of sampling the image doesn't depend on the density of the surface mesh's vertices because it is sampled after the curve root points are generated, the accuracy . However, using the *Density Mask* input instead can give better performance. Using them in combination can give the benefits of both methods.

Viewport Amount
: Factor applied on the density for the viewport.

Seed
: Random seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Guide Index
: Index of the main guide curve per curve.

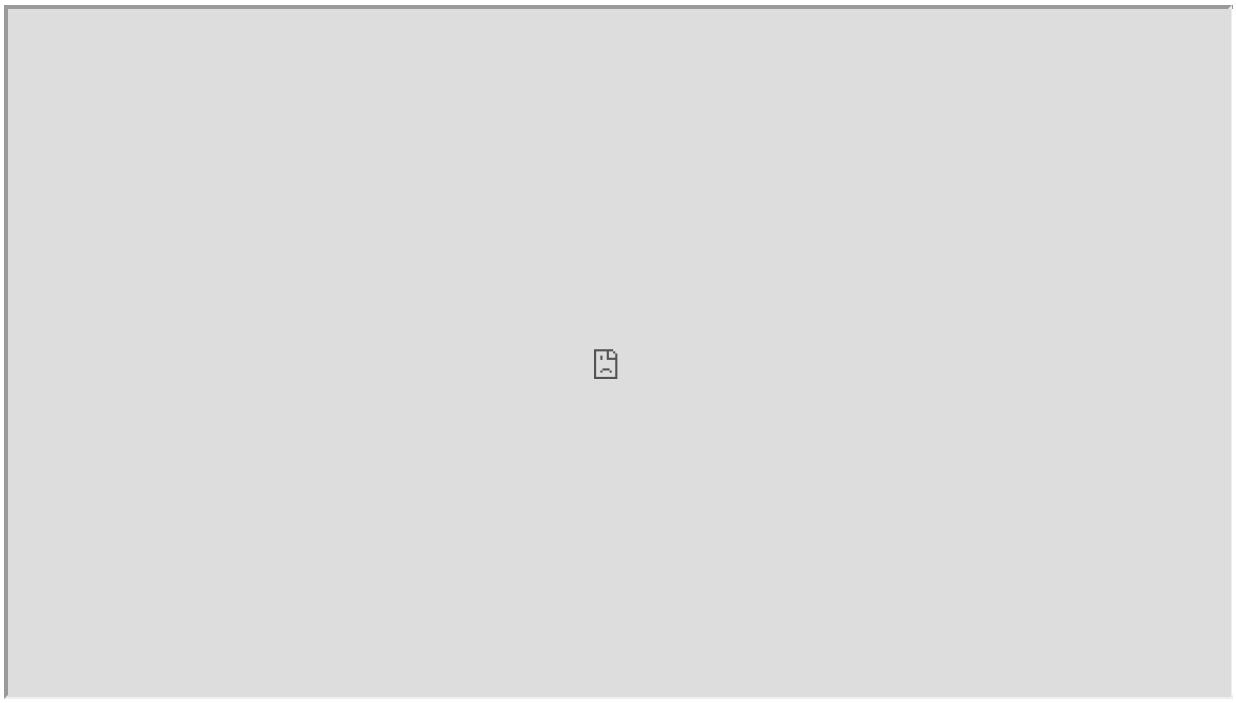
Surface Normal
: Normal direction of the surface mesh at the attachment point.

Hair Guides Nodes

- Braid Hair Curves
- Clump Hair Curves
- Create Guide Index Map
- Curl Hair Curves

Braid Hair Curves

Deforms existing hair curves into braids using guide curves.



Inputs

Geometry

Guide Index
: Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance
: Minimum distance between two guides for new guide map.

Guide Mask

Mask for which curve are eligible to be selected as guides.

Existing Guide Map
: Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn't provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor
: Factor to blend overall effect.

Subdivision
: Subdivision level applied before deformation.

Braid Start
: Percentage along each curve to blend deformation from the root.

Radius
: Overall radius of the braids.

Shape
: Shape of the braid radius along each curve.

Factor Min
: Factor of the minimum radius of the braids.

Factor Max
: Factor of the maximum radius of the braids.

Frequency
: Frequency factor of the braids. This input can vary for different points of the same curve.

Thickness
: Thickness of each strand of hair.

Thickness Shape
: Shape adjustment of the strand thickness for the braids.

Shape Asymmetry
: Asymmetry of the shape adjustment of the strand thickness.

Flare Length
: Length of the flare at the end of the braid.

Flare Opening
: Opening radius of the flare at the tip of the braid.

Hair Tie
: Geometry used for the hair tie instance (priority).

Hair Tie
: Object used for the hair tie instance.

Hair Tie Scale
: Scale of the hair tie instance.

Properties

This node has no properties.

Outputs

Geometry

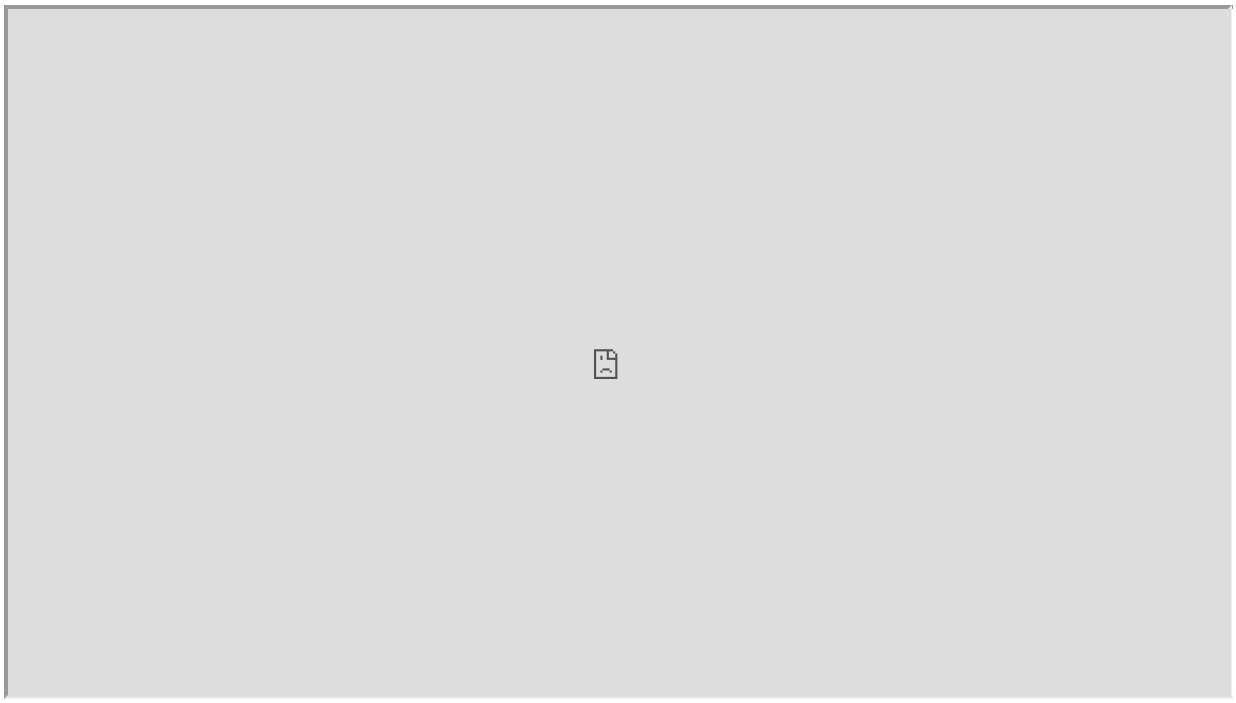
Guide Index
: Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Flare Parameter
: Parameter from 0 to 1 along the flare.

Strand Index
: Index of the group of hair in the braid that each hair curve belongs to.

Clump Hair Curves

Clumps together existing hair curves using guide curves.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Guide Index
: Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance
: Minimum distance between two guides for new guide map.

Guide Mask
: Mask for which curve are eligible to be selected as guides.

Existing Guide Map
: Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn't provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor
: Factor to blend overall effect.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Tip Spread
: Distance of random spread at the curve tips.

Clump Offset
: Offset of each clump in a random direction.

Distance Falloff
: Falloff distance for the clumping effect (0 means no falloff).

Distance Threshold
: Distance threshold for the falloff around the guide.

Seed
: Random seed for the operation.

Preserve Length
: Preserve each curve's length during deformation.

Properties

This node has no properties.

Outputs

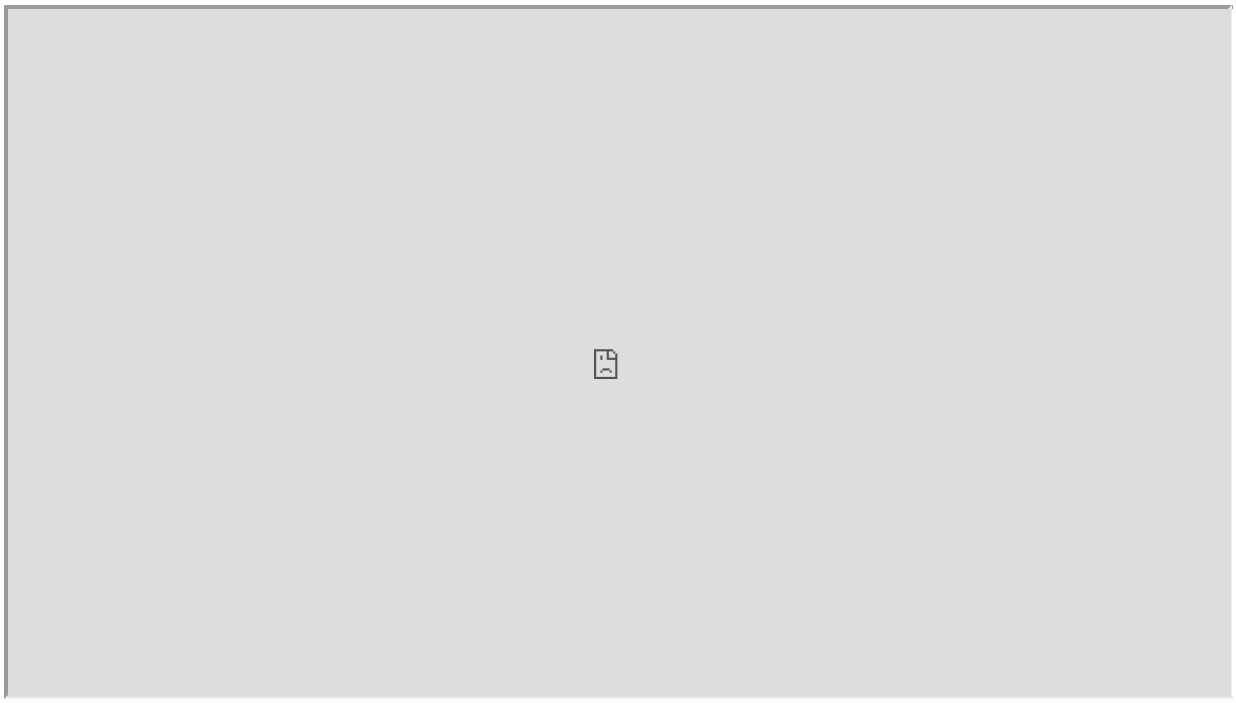
Geometry

Guide Index
: Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Create Guide Index Map

Creates an integer attribute named `guide_curve_index` that stores the nearest guide curve for every curve to its nearest guide via index.

Other nodes in the Hair Guides Nodes category can generate guide maps themselves for convenience, but the behavior is always the same as this node.



Inputs

Geometry

Guides
: Guide Curves or Points used for the selection of Guide Curves.

Guide Distance
: Minimum distance between two guides.

Guide Mask
: Mask for which curve are eligible to be selected as guides.

Group ID
: ID to group together curves for guide map creation. Curves will only choose a guide with the same ID value.

Properties

This node has no properties.

Outputs

Geometry
: Output geometry including the new map attribute and the guide selection anonymous attribute as well. This geometry includes the guide curves, they are not separated.

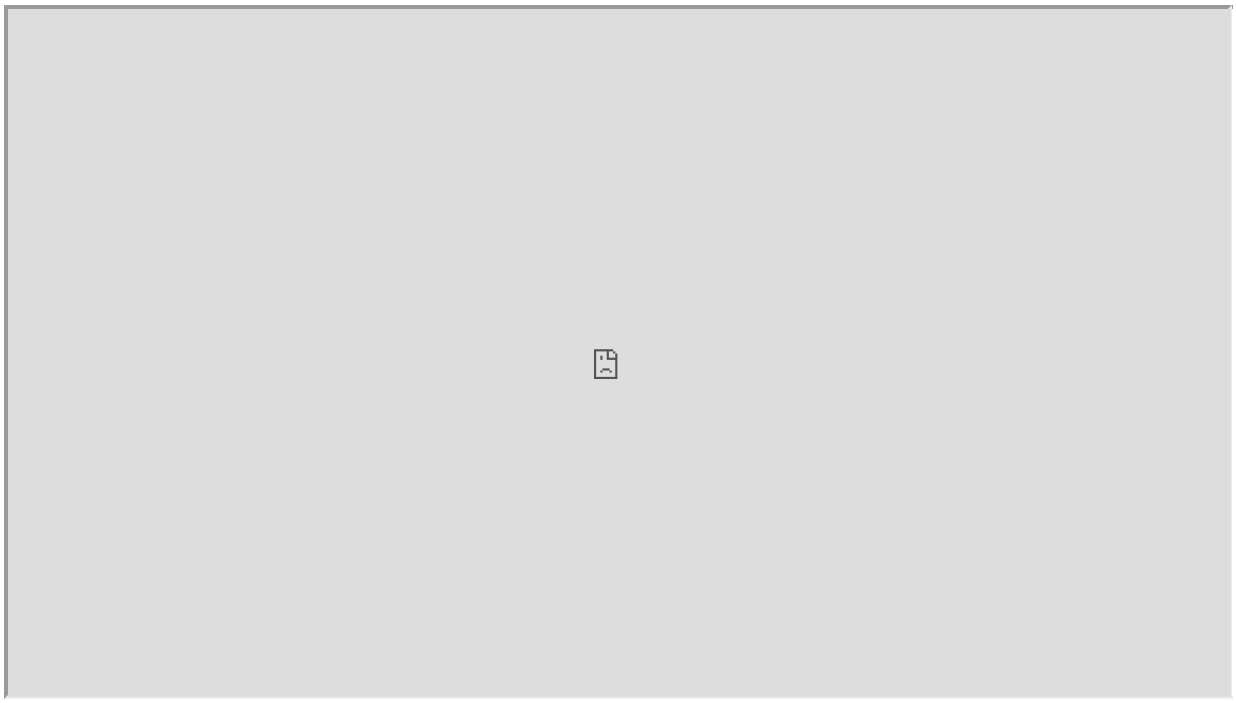
Guide Curves
: Output geometry including only the selected guide curves.

Guide Index
: The index of the closest curve with the same *Group ID* value.

Guide Selection
: A selection in the *Geometry* output set to true for only the curves that were chosen as guides.

Curl Hair Curves

Deforms existing hair curves into curls using guide curves.



Inputs

Geometry

Guide Index
: Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance
: Minimum distance between two guides for new guide map.

Guide Mask

Mask for which curve are eligible to be selected as guides.

Existing Guide Map
: Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn't provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor
: Factor to blend overall effect.

Subdivision
: Subdivision level applied before deformation.

Curl Start
: Percentage along each curve to blend deformation from the root.

Radius
: Overall radius of the curls.

Factor Start
: Factor for the radius at the curl start.

Factor End
: Factor for the radius at the curl end.

Frequency
: Frequency factor of the curls. This input can vary for different points of the same curve.

Random Offset
: Amount of random offset per curve.

Seed
: Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

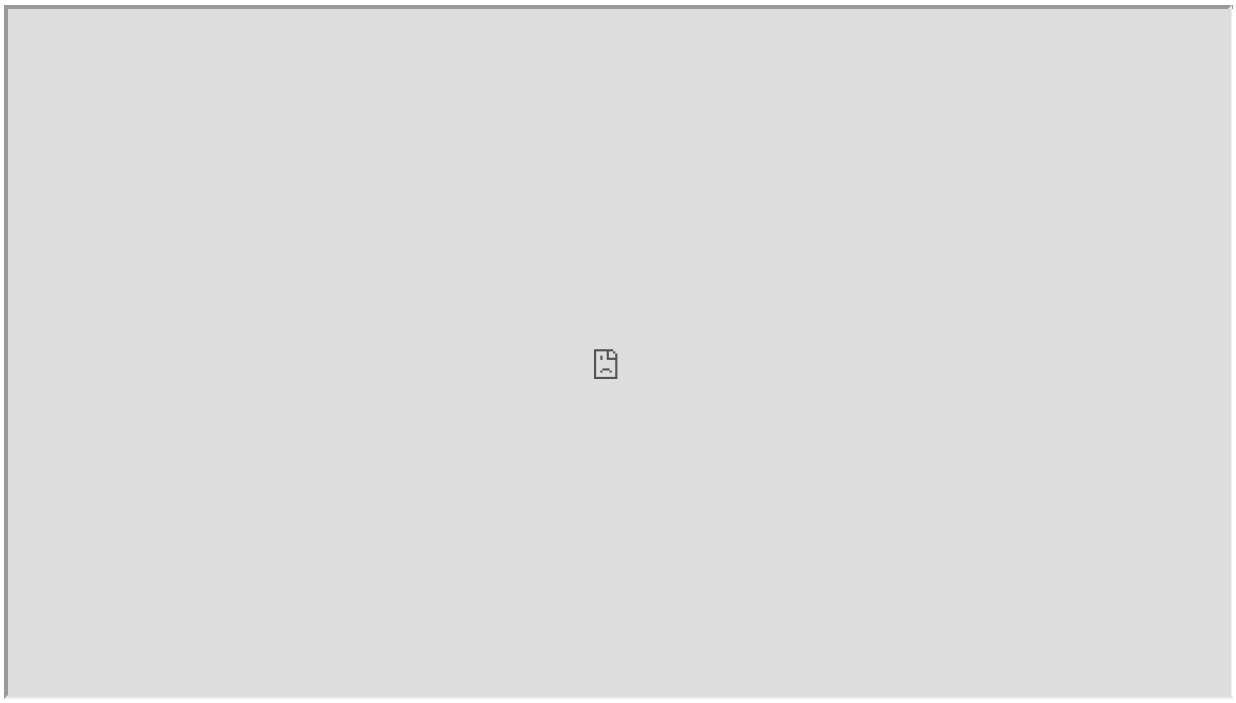
Guide Index
: Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Hair Read Nodes

- Curve Info
- Curve Root
- Curve Segment
- Curve Tip
- Hair Attachment Info

Curve Info

Reads information about each curve.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Curve Index
: Index of each Curve.

Curve ID
: ID of each Curve.

Length
: Length of each Curve.

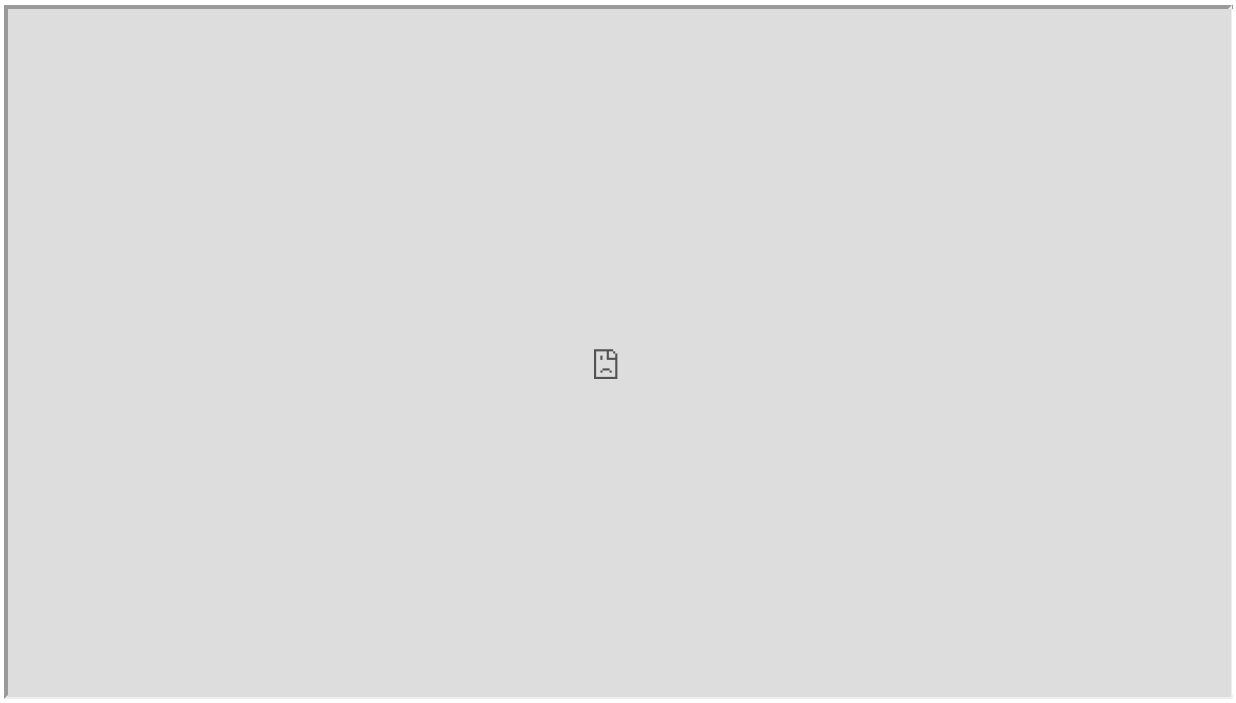
Direction
: Direction from root to tip of each Curve.

Random
: Random vector for each Curve.

Surface UV
: Attachment surface UV coordinate of each Curve.

Curve Root

Reads information about each curve’s root point.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Root Selection
: Boolean selection of curve root points.

Root Position
: Position of the root point of a Curve.

Root Direction
: Direction of the root segment of a Curve.

Root Index
: Index of the root point of a Curve.

Curve Segment

Reads information each point's previous curve segment.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

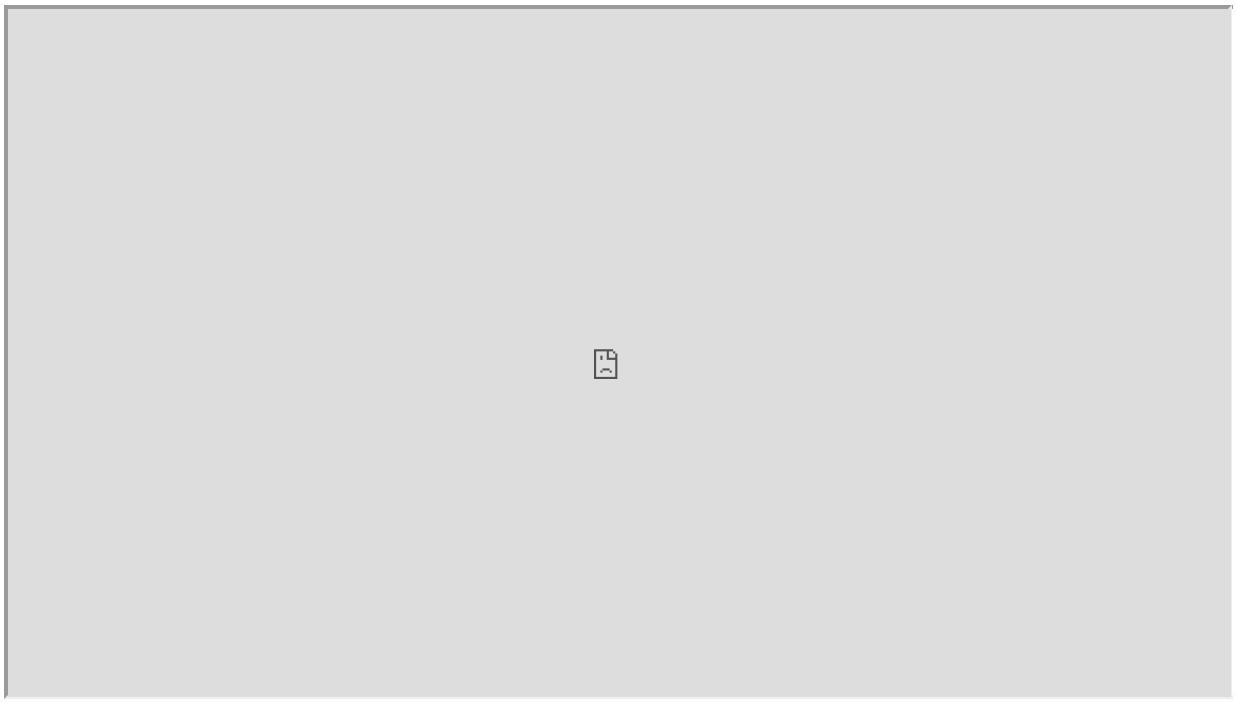
Segment Length
: Distance to previous point on Curve.

Segment Direction
: Direction from previous neighboring point on segment.

Neighbor Index
: Index of previous neighboring point on segment.

Curve Tip

Reads information about each curve’s tip point.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Tip Selection
: Boolean selection of curve tip points.

Tip Position
: Position of the tip point of a Curve.

Tip Direction
: Direction of the tip segment of a Curve.

Tip Index
: Index of the tip point of a Curve.

Hair Attachment Info

Reads attachment information regarding a surface mesh.

Inputs

Surface Geometry
: Surface geometry of the curve attachment.

Surface UV Map
: Surface UV map stored on the mesh used for finding curve attachment locations.

Properties

This node has no properties.

Outputs

Attachment UV
: Surface attachment UV coordinate stored on each curve.

Attachment is Valid
: Whether the stored attachment UV coordinate is valid.

Surface Normal
: Normal direction of the surface mesh at the attachment point.

Hair Utility Nodes

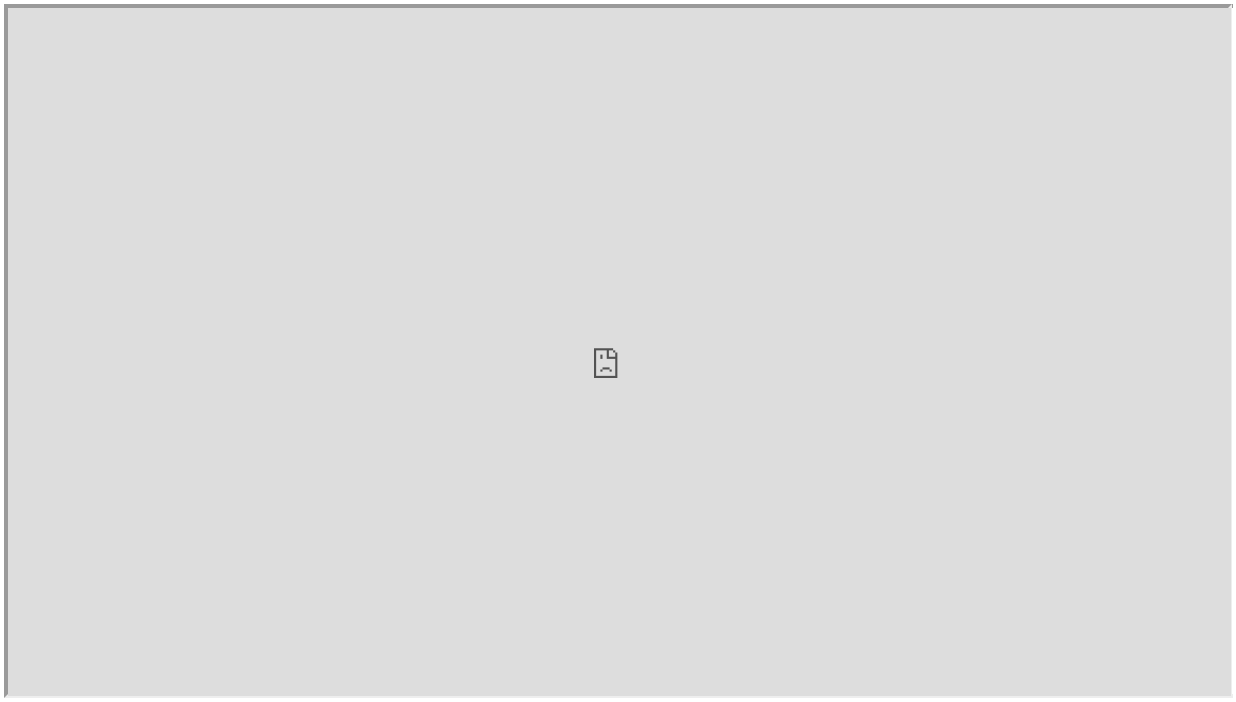
- Attach Hair Curves to Surface
- Redistribute Curve Points
- Restore Curve Segment Length

Attach Hair Curves to Surface

Attaches hair curves to a surface mesh.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.



Inputs

Geometry
: Input Geometry (only curves will be affected).

Surface
: Surface Geometry to attach hair curves to. This input takes priority over the corresponding object input, if used.

Surface
: Surface Object to attach to (needs to have matching transforms).

Surface UV Map
: Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position
: Set the surface mesh into its rest position before attachment.

> **Tip**
>
> In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Sample Attachment UV
: Sample the surface UV map at the attachment point.

Snap to Surface
: Snap the root of each curve to the closest surface point.

Align to Surface Normal
: Align the curve to the surface normal (need guide as reference).

Blend along Curve
: Blend deformation along each curve from the root.

Properties

This node has no properties.

Outputs

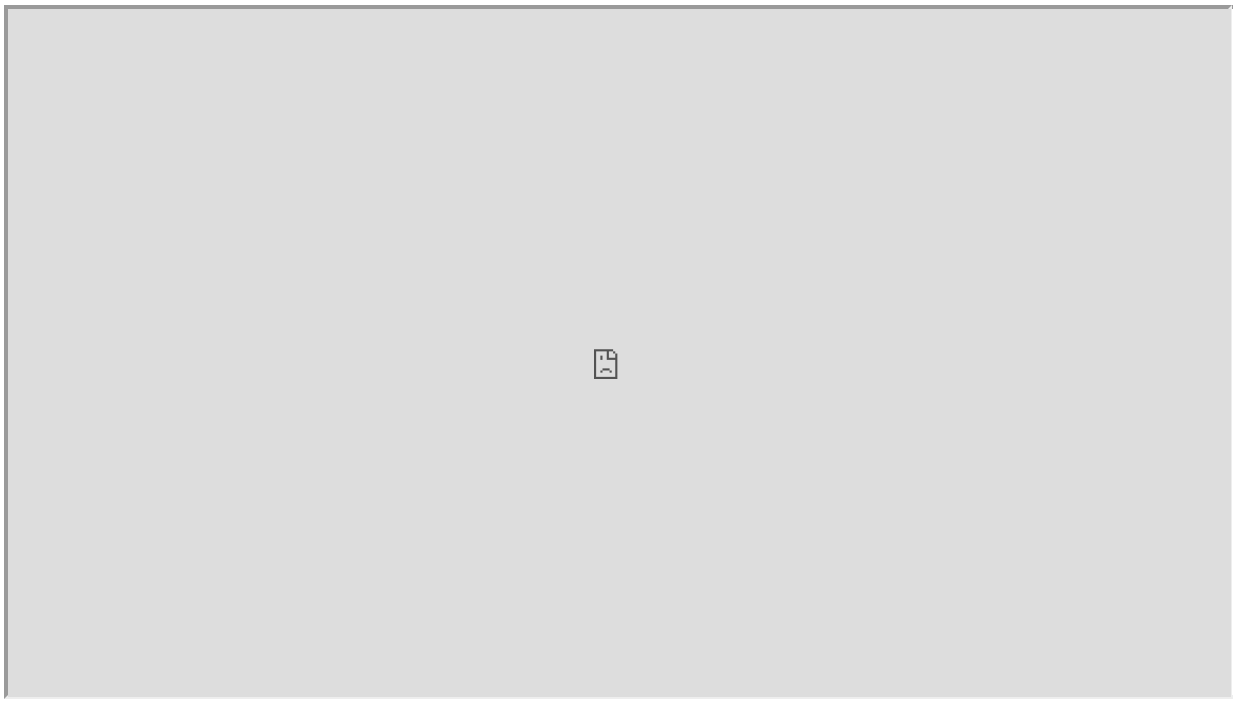
Geometry

Surface UV Coordinate
: Surface UV coordinate at the attachment point.

Surface Normal
: Surface normal at the attachment point.

Redistribute Curve Points

Redistributes existing control points evenly along each curve.



Inputs

Curves

Factor
: Factor to blend overall effect.

Feature Awareness
: Use simple feature awareness to keep feature definition.

Properties

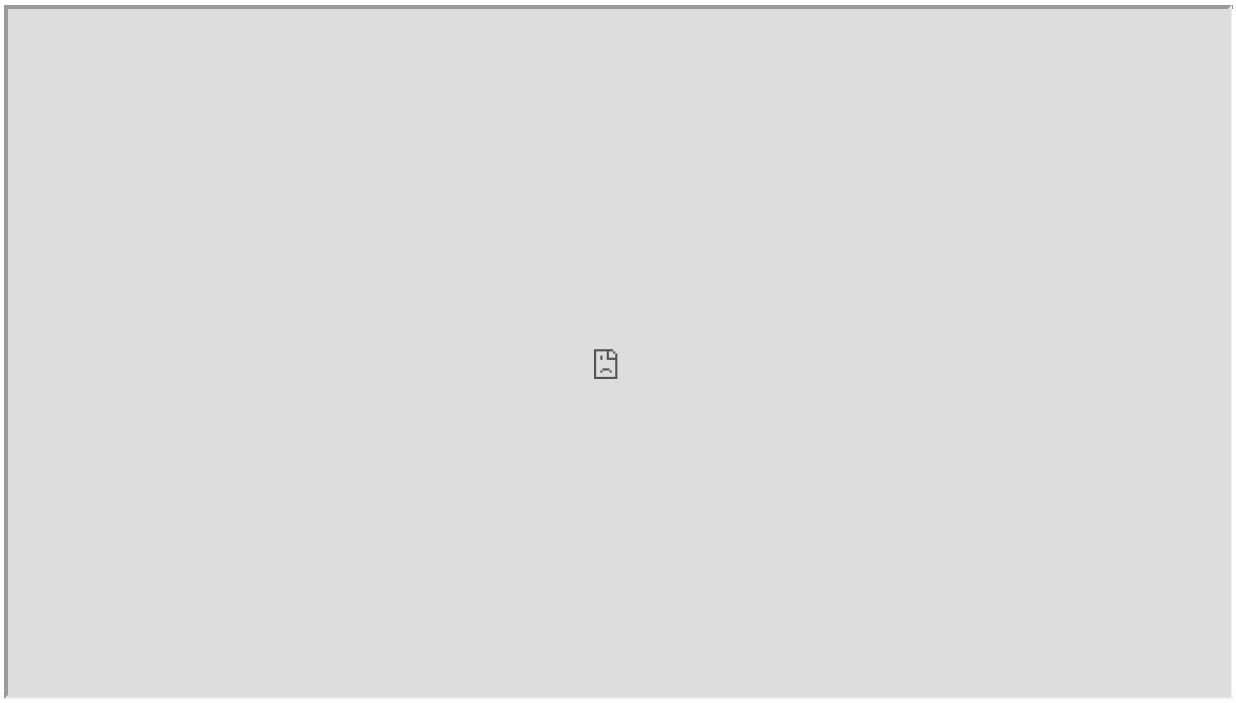
This node has no properties.

Outputs

Curves

Restore Curve Segment Length

Restores the length of each curve segment using a previous state after deformation.



Inputs

Curves

Selection
: Only affect selected elements.

Factor
: Factor to blend overall effect.

Reference Position
: Reference position before deformation.

Pin at Parameter
: Pin each curve at a certain point for the operation.

Properties

This node has no properties.

Outputs

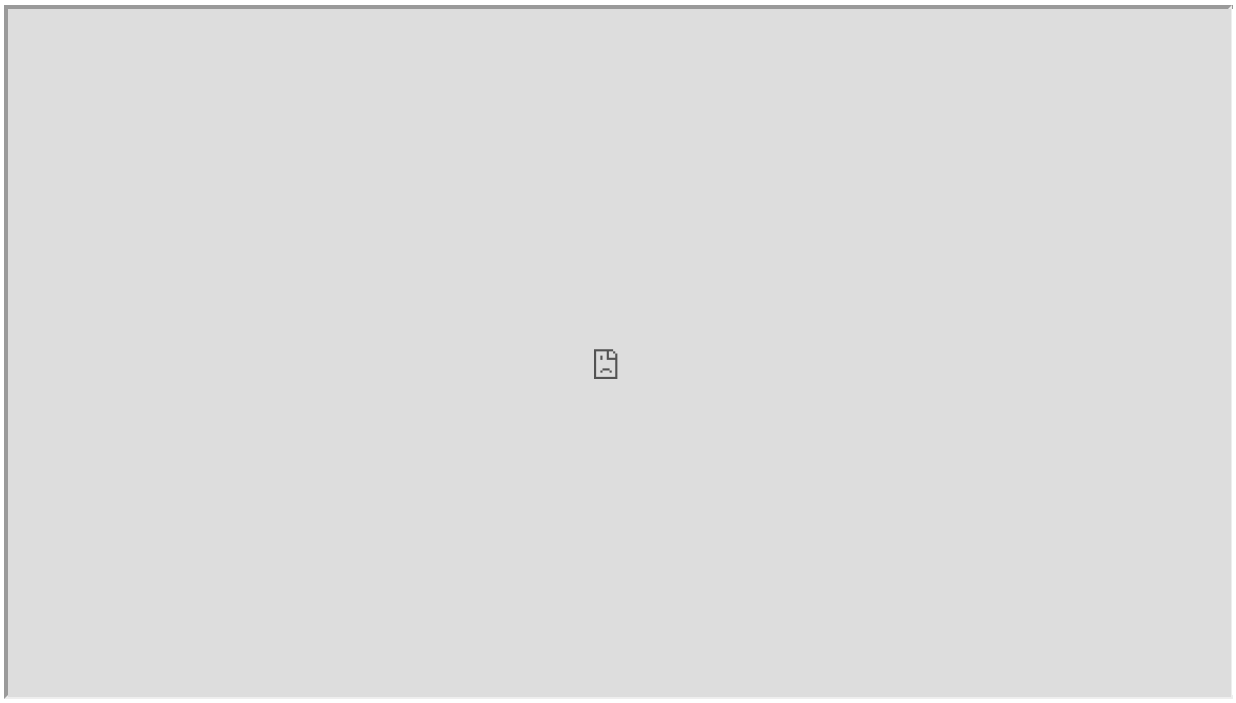
Curves

Write Hair Nodes

- Set Hair Curve Profile

Set Hair Curve Profile

Sets the radius attribute of hair curves according to a profile shape.



Inputs

Geometry

Replace Radius
: Replace the original radius.

Radius
: Base radius to be set if *Replace Radius* is enabled.

Shape
: Shape of the radius along the curve.

Factor Min

Factor of the radius at the minimum.

Factor Max
: Factor of the radius at the maximum.

Properties

This node has no properties.

Outputs

Geometry

Normals Nodes

- Smooth By Angle Node Group

Smooth By Angle Node Group

Set the sharpness of mesh edges based on the angle between the neighboring faces.

Note

This is a node group asset that is included in the bundled "Essentials" asset library.

Inputs

Mesh
: Standard geometry input.

Angle
: Maximum angle between face normals that will be considered as smooth.

Ignore Sharpness
: Smooth all edges, even if they have been marked as sharp.

Properties

This node has no properties.

Outputs

Mesh
: Standard geometry output.

Transform

- Introduction
  - Operators
- Transform Modal Map
  - Constraints
  - Snapping
  - Navigating

Introduction

Transform is the modality of operations that perform transformations in 2D and 3D elements. Transformations can include things like moving, rotating, scaling, and applying other operations to objects in the scene.

They work by changing the geometry which you can edit directly.

Operators

There are several transformation operations included in Blender. Here are some of the main operations available:

Move

This operations allows you to move elements along the X, Y, and Z axes in the scene.

Rotate

You can use this function to rotate elements around the X, Y, and Z axes.

Scale

Scaling allows you to increase or decrease the size of an object along the X, Y, and Z axes.

Align to View

This is useful for aligning objects with the view from the camera or another specific viewpoint.

Mirror

Mirrors objects along one or more axes.

Transform Modal Map

During a transformation, some hotkeys can change the behavior of the operation.

You can check editing the keys of these modal modifiers in *Blender Preferences ▸ Keymap ▸ Transform Modal Map* (at the bottom of the keymap).

Constraints

When moving, rotating or scaling, if you only want certain axes to be affected, you can restrict the transformation to those axes.

By default the constraint keys are `X`, `Y` and `Z`. This constraint can be restricted to a plane if `Shift` is pressed or automatically detected if `MMB` is pressed.

It is worth noting that if you press the same contraint hotkey a second time, you change the orientation from Local to Global or vice versa. Pressing a third time disables the constraint.

Snapping

Transform operations use the snapping settings set in the scene. However, some options can be changed during the transformation.

Snap Invert

Even if the magnetic icon is disabled, you can still enable snapping during a transformation. The default hotkey in this case is `Ctrl`.

Set Snap Base

The Snap Base is automatically determined based on the Snap Base options. However, this automatic snap detection point of origin may not always align with the user's intentions. Therefore, transform operations include a utility to set a new snap origin point during the transformation. The new Snap Base will correspond to the snap point whose target is defined by the Snap Target.

By default the hotkey is `B`.

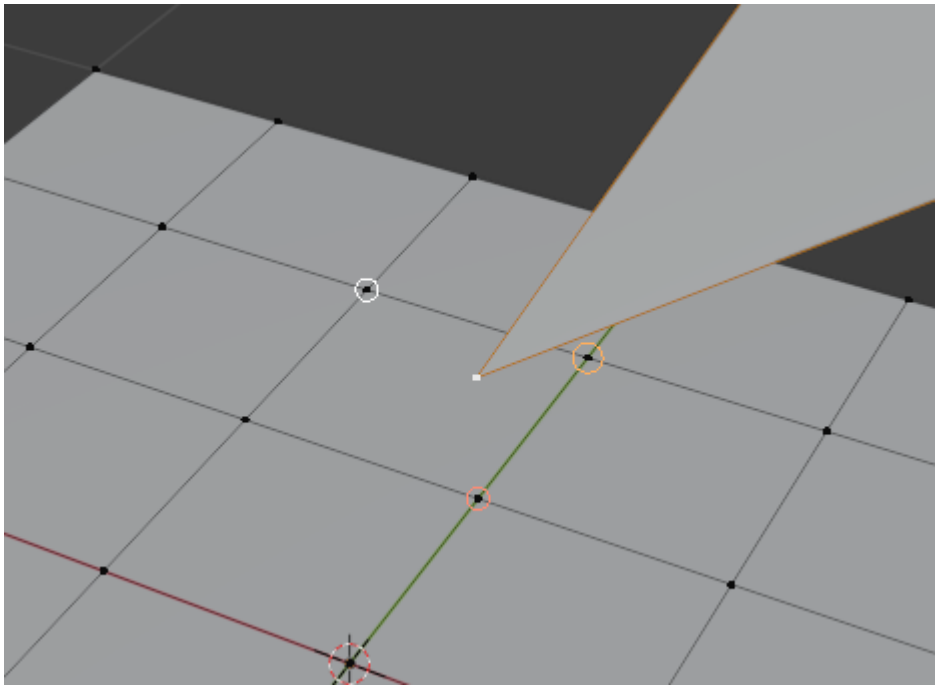
Note

If only Snap to Increment is enabled as Snap Target, the targets **Vertex**, **Edge**, **Face** and **Edge Center** will be used instead.

Add Snap Point

While you're transforming a selection with snapping enabled, you can press `A` whenever there's a highlighted snap target to mark it. With multiple such targets marked, the selection will then be snapped to their average location.

Marking a target more than once will give it more weight.



Multiple snapping targets.

Navigating

While performing a modal transformation, you can perform navigation actions such as zooming, panning, or rotating by holding `Alt` then perform the desired action.

Sculpting & Painting

- Introduction
- Brushes
  - Introduction
  - Manage Brushes
  - Brush Settings
- Selection & Visibility
  - Selection Masking
  - Select Linked
- Navigation

Modes

- Sculpting
  - Introduction
  - Brushes
  - Toolbar
  - Tools
  - Tool Settings
  - Controls
  - Editing
- Texture Paint
  - Introduction
  - Brushes
  - Tool Settings
- Vertex Paint
  - Introduction
  - Brushes
  - Vertex Paint Tools
  - Tool Settings
  - Editing
- Weight Paint
  - Introduction

    - Brushes
    - Weight Paint Tools
    - Tool Settings
    - Using Vertex Groups
    - Editing
- Curves Sculpting
    - Introduction
    - Brushes
    - Common Settings

Introduction

Sculpting and painting offers a more freeform workflow of editing via brushes. There are several modes to do this, each with their own purpose.

- Sculpting: Change and transform the topology of your mesh.
- Vertex Paint: Change the color of vertices in the active Color Attribute.
- Weight Paint: Change the weight of vertices in the active vertex group.
- Texture Paint: Change the pixels of the active image texture.

Brushes

- Introduction
  - Accessing Brushes
  - Brush Control
  - Custom Brush Shortcuts
  - Brush Assets
  - Brush Tool
- Manage Brushes
  - Asset Operators
  - Manual Storage

Brush Settings

- Brush Settings
- Texture & Texture Mask
- Stroke
- Falloff
- Cursor

Introduction

Brushes are the main way of interacting with any painting and sculpting mode. By click & dragging in the 3D Viewport (or the Image Editor when using Texture Paint), the active brush creates a stroke with a certain effect, depending on the used brush settings. Brushes are used as brush assets and stored in asset libraries, which makes it easy to reuse and share them. Typically they have a preview image and a name that indicate the effect they create.

Tip

It is highly recommended to use a Graphics Tablet for a better brush feel and additional features.

Accessing Brushes

In modes that use painting or sculpting functionality, the Asset Shelf of the 3D Viewport and Image Editor displays brush assets that can be used in that mode. Clicking a brush asset will activate the Brush Tool if necessary, with the clicked brush set.



The Asset Shelf of the 3D Viewport, providing access to brush assets.

This asset shelf is also available as popup in the Tool Settings, the Sidebar, Properties and using a shortcut.

Reference

Mode::	All Paint Modes
Header::	*Tool Settings*
Panel::	*Sidebar* ▸ *Tool* ▸ *Brush Asset*, *Properties* ▸ *Tool* ▸ *Brush Asset*
Shortcut::	`Shift-Spacebar`

Brush Control

These are the most common hotkeys for controlling the brush.

- Set brush size `F`
- Set brush strength `Shift-F`
- Rotate brush texture / Set brush weight `Ctrl-F`

After pressing these hotkeys, you can then either adjust the value interactively or by typing in numbers. Move the mouse right or left to increase/reduce the value (additionally with precision (`Shift`) and/or snapping (`Ctrl`) activated). Finally confirm (`LMB`, `Return`) or cancel (`RMB`, `Esc`).

You can also invert the brush direction/effect by holding `Ctrl`.

Custom Brush Shortcuts

To give a brush a shortcut, simply right click it in the asset shelf or brush selector popup, and select *Assign Shortcut*. To modify or remove an existing shortcut, select *Change Shortcut* or *Remove Shortcut* accordingly.

Brush Assets

Brushes are used as assets, and stored in asset libraries. This makes the brushes shared across project files. All available brush assets can be displayed in the Asset Browser, which also provides ways to organize them.

Blender comes bundled with a number of brushes in the *Essentials* asset library. These can be customized into all kinds of custom brushes by

duplicating them (see Brush Editing).

While it's possible to have brush data-blocks that are local to the file and not marked as assets, such brushes cannot be activated for actual painting or sculpting. Use the *Mark as Asset* operator to make them brush assets that can be activated.

Brush Tool

Painting or sculpting with brushes requires the brush tool to be active. Activating a brush from an asset shelf or brush selector also activates the brush tool for convenience.



The *Brush* tool.

Manage Brushes

Brush assets are stored in asset libraries to make them accessible from any Blender session. There are two ways of managing brush assets:

- Using asset operators: Create and update brush assets using utility operators from any Blender file. Storage is managed by Blender. Convenient for simple “on the fly” management of personal brush asset libraries.
- Using manual storage: Create and update brush assets by opening blend files within asset libraries, and managing brush asset data-blocks in there. Useful for careful curation of asset libraries, especially to prepare them for sharing with others.

Asset Operators

Brushes can be managed through a few operators that let Blender handle the act of saving and updating the brushes in asset libraries for you. Assets managed this way will be saved in special asset system files using a *.asset.blend* file extension.

Note

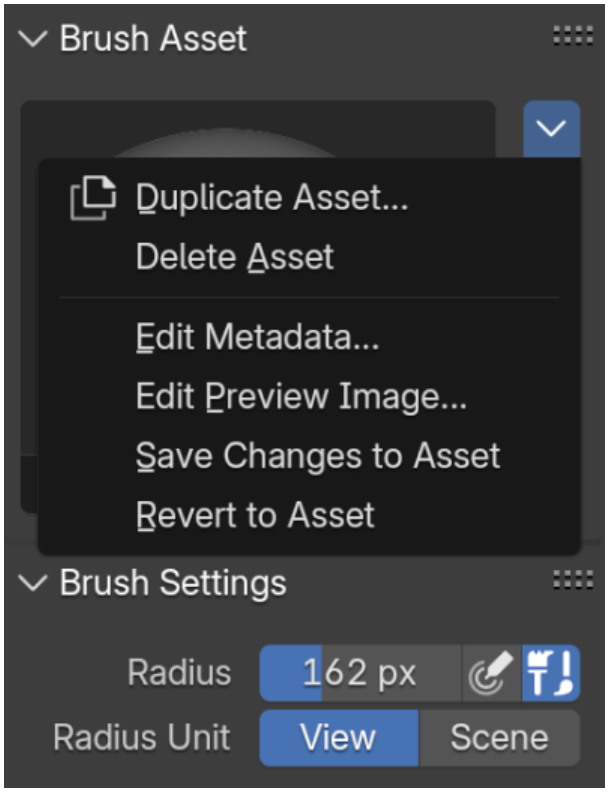
Note that only brush assets created via Duplicate Asset can be edited further using these asset operators. For others, these operations will be grayed out, and manual management is necessary.

Brushes from the *Essentials* asset library cannot be edited.

Reference

Mode::	All Paint Modes
Panel::	*Sidebar* ▸ *Tool* ▸ *Brush Asset Properties* ▸ *Tool* ▸ *Brush Asset*

Menu:: *Asset Shelf ▸ Context Menu*



Brush Asset panel in the Sidebar showing asset operators.

Duplicate Asset… (Duplicate icon)
: Creates a copy of the currently active brush as asset, and activates it. A popup is spawned to input some settings to use:

  Name
  : A custom name to use for the new brush.

  Library
  : Choose an Asset Library to store the new brush asset in. The available asset libraries are configured in the Preferences.

  Catalog
  : Choose an Asset Catalog to assign the brush asset to. Entering a non-existent name/path will create a new catalog accordingly.

Delete Asset
: Permanently remove this brush asset from the Asset Library it is stored in. This cannot be undone, so a popup will ask for confirmation.

Edit Metadata…
: Spawns a popup to change some of the available asset metadata fields:

  Catalog
  : Choose an Asset Catalog to assign the brush asset to. Entering a non-existent name/path will create a new catalog accordingly.

  Author
  : See Asset Author.

  Description
  : See Asset Description.

Edit Preview Image…
: Opens a window with the File Browser to select an image for the asset preview.

Save Changes to Asset
: Saves any changes made to the active brush to the asset library.

Revert to Asset
: Discards any unsaved changes made to the brush asset.

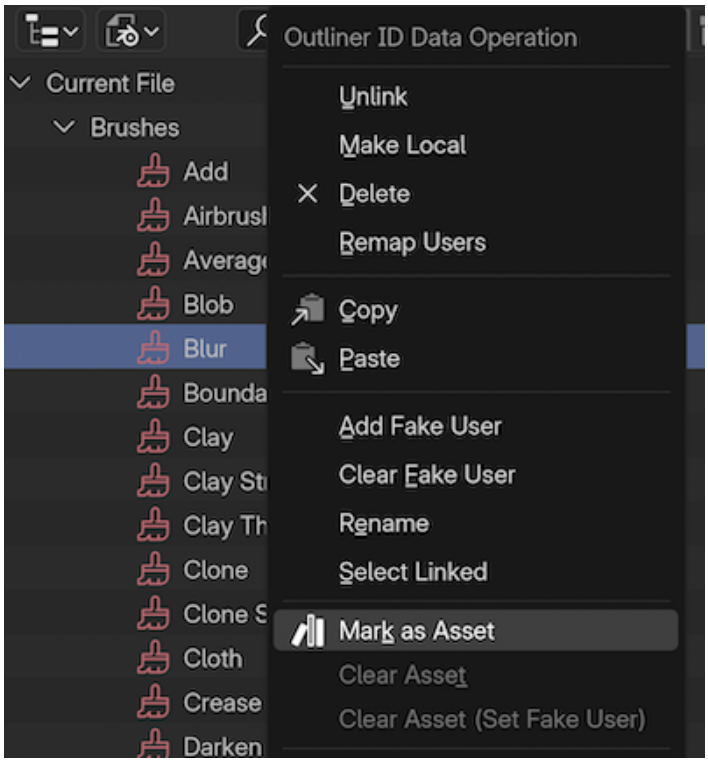
Manual Storage

See also

Life Cycle of an Asset
: Complete description of the manual asset create, edit, share and use workflow.

It is also possible to manually manage brushes in blend-files like any other asset data-block. By marking brushes as assets and saving the file in an asset library, they become available from any Blender session. This gives full control over how assets are stored, and is particularly useful for curating asset libraries that can be shared with others.

Brushes can be imported as normal data-blocks from other files (including from *.asset.blend* files from an asset library) through appending. In the Blender File mode of the Outliner, the brush will be listed under *Brushes*. Right-click the brush and select *Mark as Asset*. By saving the file inside of an asset library directory, the asset becomes available from all Blender sessions. If necessary, configure an asset library directory in the Preferences.



The *Mark as Asset* operator used on a brush in the Outliner.

Brush Settings

Each mode and brush has unique brush settings. But there is also a lot of overlap or similar settings. This page explains general and mode specific settings that are used across various brushes in more detail.

General

Radius
: This option controls the size of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse from left to right and then `LMB` to accept. Meanwhile the texture of the brush will be visible inside the circle. You can also enter the size numerically with the number keys.

  The size can be decreased/increased using `[` and `]` respectfully.

  (Size Pressure)
  : Adjusts the radius based on the stylus pressure when using a Graphics Tablet.

  (Use Unified Radius)
  : Use the same brush *Radius* across all brushes.

Radius Unit *Sculpt Mode*
: Controls how the brush *Radius* is measured.

View::	The *Radius* is measured based on how the cursor appears on the monitor i.e. “screen space”.
Scene::	The *Radius* is measured based on real world units. This means that the brush radius stays consistent, independently from zooming in and out in the viewport. The unit type and scaling can be configured in the Scene Units.

Strength

For painting brushes the *Strength* defines the maximum effect of each brush stroke. For example, higher values cause a *Paint* brush to give each stroke a higher opacity. The opacity is never stronger than the set *Strength*, no matter how often the same surface is painted during the same stroke.

For sculpting brushes on the other hand the *Strength* relates to how strong each step of the stroke is, resulting in a slower/faster buildup towards the full brush effect during the stroke.

You can change the brush strength interactively by pressing `Shift-F` and then moving the brush and then `LMB`. You can also enter the strength numerically with the number keys.

(Strength Pressure)

Adjusts the strength based on the stylus pressure when using a Graphics Tablet.

(Use Unified Strength)

Use the same brush *Strength* across all brushes.

Blend

Set the way the color or value is applied over the targeted Color Attribute, Vertex Group or Image Texture. See Color Blend Modes.

- Add Alpha: makes the image more opaque where painted.
- Erase Alpha: makes the image transparent where painted, allowing background colors and lower-level textures to show through. As you "paint", the false checkerboard background will be revealed. Using a tablet pen's eraser end will toggle on this mode.

Tip

In order to see the effects of the Erase and Add Alpha mix modes in the Image Editor, the Display Channels must be set to *Color & Alpha*

or *Alpha*. Transparent (no alpha) areas will then show a checkered background.

Weight *Weight Paint*
: The weight value that is applied to the vertex group.

  Use `Shift-X` to sample the weight value of clicked vertex. `Shift-Ctrl-X` lets you select the group from which to sample from.

Direction `Ctrl` *Sculpt Mode*
: Brush direction toggle, *Add* raises geometry towards the brush, *Subtract* lowers geometry away from the brush. This setting can be toggled with `Ctrl` while sculpting.

Normal Radius *Sculpt Mode*
: Determines the ratio of how much the brush radius is used to sample the normal direction of the sculpt plane of the brush. For example, a smaller *Normal Radius* will lead to drastic changes in the brush orientation, like for following the contours of hard surface meshes more closely. A large *Normal Radius* will lead to smoother changes in orientation, like for building overall forms on organic sculptures.

Area Radius
: The ratio between the brush radius and the radius that is going to be used to sample the area plane depth.

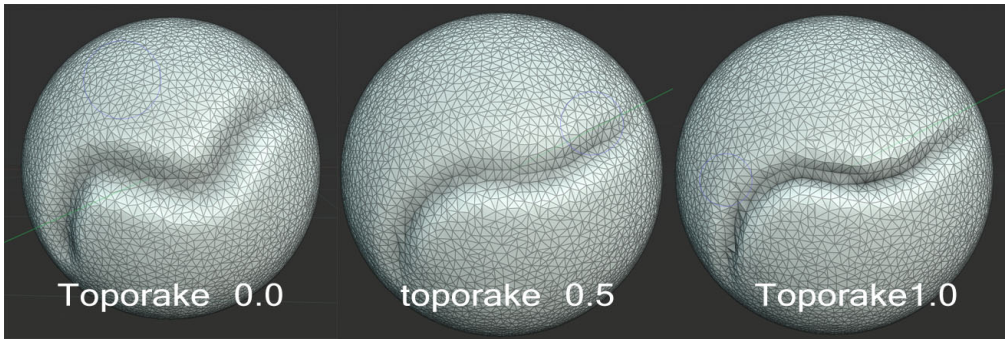
Hardness *Sculpt Mode*
: How close the brush falloff starts from the edge of the brush.

Tip Roundness
: The factor to control how round the brush is. A value of zero will make the brush square. Note, the Brush Falloff is only applied to the rounded portions of the brush.

Auto-smooth *Sculpt Mode*
: Sets the amount of smoothing to be applied to each stroke.

Topology Rake *Sculpt Mode*

The higher this setting is set, the more Dyntopo aligns mesh edges to the brush direction while tessellating the surface. This generates cleaner edge flow to help define sharp features. *Topology Rake* can have a severe performance impact so it works best on low-poly meshes.



Normal Weight `Ctrl` *Sculpt Mode*

Constrains brush movement along the surface normal. Especially useful with the *Grab* brush, can be temporarily enabled by holding `Ctrl`. E.g. *Grab* brush can be used to push a depression (hole) into the mesh when *Normal Weight* is set.

Applies to *Grab* and *Snake Hook* brushes.

Plane Offset *Sculpt Mode*

Offset for planar brushes (Clay, Fill, Flatten, Scrape), shifts the plane that is found by averaging the faces above or below.

Plane Trim *Sculpt Mode*

Ability to limit the distance that planar brushes act. If trim is enabled vertices that are further away from the offset plane than the trim distance are ignored during sculpting.

Pinch/Magnify *Sculpt Mode*

Pushes the mesh towards/away from the brush center during the stroke.

Deformation Target

How the deformation of the brush will affect the object.

Geometry:: Deform the geometry directly.
Cloth Simulation:: Deform the mesh while a cloth simulation is applied to it at the same time.

Advanced

Brush Type
: Defines the basic behavior and the available settings. Through the settings of a brush type, brushes can be created that produce vastly different effects.

  The *Essentials* asset library contains brushes for each of the brush types. Their preview image and description should give a good idea of the effect the brush produces, with the particular combination of brush type and settings. Because of this, they are usually the more useful starting point for custom brushes than the mere brush type is, which is why the brush type is part of the *Advanced* brush settings.

  Brushes and Brush Types of each mode:

  - Sculpt
  - Vertex Paint
  - Weight Paint
  - Texture Paint

Accumulate
: Causes stroke dabs to accumulate on top of each other.

Front Faces Only
: When enabled, the brush only affects vertices that are facing the viewer.

Affect Alpha *2D Painting Only*
: When this is disabled, it prevents changes to the alpha channel while painting (Only in 3D Viewport).

Anti-Aliasing *2D Painting Only*

Toggles Anti-Aliasing around the brush, this is useful if you are working with pixel art or low resolution textures.

Auto-Masking *Sculpt Mode*

The auto-masking toggles in the brush settings are the same as the sculpt mode auto-masking settings. The difference is that these toggles can be customized per brush to create specific brush behaviors.

See also

For more information on the Auto-Masking toggles, see Auto-Masking.

Sculpt Plane *Sculpt Mode*

Use this menu to set the plane in which the sculpting takes place. In other words, the primary direction that the vertices will move.

Area Plane::	The movement takes place in the direction of average normal for all active vertices within the brush area. Essentially, this means that the direction is dependent on the surface beneath the brush.
View Plane::	Sculpting in the plane of the current 3D Viewport.
X, Y, Z Plane::	The movement takes place in the positive direction of one of the global axes.

Original *Sculpt Mode*

Normal

When locked it keeps using the normal of the surface where stroke was initiated, instead of the surface normal currently under the cursor.

Plane

When locked keep using the plane origin of surface where stroke was initiated, instead of the surface plane currently under the cursor.

Color Picker

Color

The color of the brush. See Color Picker.

Press `Shift-X` on any part of the image to sample that color and set it as the brush color. Hold `Ctrl` while painting to temporally paint with the secondary color.

Swap Colors (cycle icon) `X`
: Swaps the primary and secondary colors.

(Use Unified Color)
: Use the same brush color across all brushes.

Note

Note that Vertex Paint works in sRGB space, and the RGB representation of the same colors will be different between the paint tools and the materials that are in linear space.

Gradient

A gradient can be used as a color source.

Gradient Colors
: The Color Ramp Widget to define the gradient colors.

Mode

Pressure::	Will choose a color from the color ramp according to the stylus pressure.
Clamp::	Will alter the color along the stroke and as specified by *Gradient Spacing* option. With *Clamp* it uses the last color of the color ramp after the specified gradient.

Repeat:: Similar to *Clamp*. After the last color it resets the color to the first color in the color ramp and repeats the pattern.

Color Palette

Color Palettes are a way of storing a brush's color so that it can be used at a later time. This is useful when working with several colors at once.

Palette
: A Data-Block Menu to select a palette.

New +
: Adds the current brush's primary *Color* to the palette.

Delete –
: Removes the currently selected color from the palette.

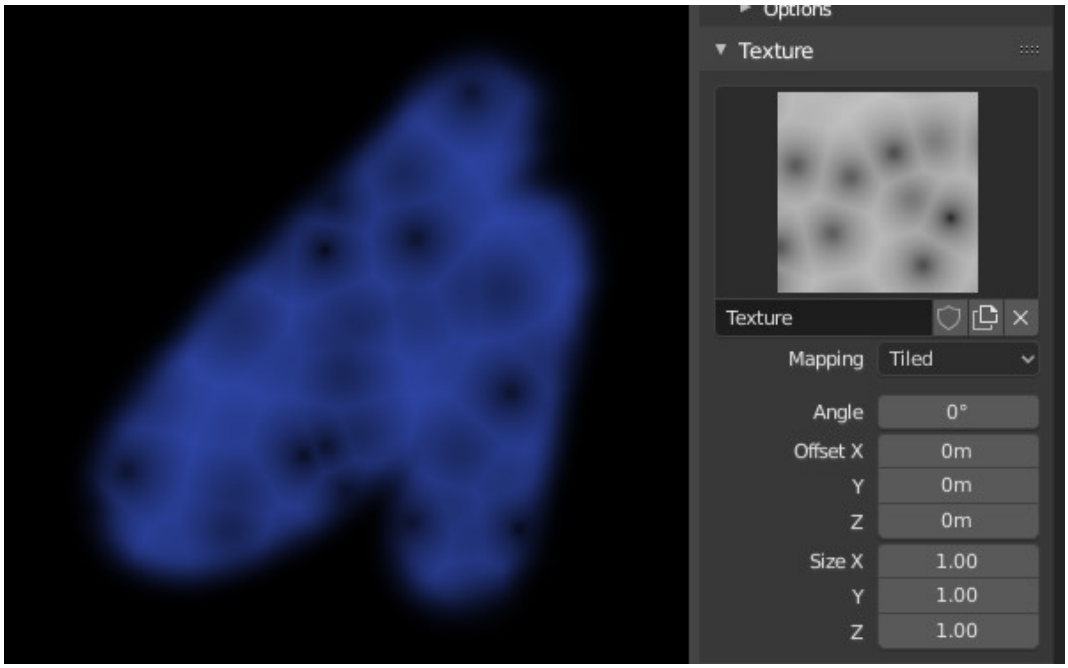
Move (up/down arrow icon)
: Moves the selected color up/down one position.

Sort
: Sort Colors by Hue, Saturation, Value, Luminance.

Color List
: Each color that belongs to the palette is presented in a list. Clicking on a color will change the brush's primary *Color* to that color.

Texture & Texture Mask

This page covers both the Texture and Texture Mask panels. Add a Texture to the brush to control the color of the brush. A Texture Mask is used to control the strength of the brush. Both the Texture and Texture Mask offer the same settings.



Example of the Texture panel and a textured brush in use.

Texture

In paint modes the texture is used as a color source, while for sculpting it is used to determine the strength of the brush.

Any image texture or procedural texture can be assigned in the texture and texture mask panels. Textures can be further edited in the properties editor (Click the properties icon for quick access)

Tip

It's recommended to load all needed images ahead of time as image textures into Blender. Then they can be easily selected by clicking on the texture and picking it from the data-block popup. Textures can also be appended/linked from other Blender files.

Mapping & Mask Mapping

How the texture is applied to the brush stroke.

Tip

It is recommended to set this to *Area Plane* or *View Plane* for the most common behavior. Ideally match this setting with the Sculpt Plane setting if in sculpt mode.

View Plane:: If *View Plane* is enabled, the current viewing angle is used to project the brush texture onto the model. This is especially useful for projection painting.

Area Plane:: Projects the brush texture along the local surface normal, which keeps the texture from stretching on steep angles. This is an ideal default for most brushes.

Tiled:: The *Tile* option repeats the texture across the screen, so moving the brush will not change where the texture is applied. The *Tile* option is most useful with tileable images, rather than procedural textures.

3D:: The *3D* option allows the brush to take full advantage of procedural textures. This mode uses vertex coordinates rather than the brush location to determine what area of the texture to use. This option is not available for the Texture Mask.

Random:: Picks a random texture coordinate to sample from for each step of the stroke.

Stencil:: This is the ideal option for stamping textures for projection painting. Stencil mapping works by projecting the texture from the camera space on the mesh or canvas. Painting is applied only inside the boundaries of the stencil. The stencil is displayed as a screen space overlay on the viewport. To transform the stencil texture use the following shortcuts (Hold `Alt` for the Texture Mask):

- Move `RMB`
- Scale `Shift-RMB`
- Rotate `Ctrl-RMB`

While using stencil scaling, `X` and `Y` are used to constrain the scaling to one axis. Pressing one of the buttons twice reverts to unconstrained scaling.

Image Aspect
: Restore the aspect ratio of the original image to reset stretching introduce by scaling, (Image textures only.) This operator can use the tiling and scale values of the brush texture if the relevant are enabled in Adjust Last Operation panel.

Reset Transform
: Restores the position of the stencil.

Pressure Masking
: Only available for the Texture Mask. It allows to clip the mask result based on pressure.

Off:: Disabled.
Ramp:: Fades out the mask effect on higher pressure.
Cutoff:: Expands the used values from the image based on stylus pressure.

Angle `Ctrl-F`
: This is the rotation angle of the texture brush. It can be changed interactively via `Ctrl-F` in the 3D Viewport. While rotating the angle

via the hotkey you can enter a value numerically as well.

Rake
: Texture angle follows the direction of the brush stroke. Useful for stamping textures repeatedly along the stroke. Not available with *3D*, *Tiled*, or *Stencil* Mapping types. The shortcut is not available in Sculpt mode.

Random
: Angle is randomized on each step of the stroke. Not available with *3D*, *Tiled*, or *Stencil* Mapping types. The shortcut is not available in Sculpt mode.

  Random Angle
  : Constraints the random deviation to a range.

Offset X, Y, Z
: Offset the texture map placement in X, Y, and Z axes.

Size X, Y, Z
: Set the scale of the texture in each axis.

Sample Bias *Sculpt Mode*
: Value added to texture samples. This can be used if the mid-level of a height map is not correct.

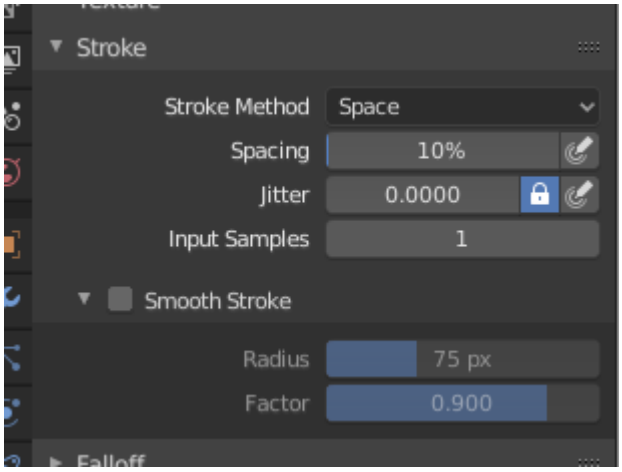
Vector Displacement *Sculpt Mode*
: Use the color channels to displace geometry in 3 vectors.

  Note

  This is only supported for the Draw brush with Area Plane mapping enabled.

Stroke

The stroke settings define the behavior of the sculpted/painted stroke. Any other brush behavior and effect is applied on top of the stroke.



Stroke panel.

Stroke Method `Alt-E`

Defines the way brush strokes are applied to the canvas.

Dots:: Apply paint on each mouse move step. This is regardless of their distance to each other, and instead depends on the stroke speed. This means that a slower stroke will have more accumulative strength applied.

Drag Dot:: Leaves only one dab on the canvas which can be placed by dragging.

Space:: Creates brush stroke as a series of dots, whose distance (spacing) is determined by the *Spacing* setting.

Spacing

Limits brush application to the distance specified by the percentage of the brush

radius.
(Spacing Pressure)
Brush *Spacing* can be affected by enabling the pressure sensitivity icon, if you are using a Graphics Tablet.

Airbrush:: Flow of the brush continues as long as the mouse click is held (spray), determined by the *Rate* setting.
Rate
Interval for how frequent the brush is applied during the stroke.

Anchored:: Creates a single dab at the brush location. Clicking and dragging will resize the dab diameter.
Edge to Edge
The brush location and orientation are determined by a two point circle, where the first click is one point, and dragging places the second point, opposite from the first.

Line:: Clicking and dragging lets you define a line in screen space. The line dabs are separated by *Spacing*, similar to space strokes. With `Alt` the line stroke is constrained to 45 degree increments.

Curve:: Defines the stroke curve with a Bézier curve (dabs are separated according to *Spacing*). This Bézier curve is stored in Blender as a "Paint Curve" data-block.
Use `Ctrl-RMB` to create the initial control point of the curve.
Paint Curves
Paint Curves are reusable and can be stored and selected by using the Data-Block Menu menu.

- **Add Points**
  You can define additional curve control points by using `Ctrl-RMB`. The handles can be defined by dragging the mouse. The stroke flows in the direction of the first control point to the second control point, and so on.

- **Transforming Points**
  The control points and handles can be dragged with `RMB` (In right click select with `LMB`). To make sure that the handles of a control point are symmetrical, drag them using `Shift-RMB`. A few transform operators are supported such as moving (`G`), rotating (`R`) and scaling (`S`).

- **Selection**
  The handles can be selected individually by using `LMB` (In right click select with `RMB`), extend the selection by `Shift-LMB` and deselect/select all by using `A`.

- **Delete Points :kbd:`X`**
  To delete a curve point, use `X`.

Draw Curve `Return`
: To confirm and execute the curved stroke, press `Return` or use the Draw Curve button.

Spacing Distance *Sculpt Mode Only*
: Method used to calculate the distance to generate a new brush step.

View:: Calculates the brush spacing relative to the view.

Scene:: Calculates the brush spacing relative to all three dimensions of the scene using the stroke location. This avoids artifacts when sculpting across curved surfaces and keeps the spacing much more consistent.

Adjust Strength for Spacing

Keep the brush strength consistent, even if the spacing changes. Available for the *Space*, *Line*, and *Curve* stroke methods.

Dash Ratio

Ratio of samples in a cycle that the brush is enabled. This is useful to create dashed lines in texture paint or stitches in Sculpt Mode. Available for the *Space*, *Line*, and *Curve* stroke methods.

Dash Length

Length of a dash cycle measured in stroke samples. This is useful to create dashed lines in texture paint or stitches in Sculpt Mode. Available for the *Space*, *Line*, and *Curve* stroke methods.

Jitter

Jitter the position of each step in the brush stroke.

(Jitter Pressure)

Brush *Jitter* can be affected by enabling the pressure sensitivity icon, if you are using a Graphics Tablet.

Jitter Unit

Controls how the brush *Jitter* is measured.

View::	The *Jitter* is relative to the view direction i.e. "screen space".
Scene::	The *Jitter* is measured relative to all three dimensions of the scene. The unit type and scaling can be configured in the Scene Units.

Input Samples

Recent mouse locations (input samples) are averaged together to smooth brush strokes.

Use Unified Input Samples

Use the same brush *Input Samples* across all brushes.

Stabilize Stroke

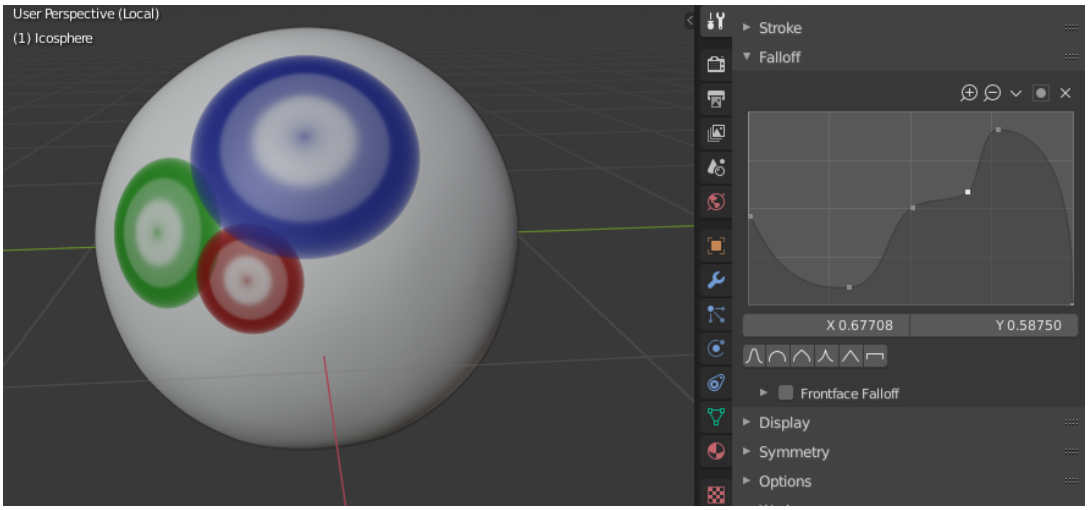
Stabilize Stroke makes the stroke lag behind the cursor and creates a smoothed curve to the path of the cursor. This can be enabled pressing `Shift S` or by clicking the checkbox found in the header.

Radius
: Minimum distance from the last point before the stroke continues.

Factor
: A smooth factor, where higher values result in smoother strokes but the drawing sensation feels like as if you were pulling the stroke.

Falloff

The Falloff allows you to control the *Strength* falloff of the brush. The falloff is mapped from the center of the brush (left part of the curve) towards its borders (right part of the curve). Changing the shape of the curve will make the brush softer or harder. Read more about using the Curve Widget.

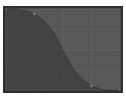


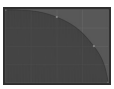
Brush curve example.

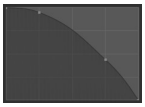
Curve Preset

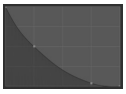
Custom:: You can choose how the strength of the falloff is determined from the center of the brush to the borders by manually manipulating the control points within the curve widget. There are also a couple of preset custom curves displayed at the bottom of the curve widget that can be used on their own or as a starting point for tweaking.

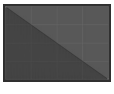
Custom Preset types.


Smooth.


Sphere.


Root.




Linear.



Sharp. Constant.

Smooth:: The center strength, the border strength, and the falloff transition between them are evenly distributed.

Smoother:: Similar to *Smooth* but produces a wider center point of the brush before tapering off.

Sphere:: The strength of the brush is predominately at its strongest point with a steep falloff near the border of the brush.

Root:: Similar to a Sphere but the center is a more concentrated point.

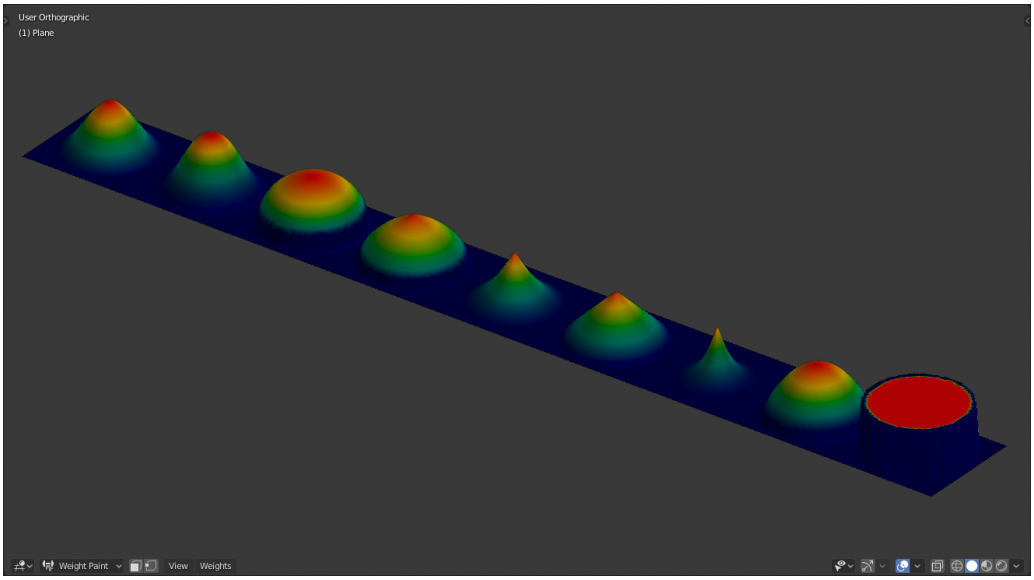
Sharp:: The center of the brush is the strongest point then exponentially tapers off to a lower strength, creating a fine point.

Linear:: With the center being the strongest, the strength will consistently weaken as it reaches the border of the brush.

Sharper:: Similar to *Sharp* but the center point is more condensed.

Inverse Square:: A hybrid between *Smooth* and *Sphere*.

Constant:: The strength of the brush remains unified across the entire brush. This will create a sharp edge at the border of the brush.



(From Left to Right) Smooth, Smoother, Sphere, Root, Sharp, Linear, Sharper, Inverse square, Constant.

Falloff Shape
: Use projected or spherical falloff. Note, this is not supported in Texture Paint Mode.

Sphere::	Applies brushes influence in a sphere, outwards from the center.
Projected::	This turns the brush influence into a cylinder (the depth along the view is ignored) instead of a sphere. It can be used along the outline of a mesh to adjust its silhouette.

Front-Face Falloff

As faces point away from the view the brush strokes fade away to prevent harsh edges.

Normal Falloff / Front-Face Falloff
: If disabled, the normal of the surface has no effect on the falloff.

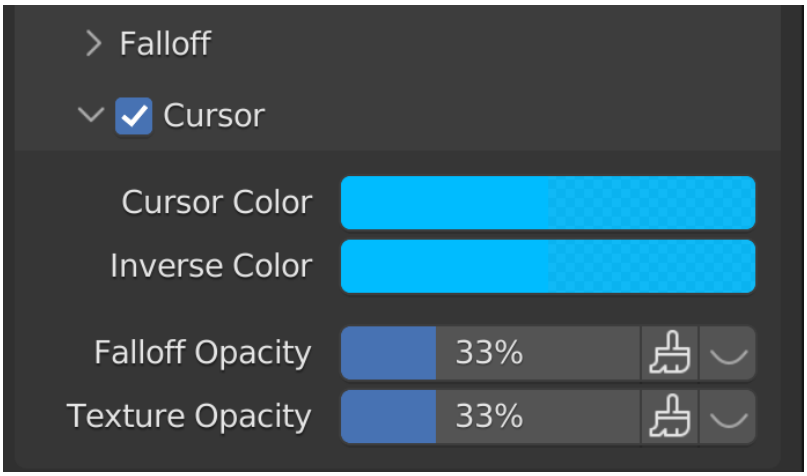
Angle
: The angle at which the falloff begins.

Cursor

Reference

Mode:: All Paint Modes
Header:: *Tool Settings* ▸ *Brush Settings* ▸ *Cursor*
Panel:: *Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Cursor*

While painting or sculpting a special cursor is shown to display information about the active brush. The cursor is shown as a circle in the 3D Viewport, the radius of the circle matches the size of the brush.



Cursor options.

The cursor can be disabled by toggling the checkbox in the panel's header.

Cursor Color
: Set the color of the brush ring while performing an add/positive stroke.

Inverse Color
: In some paint/sculpt modes the brush can be negative and subtract information from the paint target; these brushes can be given a separate color.

Opacity Options
: Depending on the paint or sculpt mode different overlays are shown within the cursor to give information on how the brush is textured. This

is most commonly used to show the brush falloff with a gradient from the circle center to the perimeter.

Alpha
: You can change the amount of transparency used when showing the texture using the slider.

Override Overlay (brush icon)
: Allows you to turn off the viewport overlay during strokes.

View (eye icon)
: Toggles whether to show or hide the given brush texture overlay.

Selection & Visibility

Selection Masking

If you have a complex mesh, it is sometimes not easy to paint on the intended vertices. Suppose you only want to paint on a small area of the Mesh and keep the rest untouched. This is where “selection masking” comes into play. When this mode is enabled, a brush will only paint on the selected vertices or faces. The option is available from the header of the 3D Viewport (see icons surrounded by the yellow frame):



> You can choose between *Face Selection masking* (left button), *Vertex selection masking* (middle button), and *Bone selection* (right button). The latter is only available when the mesh has an Armature modifier.

Selection masking has some advantages over the default paint mode:

- The original mesh edges are shown, even when modifiers are active.
- You can select and deselect faces instead without the need to switch to Edit Mode.

Details About Selecting

The following standard selection operations are supported:

- `Alt-LMB` – Single faces
- `Shift-Alt-LMB` – Select more or remove them from the selection.
- `A` – All faces, `A A` to deselect.
- `B` – Box selection.
- `C` – Circle select with brush.
- `Ctrl-I` – Invert selection.

- `L` – Pick linked (under the mouse cursor).
- `Ctrl-L` – Select linked.
- `Ctrl-NumpadPlus` – Extend Selection
- `Ctrl-NumpadMinus` – Shrink Selection

The following only work for face selection and with the selection tool active:

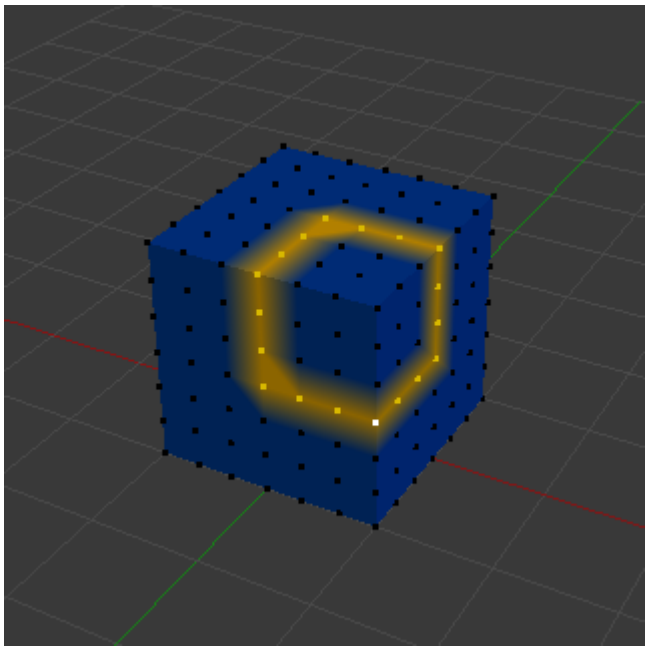
- `Alt-LMB` – Loop Select

Vertex Selection Masking

Reference

Mode::	Vertex and Weight Paint Modes
Header::	*Vertex Selection*
Shortcut::	2

In this mode you can select one or more vertices and then paint only on the selection. All unselected vertices are protected from unintentional changes.



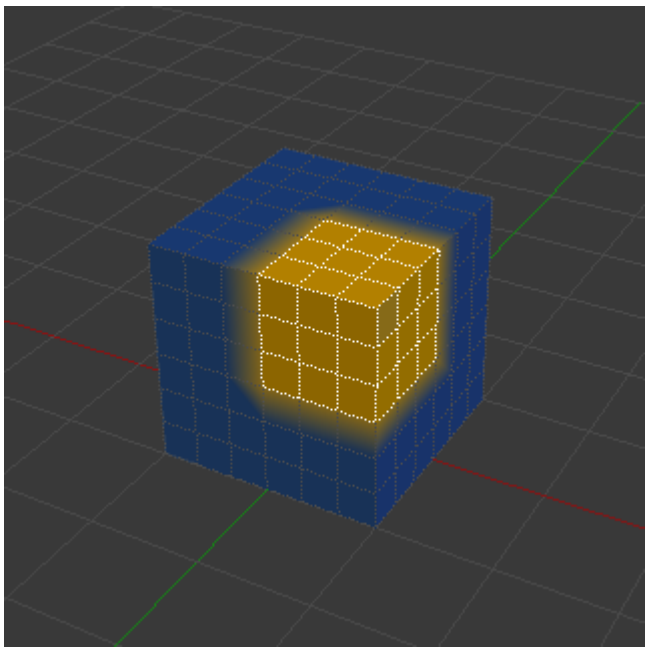
Vertex Selection masking.

Face Selection Masking

Reference

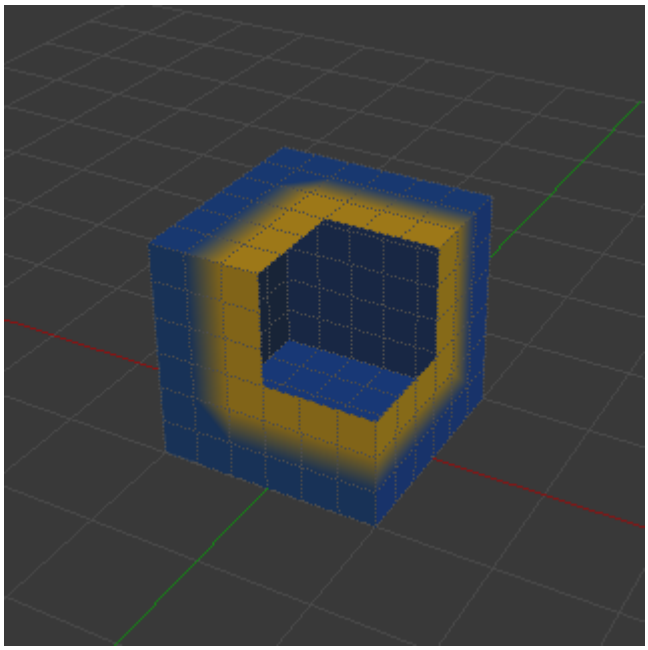
Mode::	Texture, Vertex, and Weight Paint Modes
Header::	*Paint Mask*
Shortcut::	1

The *Face Selection masking* allows you to select faces and limit the paint tool to those faces, very similar to Vertex selection masking.



Face Selection masking.

Hide/Unhide Faces



Hidden faces.

You also can hide selected faces as in Edit Mode with the keyboard Shortcut `H`, then paint on the remaining visible faces and finally unhide the hidden faces again by using `Alt-H`.

Hide/Unhide Vertices

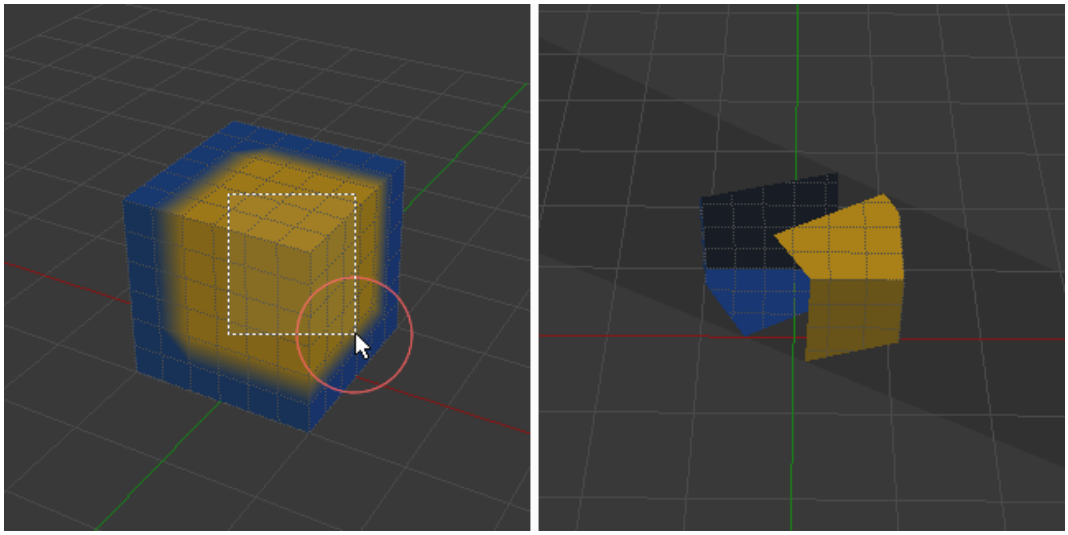
You cannot specifically hide only selected faces in vertex mask selection mode. However, the selection is converted when switching selection modes. So a common trick is to:

1. Switch to Face selection mask mode to have the selection converted to faces.
2. Refine your selection next or just hide the faces.
3. Switch back to Vertex Selection mask mode.

Hidig faces will make sure that vertices that belong to visible faces remain visible.

The Clipping Region

To constrain the paint area further you can use the *Clipping Region*. Press `Alt-B` and `LMB`-drag a rectangular area. The selected area will be "cut out" as the area of interest. The rest of the 3D Viewport gets hidden.



The Clipping Region is used to select interesting parts for local painting.

You make the entire mesh visible again by pressing `Alt-B` a second time.

All paint tools that use the view respect this clipping, including box select, and of course brush strokes.

There are two helpful reminders that a Clipping Region is used:

1. The clipping region is drawn as a grey box in the 3D Viewport
2. The Text Info overlay will state that the perspective is "Clipped"

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Linked* ▸ *Linked*
Shortcut::	`Ctrl-L`, `Shift-L`

Select geometry connected to already selected elements. This is often useful when a mesh has disconnected, overlapping parts, where isolating it any other way would be tedious. Pressing `Shift-L` will deselect linked any linked elements.

With `L` you can also select connected geometry directly under the cursor.

Navigation

There are different preferences for navigating the 3D Viewport in Blender. For painting and sculpting specific workflows it is recommended to use any of the following methods.

Center View to Mouse `Alt MMB`
: Center the View on the surface directly under the mouse position. This way the rotation point of the viewport can be manually changed to any point you wish to orbit around.

Center on Last Stroke `NumpadPeriod`
: Center the View on the average position of the last stroke.

Various preferences can also make navigation more convenient. These can be found in the “Navigation” tab of the preferences.

Orbit Method = Trackball
: Tumble the view based on the mouse position in your 3D Viewport while rotating. This makes it very easy to tilt the viewport freely, instead of having the Z axis of the viewport locked.

Zoom to Mouse Position
: Use the mouse position to zoom towards and rotate around the surface that is pointed at. This can be an alternative to the repeated manual use of the Center View to Mouse operator.

  The disadvantage is that this navigation preference can lead to accidental navigation around backfacing geometry or very distant geometry.

Sculpting

- Introduction
  - General
  - The Brush
  - Gesture Tools
  - Visibility, Masking & Face Sets
  - Filters
  - Transforming
  - Painting
  - Working with Multiple Objects
  - Adaptive Resolution
  - Cloth Sculpting
- Brushes
  - Brushes
  - Draw
  - Draw Sharp
  - Clay
  - Clay Strips
  - Clay Thumb
  - Layer
  - Inflate
  - Blob
  - Crease
  - Smooth
  - Flatten
  - Fill
  - Scrape
  - Scrape Multiplane
  - Pinch
  - Grab
  - Elastic Deform
  - Snake Hook
  - Thumb
  - Pose

    - Nudge
    - Rotate
    - Relax Slide
    - Boundary
    - Cloth
    - Density
    - Mask
    - Draw Face Sets
    - Erase Multires Displacement
    - Smear Multires Displacement
    - Paint
    - Smear
- Toolbar
    - Sculpting Tools
    - Gesture Tools
    - Filter Tools
    - Single Click Tools
    - General Tools
- Tools
    - Brush
    - Mask Gesture Tools
    - Hide Gesture Tools
    - Face Set Gesture Tools
    - Trim Gesture Tools
    - Line Project
    - Mesh Filter
    - Cloth Filter
    - Color Filter
    - Edit Face Set
    - Mask by Color
    - Transforms
- Tool Settings
    - Brushes
    - Brush Settings
    - Dyntopo
    - Remesh
    - Symmetry

    - Options
- Controls
    - Auto-Masking
- Editing
    - Sculpt
    - Mask
    - Face Sets
    - Expand

Introduction

- General
- The Brush
  - Common Brushes
- Gesture Tools
  - Box Gestures
  - Lasso Gestures
  - Line Gestures
  - Polyline Gestures
- Visibility, Masking & Face Sets
  - Visibility Control
  - Masks
  - Face Sets
  - Auto-Masking
  - Display Settings
- Filters
- Transforming
- Painting
- Working with Multiple Objects
- Adaptive Resolution
  - Voxel Remesher
  - Dyntopo
  - Multiresolution
- Cloth Sculpting

General

Sculpt Mode is similar to *Edit Mode* in that it is used to alter the shape of a model, but Sculpt Mode uses a very different workflow: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is primarily changed using brushes.



Sculpting Mode Example.

Sculpt Mode is accessed from the mode menu of the 3D Viewport header or with the pie menu via `Ctrl-Tab`. Once inside Sculpt Mode, the Toolbar and Tool Settings of the 3D Viewport will change to Sculpt Mode specific panels. The cursor will change to a circle, to indicate the size of the brush.

Warning

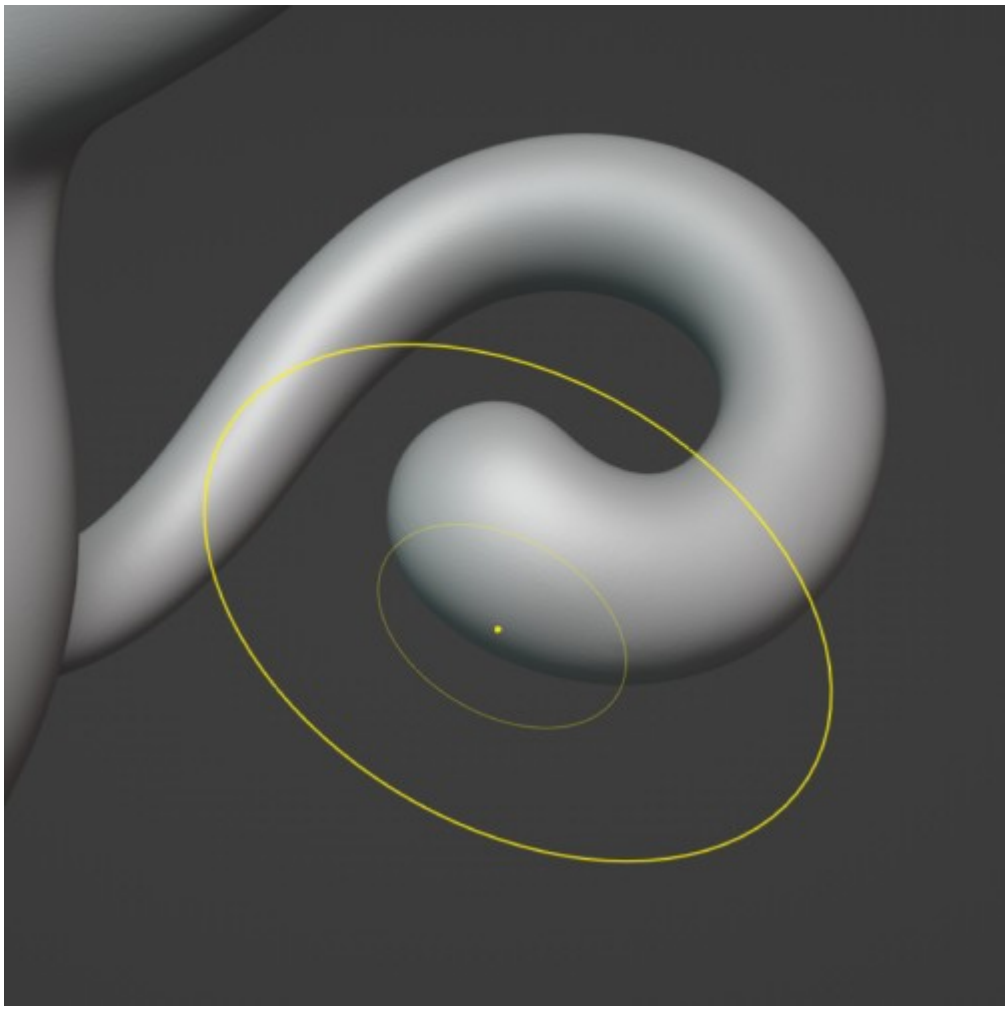
To have predictable brush behavior, make sure to apply the scale of your mesh.

The following pages will briefly explain the fundamental features and concepts of *Sculpt Mode*, including various links to other pages for more details.

The Brush

Sculpt Mode is very recognizable by the behavior and visualization of the brush. All the usual brush controls still apply, yet the brush for sculpting is displayed in 3D. This means that the brush will follow the curvature of the surface by orienting the radius to match the topology Normal.

The inner ring of the brush cursor is used to visualize the strength of the brush.



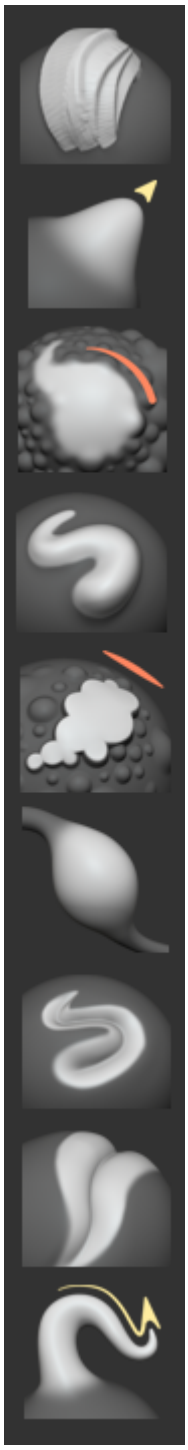
Note

How closely the cursor follows the curvature of the mesh can be changed in the Brush Settings with “Normal Radius”. This can make hard surface sculpting easier, for example with the Scrape brush.

The brush is also used for other tools in the toolbar to better display how that tool works. For example, the Box Trim and Lasso Trim tools are able to use the current brush radius for how deep geometry is trimmed or added.

Common Brushes

There are many brushes to choose from but these are the most common brushes to be used during sculpting. More information on sculpting brushes in the Toolbar.



[Clay Strips]

Block out broad shapes and build up volumes before refining them further.

[Grab]

Move geometry across the screen for general shaping.

Smooth
Smooth and shrink surfaces to remove noise or flatten shapes.

Draw
Generic adding and subtracting on surfaces. This brush is often customized with different stroke methods and textures for various effects.

Scrape
Scrape and fill surfaces either for hard surface sculpting or more aggressive smoothing.

Inflate
Inflate or shrink volumes or surfaces. Especially useful for controlling the thickness of cylindrical shapes.

Draw Sharp
Same as *Draw* but with a much sharper falloff. Useful for adding creases, cracks and other sharp edges.

Crease
A mix of the *Draw* and *Pinch* brushes. Useful for creating detailed creases or sharpening existing creases for additional polish.

Snake Hook.
Similar to *Grab* but this brush will dynamically let go and pick up geometry during the stroke. The dragged geometry is also following the angle of the stroke, making it very useful for pulling geometry out. Ideally used together with Dyntopo.

Gesture Tools

Separate from brushes and filters, Sculpt mode also has a set of tools that perform actions to a drawn selection area. These tools are similar to the selection tools (e.g. box selection and lasso selection in other areas of Blender).

These tools do not provide a selection of elements that are then modified, they directly modify the underlying mesh.

Box Gestures

Dragging creates a rectangular area defined by where `LMB` was pressed and where `LMB` is released.

Controls

Move `Spacebar`
: Hold to reposition the selection area.

Lasso Gestures

Dragging creates a freeform area that follows the cursor defined by where `LMB` was pressed and where `LMB` is released.

Controls

Move `Spacebar`
: Hold to reposition the selection area.

Tool Settings

Stabilize Stroke

Helps to reduce jitter of the strokes while drawing by delaying and correcting the location of points.

Radius
: Minimum distance from the last point before the stroke continues.

Factor
: A smooth factor, where higher values result in smoother strokes but the drawing sensation feels as if you were pulling the stroke.

Line Gestures

Dragging creates a line. The resulting action acts upon everything on the highlighed side of the line. The area acted upon is extended in both directions of the viewport.

Controls

Flip `F`
: Toggles the side of the line that the tool affects.

Snap `Ctrl`
: Hold to constrain the rotation of the line to user-specified intervals. Defaults to 5 degree increments, customizable via the Snapping menu indicated by the magnet icon in the header.

Move `Spacebar`
: Hold to reposition the line.

Tool Settings

Limit to Segment
: The affected area will not extend the length of the drawn line. This helps defining a smaller area instead of extending the line infinitely long.

Polyline Gestures

Clicking places a point in the viewport. Each time `LMB` is pressed, a new point of the polygon is created. Pressing `LMB` on the starting point, pressing `LMB` twice, or pressing `Return` closes the selection area.

Controls

Move `Spacebar`
: Hold to reposition the selection area.

Visibility, Masking & Face Sets

Visibility Control

Parts of the mesh can be hidden in Sculpt Mode. Because hidden faces cannot be sculpted, hiding makes it easier to isolate what you want to work on. Hiding geometry also improves the viewport performance.

Hiding is shared between all modes, except Object Mode (i.e. hiding/showing of faces in one mode will hide the same faces in other modes too).

Unlike Selection Masking in other painting modes, Sculpt Mode primarily uses Masks and Face Sets to easily control the mesh visibility and which faces can currently be edited. The exception is the Clipping Region, which can be used in any mode.

The most common shortcuts are `H` to hide the face set under the cursor and `Shift-H` to isolate the face set under the cursor (or show everything).

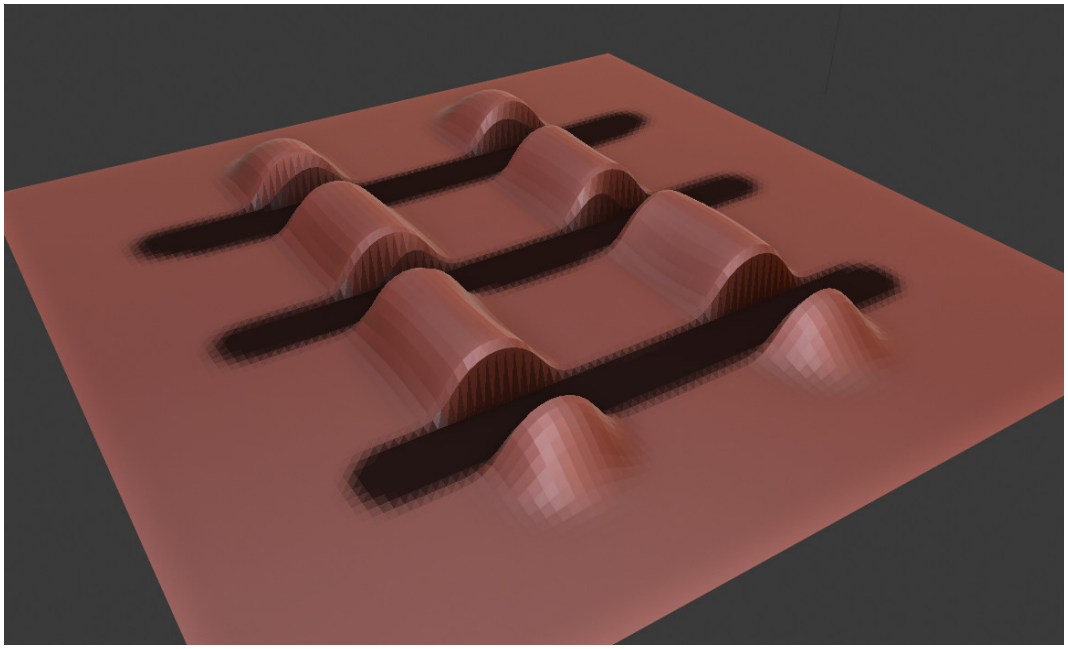
Inverting the visibility and showing all is also available in the `Alt-W` pie menu.

Modifying visibility can also be done via the Hide Gesture Tools.

See also

More information for controlling the visibility at Show & Hide.

Masks



A mask is used to control which vertices of the mesh are influenced by sculpting and painting. The mask can for example be created/edited via the Mask Gesture Tools and Mask by Color tool.

Internally, masks are stored using the `sculpt_mask` Attributes.

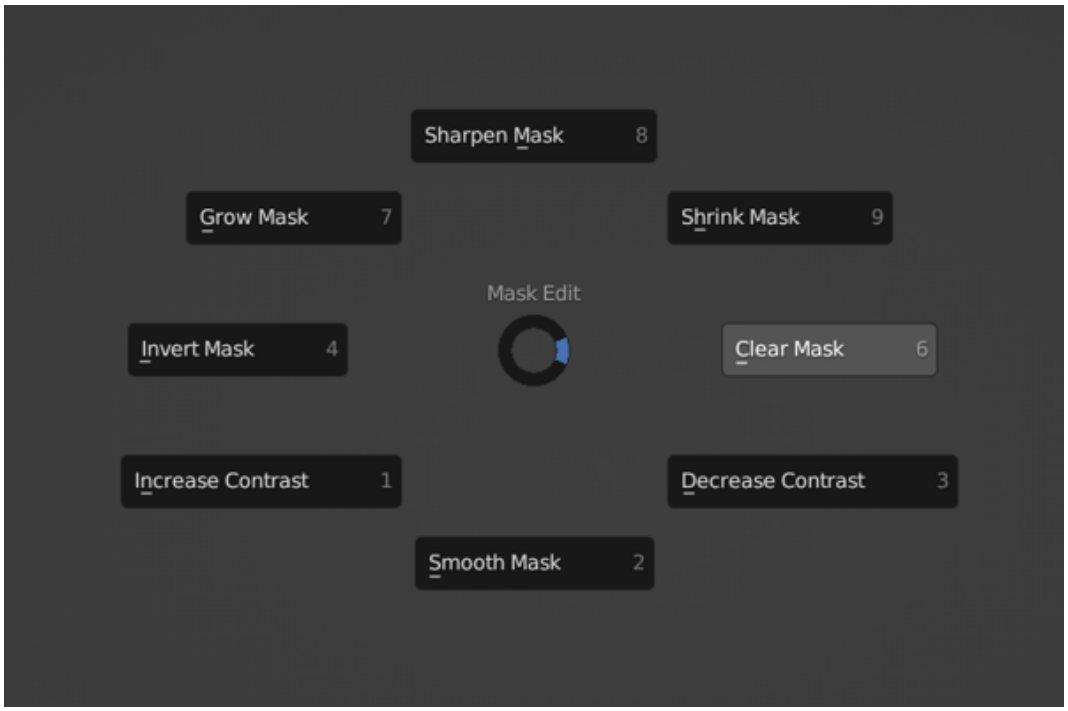
Clear & Invert

Creating masks follows a slightly different mental model than selecting in other modes. For example `Shift-LMB` is used for smoothing instead of adding to a mask.

Masking is also conceptually inverted to selection (i.e. You **cannot** edit masked vertices. But you **can** edit selected vertices).

Instead a mask is typically always added to the current mask with `LMB` and subtracted with `Ctrl-LMB`. So if you wish to edit the masked surfaces, you'll need to use the Invert operator, In the case of masking everything that is visible, the best workflow is to first Clear and then Invert the mask.

Both these operators can be quickly accessed in the `A` pie menu.



See also

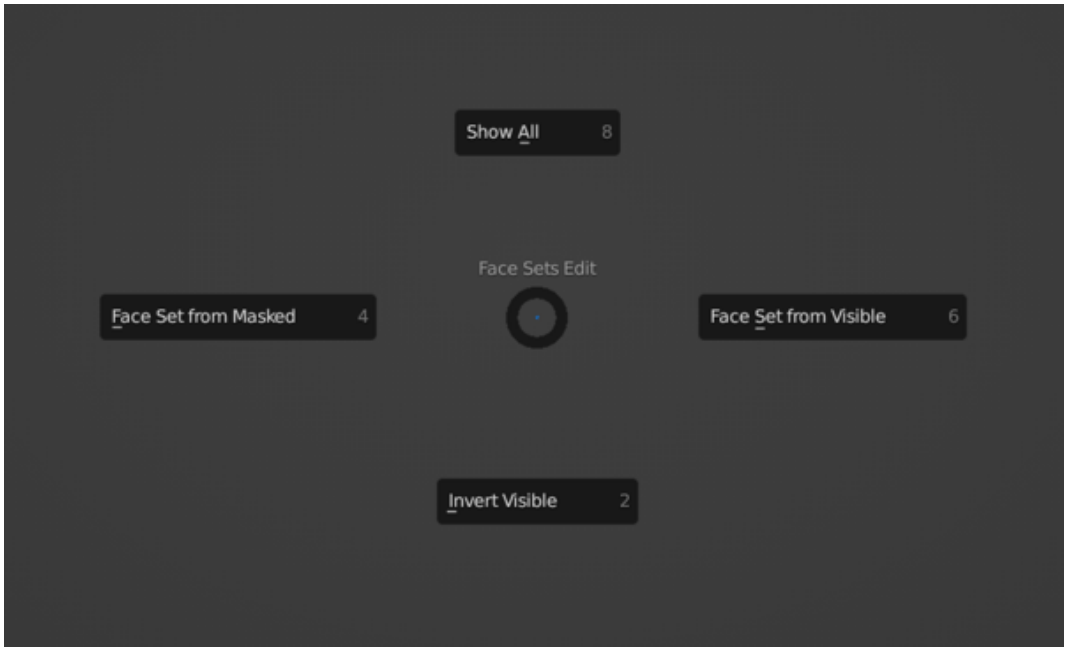
More information about editing and using masks at the Mask Menu

Face Sets



Face sets are used to group your mesh into differently colored faces, which can then be quickly hidden or shown like mentioned above. They can also be used for fast mask creation via the Mask Expand. Face Set Expand is also useful for creating, editing and joining face sets.

More options can be found in the `Alt-W` pie menu.



Otherwise Face Sets can be created/edited with the Draw Face Sets brush, Mask Gesture Tools. They can also be edited with the Edit Face Set tool.

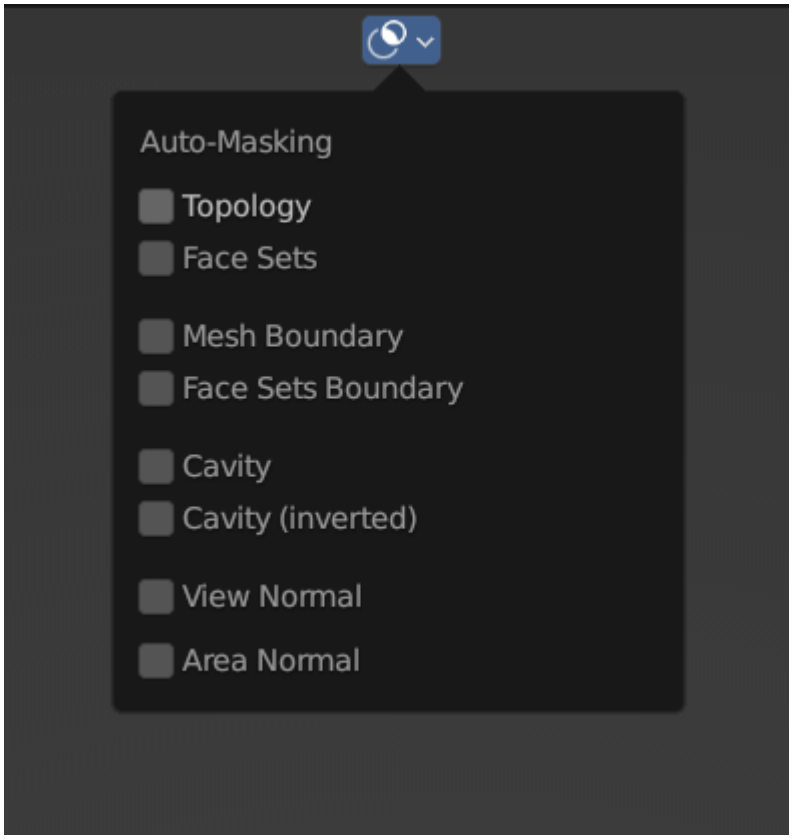
See also

More information about editing and using face sets at the Face Sets Menu

Internally, face sets are stored using the `sculpt_face_set` Attributes.

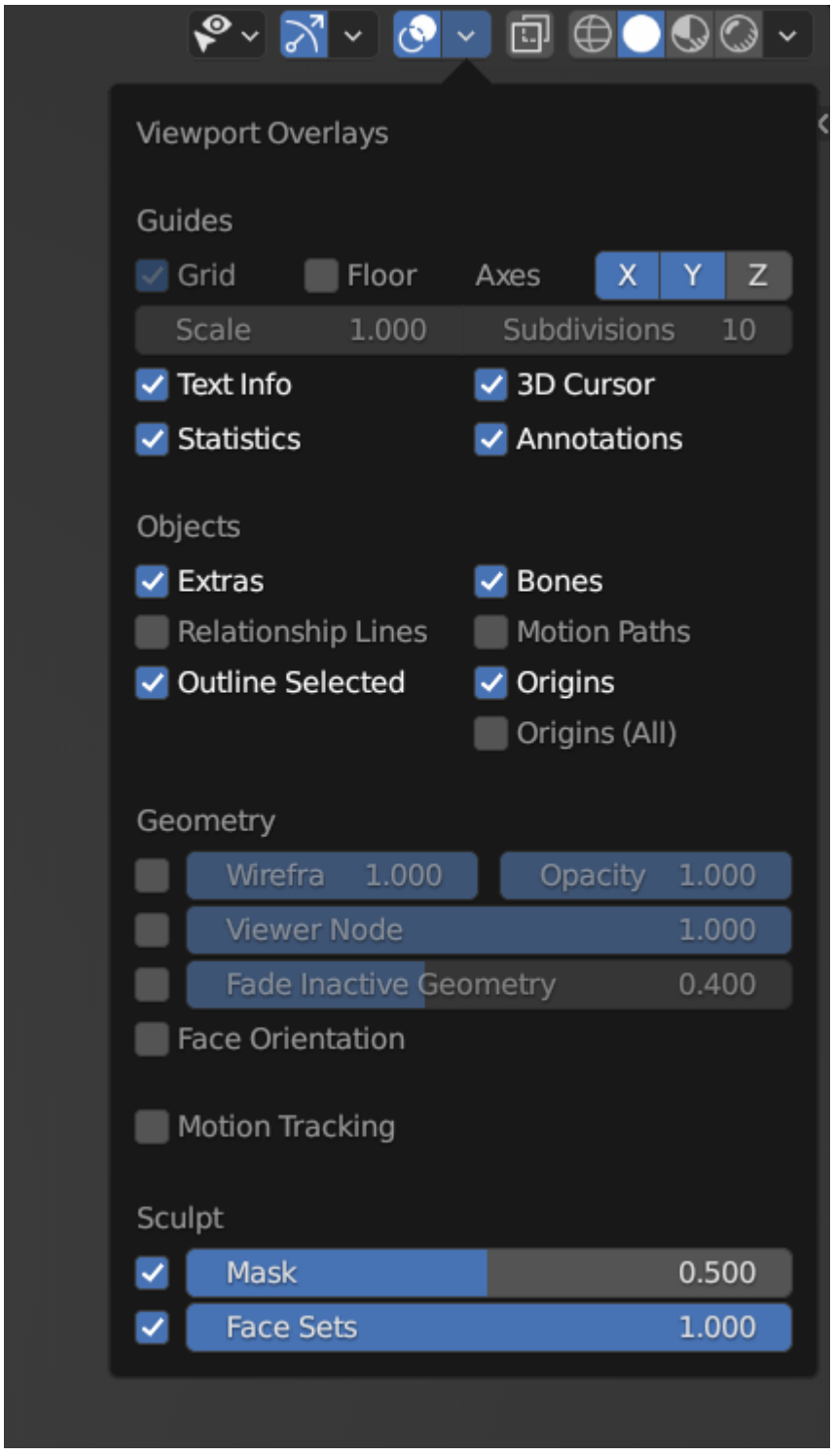
Auto-Masking

Auto-Masking is also a fast way of only editing specific geometry without having to manually create a new mask or hide geometry. This feature is especially useful in combination with face sets.



Display Settings

The mask and face sets display can be toggled and adjusted in the Display Settings.

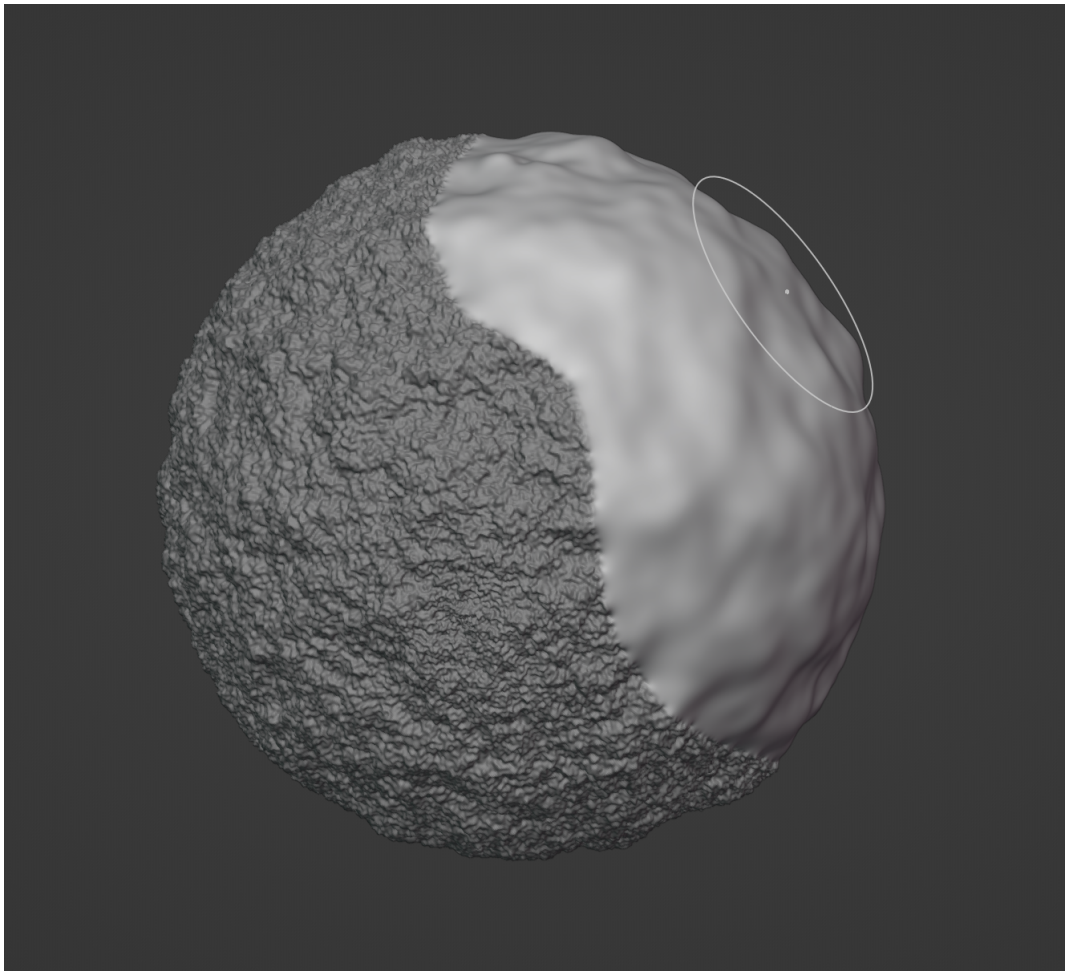

Viewport Overlays
Guides
Grid
Floor
Axes
X
Y
Z
Scale 1.000
Subdivisions 10
Text Info
3D Cursor
Statistics
Annotations
Objects
Extras
Bones
Relationship Lines
Motion Paths
Outline Selected
Origins
Origins (All)
Geometry
Wirefra 1.000
Opacity 1.000
Viewer Node 1.000
Fade Inactive Geometry 0.400
Face Orientation
Motion Tracking
Sculpt
Mask 0.500
Face Sets 1.000

Note

When Xray shading is enabled, masks and face sets will not be displayed.

Filters

Filters are tools which provide an alternative way of sculpting, because they do not rely on a brush radius. Instead they will affect any vertices that are visible and not masked.



The strength is controlled by click & dragging from left to right. The position of the cursor can be used to only affect specific areas, if auto-masking is used.

Many of the same brush types are also available as a filter type. This way much of the mesh can simultaneously be smoothed, colored or have some cloth simulation applied.

Tip

A common example for using the Mesh Filter is to smooth everything after increasing the resolution with the Voxel Remesher or Dyntopo.

See also

More information at Mesh Filter, Cloth Filter, Color Filter and Mask Filters.

Transforming

Transform tools to move, rotate and scale are also available in Sculpt Mode, but with an important difference to other modes. Sculpt Mode uses its own pivot point, which can be manually positioned `Shift-RMB` or automatically positioned with Mask Expand. This ensures that the pivot point can be more freely placed and always moves with the transformed geometry.

Optionally instead of keeping the transform tools active, you can enable the viewport gizmos to have access to the gizmo at all times.

Note

The gizmo can in some cases block areas from being sculpted on. In that case move the pivot point somewhere else to be able to click on the desired surface.

Apart from the transform tools there are also special brushes to move, rotate and scale the topology like Pose, Boundary and Elastic Deform.

Painting

Sculpt Mode also allows painting your geometry via Color Attributes such as Vertex Colors. This ensures that the most common actions related to the sculpting workflow are contained in the same mode, to avoid unnecessary mode switching.



Other sculpt mode features such as face sets, masking and filters can also be used with painting tools.

The painting functionality in Sculpt Mode is limited to a Paint and Smear brush, as well as a Color Filter and Mask by Color tool.

Just like any other brush, `Shift` can be used to smooth. In the case of painting brushes it will blur the colors within the brush radius instead.

Note

Once any painting tool is executed, the viewport color shading is switched to “Attribute”. This ensures that color attributes are shown on all objects once painting is needed.

Working with Multiple Objects

Unlike Edit Mode, there is no multi-object editing supported for Sculpt Mode. Since sculpting often involves editing many separate objects, it is recommended to use the shortcut `Alt Q` while pointing at other objects, for Switching Objects quickly.



The advantage of using multiple objects is that each can have its own origin and modifiers. Splitting the geometry among multiple objects can also improve the sculpt mode performance. Alternatively objects can also be joined so there is no need to switch objects.



In the case that Face Sets were already used, joining objects or creating new geometry in Edit Mode will automatically assign new Face Sets. This makes it immediately possible to target each new geometry, for example via auto-masking. If no Face Sets are created, use the Initialize Face Sets operator to create them.

Face Sets and Masked geometry can also be extracted via Expand Mask or sliced into a new object via Mask Slice.

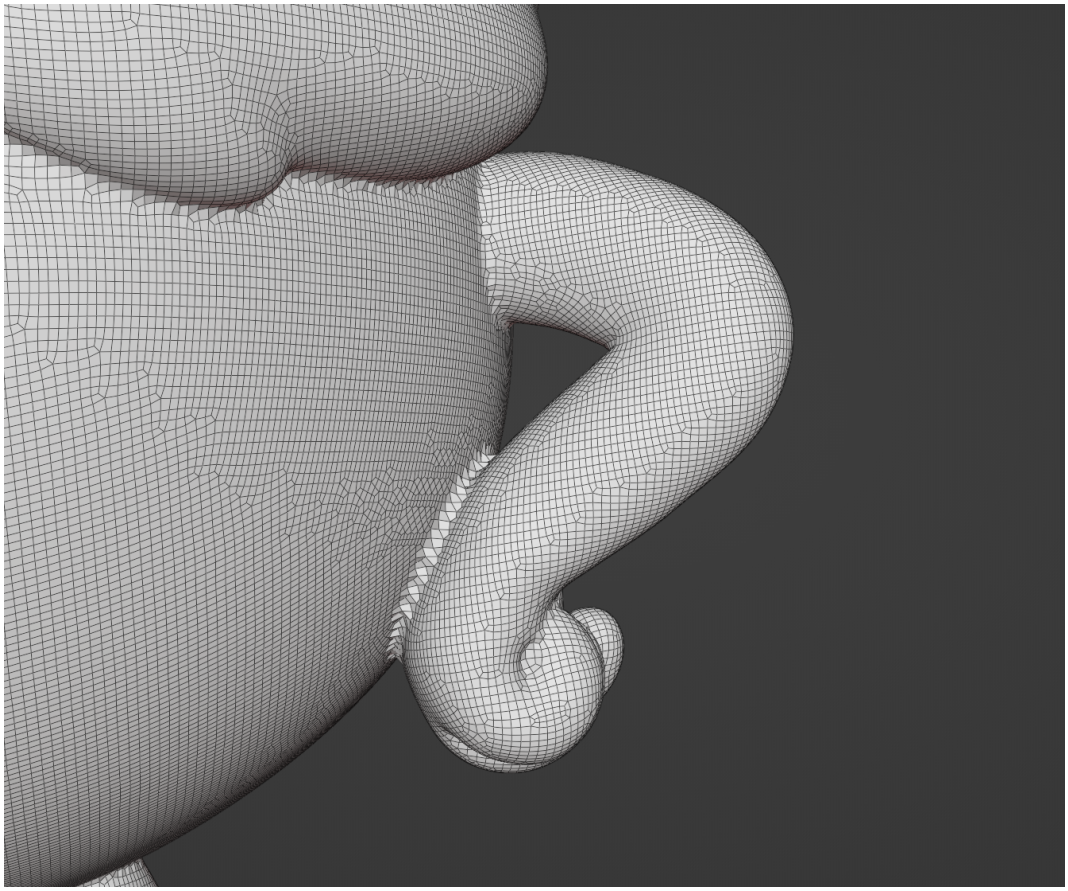
Adaptive Resolution

In order for sculpting to give accurate and predictable results, Blender needs enough geometry. Instead of starting out with a highly subdivided mesh, add geometry dynamically by using either of the following adaptive sculpting methods.

Voxel Remesher

“Voxel remeshing” rebuilds the geometry with a perfectly even distributed topology. Depending on the set voxel size, this will lead to a lower or higher resolution.

This technique is especially useful to block out the initial shape of an object. It also has the advantage of removing any overlapping geometry and creating a manifold volume as a result.



Any currently used mask, face sets and color attributes will be re-projected on the remeshed result. Reaching high vertex counts should still be achievable with this technique, depending on the used hardware.

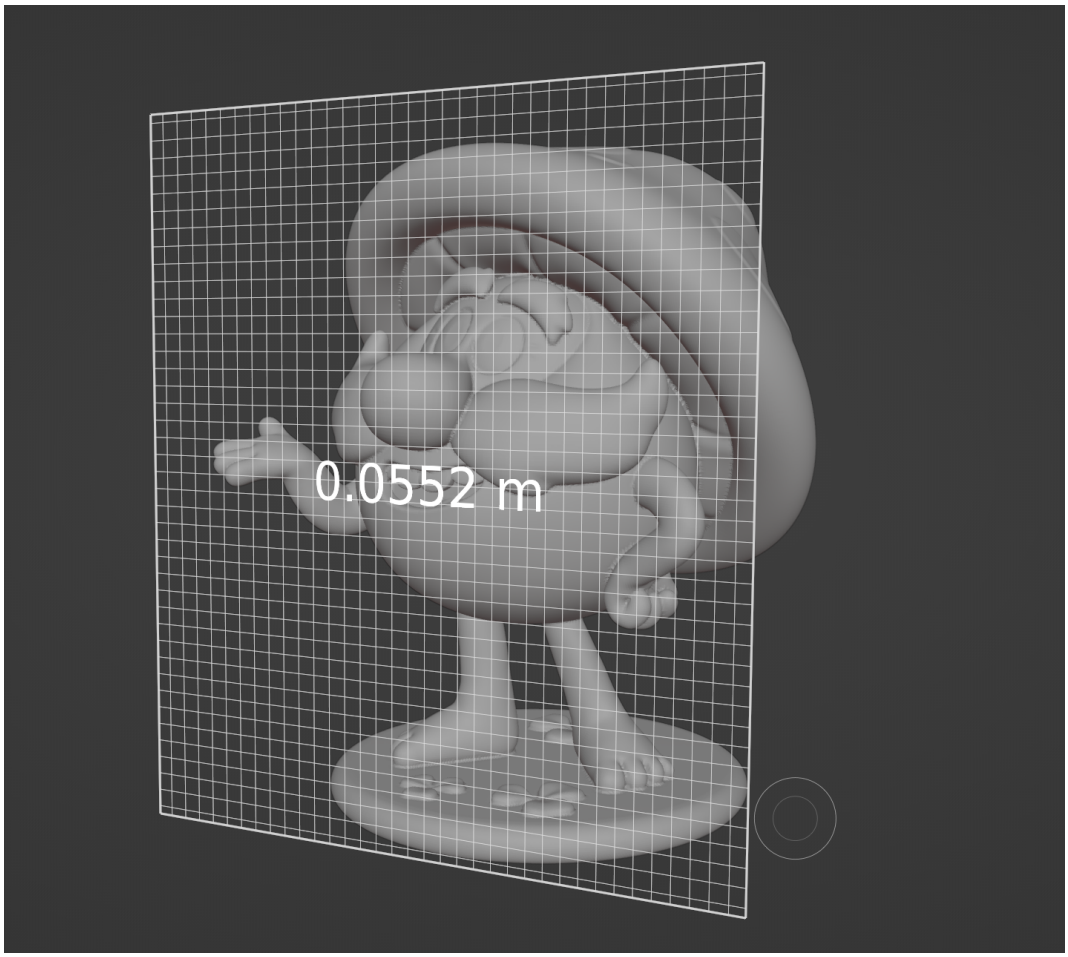
Note

This technique will not work on objects that do not have an enclosed volume. Make sure to fill any holes in the mesh before remeshing. Or avoid any holes in the mesh/volume that are larger than the defined voxel size.

Tip

If in doubt, you can fill all holes in edit mode or by using the Mask Slice and Fill Holes operation to fill all holes in the mesh. If nothing is masked, it only fills any holes.

To more easily access this feature, use the shortcuts `R` to define the resolution, and `Ctrl-R` to execute the remeshing.

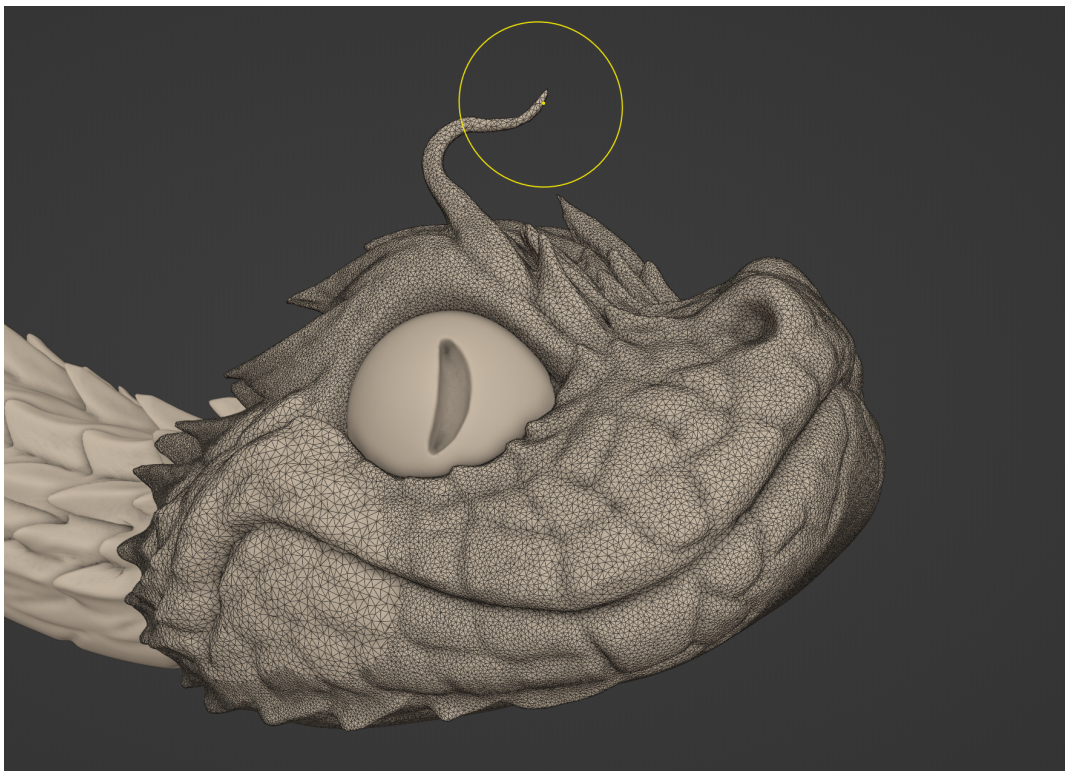


See also

More information at Remesh.

Dyntopo

Dynamic topology (aka Dyntopo) is a dynamic tessellation sculpting method that automatically adds and removes topology under the brush.



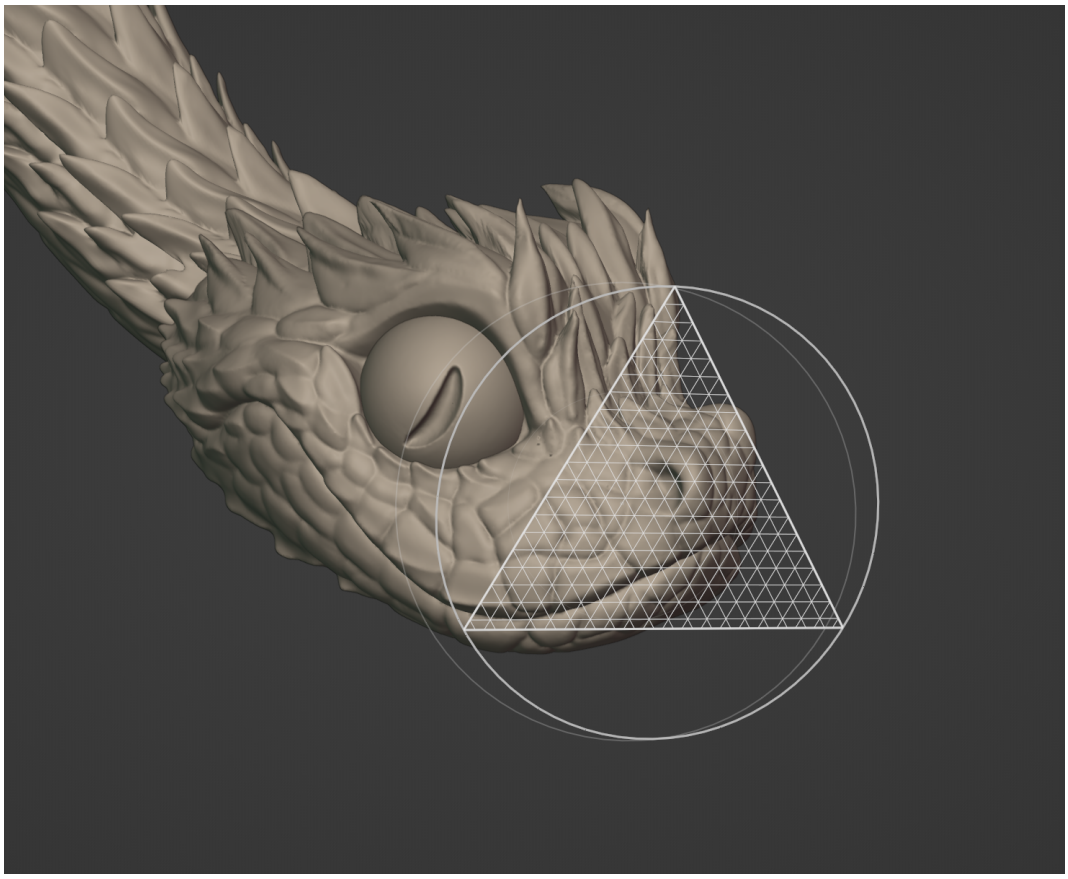
Unlike the Voxel Remesher, this makes it possible to sculpt complex shapes without thinking about the resolution or topology. It also allows to define a different resolution wherever necessary. Much more complex base mesh sculpting is especially useful with this technique.

The disadvantages of this technique are a slower performance and limited support for some sculpt mode features. Custom attributes like Color Attributes, UV Maps and Face Sets are also lost or corrupted when using Dyntopo.

This feature shares the same shortcuts with voxel remeshing when enabled. Use `R` to define the resolution and `Ctrl-R` to flood fill the resolution (if Constant Detail is used).

Note

Because Dyntopo and the Voxel Remesher are mutually exclusive and cannot be used at the same time, both use the same shortcut to define the remeshing resolution.



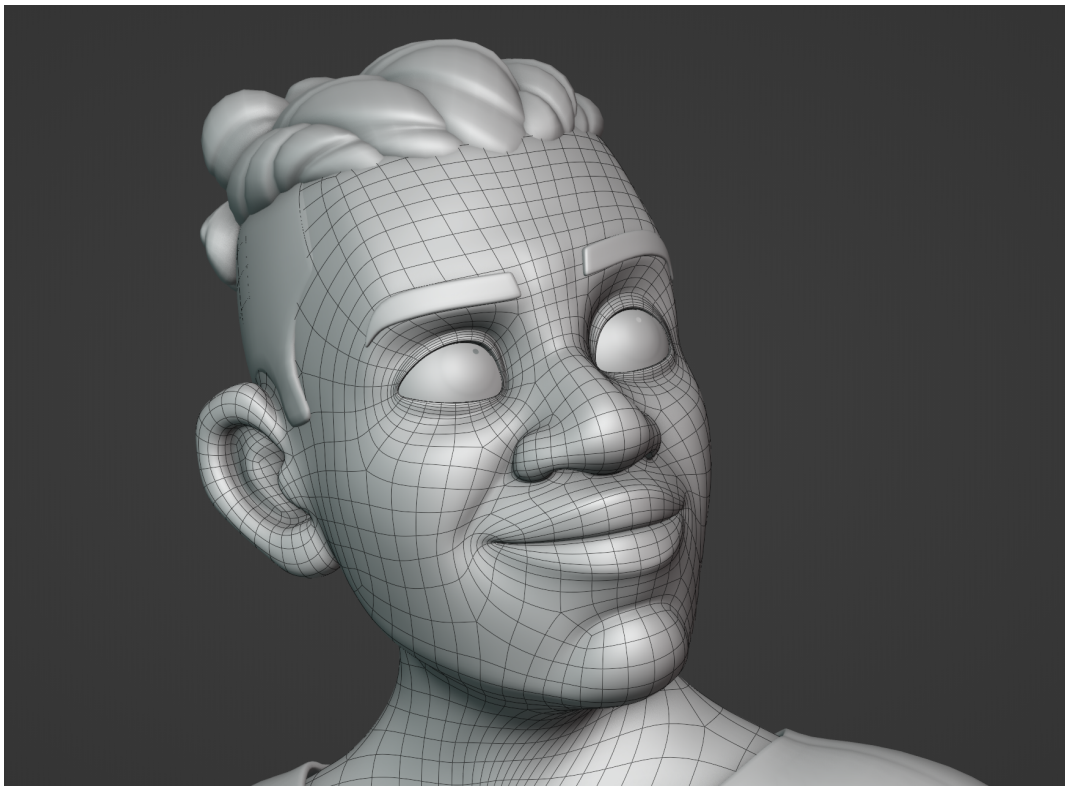
Brushes like Density, Snake Hook and Clay Strips work especially well with this feature.

See also

More information at Dyntopo.

Multiresolution

The Multiresolution Modifier can be used for subdivision based sculpting. This means the object will be subdivided, similar to the Subdivision Surface Modifier, only that the subdivisions can be freely sculpted for very high resolution detailing.



Note

For this technique it is highly recommended to use on a clean topology base mesh. This means the base mesh should be only made of quads and avoid non-manifold faces, as well as poles with two connected edges. More information at Quad Remeshing for an automatic retopology method.

This technique has the advantage of sculpting with multiple resolutions, meaning you have the ability to sculpt on any level of subdivision. This allows to add a much higher resolution of details for rendering and sculpting, while displaying lower resolutions for better viewport

performance. It also allows sculpting on lower resolutions any time for broader changes.

As an example, you can sculpt general proportions in subdivision level 1, add high resolution details in level 4 and switch back to subdivision 1 to correct the shape further.

The disadvantages are that you may end up with some mesh distortions because the topology is not dynamic like voxel remeshing and dyntopo. The topology should also not be changed once already subdivided, since any edits to the base mesh will result in corrupted subdivision details.

Tip

Pay attention to the topology that you sculpt and how much it gets stretched. If more resolution is needed you can always subdivide another time, but there will be worse performance and slower level switching once more than 5 subdivisions are used. Alternatively use the Slide Relax brush to slide topology to where it is needed.

Additional brushes like the Multires Eraser and Multires Smear are recommended for adjustments.

Here are general shortcuts to use the feature.

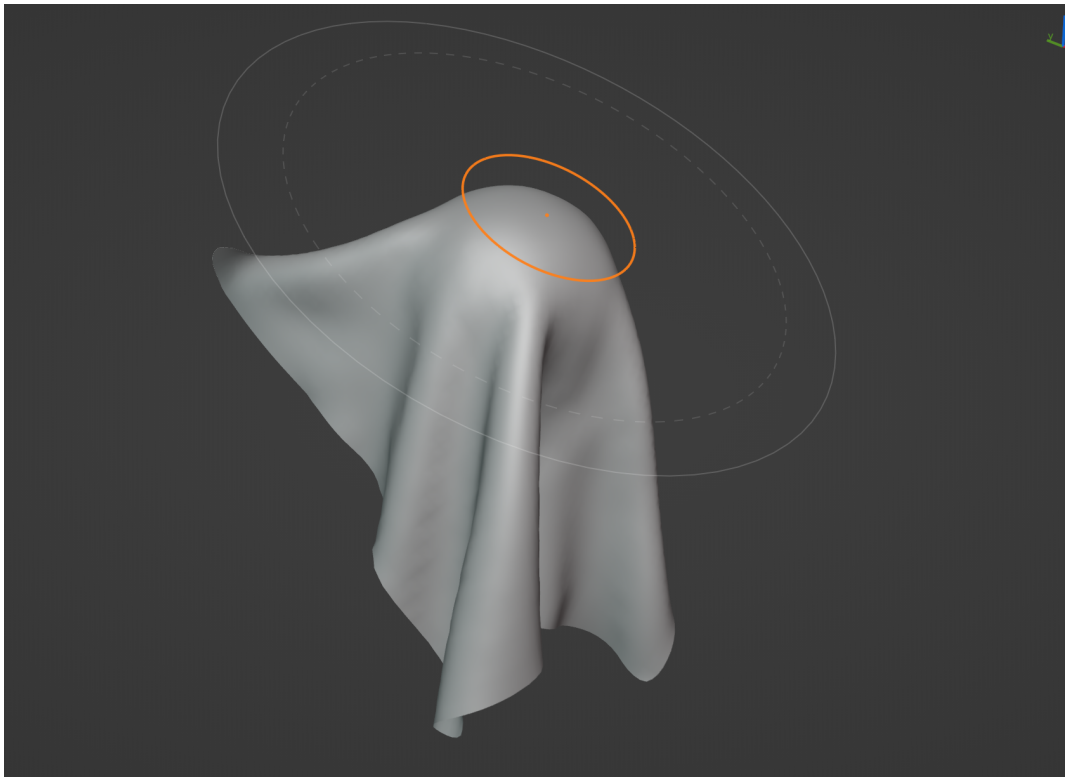
- Step up one multires level `Alt-2`
- Step down one multires level `Alt-1`
- Set multires level / Create multires modifier `Ctrl-0` to `Ctrl-5`

See also

More information at Multiresolution Modifier.

Cloth Sculpting

Instead of sculpting cloth manually or creating complex physics simulation setups, there are various tools directly in sculpt mode that offer a simplified Cloth Physics Simulation.



This has various advantages but is especially useful for base mesh creation and larger clothing folds and draping. Detailing is possible, but the slower performance on high resolution meshes and simplified cloth physics might not lead to desirable results.

The resolution of the topology is mainly responsible for the size of the folds and detail level of the simulation. So an optimal and evenly distributed topology is important.

Many sculpting features are supported, so for example Masked vertices are pinned in the simulation. Another example is with auto-masked face set

boundaries. The sculpt mode gravity factor is also applied on the cloth physics.

The main brushes and tools for this feature are the Cloth Brush and Cloth Filter, but other transform brushes like Pose and Boundary also support cloth sculpting in the brush settings.

A demo file for trying out the various brushes and tools is available here.

Brushes

Brushes for *Sculpt Mode* bundled in the *Essentials* library.

- Brushes
- Draw
- Draw Sharp
- Clay
- Clay Strips
- Clay Thumb
- Layer
- Inflate
- Blob
- Crease
- Smooth
- Flatten
- Fill
- Scrape
- Scrape Multiplane
- Pinch
- Grab
- Elastic Deform
- Snake Hook
- Thumb
- Pose
- Nudge
- Rotate
- Relax Slide
- Boundary
- Cloth
- Density
- Mask
- Draw Face Sets
- Erase Multires Displacement
- Smear Multires Displacement

- Paint
- Smear

Brushes

This is a list of all provided ‘Essentials’ brush assets that come with Blender. These are based on various Brush Types which are mentioned for each brush..

Add/Subtract Brushes

These brushes generally push vertices outwards and inwards and are the most customizable to achieve a wide variety of effects. They typically don’y use a color in their thumbnail.



Draw

> Brush Type: Draw
> Shortcut: `V`
>
> The standard brush for pushing vertices inwards and outwards from the surface direction.

Draw Sharp

> Brush Type: Draw Sharp
> Shortcut: `Shift V`
>
> Same as *Draw* but with a much sharper Falloff. Useful for creating creases and sharp angles.

Clay

> Brush Type: Clay

Similar to the *Draw* brush but with a flattening effect and subtle smoothing. Useful for polishing and building volumes.

Clay Strips

Brush Type: Clay Strips
Shortcut: `C`

The same as the *Clay* brush, but more aggressive with a square falloff. A common standard for building rough volumes.

Clay Thumb

Brush Type: Clay Thumb

The same as the *Clay* brush, but specifically for emulating the effect of running your thumb over surfaces. Pushes geometry in and sideways.

Layer

Brush Type: Layer

Draw with a fixed height. Useful for adding flat layers to a surface.

Inflate/Deflate

Brush Type: Inflate
Shortcut: `I`

Moves the mesh in multiple direction. Useful for inflating or shrinking surfaces and volumes.

Blob

Brush Type: Blob

Magnifies the mesh as you draw. Useful for an additional inflation effect on the stroke.

Crease Polish

Brush Type: Crease
Shortcut: `Shift C`

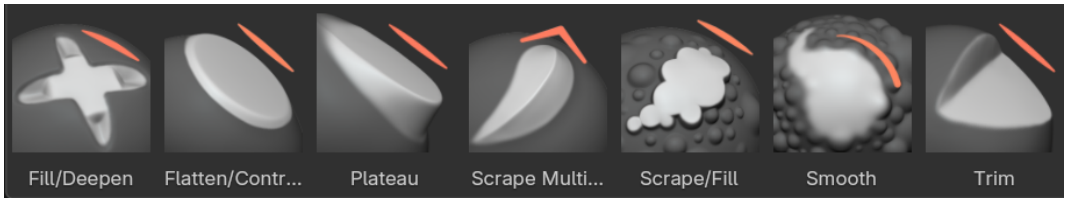
A Draw brush with a pinching effect. Useful for polishing existing creases or carefully creating new ones.

Crease Sharp

Brush Type: Crease

Much sharper and stronger Crease brush. Great for creating thin and deep pinches.

Contrast Brushes



Recognizable by their red thumbnail and cursor. These brushes generally flatten or heighten the contrast of the surface.

Smooth

Brush Type: Smooth
Shortcut: `S`

Smooths out irregularities in the surface and shrinks volumes by averaging the vertices positions. An essential brush that is frequently used.

Flatten/Contrast

Brush Type: Flatten

Pushes vertices to an average height to create a flat surfaces. Alternatively pushes them away from the center for more contrast.

Plateau

Brush Type: Flatten

Similar to Flatten but with a locked orientation and depth to create a consistently flat surface.

Fill/Deepen

Brush Type: Fill

Pushes surfaces upwards towards a flat plane. Useful for filling in holes and crevices. Alternatively deepens existing holes when holding ‘Ctrl’.

Scrape/Fill

Brush Type: Scrape
Shortcut: `Shift T`

Pushes surfaces inwards. Alternatively fills surfaces while holding ‘Ctrl’. This is the most common brush for flattening meshes.

Trim

Brush Type: Scrape

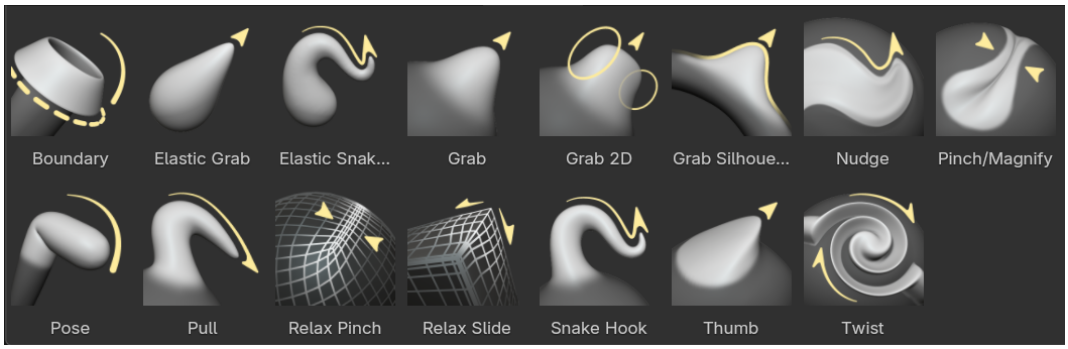
Pushes surfaces inwards toward a locked direction. The depth can be defined by going deeper towards surfaces along the stroke.

Scrape Multiplane

Brush Type: Scrape Multiplane

Scrapes the mesh with two angled planes at the same time, producing a sharp edge between them.

Transform Brushes



Recognizable by their yellow icon and cursor. These brushes generally move, pinch and magnify the mesh.

Pinch/Magnify

> Brush Type: Pinch
> Shortcut: `P`
>
> Pulls vertices towards the center of the brush. Useful for polishing angles and creases. Alternatively pushes them away from the center.

Grab

> Brush Type: Grab
> Shortcut: `G`
>
> Moves vertices along with the mouse. An essential brush for building shapes and adjusting proportions.

Grab 2D

> Brush Type: Grab
>
> Similar to Grab but with an infinitely projected falloff. Useful for grabbing broader shapes and giving a similar feel to using Liquify tools in image painting applications.

Grab Silhouette

> Brush Type: Grab

Similar to Grab but only affects vertices with the normal facing sideways away from the view. Very useful for adjusting outer silhouettes of thin objects.

Elastic Grab

Brush Type: Elastic Deform

Used to simulate realistic deformations from grabbing of Elastic objects.

Elastic Snake Hook

Brush Type: Snake Hook

Similar to Elastic Grab but rotates affected geometry based on the stroke direction.

Snake Hook

Brush Type: Snake Hook
Shortcut: K

Pulls vertices along with the stroke to create long, snake-like forms. Geometry is rotated and magnified to allow continuous pulling. Much more useful while having Dyntopo enabled.

Pull

Brush Type: Snake Hook

Iteratively picks up and lets go of geometry like the Snake Hook, but much softer. Useful for subtle small scale deforming over longer strokes.

Thumb

Brush Type: Thumb

Same as *Grab* but moves vertices along the surface direction. Useful for preserving specific surfaces.

Pose

Brush Type: Pose

Simulating an armature-like deformations. Useful for quick posing and transformations.

Nudge

Brush Type: Nudge

Similar as *Thumb* but dynamically picks up vertices like the *Snake Hook*. Useful for nudging something along the mesh surface.

Twist

Brush Type: Rotate

Rotates vertices within the brush in the direction mouse.

Relax Slide

Brush Type: Relax Slide

Slides the topology of the mesh in the direction of the stroke while preserving the geometrical shape of the mesh. Alternatively smoothes the mesh on 'Shift'. Also useful for redistributing topology where it is needed.

Relax Pinch

Brush Type: Relax Slide

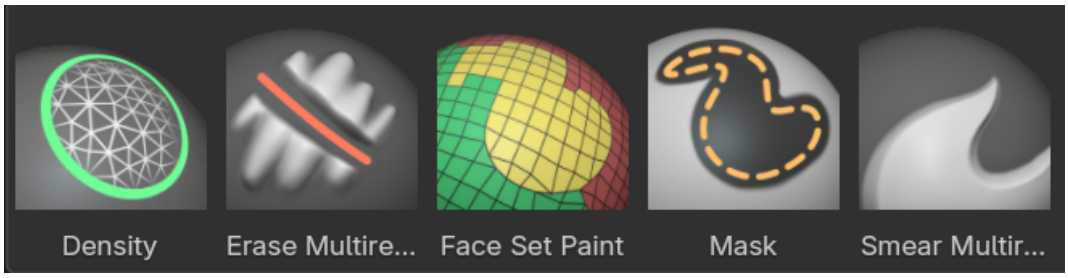
Similar to the Relax Slide brush but pinches/relaxes geometry instead.

Boundary

Brush Type: Boundary

Transform specifically mesh boundaries with various deformations.

Utility Brushes



No clear color assignment. These brushes are general purpose brushes or specific.

Density

Brush Type: Density

Cleans up geometry by collapsing short edges. Specifically for use with Dyntopo.

Mask

Brush Type: Mask
Shortcut: `M`

Paints a selection on parts of the mesh to be unaffected by other brushes.

Draw Face Sets

Brush Type: Draw Face Sets

Paint new, smooth or extend existing Face Sets.

Erase Multires Displacement

Brush Type: Erase Multires Displacement

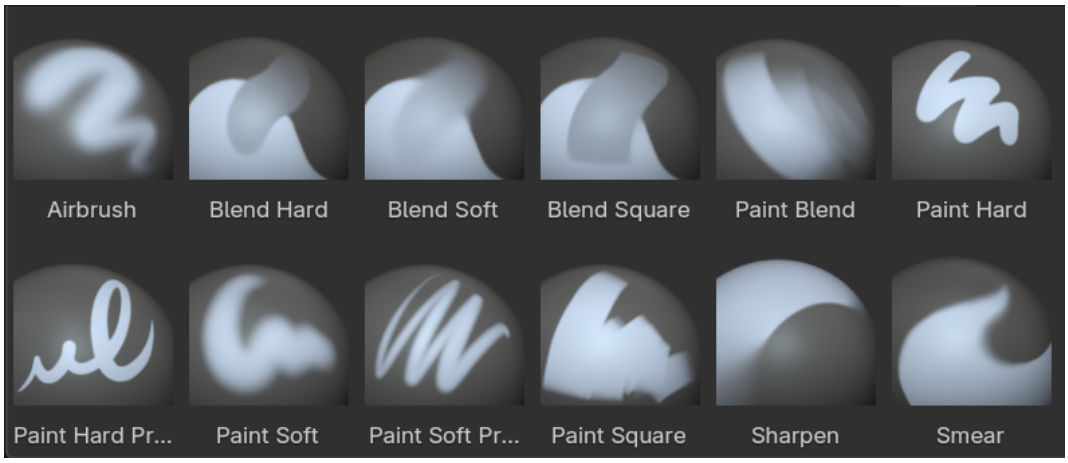
Remove displacement information on a Multiresolution modifier.

Smear Multires Displacement

Brush Type: Smear Multires Displacement

Smear displacement information on a Multiresolution modifier.

Painting Brushes



Recognizable by their blue thumbnails. These brushes are used for painting color attributes within sculpt mode.

Paint Hard
: Brush Type: Paint

  A simple hard round falloff.

Paint Soft
: Brush Type: Paint

  A soft round falloff with pressure sensitivity for only the strength.

Paint Hard Pressure
: Brush Type: Paint

  A hard round falloff with pressure sensitivity for the brush radius.

Paint Soft Pressure
: Brush Type: Paint

  A soft round falloff with pressure sensitivity for both radius and strength.

Paint Square

Brush Type: Paint

A hard square brush falloff.

Airbrush

Brush Type: Paint

A soft round brush that builds up over time instead of stroke distance.

Blend Hard

Brush Type: Paint

Similar to Average brushes in other modes with a hard round falloff. Used to blend colors along the stroke.

Blend Soft

Brush Type: Paint

Same as Blend Hard but with a soft round falloff.

Blend Square

Brush Type: Paint

Same as Blend Hard but with a hard square falloff.

Paint Blend

Brush Type: Paint

A mix of a Paint and Blend brush. On low pen pressure the brush averages colors and with high pen pressure it paints colors.

Smear

Brush Type: Smear

Smears colors along the stroke.

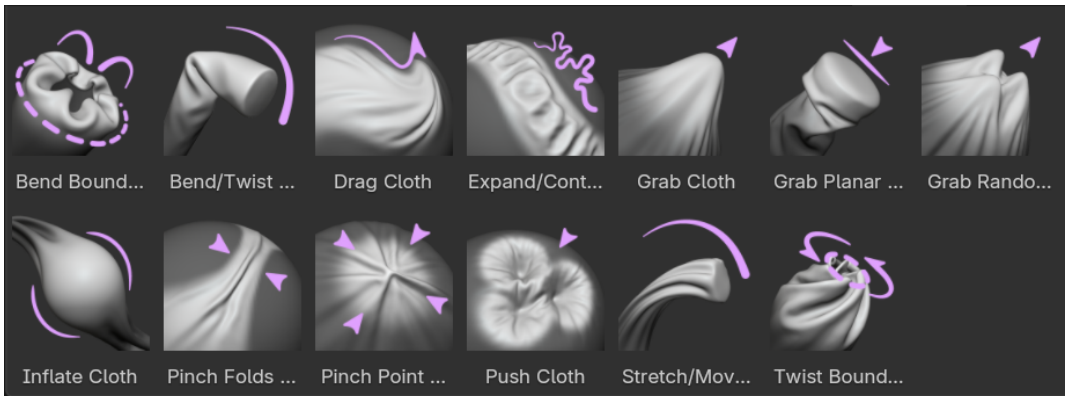
Sharpen

Brush Type: Smear

Pinches the colors inwards to create sharp edges or points.

Simulation Brushes

These brushes are similar to regular brushes but with an additional cloth simulation applied. These are ideally used on a relatively low resolution, since the mesh density defines the size of cloth dynamics.



Drag Cloth

Brush Type: Cloth

Nudges the geometry along the surface while minimally affecting the overall shape of the object.

Push Cloth

Brush Type: Cloth

Pushes geometry inwards or outwards.

Grab Cloth

Brush Type: Cloth

Grabs geometry within the brush radius firmly, while surrounding geometry is being simulated to follow.

Grab Planar Cloth

Brush Type: Cloth

Similar to Grab Cloth but with a line as the brush radius instead of a circle.

Grab Random Cloth

Brush Type: Cloth

Similar to Grab Cloth but with a noise texture applied to create more random variation.

Inflate Cloth

Brush Type: Cloth

Inflates the geometry outwards or inwards.

Expand/Contract Cloth

Brush Type: Cloth

Creates compression or stretching on geometry.

Pinch Point Cloth

Brush Type: Cloth

Pinches geometry to the center point of the radius, creating folds from all sides.

Pinch Folds Cloth

Brush Type: Cloth

Pinches only from two perpendicular sides along the stroke direction, creating parallel folds along the stroke.

Bend/Twist Cloth

Brush Type: Pose

A pose brush that rotates geometry.

Stretch/Move Cloth

Brush Type: Pose

A pose brush that translates and scales geometry.

Bend Boundary Cloth

Brush Type: Boundary

Bend only open boundaries of the mesh, folding the surrounding geometry in the process.

Twist Boundary Cloth

Brush Type: Boundary

Twist open boundaries of the mesh, creating twisting folds.

Draw

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Moves vertices inward or outward, based the average vertex normals within the brush radius. This is a very default behavior for sculpting and can be used in most cases.

It is common to use this particular brush with heavy customization for creating many custom brushes.

Brush Settings

Note

More info at General brush settings and on Advanced brush settings.

VDM Displacement

Vector Displacement Maps are supported for the *Draw* brush to insert complex & overhanging shapes. Unlike regular displacement, this uses all 3 color channels of the image to displace geometry in three directions instead of just one.



An example of various VDM brushes used on a smooth head from the official demo file.

Download the demo file for more information and to try the feature out.

To use this feature, enable Vector Displacement in the texture panel. All stroke methods are supported, but the recommended behavior is *Anchored*.

Ideal images for vector displacement are open EXR files with color clamping disabled.

Note

This feature is only supported with Area Plane mapping.

Draw Sharp

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Draw brush, but it always deforms the mesh from the original coordinates and uses the *Sharper* Falloff by default.

Draw Sharp is useful on high density meshes for creating cloth wrinkles, stylized hair or hard surface edges. To further sharpen or polish sharp edges in the case that the mesh density is not enough, it's recommended to use the Pinch, Crease or Multiplane Scrape brushes.

A limitation is that the brush does not remesh the sculpted surfaces with Dyntopo enabled. Because of that, a better brush to use with Dyntopo can be Crease.

Brush Settings

General

Direction
: On *Subtract* by default to carve in creases. More info at Direction

Note

More info at General brush settings and on Advanced brush settings.

Clay

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Draw brush, but includes settings to adjust the sculpt plane on which the brush acts. That's because it behaves like a combination of the Flatten and *Draw* brushes.

This brush is useful for building and removing volumes and shapes like real clay, because it flattens details as you add/subtract from the surfaces.

If used together with Dyntopo it's easy to continuously build shapes, even in a single stroke.

Brush Settings

General

Hardness
: Slightly higher by default. This makes the profile of the brush more noticeable. More info at Hardness

Auto-Smooth
: Enabled by default for a consistent smoothing effect. With lower brush strength (for example with lower pen pressure) the smoothing effect will be more noticeable and can be used to create and then blend/polish shapes in a single stroke. Enable Pressure to modulate the use of auto-smooth even more with pen inputs. More info at Auto-Smooth

Note

More info at General brush settings and on Advanced brush settings.

Clay Strips

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Clay brush, but it uses a square tip shape instead of a round one.

Just like the *Clay* brush, it's useful for building and removing volumes and shapes like real clay, because it flattens details as you add/subtract from the surfaces.

Clay Strips is very commonly used for aggressive building of volumes and deliberate control over shapes on the surface. This brush alone can be used for a fast rough pass over the entire sculpt, with additional smoothing or polishing often required afterwards. This brush can be very versatile with varying stroke directions, repeated strokes and pen pressure to achieve various results.

If used together with Dyntopo it's easy to continuously build shapes, even in a single stroke.

Brush Settings

General

Normal Radius
: Higher by default. This ensures that the brush does not change directions to sporadically during a stroke. More info at Normal Radius

Tip Roundness

Very low by default for a square shape for more deliberate shaping. More info at Tip Roundness

Note

More info at General brush settings and on Advanced brush settings.

Clay Thumb

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Clay brush. It imitates the effect of deforming clay with the finger, accumulating material during the stroke. The sculpt plane tilts during the stroke in the front part of the brush to achieve this effect.

Brush Settings

Note

More info at General brush settings and on Advanced brush settings.

Layer

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

This brush is similar to Draw, except that the height capped. This creates the appearance of a flat layer.

It is recommended to use the Persistent setting and regularly Set Persistent Base, so that multiple strokes to not add on top of each other.

Brush Settings

General

Hardness
: Higher by default to ensure the profile of layers is more noticeable. More info at Hardness

Note

More info at General brush settings and on Advanced brush settings.

Unique

Height
: The fixed height of each stroke. This is measured using the scene scale, so it is consistent no matter the amount of zoom or or object size.

Persistent

This will ensure that multiple strokes use the same height, as if sculpting a single layer.

Set Persistent Base

This button resets a new base so that you can sculpt new layer.

Inflate

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to Draw, except that vertices are moved in the direction of their own normals. Especially useful when sculpting meshes with a lot of curvature.

Also available as a Mesh Filter to inflate all unmasked areas at once.

Brush Settings

General

Direction
: Either Inflate or Deflate sculpted areas. This is different from the typical Add & Subtract.

Note

More info at General brush settings and on Advanced brush settings.

Blob

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to Draw, but vertices are pushed outwards like an inverted pinching effect. This will lead to a more consistent spherical curvature and thickening of strokes.

Brush Settings

General

Magnify
: By default at 0.5 to push out the mesh during the stoke. More info at Pinch/Magnify

Note

More info at General brush settings and on Advanced brush settings.

Crease

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Create sharp indents or ridges by pushing or pulling the mesh, while pinching the vertices together.

Crease can also be used to sharpen and polish existing creases. Enable pressure sensitivity on Strength to regulate the add/subtract effect while pinching creases.

Brush Settings

General

Pinch
: Adds a consistent pinching effect to your stroke. If set to 0 it the brush will behave like the Draw brush. If set to 1 and the brush strength set to 0, the brush will behave like a Pinch brush.

Note

More info at General brush settings and on Advanced brush settings.

Smooth

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Smooths the positions of the vertices to either polish surfaces or remove volume from larger shapes. Because this brush is so essential, it's always accessible by holding `Shift` and sculpting.

Also available as a Mesh Filter to smooth all unmasked areas at once.

Note

The brush called *Smooth* will be used whenever holding `Shift` and sculpting. If the smoothing strength or behavior needs to be changed, switch to the *Smooth* brush and adjust the settings there.

Brush Settings

General

Strength
: The strength of the smoothing is relative to the density of the mesh. If the resolution is increased on the sculpted mesh, the strength of the smooth brush will be weaker than before and needs to be increased.

Direction `Ctrl`
: Smooth
  : Smooths the surface of the mesh.

Enhance Details
: Enhances details on the surface of the mesh by applying a smoothing operation in the opposite direction.

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation
: There are two deformation types.

Laplacian:: Smooths the surface and volumes. This is the default behavior.

Surface:: Smooths only the surface of the mesh, while preserving the volume.

Shape Preservation
: How much of the original shape is preserved while smoothing. Increasing the value reduces the effect of having multiple iterations on the strength of smoothing.

Per-Vertex Displacement
: How much the position of each individual vertex influences the final result. Increasing the value reduces the overall strength of smoothing.

Iterations
: Number of smoothing iterations per brush step.

Note
This method works by applying regular smoothing, computing the difference between

the original (blended between start of iteration and fully original based on *Shape Preservation*) and the smoothed mesh, smoothing these offsets, pushing vertices back using the smoothed offsets, and finally blending in the original mesh based on *Per-Vertex Displacement*.

Flatten

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Flatten or contrast surfaces by pulling and pushing them towards (or away from) a defined medium height. This medium height is determined via an area plane within the brush radius.

This area plane height can be further defined in the brush settings.

Brush Settings

General

Direction `Ctrl`
: Invert the direction to push surfaces away from the sculpt plane, creating more surface contrast as a result.

Note

More info at General brush settings and on Advanced brush settings.

Fill

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Flatten brush, but only pushes surfaces upwards to the medium height.

Although `Ctrl` can be held to invert the effect to a Scrape brush, if *Invert to Scrape* is enabled. When disabled, the inverted direction will push surfaces away.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Invert to Scrape
: When enabled, holding `Ctrl` while sculpting changes the brush behavior to be the same as the *Scrape* brush. When disabled, holding `Ctrl` while sculpting, will push vertices below the cursor downward.

Scrape

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Flatten brush, but only pushes surfaces downward to the medium height.

Although `Ctrl` can be held to invert the effect to a Fill brush, if *Invert to Fill* is enabled. When disabled, the inverted direction will push surfaces away.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Invert to Fill
: When enabled, holding `Ctrl` while sculpting changes the brush behavior to be the same as the *Fill* brush. When disabled, holding `Ctrl` while sculpting, will push vertices above the cursor up away from the cursor.

Scrape Multiplane

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Scrapes the mesh with two angled planes at the same time, creating a sharp edge between them. This is useful for creating & polishing hard surface objects.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Plane Angle
: The angle between the two planes of the brush, pressing `Ctrl` inverts the angle.

Dynamic Mode
: When enabled, the angle is dynamically updated based on the surface under the brush. The *Plane Angle* then controls how much the angle will increase when applying pen pressure. When pressing `Ctrl`, it locks the plane angle to 0 degrees.

Show Cursor Preview

Displays a preview of the two planes and the angle they form during the stroke.

Pinch

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ‣ *Tool* ‣ *Brush Settings* ‣ *Advanced* ‣ *Brush Type*

Pulls geometry towards the center of the brush. When inverted via `Ctrl` it will *Magnify* geometry by pushing them away from the center of the brush.

Brush Settings

General

Note

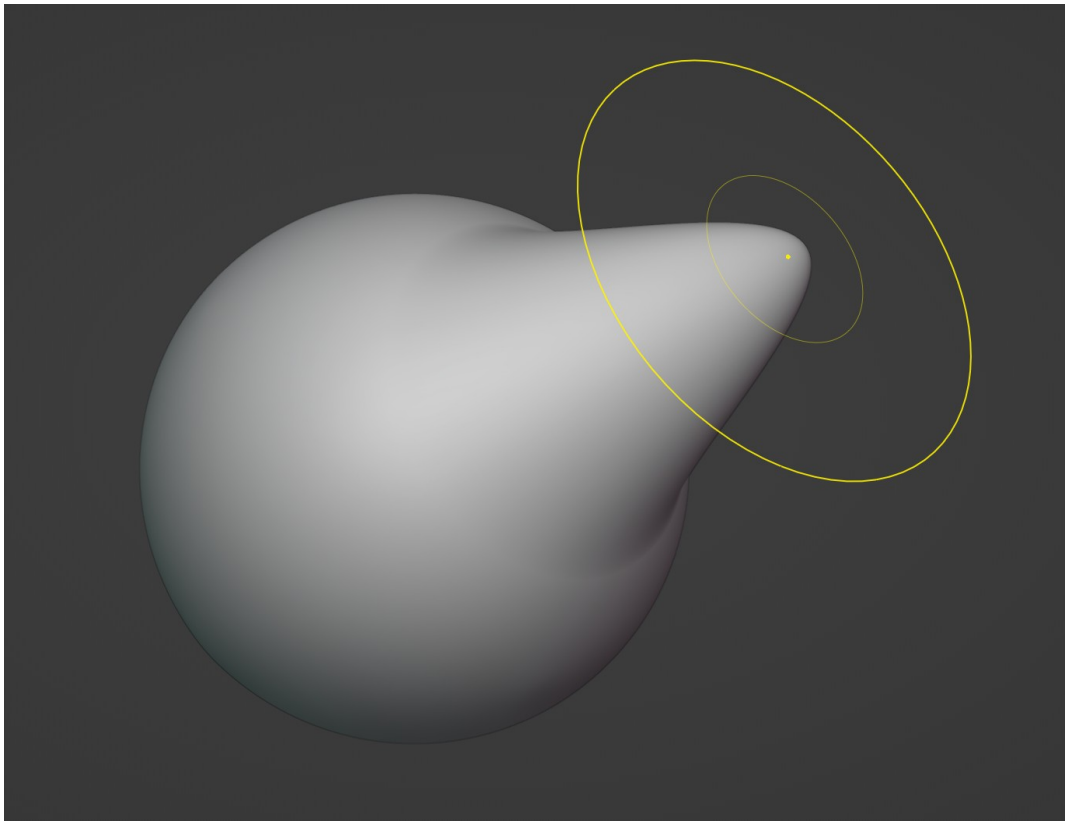
More info at General brush settings and on Advanced brush settings.

Grab

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ‣ *Tool* ‣ *Brush Settings* ‣ *Advanced* ‣ *Brush Type*

Drag geometry across the screen, following the cursor. *Grab* only moves the vertices that are under the brush radius at the start of the stroke. This is an essential sculpting brush to be used frequently to build shapes and adjust proportions.



Note

The effect is similar to moving geometry in Edit Mode with Proportional Editing enabled, except that *Grab* can make use of other Sculpt Mode options and brush settings.

Brush Settings

General

Radius
: Pressure Sensitivity is not supported for this brush type. More info at Radius.

Strength
: Pressure Sensitivity is not supported for this brush type. More info at Strength.

Normal Radius
: For this brush, this setting is a purely visual change. It does not alter the brush behavior. More info at Normal Radius.

Auto-Smooth
: This setting is not supported. More info at Auto-Smooth.

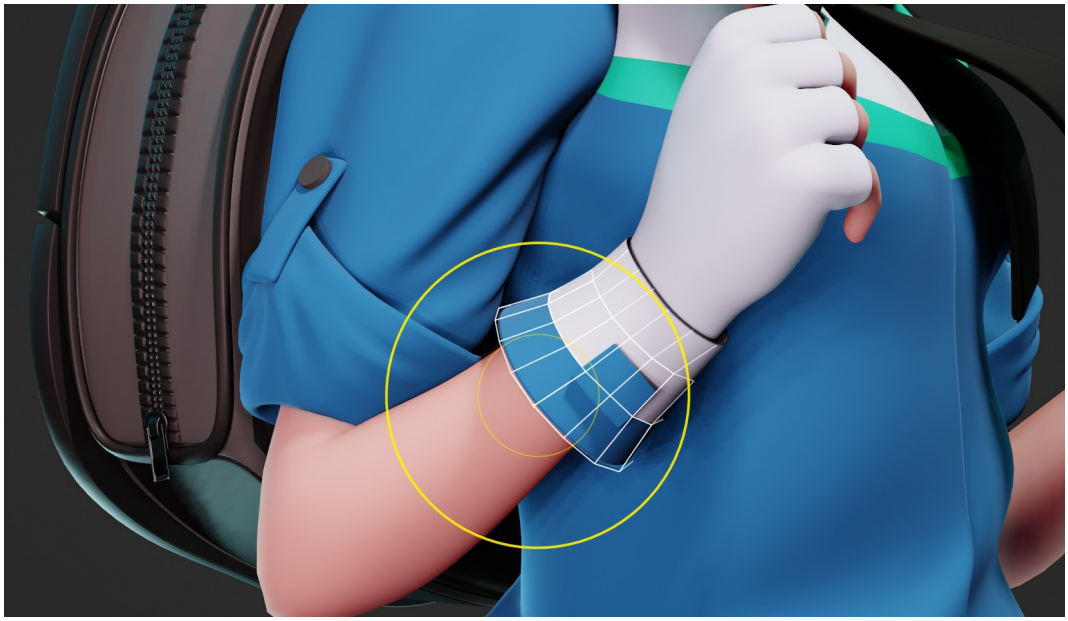
Note

More info at General brush settings and on Advanced brush settings.

Unique

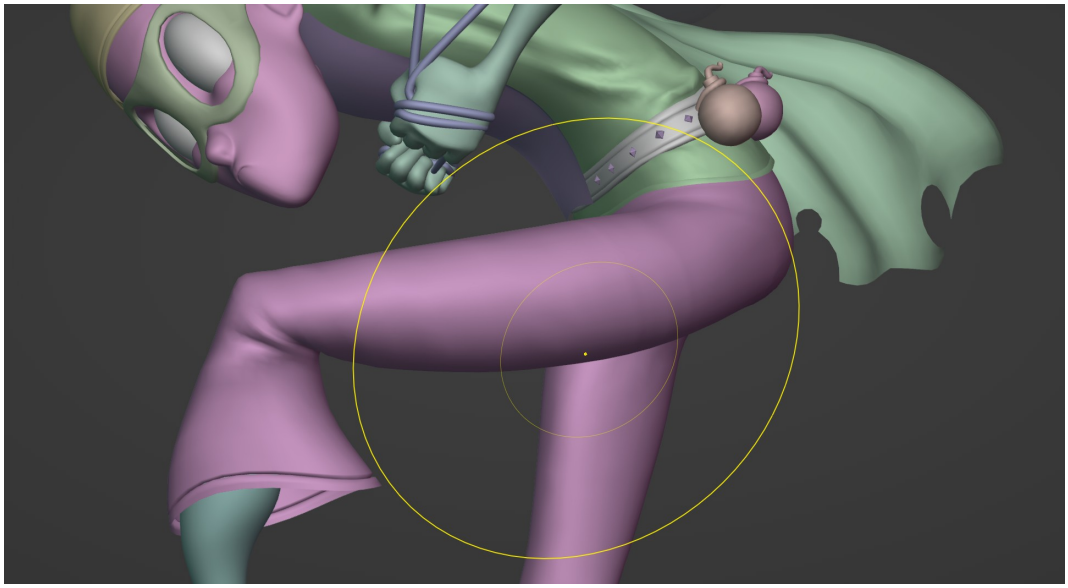
Grab Active Vertex
: Applies the maximum strength of the brush to the highlighted active vertex, making it easier to manipulate low-poly models or meshes with modifiers.

Enabling this option also enables a white wireframe overlay within the brush radius. This helps to visualize the real base geometry that is being manipulated while sculpting with Modifiers.



Grab Silhouette

Preserves the object's silhouette shape by only grabbing vertices on one side of the mesh curvature. The shape of the silhouette is determined by the orientation of the 3D Viewport and the start of the stroke.



Note how in the image only the bottom side of the leg is pulled down, despite the size of the brush.

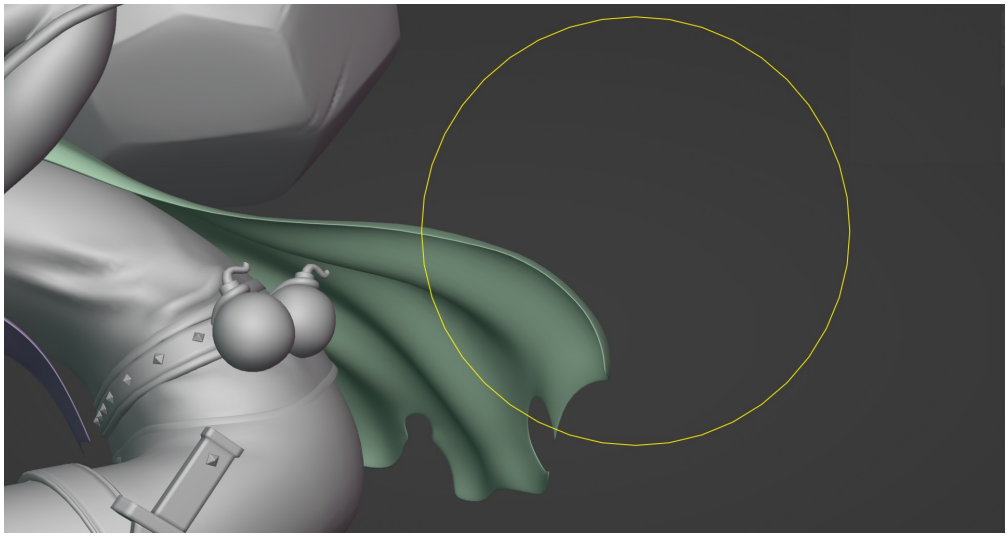
Tip

This setting is also useful for grabbing a single side of a crease and pushing it further inwards, creating a more pinched crease.

Additional Workflows

2D Grab Brush

If the Falloff Shape is set to *Projected*, the brush can grab infinitely deep into the viewport. This is especially useful for much broader changes to a sculpt.



The stroke can also be started outside of the mesh (like in empty 3D space) and grab the vertices within the brush radius. This can be useful for sculpting flat and tube-like meshes.

Elastic Deform

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ‣ *Tool* ‣ *Brush Settings* ‣ *Advanced* ‣ *Brush Type*

Used to simulate realistic deformations such as grabbing or twisting of Elastic objects. For example, this tool works great for modeling the shape of organic objects such as humans or animals. When pressing `Ctrl`, the brush deforms vertices along the normal of the active vertex.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation
: The surface alteration that is used in the brush.

Grab::	Used to drag a group of vertices around.
Bi-scale Grab::	Like *Grab* but the falloff is more localized to the center of the brush.
Tri-scale Grab::	Like *Bi-scale Grab* but the falloff is more localized to the center of the brush.
Scale::	Displaces vertices away from the active vertex.

Twist:: Vertices are rotated around the active vertex.

Volume Preservation

Higher values preserve volumes more, but also lead to more bulging. (This value determines the poisson ratio for elastic deformation)

Snake Hook

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Pulls vertices along with the movement of the brush to create long, snake-like forms. During the stroke, geometry will be dynamically picked up & let go.

When the *Rake* setting is used, the brush can also be used to rotate geometry via dragging.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Magnify
: Pulled geometry tends to lose volume along the stroke. With *Magnify* value greater than 0.5 this is prevented. More info at Pinch/Magnify

Rake
: Rotates geometry along the direction of the stroke.

Deformation

Deformation type that is used by the brush.

Radius Falloff:: Applies the brush falloff to the tip of the brush.

Elastic:: Modifies the entire mesh using an Elastic deformation. More info in the Elastic Deform brush.

Thumb

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Grab brush, but instead only moves the geometry along the surface (using the area plane). This is useful for grabbing surfaces with a very specific direction, or without making too impactful changes to the overall object shape.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Pose

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Deform a model simulating armature-like workflow. This can either be useful for posing a model without a rig, adjusting the proportions of a mesh or other fast deformations.

The brush will automatically determine an origin point, indicated with a while line on the brush cursor.

If the Deformation Target is changed, the brush can also be used for cloth sculpting.

Brush Settings

General

Only Radius and Auto-Masking has an impact on the brush behavior for this brush.

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation

Deformation type that is used by the brush.

Rotate/Twist:: Rotates the mesh around the pose origin. When pressing `Ctrl`, the brush applies a twist rotation instead (and disables any IK segments that are used).

Scale/Translate:: Scale the mesh based on the pose origin. While holding `Ctrl` the brush moves the mesh.

Squash/Stretch:: Works similar to *Scale/Translate* however, it applies different scale values along different axes to achieve the stretching effect.

Rotation Origins

Method to set the rotation origin for the pose origin or individual IK segments.

Topology:: Sets the rotation origin automatically using the topology and shape of the mesh.

Face Sets:: Creates a pose segment per Face Set, starting from the active face set. This can lead to the most accurate and desirable results.

Face Sets FK:: Simulates a Forward Kinematics deformation using the Face Set under the cursor as control.

Pose Origin Offset

Offset of the pose origin in relation to the brush radius. This is useful to manipulate areas with a lot of complex shapes like fingers.

Smooth Iterations

Controls the smoothness of the falloff of the deformation.

Pose IK Segments

Controls how many IK segments are going to be created for posing. This can be seen by a divided white line on the cursor. This is also useful for making curved deformations with the pose brush, like hair clumps and tails.

Lock Rotation when Scaling

When using *Scale/Translate Deformation*, do not rotate the segment; only scaling is applied.

Keep Anchor Point
: Keeps the position of the last segment in the IK chain fixed. If this is disabled, the mesh can be dragged around more freely, creating snake like shapes.

Connected Only
: The brush will only affect topologically connected elements. Disabling this will allow deforming multiple disconnected meshes at the same time, for example characters with clothing & shoes.

  Disabling this setting can have a big impact on performance, as neighboring elements will be merged internally. Keeping the *Max Element Distance* as low as possible will help counteract the performance impact.

Max Element Distance
: Maximum distance to search for disconnected loose parts in the mesh.

Nudge

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Moves vertices in the direction of the brush stroke.

Similar to the Snake Hook brush, but instead only moves the geometry along the surface (using the area plane). This is useful for grabbing geometry along curved surfaces, without making too impactful changes to the overall object shape.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Rotate

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ‣ *Tool* ‣ *Brush Settings* ‣ *Advanced* ‣ *Brush Type*

Rotates geometry in the direction in which the cursor is moved. The initial drag direction is the zero angle and by rotating around the center you can create a vortex/swirl effect.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Relax Slide

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

This brush deforms the topology of the mesh while minimizing changes to the geometrical shape of the mesh. By default it will drag geometry, but this can be changed in the *Deformation* settings.

This brush is especially useful for redistributing topology to areas that require more detail, or sliding geometry to somewhere where they should be.

Holding `Shift` changes the brush effect to Relax geometry, creating an even distribution of topology.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation
: Deformation type that is used by the brush.

Drag::	Slides the topology of the mesh in the direction of the stroke.
Pinch::	Slides the topology of the mesh towards the center of the stroke.
Expand::	Slides the topology of the mesh away from the center of the stroke.

Boundary

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Similar to the Pose brush but deforms the open boundaries of a mesh. The tool detects the mesh boundary closest to the active vertex and propagates the deformation using the brush Falloff into the mesh.

The main use cases of this brush are the *Bend* and *Expand* geometry, which leads to the best results on evenly distributed quad based topology. Use the *Inflate*, *Grab*, *Twist*, and *Smooth* deformation modes, to further adjustments and tweaks to the result (which do not depend that much on a clean topology).

Tip

Boundaries to hidden geometry will also be counted as an open boundary.

The boundary origin is displayed via a white line, which indicates the reach of the deformation. The targeted boundary that will be deformed is highlighted in the brush cursor color.

If the Deformation Target is changed, the brush can also be used for cloth sculpting.

Note

Evenly distributed and quad based topology will lead to much better results. Triangles and N-gons are also supported but may lead to

unpredictable outcomes.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation
: Deformation type that is used by the brush.

Bend::	Rotates the boundary around the local Y axis. Useful for creating folding shapes, like sleeves.
Expand::	Moves/extends the mesh boundary in the local X direction. Useful for extending the boundaries along the surface.
Inflate::	Works similar to the Inflate tool but, the vertices that are inflated are constrained to the mesh boundary.
Grab::	Works similar to the Grab tool but, the vertices that are grabbed are constrained to the mesh boundary.
Twist::	Rotates the active boundary around the local Z axis. Useful for creating folds like on a skirt.
Smooth::	Works similar to the Grab tool but, the vertices that are smoothed are constrained to the mesh boundary.

Boundary Falloff
: How the brush Falloff is applied across the boundary.

Constant::	Applies the same deformation in the entire boundary.
Brush Radius::	Applies the deformation only within the brush radius.
Loop::	Applies the brush falloff in a loop pattern along the boundary.
Loop and Invert::	Applies the falloff radius in a loop pattern, inverting the direction back & forth.

Boundary Origin Offset

Offset of the boundary origin in relation to the brush radius.

Cloth

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

This brush simulates cloth physics on the mesh under the brush cursor. There are various deformation types and settings to customize the brush.

It's also easy to sculpt the mesh with other brushes and tools in between using the cloth brushes.

Note

Using a relatively small brush size makes the calculations much faster, while larger brush sizes might be too slow to get a usable brush.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Persistent

Allows the cloth brush to not accumulate deformation after each stroke. This is convenient to always simulate the cloth based on the same initial shape, but applying different forces to it.

When disabled, deformations accumulate after each stroke.

Set Persistent Base
: Resets the base mesh so that you can add another layer of deformations.

Simulation Area
: Selects the part of the mesh that is going to be simulated when the stroke is active. This can greatly affect performance depending on the complexity of the mesh.

  Local
  : Simulates only a specific area around the brush limited by a fixed radius.

  Global
  : Simulates the entire mesh.

  Dynamic
  : The active simulation area moves with the brush while still being limited by a fixed radius.

Simulation Limit
: The Factor added relative to the size of the radius to limit the cloth simulation effects.

Simulation Falloff
: The area to apply deformation falloff to the effects of the simulation. This setting is a factor of the *Simulation Limit* and is shown as a dashed line around the cursor.

Pin Simulation Boundary
: Lock the position of the vertices in the simulation falloff area to avoid artifacts and create a softer transition with unaffected areas.

Deformation

The type of cloth deformation that is used by the brush.

Drag:: Simulates pulling the cloth to the cursor, similar to placing a finger on a table cloth and pulling.

Push:: Simulates pushing the cloth away from the cursor, similar to placing a finger on a table cloth and pushing.

Pinch Point:: Simulates pulling the cloth into a point.

Pinch Perpendicular:: Simulates pulling the brush into a line.

Inflate:: Simulates air being blown under the cloth so that the cloth lifts up.

Grab:: Simulates picking up and moving the cloth.

Expand:: Simulates stretching the cloth out.

Snake Hook:: Simulates moving the cloth without producing any artifacts in the surface and creates more natural looking folds than any of the other deformation modes. This is accomplished by adjusting the strength of the deformation constraints per brush step to avoid affecting the results of the simulation as much as possible.

Force Falloff

Shape used in the brush to apply force to the cloth.

Radial:: Applies the force as a sphere.

Plane:: Applies the force as a plane.

Cloth Mass

Mass of each simulation particle.

Cloth Damping

How much the applied forces are propagated through the cloth.

Soft Body Plasticity

The amount the cloth preserves its original shape, acting as a Soft Body.

Use Collisions

Enables the detection of collisions with other objects during the simulation. In order for the sculpt object to collide with objects, the collision object must have Collision Physics activated.

Density

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

The Density brush is specifically meant for use with Dyntopo to remove/add detail in the mesh.

Even when the Refine Method is set to *Subdivide Edges*, this brush is always able to collapse edges. This ensures that while focusing on adding detail to your sculpt, the *Density* brush can always be used to simplify and polish surfaces.

This brush has no effect if Dyntopo is disabled.

Tip

In combination with auto-smooth the brush can polish surfaces while it remeshes them. On tube-like geometry it can also shrink and dissolve volumes completely.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Mask

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Paint a selection on parts of the mesh to be unaffected by other brushes & tools. The mask values are shown as a gray-scale overlay.

Note

More information in the Masking Introduction.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Mask Tool
: The mask brush has two modes:

Draw::	Mask drawing.
Smooth::	Holding `Shift` will instead smooth existing masks.

Draw Face Sets

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Draw new or extend existing Face Sets with each stroke.

Holding `Ctrl` will continue drawing the same face set as the one under the cursor. Holding `Shift` will relax or smooth the edges of the face sets by modifying the underlying topology so edges flow along the perimeter of the face sets. This will remove the jagged lines visible after drawing or creating a face set.

Note

More information in the Face Set Introduction.

Brush Settings

General

While a lot of the general brush settings are supported, it's not needed to change them from the default, as the brush purpose is very simple.

Note

More info at General brush settings and on Advanced brush settings.

Erase Multires Displacement

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

This brush deletes displacement information of the Multires Modifier, resetting the mesh to a regular subdivision surface result.

This can be used to reset parts of the sculpt or to fix reprojection artifacts after applying a Shrinkwrap Modifier.

Tip

This brush works best after using Apply Base.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Smear Multires Displacement

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

This tool deforms displacement information of the Multires Modifier, moving the displaced vertices without affecting the base mesh.

Smearing effect can be used multiple times over the same area without generating any artifacts in the topology.

Tip

This brush works best after using Apply Base.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

Deformation
: Deformation type that is used by the brush.

Drag::	Pulls the displacement values in the direction of the brush.
Pinch::	Pulls the displacement values towards the center of the brush, creating hard surface effects without pinching the topology.
Expand::	Pushes the displacement values away from the brush center, smoothing the displacement.

Paint

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ▸ *Tool* ▸ *Brush Settings* ▸ *Advanced* ▸ *Brush Type*

Paints on the active color attribute. Hold `Shift` to blur painted colors instead.

Color attribute's can be managed in the pallette pop-over in the middle of the header.

Note

More information in the Painting Introduction.

Brush Settings

General

Strength
: This settings has a different effect on this brush. Instead of defining the strength of each individual step in the stroke, it determines the overall Opacity of the applied color.

  Use the *Flow* setting instead for faster increasing of strength.

Note

More info at General brush settings and on Advanced brush settings.

Unique

Flow
: Amount of paint that is applied per stroke sample. Used to create fast/slow accumulation effect.

Wet Mix
: Amount of paint that is picked from the surface into the brush color. Can achieve the effect of a wet canvas.

Wet Persistence
: Amount of wet paint that stays in the brush after applying paint to the surface.

Wet Paint Radius
: Ratio between the brush radius and the radius that is going to be used to sample the color to blend in wet paint.

Density
: Amount of random elements that are going to be affected by this brush. Use this for a more detailed airbrush effect. This works best on a high resolution.

Tip Scale X
: Scale of the brush tip in the X axis. This is useful for a achieving a painting stroke like a marker or paint roller.

Smear

Reference

Mode::	Sculpt Mode
Brush::	*Sidebar* ‣ *Tool* ‣ *Brush Settings* ‣ *Advanced* ‣ *Brush Type*

Smears the painted colors of the active color attribute. It takes the colors under the cursor, and blends them in the direction of your stroke.

Brush Settings

General

Note

More info at General brush settings and on Advanced brush settings.

Unique

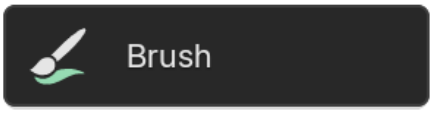
Deformation

Drag::	Smear colors along the direction of the stroke.
Pinch::	Smear colors inwards towards your brush center.
Expand::	Smear colors outwards away from your brush center.

Toolbar

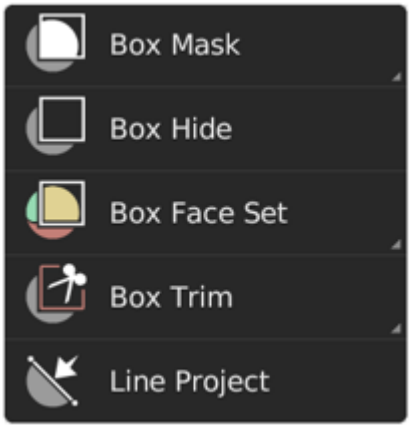
The amount of tools in sculpt mode is very extensive. This is an overview of all of them, categorized by their general functions.

Sculpting Tools



Brush
: Tool to use for any of the *Sculpt* mode brushes.

Gesture Tools



General gesture tools to apply an operation via box, lasso, line and polyline shapes. See Gesture Tools for more information.

Mask Gesture Tools
: Create a mask via a gesture.

Hide Gesture Tools
: Hides/Shows geometry via a gesture.

Face Set Gesture Tools
: Create a face set via a gesture.

Trim Gesture Tools
: Perform a Boolean operation via a gesture.

Line Project
: Flatten the geometry towards a drawn line.

Filter Tools

Tools for applying effects on the entire unmasked and visible mesh.



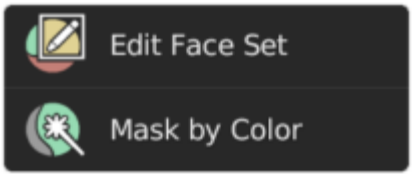
Mesh Filter
: Apply a deformation to all unmasked vertices.

Cloth Filter
: Applies a cloth simulation to all unmasked vertices.

Color Filter
: Changes the active color attribute on all unmasked vertices.

Single Click Tools

Simpler tools that apply an operation on surfaces that are clicked on.



Edit Face Set
: Modifies the face set under the cursor.

Mask by Color
: Create a mask from any color from the color attribute by clicking on it.

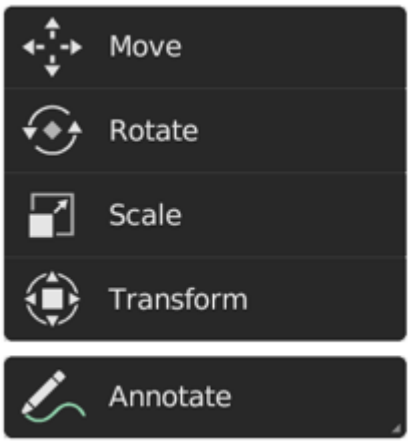
General Tools

General transform and annotate tools like in other modes.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.



Transform

Adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line

Draw straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previous drawn annotations.

Tools

- Brush
- Mask Gesture Tools
- Hide Gesture Tools
- Face Set Gesture Tools
- Trim Gesture Tools
- Line Project
- Mesh Filter
- Cloth Filter
- Color Filter
- Edit Face Set
- Mask by Color
- Transforms

Brush

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar* ‣ *Brush*

Tool to use for any of the *Sculpt* mode brushes. Activating a brush from an asset shelf or brush selector will also activate this tool for convenience.

See the list of Essentials Brushes (based on available Brush Types) fore more details.

Mask Gesture Tools

Reference

Mode:: Sculpt Mode

Mask gesture tools apply a constant value to all selected vertices within the selection area. By default, these tools fully mask each vertex. Holding `Ctrl` while performing the selection clears the mask.

All mask gesture tools can be activated in the Toolbar and are comprised of the following:

Box Mask

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Box Mask*
Shortcut:: `B`

Creates a new Mask based on a box gesture.

Lasso Mask

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Lasso Mask*

Creates a new Mask based on a lasso gesture.

Line Mask

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Line Mask*

Creates a new Mask based on a line gesture.

Polyline Mask

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Polyline Mask*

Creates a new Mask based on a polyline gesture.

Tool Settings

Front Faces Only
: Only creates a mask on the faces that face towards the view.

Hide Gesture Tools

Reference

Mode:: Sculpt Mode

Hide gesture tools hide all selected vertices within the selection area and any of their connected edges and faces. Holding `Ctrl` while performing the selection reveals the vertices, edges, and faces.

Pressing `LMB` with any of these tools without also dragging reveals all elements of a mesh.

All hide gesture tools can be activated in the Toolbar and are comprised of the following:

Box Hide

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Box Hide*

Hides vertices and connected edges and faces based on a box gesture.

Lasso Hide

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Lasso Hide*

Hides vertices and connected edges and faces based on a lasso gesture.

Line Hide

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Line Mask*

Hides vertices and connected edges and faces based on a line gesture.

Polyline Hide

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ▸ Polyline Mask*

Hides vertices and connected edges and faces based on a polyline gesture.

Note

The Polyline Hide tool does not support showing all vertices via pressing `LMB`.

Tool Settings

Visibility Area
: Determines whether all vertices inside or outside the selected area should be affected.

  Inside:: All vertices and connected elements inside the selection area will be hidden.
  Outside:: All vertices and connected elements outside the selection area will be hidden.

Face Set Gesture Tools

Reference

Mode:: Sculpt Mode

Face Set gesture tools apply a single new Face Set to all faces within the selection area.

All Face Set gesture tools can be activated in the Toolbar and are comprised of the following:

Box Face Set

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ‣ Box Face Set*

Creates a new Face Set based on a box gesture.

Lasso Face Set

Reference

Mode:: Sculpt Mode
Tool:: *Toolbar ‣ Lasso Face Set*

Creates a new Face Set based on a lasso gesture.

Line Face Set

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Line Face Set*

Creates a new Face Set based on a line gesture.

Polyline Face Set

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Polyline Face Set*

Creates a new Face Set based on a polyline gesture.

Tool Settings

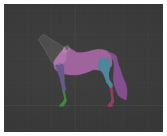
Front Faces Only
: Only creates a face set on the faces that face towards the view.

Trim Gesture Tools

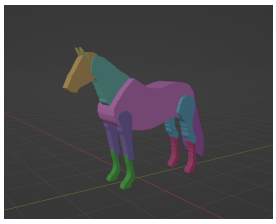
Reference

Mode:: Sculpt Mode

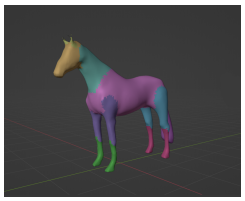
Trim gesture tools add or remove geometry based on a selection area. This tool is especially useful for sketching an early base mesh for further sculpting with the voxel remesher.



Using Lasso Trim set to *Join*



The symmetrized mesh.



Sculpting with voxel remeshing.

New geometry is assigned to a new Face Set. When removing geometry, the new interior geometry along the selection will be assigned a new face set instead.

Note

It is not recommended to use this tool on a mesh above 100k vertices when using *Difference* or *Union* as the Trim Mode with the *Exact* Solver. This tool is using a Boolean operation so it might take a long time to process. For higher resolution meshes it is recommended to instead use the Line Project tool or the *Fair Positions* mode of the Edit Face Set tool to trim geometry.

Box Trim

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Box Trim*

Performs a Boolean operation based on the area defined by a box gesture.

Lasso Trim

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Lasso Trim*

Performs a Boolean operation based on the area defined by a lasso gesture.

Line Trim

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Line Trim*

Performs a Boolean operation based on the area defined by a line gesture.

Note

The Line Trim tool does not support adding geometry. Only *Difference* mode is supported.

Polyline Trim

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Polyline Trim*

Performs a Boolean operation based on the area defined by a polyline gesture.

Tool Settings

Solver
: Algorithm used to calculate the Boolean intersections.

  Fast:: Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.

  Exact:: Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Fast Solver*.

Trim Mode
: Geometry can be either added or removed by choosing one of these modes.

  Difference:: Removes geometry, filling any holes that are created.

  Union:: Creates a geometry and joins any intersections with existing geometry.

  Join:: Similar to *Union* but joins the mesh as separate geometry, without performing any Boolean operations with existing geometry.

Shape Orientation
: The method used to orientate the trimming shape.

  View:: Use the view to orientate the trimming shape.

Surface:: Use the surface normal to orientate the trimming shape.

Extrude Mode

Fixed:: Aligns new geometry orthogonally for 90 degree angles in depth.

Project:: Aligns new geometry with the perspective of the current view for a tapered result.

Use Cursor for Depth

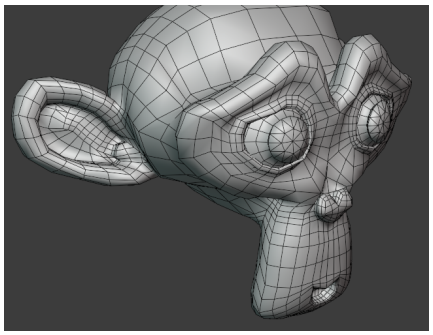
Use cursor location and radius for the dimensions and position of the trimming shape. If not set, the tool uses the full depth of the object from the camera view.

Line Project

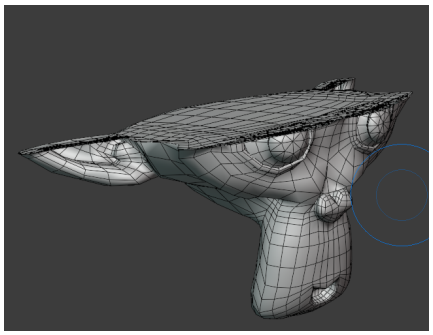
Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Line Project*

This tool flattens the geometry along a plane determined by the camera view and a drawn line. The region of the mesh being flattened is visualized by the side of the line that is shaded.



Before Line Project.



After Line Project.

Usage

Use the tool by:

1. Orient the 3D Viewport to define the direction in depth.
2. LMB and hold while moving the cursor to define direction of the line projection.
3. Adjust the operation with extra *Controls* shortcuts.
4. Release LMB to confirm.

Controls

Flip `F`
: Changes the side of the line that the tool projects geometry.

Snap `Ctrl`
: Constrains the rotation of the line to 15 degree intervals.

Move `Ctrl-Spacebar`
: Reposition the line.

Tool Settings

Limit to Segment
: The affected area will not extend the length of the drawn line. This helps defining a smaller area instead of extending the line infinitely long

Mesh Filter

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Mesh Filter*

Applies a deformation to all vertices in the mesh at the same time. Masking, auto-masking and visibility will be taken into account.

To use this tool, click and drag away from left to right or from right to left for a negative effect.

Tool Settings

Filter Type
: These are all of the available filter deformations.

Smooth:: Smooths the positions of the vertices to either polish surfaces or remove volume from larger shapes. Especially useful to fix most of the artifacts of the voxel remesher. This filter works similar to the Smooth brush.

Scale:: Increases the size of the mesh. This filter works similar to the Scale Transform.

Inflate:: Displaces vertices uniformly along their normal. This filter works similar to the Inflate brush.

Sphere:: Morphs the mesh progressively into a sphere. This filter works similar to the To Sphere Transform.

Random::

Randomly moves vertices along the vertex normal. This filter works similar to the Randomize Transform.

Relax:: Tries to create an even distribution of quads without deforming the volume of the mesh. This filter works the same as holding `Shift` with the Slide Relax brush.

Relax Face Sets:: This will remove the jagged lines visible after drawing or creating a face set. This filter works the same as holding `Shift` with the Draw Face Set brush.

Surface Smooth:: Eliminates irregularities of the mesh by making the positions of the vertices more uniform while preserving the volume of the object. This filter works similar to the *Surface* deformation type of the Smooth brush.

Shape Preservation
: How much of the original shape is preserved when smoothing.

Per-Vertex Displacement
: How much the position of each individual vertex influences the final result.

Sharpen:: Sharpens and smooths the mesh based on its curvature, resulting in pinching hard edges and polishing flat surfaces. Especially useful when sculpting hard surfaces and stylized models with creasing and flattening brushes.

Smooth Ratio
: How much smoothing is applied to polished surfaces.

Intensify Details
: Increases the high frequency surface details of the mesh by intensifying the difference between creases and valleys.

Curvature Smooth Iterations

The number of times the smoothing operation is applied per brush step. Controls how much smooth the resulting shape is, ignoring high-frequency details.

Enhance Details:: Increases the high frequency surface details of the mesh by intensifying the difference between creases and valleys. This filter works similar to the inverted direction of the Smooth brush.

Erase Displacement::
Deletes displacement information of the Multires Modifier, resetting the mesh to a regular subdivision surface result. This can be used to reset parts of the sculpt or to fix reprojection artifacts after applying a Shrinkwrap Modifier. Negative strokes will intensify the displacement details, this method works similar to *Enhance Details* and can give better results in some circumstances.

Strength
The amount of effect the filter has on the mesh. At certain object scales it can be useful to change this value.

Deformation Axis
Apply the deformation only on the selected axis.

Orientation
Orientation of the axis to limit the filter displacement.

Local:: Use the local axis to limit the displacement.
World:: Use the global axis to limit the displacement.
View:: Use the view axis to limit the displacement.

Cloth Filter

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Cloth Filter*

This tool works similar to the Cloth Brush, however, it applies a cloth simulation to all vertices in the mesh at the same time. Click and drag away from the object for a positive effect and towards for a negative effect.

Tip

Vertices can be "pinned" by masking vertices that should remain stationary, or by using Face Sets.

Tool Settings

Filter Type
: Operation that is going to be applied to the mesh.

Gravity::	Applies gravity to the simulation.
Inflate::	Inflates the cloth.
Expand::	Expands the cloth's dimensions.
Pinch::	Pinches the cloth to the point where the cursor was when the filter started.
Scale::	Scales the mesh as a Soft Body using the distance to the origin of the object as scale. This creates filter produces folds in the surface. The orientation of the folds can be controlled using the *Force Axis* and *Orientation*.

Strength

The amount of effect the filter has on the mesh.

Force Axis
Apply the force along the selected axis.

Orientation
Orientation of the axis to limit the filter force.

Local::	Use the local axis to limit the force and set the gravity direction.
World::	Use the world axis to limit the force and set the gravity direction.
View::	Use the view axis to limit the force and set the gravity direction.

Cloth Mass
Mass of each simulation particle.

Cloth Damping
How much the applied forces are propagated through the cloth.

Use Face Sets
Only applies the cloth forces to the vertices assigned to the Face Set that are under the mouse.

Use Collisions
Enables the detection of collisions with other objects during the simulation. In order for the sculpt object to collide with object, the collision object must have Collision Physics activated.

Color Filter

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Color Filter*

Apply color corrections or effects on the active color attribute on all vertices in the mesh at the same time.

To use this tool, click and drag away from left to right or from right to left for a negative effect.

Tool Settings

Filter Type

Fill::	Fills in a single color.
Hue::	Shifts the Hue of each color.
Saturation::	Increases or decreases the saturation.
Value::	Increases or decreases the values.
Brightness::	Increases or decreases the brightness.
Contrast::	Increases or decreases the contrast.
Smooth::	Blurs or sharpens the colors.
Red::	Increases or decreases the red channel.
Green::	Increases or decreases the green channel.
Blue::	Increases or decreases the blue channel.

Fill Color
: Set a color that will be used for the fill filter type.

Strength
: The amount of effect the filter has on the color attribute.

Edit Face Set

Reference

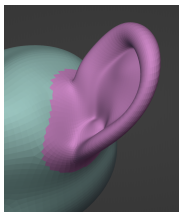
Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Edit Face Set*
Operator::	Grow/Shrink Face Sets

Edits the Face Set under the cursor.

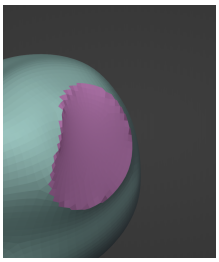
Tool Settings

Mode
: The operation to apply to the face set.

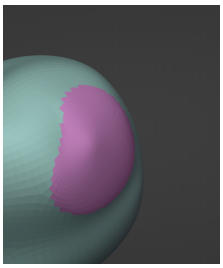
Grow Face Set::	Grows the face sets boundary by one face based on mesh topology. This is also available as a shortcut operator via `Ctrl-W`.
Shrink Face Set::	Shrinks the face sets boundary by one face based on mesh topology. This is also available as a shortcut operator via `Ctrl-Alt-W`.
Delete Geometry::	Deletes the faces that are assigned to the face set.
Fair Positions::	Creates a perfectly flat and smooth geometry patch from the face set. This is the ideal way to trim parts of your mesh if the vertex count is too high for other operations, or the vertex IDs must not be altered (Like when using Multires sculpting).
Fair Tangency::	Creates a smooth as possible geometry patch from the face set by minimizing changes in vertex tangents. This is ideal for creating smooth curved surfaces on complex topology, where just using the smooth brush will not lead to desired results



Before fairing.



After using Fair Positions.



After using Fair Tangency.

Strength

The amount of effect the filter has on the mesh. This setting is only available for the fairing operations.

Modify Hidden

Apply the edit operation to hidden face sets.

Mask by Color

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Mask by Color*

Click on any color on the mesh to create a new mask (based on the active color attribute).

Tool Settings

Threshold
: How much changes in color affect the mask generation. A smaller threshold includes fewer similar colors. A larger threshold includes much more similar colors.

Contiguous
: Mask only contiguous color areas. Colors that don't touch the one that you click on will not be masked.

Invert
: Invert the generated mask.

Preserve Previous Mask
: Preserve previous mask and add or subtract the new one generated by the colors.

Transforms

Move

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ‣ Move*

Translation tool.

Rotate

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ‣ Rotate*

Rotation tool.

Scale

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ‣ Scale*

Scale tool.

Transform

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Transform*

Tool to adjust the objects translation, rotations and scale.

Tool Settings

Each tool has the following settings to change how the unmasked mesh will be transformed.

Transform Mode
: How the transformation is going to be applied to the target.

All Vertices::	Applies the transformation to all vertices in the mesh.
Elastic::	Applies the transformation while dynamically simulating elasticity. Instead of applying this to all vertices, it uses the radius of the cursor as the area of effect.

Tool Settings

- Brushes
  - Introduction
  - Manage Brushes
  - Brush Settings
- Brush Settings
- Dyntopo
- Remesh
  - Known Limitations
- Symmetry
- Options
  - Gravity

Brush Settings

Information on brush settings for every mode can be found in these pages:

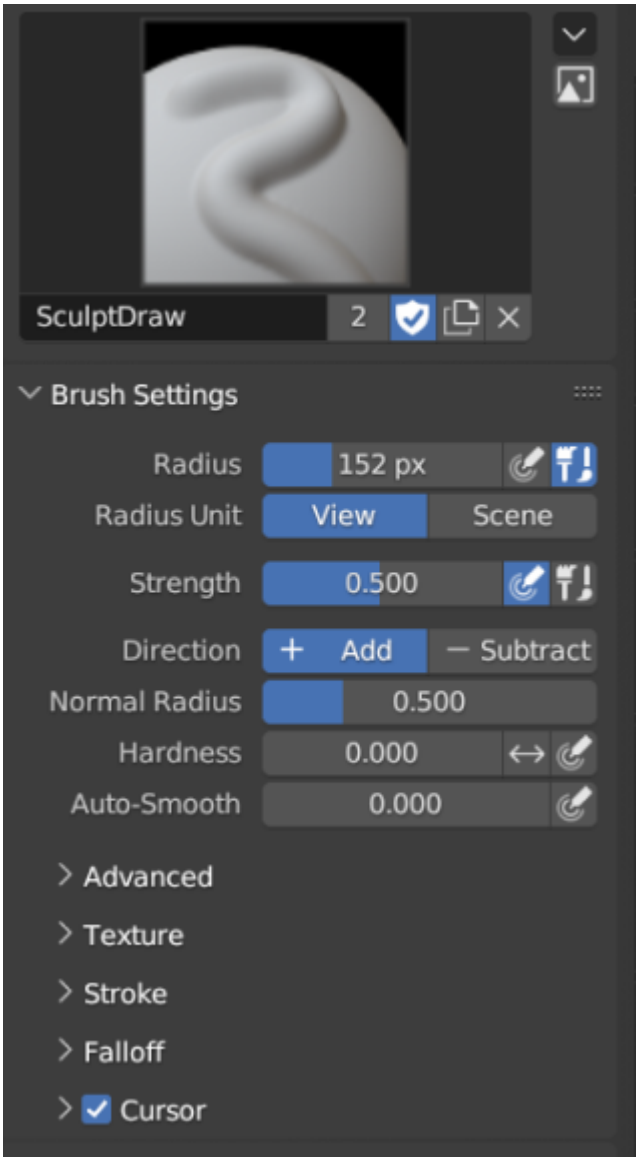
Brush
: General and advanced settings.

Texture
: Color and mask texture settings.

Stroke
: Stroke methods and settings.

Falloff
: Falloff curves and settings.

Cursor
: Cursor and appearance settings.



Brush settings panel.

Dyntopo

Reference

Mode:: Sculpt Mode
Panel:: *Sidebar ▸ Tool ▸ Dyntopo*

Dynamic Topology (aka Dyntopo) can be toggled with the checkbox. With dynamic topology active, most brushes will subdivide the mesh during the stroke.

For a general explanation of dynamic topology, visit the Introduction.

Detail Size/Percentage, Resolution R
: Each Detail Type's detail is set here. Depending on the Detail Type being used this property will rather show as a pixel count (px), or percentage.

  R allows you to interactively set the detail, with a preview of the detail's density in the 3D Viewport.

  Sample Detail Size (pipette icon)
  : When using *Constant Detail*, it is possible to sample the detail value of a certain mesh area by clicking the pipette icon next to the detail setting and then clicking on the area.

Refine Method
: Setting the option will determine which of the methods will be used when altering the topology.

  Subdivide Edges:: Just like the Subdivide tool, this method will only subdivide topology to match the detail given.
  Collapse Edges:: When topology is too dense, and is smaller than the detail given, edges will be collapsed to fit the detail size appropriately.

Subdivide Collapse::
: This method combines the two methods, subdividing edges smaller than the detail size, and collapsing topology.

Detailing
: Dyntopo uses the following different detail methods to create dynamic detail on an object.

Relative Detail:: This method uses a detail size based on the number of pixels, and in turn will create topology in that size. Zoom out big details, zoom in small fine details.

Constant Detail:: To keep detail uniform across the entire object, Constant Detail can be used. The detail is based on the percentage of a single unit.

Brush Detail:: Giving more control over the topology, with this method you can create topology based on the brush size. You can increase and lower topology by resizing the brush itself. The detail size is based the size of the brush itself, where full detail will create topology the size of the brush radius itself.

Manual Detail:: Similar to constant detail, this value sets a percentage value uniform across the object, but only applies detailing changes when using Flood Fill.

Detail Flood Fill `Ctrl-R`
: When using *Constant* or *Manual Detailing*, this option is made available, allowing you to fill the entire object with a uniform detail, based on the detail size.

Remesh

Reference

Mode::	Sculpt Mode
Header::	*Tool Settings* ▸ *Remesh*
Panel::	*Sidebar* ▸ *Tool* ▸ *Remesh*
Shortcut::	`Ctrl-R`

For a general explanation to remeshing, visit the Introduction.

Voxel Size `R`
: The resolution or the amount of detail the remeshed mesh will have. The value is used to define the size, in object space, of the Voxel. These voxels are assembled around the mesh and are used to determine the new geometry. For example a value of 0.5m will create topological patches that are about 0.5m (assuming *Preserve Volume* is enabled). Lower values preserve finer details but will result in a mesh with a much more dense topology.

  The voxel size also be adjusted from the 3D Viewport using `R`. Using the shortcut displays an interactive grid overlay showing the resulting voxel size. Moving the mouse closer to center of the grid decreases the voxel size while moving away from the center increase the voxel size. Holding `Shift` increases the precision; adjusting the voxel size in small increments.

  Sample Voxel Size
  : Used to adjust the *Voxel Size* by picking an area of the mesh to match the denseness of polygons after the remesh operation.

Adaptivity
: Reduces the final face count by simplifying geometry where detail is not needed. This introduce triangulation to faces that do not need as much

detail. Note, an *Adaptivity* value greater than zero disables *Fix Poles*.

Fix Poles
: Tries to produce less poles at the cost of some performance to produce a better topological flow.

Preserve
: Volume
: Tells the algorithm to try to preserve the original volume of the mesh. Enabling this could make the operator slower depending on the complexity of the mesh.

: Paint Mask
: Reprojects the paint mask onto the new mesh.

: Face Sets
: Reprojects Face Sets onto the new mesh.

: Color Attributes
: Reprojects the Color Attributes onto the new mesh.

Voxel Remesh
: Performs the remeshing operation to create a new manifold mesh based on the volume of the current mesh. Performing this will lose all mesh object data layers associated with the original mesh.

See also

Remesh modifier

Known Limitations

- Remeshing only works on the original mesh data and ignores generated geometry from modifiers, shape keys, rigging, etc.
- Remeshing will not work with the Multiresolution Modifier.

Symmetry

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar ▸ Tool ▸ Symmetry*

Mirror
: Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode and not in Object Mode.

Lock
: These three buttons allow you to block any modification/deformation of your model along selected local axes, while you are sculpting it.

Tiling
: Using this option allows you to seamlessly tile your strokes along the given axes. This allows to create repeating patterns.

Feather
: Reduces the strength of the stroke where it overlaps the planes of symmetry.

Radial X, Y, Z
: These settings allow for radial symmetry in the desired axes. The number determines how many times the stroke will be repeated within 360 degrees around the central axes.

Tile Offset X, Y, Z
: The offset allows the option to alter the tile size along all three axes. The default tile size is set to one unit.

Symmetrize

Direction

Determines which direction the model will be symmetrized.

Merge Distance

A parameter of the *Symmetrize* operator to control the distance within which symmetrical vertices are merged.

Symmetrize

Uses direction orientation to symmetrize. Since Dyntopo adds details dynamically it may happen that the model becomes asymmetric, so this a good tool for that.

Options

Reference

Mode::	Sculpt Mode
Tool::	*Toolbar* ‣ *Options*

Display
: Fast Navigate
  : For multiresolution models, shows low resolution while navigating in the viewport.

: Delay Viewport Updated
  : Update the geometry when it enters view. This provides for faster navigation.

: Use Deform Only
  : Limits the activated modifiers on the active object to Deform Modifiers, and Multiresolution. Constructive modifiers (like Subdivision Surface, Mirror and other) get deactivated, because they could give inaccurate results.

See also

See the Display options.

Gravity

Factor
: Setting the factor allows you to add gravity to your brush strokes, giving it a draping effect.

Orientation

Using another object, the gravity can be oriented to the set object's local Z axis, changing the direction of the gravity.

Controls

Auto-Masking

Reference

Mode::	Sculpt Mode
Tool::	*Header ▸ Auto-Masking*
Shortcut::	`Alt-A`

These properties automatically mask geometry based on geometric features of the mesh. It's an quick alternative to frequent manual masking. These masks are initialized on every new stroke or tool usage. They are also never visible as an overlay.

Note, these properties are applied across all sculpt brushes, however, they can also be configured per brush in the Advanced Brush Settings.

These properties can be accessed via a Pie Menus by pressing `Alt-A`.

All auto-masking modes can be combined, which makes the generated auto-mask more specific. For example it's possible to auto-mask a specific face set, while excluding disconnected topology and face set boundaries, and only affect faces that are oriented towards the view via View Normal.

Topology
: Only vertices that are topologically connected to where you started the stroke/tool on are affected. So loose geometry islands will be auto-masked.

  Additionally for the Grab and Thumb brushes, anything that is not connected within the brush radius will be auto-masked. So even if geometry is connected somewhere, it is considered separate if the connection is not within the radius.

Face Sets

Only vertices that are part of the same face set that you started the stroke/tool on are affected.

Tip

If no topology or face set is visible under the cursor at the start of the stroke, the previously auto-masked area will be targeted. This is especially useful with the “Projected” falloff shape in the Falloff Settings.

Mesh Boundary

Vertices that are part of open boundary edges are not affected. This also includes boundary edges to hidden faces.

Propagation Steps

Increases the soft gradient towards the auto-masked boundary edges. Each step iterates the distance one edge further. This setting is used for both Mesh Boundary and Face Sets Boundary.

Create Mask

This will execute the Mask From Mesh Boundary operator with the current auto-masking settings. This is very useful to visualize the current auto-mask, or to edit the mask further manually.

Face Sets Boundary

Vertices that are part of a boundary between face sets are not affected. This also includes boundary edges to hidden faces. Propagation Steps are shared with Mesh Boundary auto-masking.

Create Mask

This will execute the Mask From Face Sets Boundary operator with the current auto-masking settings. This is very useful to visualize the current auto-mask, or to edit the mask further manually.

Cavity

Vertices that are the peaks of the surface curvature are not affected. While this auto-mask is primarily meant for painting, it can also be used for regular sculpting.

Factor
: The overall contrast of how strong the cavity is applied. The value of 0.5 is the default, but better results can also be achieved on 0.2 if a Custom Curve is used as well.

Blur
: The number of times the cavity mask is blurred. A value of 0 will give the pure cavity auto-mask. Anything higher than 6 will likely have a less visible effect and decrease performance. Even though the value is capped to 10, it can be increased up to 25 if typing in the value.

Custom Curve
: Use a custom curve to fine tweak the cavity auto-mask. This is very useful if only small crevices or flat surfaces should be affected. Or for example if the contrast should be increased/decreased in a specific way.

Create Mask
: This will execute the Mask From Cavity operator with the current auto-masking settings. This is very useful to visualize the current auto-mask, or to edit the mask further manually.

Cavity (Inverted)
: This is the same as "Cavity", but inverted. This means the valleys/crevices of the surface curvature will not be affected. It cannot be used at the same time as Cavity and shares all of its settings. Enable this to quickly invert the cavity auto-mask.

View Normal
: Only vertices with a Normal that face the viewer are affected. This is similar to the "Front Faces Only" toggle in the Brush Setting, to only affect visible geometry. The advantage of this auto-mask is that it has more options and works on sculpt mode as a whole.

Occlusion

Change the View Normal behavior to only affect vertices that are not occluded by other faces. This setting is incompatible with the other Limit and Falloff sliders. It also causes a much slower performance.

Limit

Determines the range of angles that will be affected. 90 degrees encompasses all that is visible.

Falloff

Extends the angular range of the Limit slider with a soft falloff gradient. This falloff will visually extend the limit range further.

Area Normal

Very similar to the View Normal, but uses the Normal of the surface that you started the stroke/tool on. This way any direction can be chosen for what vertices will be affected. It has the same Limit and Falloff sliders as the View Normal auto-mask.

Editing

- Sculpt
  - Transform
  - Show & Hide
  - Fairing
  - Trimming
  - Mesh Filters
  - Sample Color
  - Set Pivot
  - Rebuild BVH
  - Dynamic Topology Toggle
  - Transfer Sculpt Mode
- Mask
  - Invert Mask
  - Fill Mask
  - Clear Mask
  - Box Mask
  - Lasso Mask
  - Mask Filters
  - Expand Mask
  - Mask Extract
  - Mask Slice
  - Mask From Cavity
  - Mask From Mesh Boundary
  - Mask From Face Sets Boundary
  - Random Mask
  - Display Settings
  - Clear Sculpt-Mask Data
- Face Sets
  - Face Set from Masked
  - Face Set from Visible
  - Face Set from Edit Mode Selection
  - Initialize Face Sets
  - Grow/Shrink Face Sets

    - Expand Face Set
    - Extract Face Set
    - Randomize Colors
    - Display Settings
- Expand
    - Expand Mask by Topology
    - Expand Mask by Normals
    - Expand Face Set by Topology
    - Expand Active Face Set

Sculpt

This page details the general hotkey operators and menu operators in sculpt mode.

Transform

Move
: Change the position of the object.

Rotate
: Change the orientation of the mesh.

Scale
: Increase/decrease the size of the mesh.

Sphere
: Morph the mesh to a spherical shape.

See also

Transform Tools.

Show & Hide

Reference

Mode::	Sculpt Mode
Menu::	*Sculpt*

Some very common hotkey operators to control the visibility based on face sets. These are not part of any menu and have to be used via the shortcuts.

More visibility operators can be found in the Face Sets Menu and the Pie Menu shortcut `W`. (Since visibility is often toggled via face sets.)

Box Hide
: Draw a box to hide faces of a mesh.

Box Show
: Draw a box to reveal hidden faces. This works similar to the Box Select tool.

Toggle Visibility `Shift-H`
: Hide all face sets except the active one (under the cursor). If face sets are already hidden, then this operator will show everything.

Hide Active Face Set `H`
: Hide the face set under the cursor. Press `Shift-H` afterwards to show everything.

Show All `W`, `Alt-H`
: Reveal all hidden geometry.

Invert Visible
: Hides all visible geometry and makes all hidden geometry visible.

Hide Masked
: Hides all masked vertices.

Grow/Shrink Visibility `PageUp`, `PageDown`
: Grows or shrinks the visible area of the mesh along its surface.

See also

For a more general introduction see Visibility, Masking & Face Sets.

Fairing

These operators smooths geometry patches based of a Face set.

See also

Edit Face Set Tool

Fair Positions
: Creates a perfectly flat and smooth geometry patch from the face set. This is the ideal way to trim parts of your mesh if the vertex count is too high for other operations, or the vertex IDs must not be altered (Like when using Multires sculpting).

Fair Tangency
: Creates a smooth as possible geometry patch from the face set by minimizing changes in vertex tangents. This is ideal for creating smooth curved surfaces on complex topology, where just using the smooth brush will not lead to desired results

Trimming

The trimming operators add or remove geometry from the mesh based on a gesture input. These operators are especially useful for sketching an early base mesh for further sculpting with the voxel remesher.

Line Project
: Flattens the geometry along a plane determined by the camera view and a drawn line. The region of the mesh being flattened is visualized by the side of the line that is shaded.

Box Trim
: Removes geometry based on a box selection.

Lasso Trim
: Removes geometry based on a lasso selection.

Box Add

Adds geometry based on a box selection.

Lasso Add
: Adds geometry based on a lasso selection.

Mesh Filters

Applies a deformation to all vertices in the mesh at the same time. Masking, auto-masking and visibility will be taken into account.

To use these operators, click and drag away from left to right or from right to left for a negative effect.

See also

Mesh Filter Tool

Smooth
: Smooths the positions of the vertices to either polish surfaces or remove volume from larger shapes. Especially useful to fix most of the artifacts of the voxel remesher. This filter works similar to the Smooth brush.

Surface Smooth
: Eliminates irregularities of the mesh by making the positions of the vertices more uniform while preserving the volume of the object. This filter works similar to the *Surface* deformation type of the Smooth brush.

Inflate
: Displaces vertices uniformly along their normal. This filter works similar to the Inflate brush.

Relax Topology
: Tries to create an even distribution of quads without deforming the volume of the mesh. This filter works the same as holding `Shift` with the Slide Relax brush.

Relax Face Sets
: This will remove the jagged lines visible after drawing or creating a face set. This filter works the same as holding `Shift` with the Draw Face Set brush.

Sharpen
: Sharpens and smooths the mesh based on its curvature, resulting in pinching hard edges and polishing flat surfaces. Especially useful when sculpting hard surfaces and stylized models with creasing and flattening brushes.

Enhance Details
: Increases the high frequency surface details of the mesh by intensifying the difference between creases and valleys. This filter works similar to the inverted direction of the Smooth brush.

Erase Multires Displacement
: Deletes displacement information of the Multires Modifier, resetting the mesh to a regular subdivision surface result. This can be used to reset parts of the sculpt or to fix reprojection artifacts after applying a Shrinkwrap Modifier.

  Negative strokes will intensify the displacement details, this method works similar to *Enhance Details* and can give better results in some circumstances.

Randomize
: Randomly moves vertices along the vertex normal. This filter works similar to the Randomize Transform.

Sample Color

Reference

Mode:: Sculpt Mode
Menu:: *Sculpt ▸ Sample Color*

Shortcut:: `Shift-X`

Adjust the brush color of the Paint tool to the color under the mouse cursor.

Set Pivot

Reference

Mode:: Sculpt Mode
Menu:: *Sculpt ▸ Set Pivot*

Like Object and Edit Mode, Sculpt Mode also has a Pivot Point. This is because the basic move, rotate and scale transforms are also supported in Sculpt Mode. But the pivot point in Sculpt Mode is unique. It always moves together with the transformed mesh and can be both manually & automatically placed.

Origin
: Sets the pivot to the origin of the sculpt.

Unmasked
: Sets the pivot position to the average position of the unmasked vertices.

Mask Border
: Sets the pivot position to the center of the mask's border. This operation will automatically happen when using Expand.

Active Vertex
: Sets the pivot position to the active vertex position.

Surface `Shift-RMB`
: Sets the pivot position to the surface under the cursor.

Tip

For more convenient placement of the pivot point it's recommended to use the shortcut assigned to *Surface*.

See also

For a more general introduction see Transforming.

Rebuild BVH

Reference

Mode:: Sculpt Mode
Menu:: *Sculpt ▸ Rebuild BVH*

Recalculates the BVH used by Dyntopo to improve performance, which might degrade over time while using Dyntopo.

See also

For a more general introduction see Adaptive Resolution.

Dynamic Topology Toggle

Toggles Dyntopo.

Transfer Sculpt Mode

Reference

Mode:: Sculpt Mode
Menu:: *Sculpt ▸ Transfer Sculpt Mode*
Shortcut::

`Alt-Q`

Switches Sculpt Mode from the Active object to the object under the mouse. See Switching Objects for more information.

See also

For a more general introduction see Working with Multiple Objects.

Mask

This page details the mask related shortcut operators and menu operators in sculpt mode. Other related information to masks can also be found at the bottom of the page.

Reference

Mode::	Sculpt Mode
Menu::	*Mask*
Shortcut::	`A`

Masks can be edited across all visible faces. Using `A` opens a pie menu to choose the most common operations.

Invert Mask

Reference

Mode::	Sculpt Mode
Menu::	*Mask* ▸ *Invert Mask*
Shortcut::	`A`, `Ctrl-I`

Inverts the visible mask. This is often useful when the masked vertices are the surfaces you want to sculpt/paint. In that case create a mask and then invert it.

Fill Mask

Reference

Mode::	Sculpt Mode
Menu::	

Mask ▸ Fill Mask

Fully masks the visible geometry. Alternatively it is common to clear and then invert a mask via A to achieve the same effect.

Clear Mask

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Clear Mask*
Shortcut::	A, Alt–M

Removes the mask on all visible vertices. To completely remove the mask data, see Clear Sculpt-Mask Data.

Box Mask

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Box Mask*
Shortcut::	B

Works like the Box Mask tool, it creates a rectangular mask region. Hold Shift or press MMB to clear the mask of the selected region.

Lasso Mask

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Lasso Mask*
Shortcut::	Ctrl–RMB

Can be used to create a free-form mask, similar to the Lasso Mask tool. This is very commonly used.

Tip

To clear the mask of areas with the *Lasso Mask*, first invert the mask, use *Lasso Mask*, and then invert the mask back.

Mask Filters

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Smooth/Sharpen Mask, Grow/Shrink Mask, Increase/Decrease Contrast*
Shortcut::	A

Similarly to other Filter Tools, mask filters are operations that are applied to the whole mask.

Type
: Smooth/Sharpen Mask
  : Changes the sharpness of the mask edge. Using this can be faster and more consistent than smoothing the mask with the Mask brush.

: Grow/Shrink Mask
  : Further grow or shrink the mask along the surface of the mesh.

: Increase/Decrease Contrast
  : Changes the contrast of the mask.

In the Adjust Last Operation panel there are further options to add iterations for a stronger effect.

Iterations
: The number of times the filter is applied.

Auto Iteration Count
: Use an automatic number of iterations based on the number of vertices of the sculpt. Disable this option to set the Iterations manually.

Tip

An alternative to Iterations is to use Repeat Last via the shortcut `Shift-R`.

Expand Mask

Note

More info on Mask Expand along Topology at the Expand page.

Mask Extract

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Mask Extract*

Creates a duplicate mesh object based on masked geometry. The extracted geometry is also further processed by default for a cleaner result.

Threshold
: Minimum mask value to consider the vertex valid to extract a face from the original mesh.

Add Boundary Loop
: Creates and extra boundary loop on the edges of the geometry, making it easier smooth the boundaries and apply additional modifiers.

Smooth Iterations
: Smooth iterations applied to the extracted mesh.

Project to Sculpt
: Project the extracted mesh on to the original sculpt object.

Extract as Solid
: Adds a Solidify Modifier to the newly created mesh object.

Mask Slice

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Mask Slice*

Removes the masked vertices from the mesh.

Threshold
: Minimum mask value to consider the vertex valid to extract a face from the original mesh.

Fill Holes
: Fills concave holes with geometry that might have resulted from the *Mask Slice* operation.

> **Tip**
>
> If nothing is masked, this operation can be used to just fill all holes. Especially when using Trim Tools tools and the Voxel Remesher

Slice to New Object
: Create a new object from the masked geometry.

Mask From Cavity

Reference

Mode:: Sculpt Mode
Menu:: *Mask ▸ Mask from Cavity*

Generates a mask based on the cavity of the surface. The settings of the operation can be changed in the Adjust Last Operation panel.

Mode
: Choose how the newly created mask is mixed with the existing one. By default it will replace the old mask via “Mix”.

Mix Factor
: The factor of the mix effect. Choose how strong the new mask is applied on the existing one.

Automask Settings
: The same settings as the Auto-Masking settings are applied.

Factor
: Same as Auto-Masking.

Blur
: Same as Auto-Masking.

Invert
: Same as Auto-Masking.

Custom Curve
: Same as Auto-Masking.

Mask From Mesh Boundary

Reference

Mode:: Sculpt Mode
Menu:: *Mask ▸ Mask from Mesh Boundary*

Generates a mask based on the topological islands of the mesh. The settings of the operation can be changed in the Adjust Last Operation panel.

Mode
: Choose how the newly created mask is mixed with the existing one. By default it will replace the old mask via “Mix”.

Mix Factor
: The factor of the mix effect. Choose how strong the new mask is applied on the existing one.

Automask Settings
: The same settings as the Auto-Masking settings are applied.

Propagation Steps
: Same as Auto-Masking.

Mask From Face Sets Boundary

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Mask from Face Sets Boundary*

Generates a mask based on the face set islands of the mesh. The settings of the operation can be changed in the Adjust Last Operation panel.

Mode
: Choose how the newly created mask is mixed with the existing one. By default it will replace the old mask via “Mix”.

Mix Factor
: The factor of the mix effect. Choose how strong the new mask is applied on the existing one.

Automask Settings
: The same settings as the Auto-Masking settings are applied.

Propagation Steps
: Same as Auto-Masking.

Random Mask

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Random Mask*

Generates a mask with random values for the entire object based on different mesh data.

Per Vertex
: Assigns a random mask value for each vertex.

Per Face Set
: Assigns a random mask value for each Face Set.

Per Loose Mask
: Assigns a random mask value for each disjoint part of the mesh.

Display Settings

Reference

Mode::	Sculpt Mode
Popover::	*Viewport Overlays – Sculpt ▸ Mask*

The mask display can be toggled as a viewport overlay. In the overlay popover, the opacity of the mask overlay can be adjusted to make it more or less visible on the mesh.

Clear Sculpt-Mask Data

Reference

Mode::	Object/Edit Mode
Menu::	*Properties* ▸ *Object Data* ▸ *Geometry Data* ▸ *Clear Sculpt-Mask Data*

Completely frees the mask data layer from the mesh. While not a huge benefit, this can speed-up sculpting if the mask is no longer being used.

Face Sets

This page details the face set related hotkey operators and menu operators in sculpt mode.

Tip

There is a face set pie menu that can be accessed with `Alt-W`.

Face Set from Masked

Reference

Mode:: Sculpt Mode
Menu:: *Face Sets ▸ Face Set from Masked*

Creates a new face set from Masked Geometry.

Face Set from Visible

Reference

Mode:: Sculpt Mode
Menu:: *Face Sets ▸ Face Set from Visible*

Creates a new face set from all visible geometry.

Face Set from Edit Mode Selection

Reference

Mode:: Sculpt Mode
Menu:: *Face Sets ▸ Face Set from Edit Mode Selection*

Creates a new face set corresponding to the Edit Mode face selection.

Initialize Face Sets

Reference

Mode:: Sculpt Mode
Menu:: *Face Sets ▸ Initialize Face Sets*

Initializes all face sets on the mesh at once based off one of several mesh attribute properties.

Mode
: The mesh data attribute used to define the boundaries for the face sets.

  By Loose Parts:: Creates a new face set per discontinuous part of the mesh.
  By Face Set Boundaries:: Creates a face set for each isolated face set. This mode is useful for splitting the patterns created by Face Set Expand into individual Face Sets for further editing.
  By Materials:: Creates a face set per Material Slot.
  By Normals:: Creates face sets for Faces that have similar Normals.
  By UV Seams:: Creates face sets using UV Seams as boundaries.
  By Edge Creases:: Creates face sets using Edge Creases as boundaries.
  By Edge Bevel Weight:: Creates face sets using Bevel Weights as boundaries.
  By Sharp Edges:: Creates face sets using Sharp Edges as boundaries.

Threshold
: The minimum value to consider a certain attribute a boundary when creating the face sets.

Grow/Shrink Face Sets

Reference

Mode::	Sculpt Mode
Menu::	*Face Sets* ‣ *Grow/Shrink Face Sets*
Tool::	Edit Face Set
Shortcut::	`Ctrl-W`, `Ctrl-Alt-W`

Expands or contracts the face set under the cursor by adding or removing surrounding faces.

Expand Face Set

Note

More info on Face Set Expand at the Expand page.

Extract Face Set

Reference

Mode::	Sculpt Mode
Menu::	*Face Sets* ‣ *Extract Face Set*

Creates a new mesh based on the selected face set. Once the operator is initiated, hover over the face set and `LMB` to create the new mesh. After the operator is finished the new mesh will be selected in Object Mode.

Randomize Colors

Reference

Mode::	Sculpt Mode
Menu::	*Face Sets* ‣ *Randomize Colors*

Generates a new set of random colors to render the face sets in the 3D Viewport.

Display Settings

Reference

Mode::	Sculpt Mode
Popover::	*Viewport Overlays – Sculpt* ‣ *Face Sets*

The face sets display can be toggled as a viewport overlay. In the overlay popover, the opacity of the face sets overlay can be adjusted to make it more or less visible on the mesh.

Expand

This is a multi-purpose modal operator to intuitively create and edit masks/face sets. When executed, it uniformly expand outwards a pattern from the vertex under the cursor.

Note

These operators are meant to be used interactively through the shortcut.

There is also a full showcase of the Expand features and use cases.



A preview of *Expand Mask by Topology*

Expand Mask by Topology

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Expand Mask by Topology*
Shortcut::	`Shift-A`

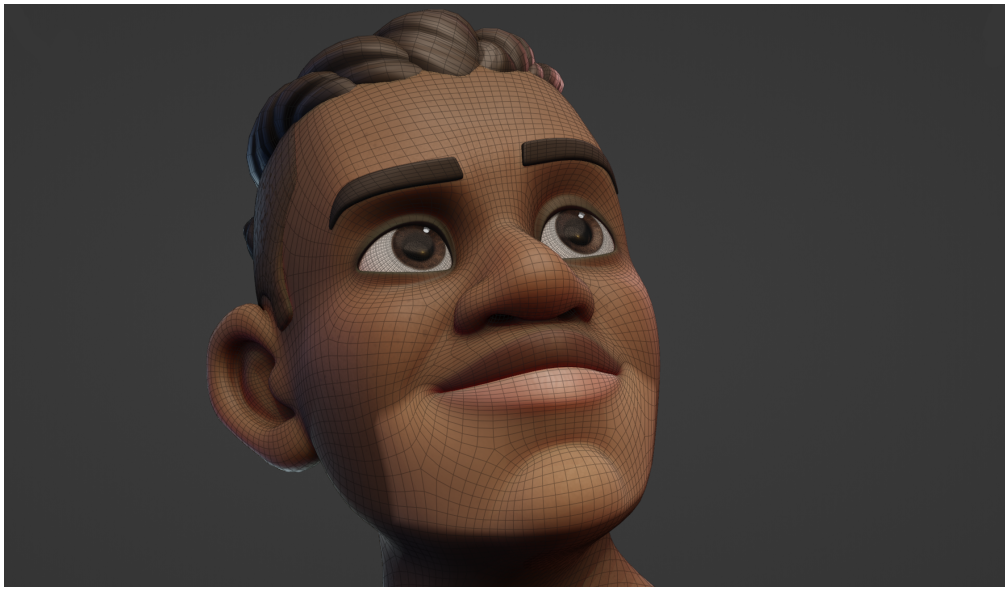
Expands a mask from the active vertex.

Usage

Mask from A to B

Start the operator and expand a mask from an origin to your mouse cursor distance. Then confirm with `LMB` or `Return`

By default the expansion will use a *Geodesic* falloff `1` to create perfectly accurate distances along the surfaces. Use other falloff types via `2`, `3` & `4` to expand via triangles, whole faces or scene distances instead.



The typical result when using the *Diagonals* falloff to expand along the quads of the face.

Start Expand while pointing at an open boundary to expand from the entire boundary loop. This will always use the *Topology* falloff mode.

Fill Connected Meshes

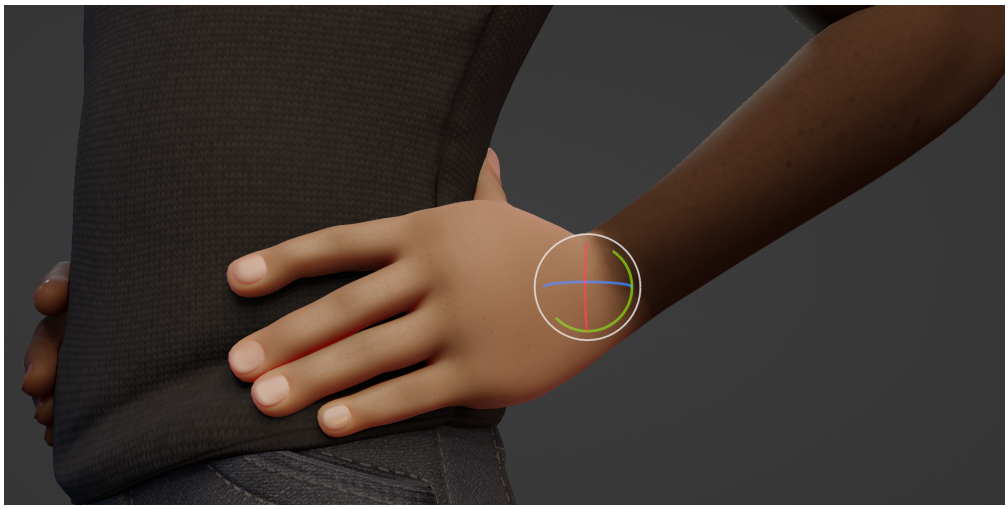
Move your cursor outside of the boundaries of the mesh to mask the entire connected mesh. This can be done repeatedly to quickly mask separate meshes.

Fill Face Sets

Hold `Ctrl` to snap to face sets under your cursor and fill them. Any face set that was already covered in the expansion will be filled as well.

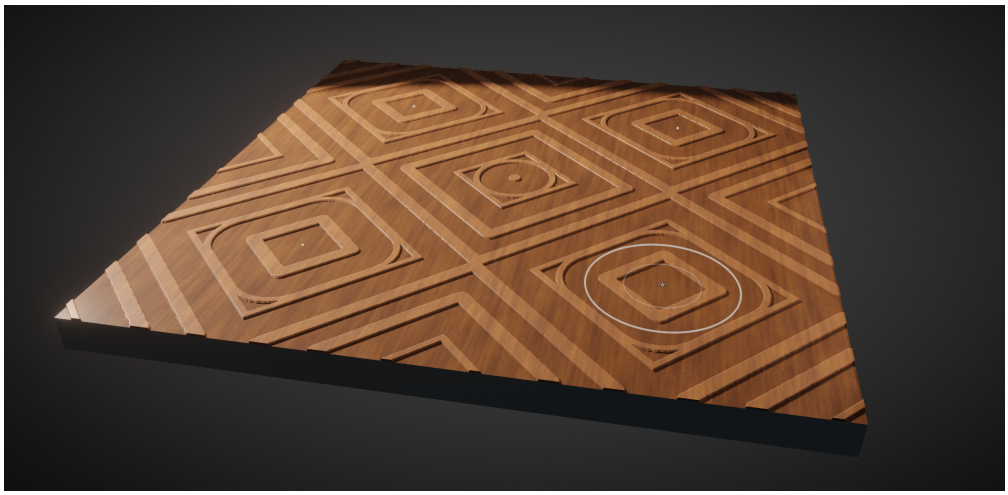
Automatically Set Transform Pivot

While using any Transform tool, the pivot point will automatically snap the border of an Expand result. This way areas (Like limbs) can be masked and then immediately rotated or otherwise transformed.



Pattern Creation

The different falloff types can be used for circular, triangular and square patterns. More loops can also be added/removed via `W` & `Q` to repeat the pattern across the mesh.



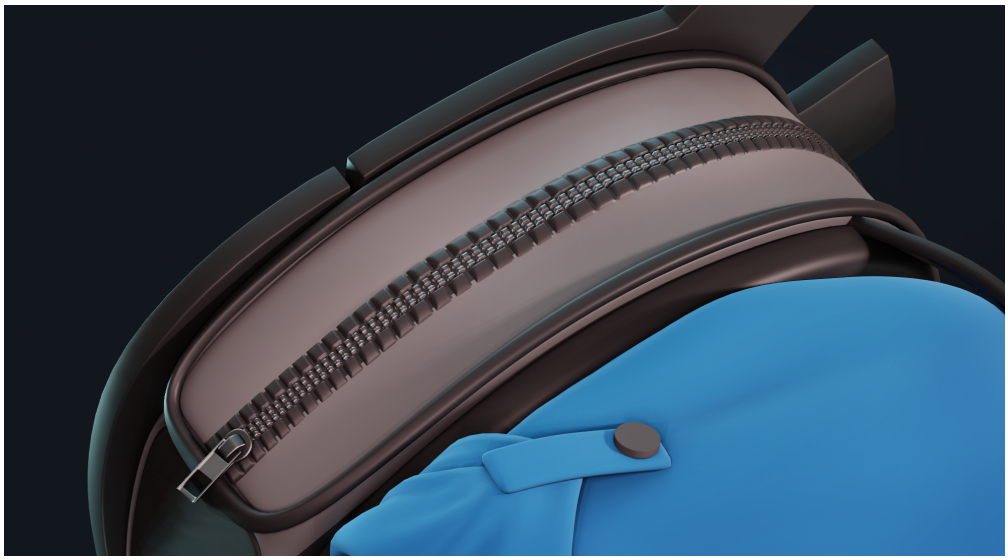
An example of using expand with mirror options, loops and a recursion to create wood carving patterns.

Tip

Mirror options can also be combined with the expansion.

Linear gradients `G` or brush falloff gradients `B` will help to add slanted surfaces to the patterns.

A “Recursions” with `R` or `Alt-R` will start a new expansion along the border of the current expansion. Doing this multiple times, can help for increasingly random patterns or advanced pattern creation.



An example of using loops and gradients with multiple expanded masks.

Tip

Remember that Expand only affects visible geometry. So if a pattern should only be created on a part of the mesh, hide the other geometry first.

Use the Mesh Filter to deform the geometry and the Color Filter to add colors, to apply the patterns on the sculpt.

Expanding Textures

Textures can be used to affect the shape and gradients of the expanded mask. This feature can be combined with loops and recursion to create unique results.

To use a texture, you need to assign it to your currently active brush in the Texture Brush Settings. The texture can be edited/created in the Texture Properties.

Note

This texture only works when the Mapping is set to *3D*.

Use `Y` and `T` to increase or decrease the affect the texture has on the edge of the mask.

Controls

Invert:: `F` Flips between expanding a positive mask (value of one) or a negative mask (value of zero). In the case of face sets, this option flips between including areas inside the masked area or areas outside the masked area. .. needs visual technical examples for both

Toggle Preserve State::

`E` Accumulate the new mask on top of the previous one or replace it. For Face Sets, this will toggle between creating Face Sets boundaries or replacing the existing Face Sets.

Move Origin:: `Spacebar` Moves the initial vertex used for calculating the falloff. .. needs visual technical example

Geodesic Falloff:: `1` Uses a falloff based on the Geodesic distances from the edge boundary to the active vertex.

Topology Falloff:: `2` Uses a falloff based on a flood fill using edges.

Diagonals Falloff:: `3` Uses a falloff based on a flood fill using polygon diagonals and edges.

Spherical Falloff:: `4` Uses a falloff based on the Euclidean distances from the edge boundary to the active vertex. .. needs visual technical examples

Toggle Gradient:: `G` Enables linear gradient of values from the origin to the current active vertex.

Toggle Brush Gradient::

`B` Similar to linear gradient but uses the current brush Falloff to define the shape of the falloff. .. needs visual technical examples

Geodesic Recursive Step::

`R` Start a new expansion with a Geodesic falloff from the boundary of the current falloff.

Topology Recursive Step::

`Alt-R` Start a new expansion with a topology falloff from the boundary of the current falloff. .. needs visual technical examples

Snap Expanded to Face Sets::

`Ctrl` Isolates the expanded region to the boundary of the face set under the cursor.

Loop Count Increase::

`W` Increase the number of loops or iterations the operator is run; using four loops will split the mask into four parts.

Loop Count Decrease::

`Q` Decrease the number of loops or iterations the operator is run; using four loops will split the mask into four parts. .. needs visual technical examples

Texture Distortion Increase::
`Y` Increases the falloff distance when using a texture to distort the mask shape.

Texture Distortion Decrease::
`T` Decreases the falloff distance when using a texture to distort the mask shape. ..needs visual technical examples

Expand Mask by Normals

Reference

Mode::	Sculpt Mode
Menu::	*Mask ▸ Expand Mask by Normals*
Shortcut::	`Shift-Alt-A`

Expand a mask, following the curvature of the surface. This operator uses the same internal operator as Expand meaning all the shortcuts and functionality works the same as that tool.

This operator is especially useful for hard surface sculpting.

Tip

If one expansion does not properly fill the entire desired surface, use the operator repeatedly with a different starting point.

Note

Using any of the Falloff shortcuts `1-4` will replace the curvature falloff of this operator.

Expand Face Set by Topology

Reference

Mode::	Sculpt Mode
Menu::	*Face Sets ▸ Expand Face Set by Topology*
Shortcut::	`Shift-W`

Expands a face set from the active vertex. This operator uses the same internal operator as Expand meaning all the hotkeys and functionality works the same as that tool, with the gradient features as the exception.

Usage

Saving Masks
: Expanding Face Sets has all the same use cases as expanding masks. The advantage for this one is that they will be saved for repeated usage. Face sets can be filled any time with a mask, so assigning areas face sets will save you time. (And of course face sets can be used to hide face sets)

Pivot Points for Pose Brush
: When using the Pose Brush it is most predictable when using it with Face Sets to define the face set boundaries as pivot point locations. Face Sets can be expanded from a point or from a boundary between hidden face sets to create them quickly. Alternatively Grow/Shrink Face Sets or use the Expand Active Face Set to dynamically grow/shrink them.

Cloth Sculpting
: Tools like the Cloth Filter and Cloth Brush work especially well when only simulating small areas at a time. Face Sets can very easily be created with Expand to assign areas of action.

Expand Active Face Set

Reference

Mode::	Sculpt Mode
Menu::	*Face Sets ▸ Expand Face Set by Topology*
Shortcut::	`Shift-Alt-W`

Expands an existing face set with a geodesic falloff. This operator uses the same internal operator as Expand meaning all the hotkeys and functionality works the same as that tool.

Note

Using any of the Falloff shortcuts `1-4` the operator to switch to Expand Face Set by Topology.

Usage

Resizing Face Sets
: Resize a Face Set along the surface distances. It is an alternative to Grow/Shrink Face Sets which follows the topology instead of geodesic distances.

Joining Face Sets
: When holding `Ctrl` the expansion will instead snap to other Face Sets. This is a fast way of joining multiple face sets into one.

Texture Paint

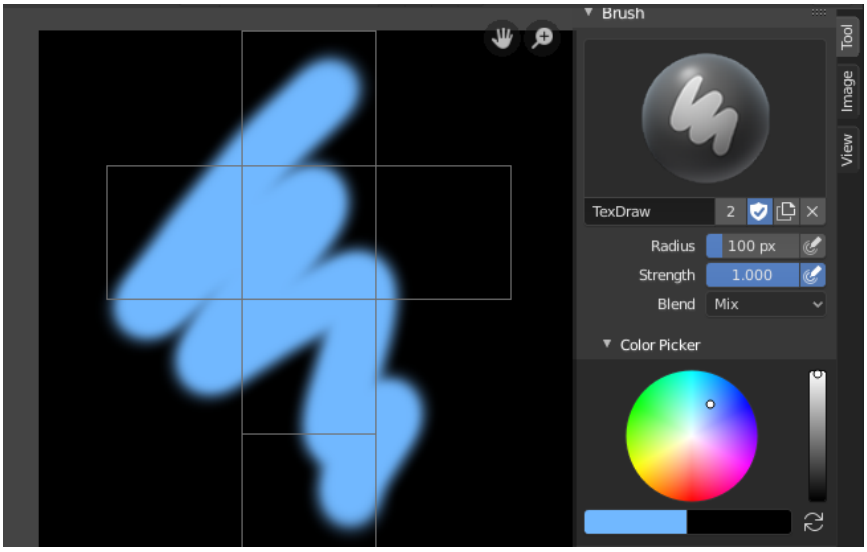
- Introduction
  - Getting Started
  - Texture Preview
  - Saving
  - Using an External Image Editor
  - Known Limitations
- Brushes
  - Brush Types
- Tool Settings
  - Texture Slots
  - Brushes
  - Brush Settings
  - Mask
  - Symmetry
  - Options
  - Tiling

Introduction

A UV texture is a picture (image, sequence or movie) that is used to color the surface of a mesh. The UV texture is mapped to the mesh through one or more UV maps. There are three ways to establish the image used by the UV texture:

1. Use any image editing program to create an image. In the Image Editor, select the UV texture and load the image. Blender will then use that texture's UV map to transfer the colors to the faces of the mesh.
2. Paint a flat image in the Image Editor onto the currently selected UV texture, using its UV map to transfer the colors to the faces of the mesh.
3. Paint the mesh in the 3D Viewport, and let Blender use the currently selected UV map to update the UV texture (as discussed below).

Blender features a built-in paint mode called *Texture Paint* which is designed specifically to help you edit your UV textures and images quickly and easily in either the Image Editor or the 3D Viewport. Since a UV texture is just a special-purpose image, you can also use any external paint program, like GIMP or Krita.



Since a mesh can have layers of UV textures, there may be many images that color the mesh. However, each UV texture only has one image.

Texture Paint works in both a 3D Viewport and the

Image Editor. In the 3D Viewport in Texture Paint Mode, you paint directly on the mesh by projecting onto the UVs.

Texture painting in Blender.

Tip

Memory Optimization

Texture Paint is fast and responsive when working in the 3D Viewport and when your image is sized as a square where the side lengths are a power of two, e.g. 256×256, 512×512, 1024×1024, etc.

Getting Started

The object to be painted on must first be unwrapped. UVs can be added traditionally, with standard Unwrapping Tools, or by adding Simple UVs in Texture Paint mode.

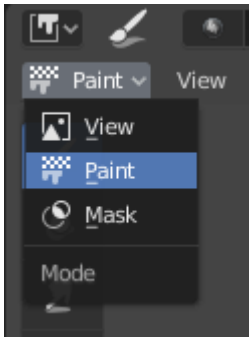
Note

When no UV layers can be detected, Blender will display a warning message.

Once you have unwrapped your model to a UV map, you can begin the texturing process. To use texture paint you may do any of the following:

- Activate the Texture Paint workspace. Here the 3D Viewport has the Texture Paint Mode enabled and the Image Editor is already switched to Paint mode.
- In the 3D Viewport, select Texture Paint Mode from the mode selector in the header, and you can paint directly onto the mesh.
- In the Image Editor, switch the mode to Paint (shown in the image to the right).

Once you enable Texture Painting, your mouse becomes a brush. As soon as you enable Texture Painting or switch to Texture Paint Mode, different tools become available in the Toolbar.



Enabling Paint mode.

In the Image Editor, you paint on a flat canvas that is wrapped around the mesh using UV coordinates. Any changes made in the Image Editor show up immediately in the 3D Viewport, and vice versa.

To work with the UV layout (for example, to move coordinates) you must use the UV Editor.

A full complement of brushes and colors can be selected from the Sidebar region in the Image Editor. Brush changes made in either panel are immediately reflected in the other panel. However, the modified texture will **not** be saved automatically; you must explicitly do so with Save Image.

Texture Preview

If your texture is already used to color, bump map, displace, alpha-transparent, etc., a surface of a model in your scene (in other technical words, is mapped to some aspect of a texture via a texture channel using UV as a map input), you can see the effects of your painting in the context of your scene as you paint.

To do this, set up side-by-side areas, one Area in 3D Viewport set to *Texture* shading option, and in the second Area the Image Editor loaded with your image. Position the 3D Viewport to show the object that is UV-mapped to the loaded image. In the image to the right, the texture being painted is mapped to the “Normal” attribute, and is called “bump mapping”, where the grayscale image is used to make the flat surface appear bumpy. See Texture Mapping Output for more information on bump mapping.

Saving

If the header menu item Image has an asterisk next to it means that the image has been changed, but not saved. Use Save Image or Save Image As to save your work with a different name or overwrite the original image.

Note

UV Textures

Since images used as UV textures are functionally different from other images, you should keep them in a directory separate from other images.

The image format for saving is independent of the format for rendering. The format for saving a UV image is selected in the header of the File Browser, and defaults to `PNG` (`.png`).

If Packing is enabled in the File Browser's header, or if you manually pack, saving your images to a separate file is not necessary.

Using an External Image Editor

If you use an external program to edit your UV texture, you must:

1. Run that paint program (GIMP, Krita, etc.).
2. Load the image or create a new one.
3. Change the image.
4. And re-save it within that program.
5. Back in Blender, you reload the image in the Image Editor.

You want to use an external program if you have teams of people using different programs that are developing the UV textures, or if you want to apply any special effects that Texture Paint does not feature, or if you are much more familiar with your favorite paint program.

Known Limitations

UV Overlap

In general overlapping UVs are not supported (as with texture baking).

However, this is only a problem when a single brush stroke paints onto multiple faces that share a texture.

Perspective View & Faces Behind the View

When painting onto a face which is partially behind the view (in perspective mode), the face cannot be painted on. To avoid this, zoom out or use an orthographic viewport.

Perspective View & Low Poly

When painting onto a face in perspective mode onto a low-poly object with normals pointing away from the view, painting may fail; to workaround disable the Normal Falloff option in the stroke settings.

Typically this happens when painting onto the side of a cube (see Blender bug #34665).

Brushes

Brush Types

See Brush Type.

Available brush types are listed here, together with brushes from the *Essentials* asset library using them.

Draw

Brushes: Paint Hard, Paint Soft, Paint Hard Pressure, Paint Soft Pressure, Erase Hard, Erase Soft, Erase Hard Pressure

The normal brush, draws a swath of color.

Soften

Brushes: Blur

Uses a “blur effect” to soften or sharpen the image.

Direction

Soften

Is used to paint a blur effect.

Kernel Radius (2D only)

Blur radius in pixels.

Sharpen

The Sharpen tool enhances the contrast of the image as you paint over it.

Sharp Threshold

The Threshold will only apply sharpening to only those pixels that differ more than the threshold value from their surrounding pixels.

Kernel Radius (2D only)

The kernel size controls how big an area the tool searches over is while calculating that difference.

Blur Mode

The blur kernel type controls how neighboring pixels are weighted when calculating the blur effect.

Gaussian::	Gaussian will sample the pixels near the center of the brush most.
Box::	Box samples all surrounding pixels equally.

Smear

Brushes: Smear

When you click, takes the colors under the cursor, and blends them in the direction you move the mouse. Similar to the "smudge" tool of *Gimp*.

Clone

Brushes: Clone

Copies the colors from the specified image (or location of the same image) to the active image.

In 3D projective painting the clone cursor can be set with `Ctrl-LMB`. In 2D painting the clone can be moved dragging it with `RMB`.

Clone from Paint Slot (3D projective only)

Use another image as clone source, instead of using the 3D cursor position as the source in the same image.

Source Clone Slot

This allows you to select an image as a clone source.

Image (2D only)

Image used as a clone source.

Alpha (2D only)

Opacity of the clone image display.

Fill

Brushes: Fill

It can be used to fill large areas of the image with the brush color. The tool fills adjacent pixels that have a color value similar to the pixel you clicked on.

Fill Threshold (2D only)

Determines how much the color must be similar to the color of pixel you click to be filled. A low *Threshold* only fills very similar in color pixels. A higher *Threshold* fills pixels within a broader range of color.

The *Gradient* type of the Color Picker allows the use of a gradient to fill the image.

To apply the gradient with the *Fill* brush click `LMB` and drag to define the gradient line, or radius if a radial gradient is used (depending on the *Gradient Fill Mode*).

Gradient Fill Mode

Linear, Radial

Note

Overrides

For projective texturing it will bypass some options for projective painting to paint the model. This means that occluded, backfacing and normal culled faces will always get filled, regardless of whether the options are activated in the External panel.

Mask

Brushes: Mask

This brush paints gray-scale values on the mask texture specified in the Mask panel. Any masked surfaces will not be affected by other paint brushes, similar to sculpt mode masking.

Mask Value
: Mask weight, a value of zero means not masked, while one is completely masked. Hold `Ctrl` to invert the painted mask value.

Tip

A simpler alternative is to use the face selection mask. See Face Selection Masking for details.

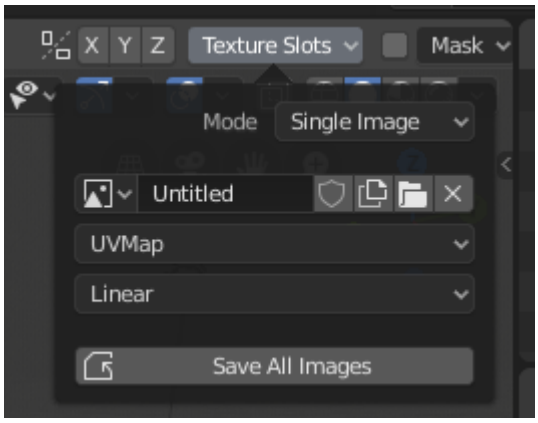
Tool Settings

- Texture Slots
- Brushes
  - Introduction
  - Manage Brushes
  - Brush Settings
- Brush Settings
- Mask
  - Stencil Mask
  - Cavity Mask
- Symmetry
- Options
  - External
- Tiling

Texture Slots

The combination of images associated with UV maps is called “slots”.

Selecting a *Paint Slots* or *Canvas Image* will also display the corresponding image in the Image Editor.



Texture Slots settings.

Mode
: The slot system includes two painting modes:

  Material
  : This mode tries to detect the slots from the materials of the mesh.

    For the Cycles renderer, all textures (*Image Texture* node) in the material’s node tree are added in the slots tab.

    Active Paint Texture Index
    : A List view of slots. Activate a certain slot to use it for painting by `LMB` click on it.

  Single Image
  : You can just select an existing image and painting will use the active UV layer for painting.

    Image
    : Allows you to select the image used as a canvas.

      New
      : Create a new image.

UV Map

Allows you to select the UV layer for painting. (Same as the currently active UV map in the mesh's *UV Maps* panel.)

Texture Filter Type

Set the interpolation mode of the texture. This can be Linear or Closest.

Save All Images

Repack (or save if external file) all edited images. Same as in the Image Editor.

Add Simple UVs

The *Add Simple UVs* does a simple cube unwrap followed by a pack operation. It's still recommended to make a custom unwrap.

This operator is available when the object does not already have a UV Map.

Brush Settings

Information on brush settings for every mode can be found in these pages:

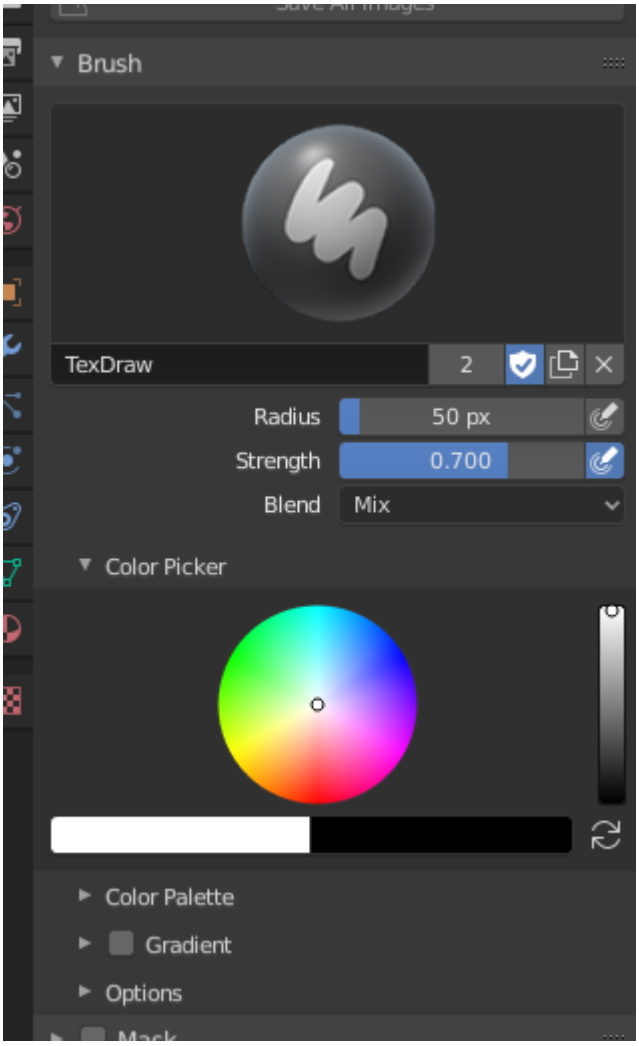
Brush
: General and advanced settings.

Texture
: Color and mask texture settings.

Stroke
: Stroke methods and settings.

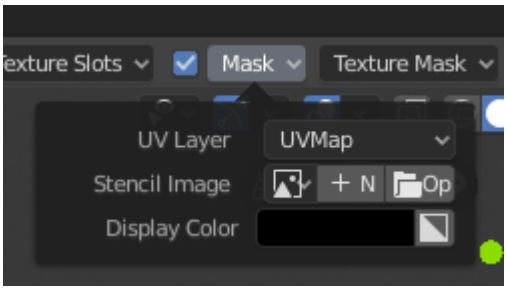
Falloff
: Falloff curves and settings.

Cursor
: Cursor and appearance settings.



Brush settings panel.

Mask



Mask settings.

Stencil Mask

Specify an additional image texture that defines masked surfaces. Masked surfaces can be defined with the Mask brush and will not be affected by painting.

The mask can be deactivated by the checkbox in the header.

Stencil Image
: Image used as a mask. See Data-Block Menu.

UV Layer
: Allows you to select the UV layer for the mask image.

Display Color
: Mask color in the viewport. See Color Picker.

: Invert Stencil (black/white icon)
: Inverts the mask.

Cavity Mask

Cavity masking means that the brush will be masked if there is a cavity or a hill on the mesh surface depending on the mesh options. The cavity algorithm is vertex-based.

Symmetry

Reference

Mode::	Texture Paint Mode
Tool::	*Toolbar ▸ Tool ▸ Symmetry*

Mirror
: Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode and not in Object Mode.

Options

Bleed
: Seam Bleed extends the paint beyond UV island bounds to avoid visual artifacts (like bleed for baking).

Dither
: Amount of dithering when painting on 8-bit images.

Occlude
: With Geometry occlusion active only exposed (not hidden by other mesh parts) pixels are affected. This also allows for 3D stencils to be used to mask out areas of the surface too.

Backface Culling
: With backface culling enabled you can only paint on the front side of faces.

See also

See the Brush Display options.

External

Screen Grab Size
: Size of the captured image for reprojecting.

Quick Edit
: Edit a snapshot of the viewport in an external image editor.

Apply
: Project edited image back onto the object.

Apply Camera Image

Project an edited render from the active camera back onto the object.

Tiling

Reference

Editor::	Image Editor
Mode::	Paint Mode
Menu::	*Sidebar ▸ Tools ▸ Tiling*

Wraps the stroke to the other side of the image as your brush moves off the opposite side of the canvas. Very handy for making seamless textures.

X
: left/right

Y
: top/bottom

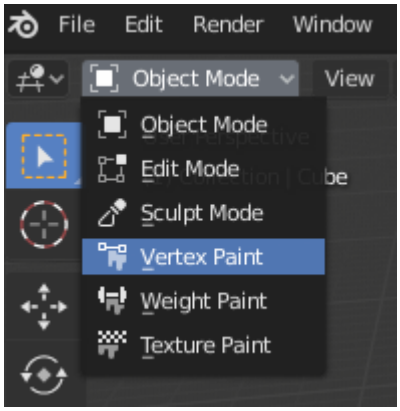
Vertex Paint

- Introduction
  - Viewing Color Attributes
- Brushes
  - Brush Types
- Vertex Paint Tools
- Tool Settings
  - Brushes
  - Brush Settings
  - Symmetry
- Editing
  - Smooth Vertex Colors
  - Dirty Vertex Colors
  - Vertex Color from Weight
  - Invert
  - Levels
  - Hue/Saturation/Value
  - Brightness/Contrast
  - Set Vertex Colors
  - Sample Color

Introduction

Vertex Painting is a simple way of painting color onto an object, by directly manipulating the color of vertices, rather than textures, and is fairly straightforward. Vertex Painting stores the color information as a Color Attribute which can be used by different render engines.

Color attribute's can be managed in the pallette pop-over in the middle of the header.



Vertex Painting Mode.

When a vertex is painted, the color of the vertex is modified according to the settings of the brush. The color of all visible planes and edges attached to the vertex are then modified with a gradient to the color of the other connected vertices. Note that the color of occluded faces is not modified.

See also

Dynamic Paint can create Color Attribute information while using physics or animation.

Viewing Color Attributes

Color Attributes can be used in a material node tree using the Color Attribute Node.

Color Attributes can be viewed in the 3D viewport using the Workbench render engine. To use such feature, set the 3D Viewport to Solid Shading and select the *Attribute* Color option.

Brushes

Brush Types

See Brush Type.

Available brush types are listed here, together with brushes from the *Essentials* asset library using them.

Paint Vertex
: Brushes: Paint Hard, Paint Soft, Paint Hard Pressure, Paint Soft Pressure, Airbrush

  Paints a specified color over the object.

Blur
: Brushes: Blur

  Smooths out the colors of adjacent vertices. In this mode the Color Value is ignored. The strength defines how much the colors are blurred.

Average
: Brushes: Average

  Smooths color by painting the average resulting color from all colors under the brush.

Smear
: Brushes: Smear

  Smudges colors by grabbing the colors under the brush and “dragging” them. This can be imagined as a finger painting tool.

Vertex Paint Tools

Brush
: Tool to use for any of the vertex paint brushes.

Annotate
: Draw free-hand annotation.

  Annotate Line
  : Draw straight line annotation.

  Annotate Polygon
  : Draw a polygon annotation.

  Annotate Eraser
  : Erase previous drawn annotations.

Tool Settings

- Brushes
  - Introduction
  - Manage Brushes
  - Brush Settings
- Brush Settings
- Symmetry

Brush Settings

Information on brush settings for every mode can be found in these pages:

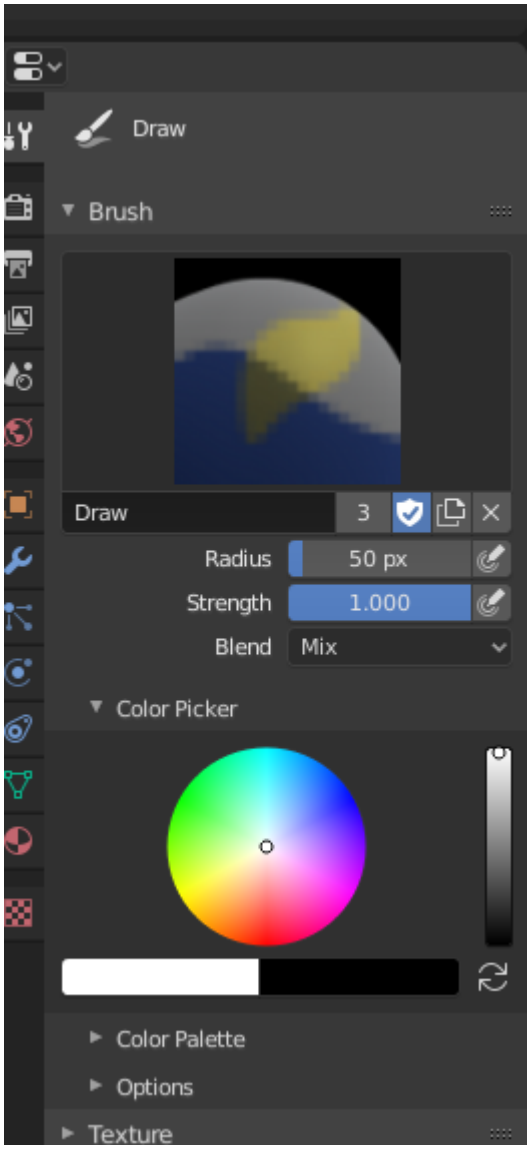
Brush
: General and advanced settings.

Texture
: Color and mask texture settings.

Stroke
: Stroke methods and settings.

Falloff
: Falloff curves and settings.

Cursor
: Cursor and appearance settings.



Brush settings panel.

Symmetry

Reference

Mode::	Vertex Paint Mode
Tool::	*Toolbar ▸ Tool ▸ Symmetry*

Mirror
: Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode and not in Object Mode.

Editing

Smooth Vertex Colors

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ‣ Smooth Vertex Colors*

Smooth colors across vertices.

Dirty Vertex Colors

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ‣ Dirty Vertex Colors*

Generate a dirt map gradient based on cavity.

Blur Strength
: Blur strength per iteration.

Blur Iterations
: Number of times to blur the colors (higher blurs more).

Highlight Angle
: Clamps the angle for convex areas of the mesh. Lower values increase the contrast but can result in clamping. 90 means flat, 180 means infinitely pointed.

Dirt Angle

Clamps the angle for concave areas of the mesh. Higher values increase the contrast but can result in clamping. 90 means flat, 0 means infinitely deep.

Dirt Only
: When active it won't calculate cleans for convex areas.

Normalize
: Choose optimal contrast by effectively lowering *Highlight Angle* and increasing *Dirt Angle* automatically. Disabling *Normalize* allows getting consistent results across multiple objects.

Vertex Color from Weight

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Vertex Color from Weight*

Converts the active weight into grayscale colors.

Invert

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Invert*

Invert RGB values.

Levels

Reference

Mode::	Vertex Paint Mode

Menu:: *Paint ▸ Levels*

Adjust the levels of the selected vertices.

Hue/Saturation/Value

Reference

Mode:: Vertex Paint Mode
Menu:: *Paint ▸ Hue/Saturation/Value*

Adjust the HSV values of the selected vertices.

Brightness/Contrast

Reference

Mode:: Vertex Paint Mode
Menu:: *Paint ▸ Brightness/Contrast*

Adjust the brightness/contrast of the selected vertices.

Set Vertex Colors

Reference

Mode:: Vertex Paint Mode
Menu:: *Paint ▸ Set Vertex Colors*
Shortcut:: `Ctrl-X`

Fill the active Color Attribute with the current paint color.

Affect Alpha
: Set color completely opaque instead of reusing existing alpha.

Sample Color

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Sample Color*
Shortcut::	`Shift-X`

Adjust the brush color of the Draw tool to the color under the mouse cursor.

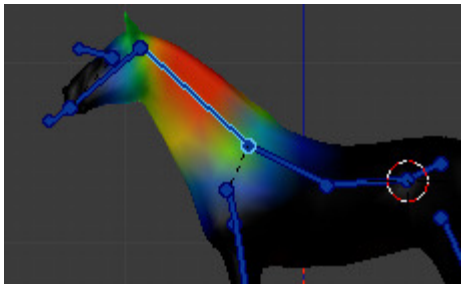
Weight Paint

- Introduction
  - The Weighting Color Code
  - Normalized Weight Workflow
- Brushes
  - Brush Types
- Weight Paint Tools
- Tool Settings
  - Brushes
  - Brush Settings
  - Symmetry
  - Options
- Using Vertex Groups
  - Vertex Groups for Bones
  - Vertex Groups for Particles
- Editing
  - Assign from Bone Envelopes
  - Assign Automatic from Bone
  - Normalize All
  - Normalize
  - Mirror
  - Invert
  - Clean
  - Quantize
  - Levels
  - Smooth
  - Transfer Weights
  - Limit Total
  - Set Weight
  - Sample Weight
  - Sample Group
  - Gradient (Linear)
  - Gradient (Radial)
  - Locks

Introduction

Vertex Groups can potentially have a very large number of associated vertices and thus a large number of weights (one weight per assigned vertex). *Weight Painting* is a method to maintain large amounts of weight information in a very intuitive way.

It is primarily used for rigging meshes, where the vertex groups are used to define the relative bone influences on the mesh. But we use it also for controlling particle emission, hair density, many modifiers, shape keys, etc.



Vertex group in Weight Paint Mode.

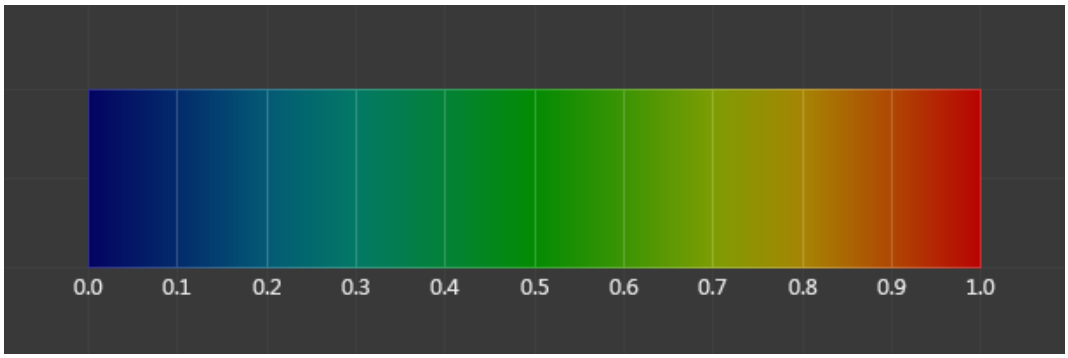
You can enter Weight Paint Mode from the Mode selector `Ctrl-Tab`. The selected mesh object is displayed slightly shaded with a rainbow color spectrum. The color visualizes the weights associated to each vertex in the active vertex group. By default *blue* means unweighted and *red* means fully weighted.

You can assign weights to the vertices of the object by painting on it with weight brushes. Starting to paint on a mesh automatically adds weights to the active vertex group (a new vertex group is created if needed).

Vertex Groups can be managed in the pallette pop-over in the middle of the header.

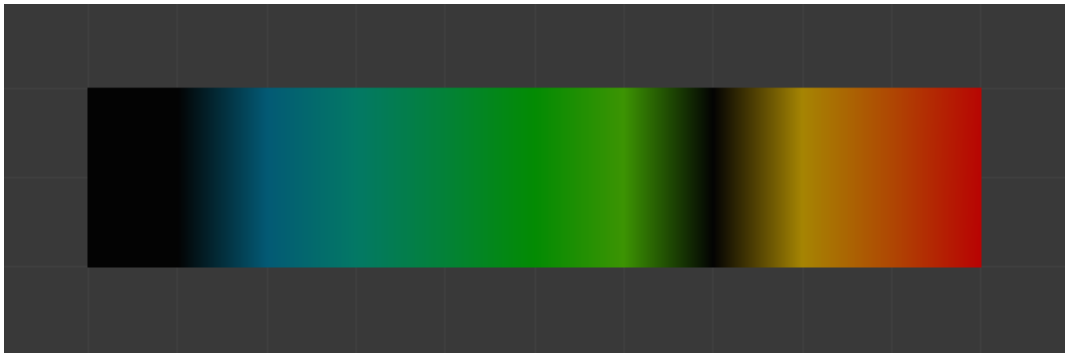
The Weighting Color Code

Weights are visualized by a gradient using a cold/hot color system, such that areas of low value (with weights close to 0.0) are displayed as blue (cold) and areas of high value (with weights close to 1.0) are displayed as red (hot). And all in-between values are displayed as rainbow colors (blue, green, yellow, orange, red).



The color spectrum and their respective weights.

In addition to the above described color code, Blender has a special visual notation (as an option) for unreferenced vertices: They are displayed as black. Thus you can see the referenced areas (displayed as cold/hot colors) and the unreferenced areas (in black) at the same time. This is most practicable when you look for weighting errors. See Options.



Unreferenced vertices example.

Note

You can customize the colors in the weight gradient by enabling Custom Weight Paint Range in the *Editing* tab of the *Preferences*.

Normalized Weight Workflow

In order to be used for things like deformation, weights usually have to be normalized, so that all deforming weights assigned to a single vertex add up to 1. The *Armature* modifier in Blender does this automatically, so it is technically not necessary to ensure that weights are normalized at the painting stage.

However, while more complicated, working with normalized weights has certain advantages, because it allows use of certain tools designed for them, and because when weights are normalized, understanding the final influence of the current group does not require knowing weights in other groups on the same vertex.

These tools are provided to aid working with normalized weights:

Normalize All
: In order to start working with normalized weights it is first necessary to normalize the existing weights. The Normalize All tool can be used for that. Make sure to select the right mode and disable *Lock Active*.

Auto Normalize
: Once the weights are initially normalized, the Auto Normalize option can be enabled to automatically maintain normalization as you paint. This also tells certain tools that the weights are supposed to be already normalized.

Vertex group locking
: Any vertex group can be locked to prevent changes to it. This can be done via the lock icon in the vertex group list, or using bone selection and the locks pie menu.

This setting prevents accidental edits to groups. However, since it is also respected by *Auto Normalize*, in the normalized weight workflow it has a more significant meaning of locking the current influence of chosen bones, so that when you paint other bones, the weight is redistributed only between the unlocked groups.

In locations affected by multiple bones, this allows more precise tweaking and re-balancing of weights by temporarily focusing on a subset of bones. This can also be aided by the Lock Relative option, which displays unlocked groups as though re-normalized with the locked groups deleted, thus making it appear as if the locked groups did not even exist.

Multi-Paint

Finally, the Multi-Paint option allows treating multiple selected bones as if they were one bone, so that the painting operations change the combined weight, preserving the ratio within the group. Combined with locking, this allows balancing between one set of bones versus the rest, excluding a third set that has its influence not affected in any way due to locks.

Technically, this option does not require the normalized workflow, but since non-normalized weights can add to more than 1, the weight display behaves best with *Auto Normalize* enabled.

Tip

For example, when dealing with a bone loop, e.g. mouth or an eye, selecting the loop with *Multi-Paint* exposes the falloff between the loop as a whole and surrounding bones, while locking the surrounding bones and using *Lock Relative* displays the falloff between bones within the loop. Thus the complex two-dimensional falloff of each bone can be viewed and edited as two independent one-dimensional gradients.

Brushes

Brush Types

See Brush Type.

Available brush types are listed here, together with brushes from the *Essentials* asset library using them.

Draw

Brushes: Paint

Paints a specified weight over the object.

Blend

The brush Blend Modes defines in which way the weight value is applied to the vertex group while painting.

Mix

In this Blending mode the Weight value defines the *target weight* that will eventually be reached when you paint long enough on the same location of the mesh. And the strength determines how many strokes you need to place at the target weight. Note that for strength = 1.0 the target weight is painted immediately and for Weight = 0.0 the brush just does nothing.

Add

In this Blending mode the specified weight value is *added* to the vertex weights. The strength determines which fraction of the weight gets added per stroke. However, the brush will not paint weight values above 1.0.

Subtract

In this Blending mode the specified weight value is *subtracted* from the vertex weights. The strength determines which fraction

of the weight gets removed per stroke. However, the brush will not paint weight values below 0.0.

Lighten
: In this Blending mode the specified weight value is interpreted as the target weight. Very similar to the Mix Blending mode, but only weights below the target weight are affected. Weights above the target weight remain unchanged.

Darken
: This Blending mode is very similar to the Lighten Blending mode. But only weights above the target weight are affected. Weights below the target weight remain unchanged.

Multiply
: Multiplies the vertex weights with the specified weight value. This is somewhat like subtract, but the amount of removed weight is now dependent on the Weight value itself.

Blur
: Smooths out the weighting of adjacent vertices. In this mode the Weight Value is ignored. The strength defines how much the smoothing is applied.

Blur
: Brushes: Blur

  Smooths out the weighting of adjacent vertices. In this mode the Weight Value is ignored. The strength defines how much the smoothing is applied.

Average
: Brushes: Average

  Smooths weights by painting the average resulting weight from all weights under the brush.

Smear
: Brushes: Smear

Smudges weights by grabbing the weights under the brush and “dragging” them. This can be imagined as a finger painting tool.

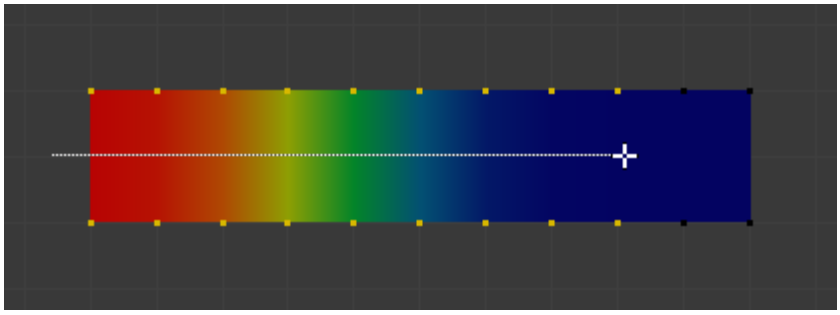
Weight Paint Tools

Brush

Tool to use for any of the weight paint brushes.

Gradient

Applies a linear/radial weight gradient; this is useful at times when painting gradual changes in weight becomes difficult. Blends the weights of selected vertices with unselected vertices.



Example of the Gradient tool being used with selected vertices.

Weight

The gradient starts at the current selected weight value, blending out to nothing.

Strength

Lower values can be used so the gradient mixes in with the existing weights (just like with the brush).

Type

The shape of the gradient.

Linear:: Create gradient that forms a straight line.
Radial:: Create gradient that forms a circle.

Note

These are also availible via shortcuts as the menu operators.

Sample

Weights

Sets the brush *Weight* as the weight selected under the cursor. The sampled weight is displayed in the tool settings.

Vertex Group

Displays a list of possible vertex groups to select that are under the cursor.

Annotate

Draw free-hand annotation.

Annotate Line

Draw straight line annotation.

Annotate Polygon

Draw a polygon annotation.

Annotate Eraser

Erase previous drawn annotations.

Tool Settings

- Brushes
  - Introduction
  - Manage Brushes
  - Brush Settings
- Brush Settings
- Symmetry
- Options

Brush Settings

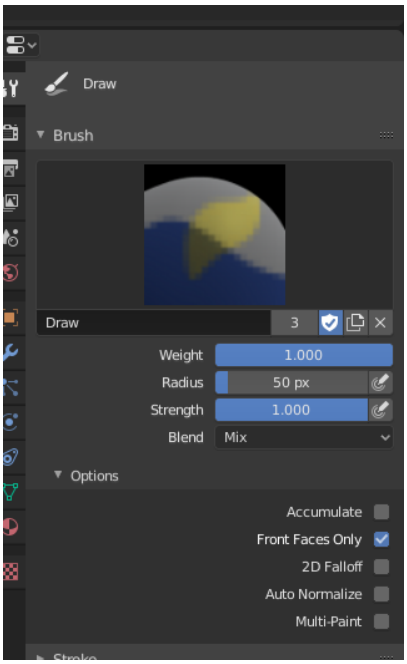
Information on brush settings for every mode can be found in these pages:

Brush
: General and advanced settings.

Stroke
: Stroke methods and settings.

Falloff
: Falloff curves and settings.

Cursor
: Cursor and appearance settings.



Brush settings panel.

Symmetry

Reference

Mode::	Vertex Paint Mode
Tool::	*Toolbar* ▸ *Tool* ▸ *Symmetry*

Mirror Vertex Groups
: Use this option for mirrored painting on groups that have symmetrical names, like with suffix ".R"/ ".L" or "_R" / "_L". If a group has no mirrored counterpart, it will paint symmetrically on the active group itself. You can read more about the naming convention in Editing Armatures: Naming conventions. The conventions for armatures/bones apply here as well.

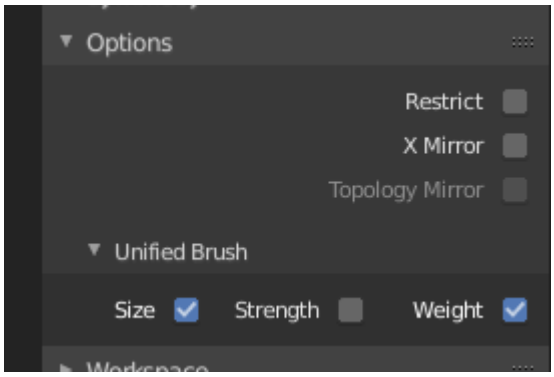
Mirror X, Y, Z
: Mirror the brush strokes across the selected local axes. Note that if you want to alter the directions the axes point in, you must rotate the model in Edit Mode and not in Object Mode.

Radial X, Y, Z
: These settings allow for radial symmetry in the desired axes. The number determines how many times the stroke will be repeated within 360 degrees around the central axes.

Topology Mirror
: Use topology-based mirroring, for when both sides of a mesh have matching mirrored topology. See here for more information.

Options

The weight paint options change the overall brush behavior.



Paint options.

Auto Normalize

Ensures that all deforming vertex groups add up to one while painting. When this option is turned off, then all weights of a vertex can have any value between 0 and 1. However, when vertex groups are used as deform groups for character animation then Blender always interprets the weight values relative to each other. That is, Blender always does a normalization over all deform bones. Hence in practice it is not necessary to maintain a strict normalization and further normalizing weights should not affect animation at all.

This option works most intuitively when used to maintain normalization while painting on top of weights that are already normalized with another tool.

Lock-Relative

Displays bone-deforming groups as if all locked deform groups were deleted, and the remaining ones were re-normalized. This is intended for use when balancing weights within a group of bones while all other bones are locked. With this option you can also temporarily view non-normalized weights as if they were normalized, without actually changing the values.

Multi-Paint

Paint on all selected vertex groups simultaneously, in a way that preserves their relative influence. This can be useful when tweaking

weights in an area that is affected by more than three bones at once, e.g. certain areas on a character's face.

This option is only useful in the *Armature* tab, where you can select multiple vertex groups by selecting multiple pose bones. Once at least two vertex groups are selected, viewport colors and paint logic switch to Multi-Paint mode, using the sum of the selected groups' weights if *Auto Normalize* is enabled, and the average otherwise. Any paint operations aimed at this collective weight are applied to individual vertex group weights in such way that their ratio stays the same.

Since the ratio is undefined if all weights are zero, Multi-Paint cannot operate on vertices that do not have any weight assigned to the relevant vertex groups. For this reason it also does not allow reducing the weight all the way to zero. When used with *X Mirror*, it only guarantees completely a symmetrical result if weights are initially symmetrical.

Tip

While Multi-Paint cannot directly paint on zero-weight vertices, it is possible to use the *Smooth Weight* tool to copy a reasonable nonzero weight distribution from adjacent vertices without leaving Multi-Paint mode or changing bone selection.

To do that, enable vertex selection, select target vertices, and apply one iteration of the tool using vertex groups from *Selected Pose Bones* with low Factor. After that simply paint on top to set the desired collective weight.

Restrict

This option limits the influence of painting to vertices (even with weight 0) belonging to the selected vertex group.

See also

See the Brush Display options.

Using Vertex Groups

Vertex Groups for Bones

This is one of the main uses of weight painting. While you can have Blender generate the weights automatically (see the skinning section), you may want to tweak them or even create them from scratch, especially around joints.

The process is as follows:

1. Select the armature and bring it into *Pose Mode* by pressing `Ctrl-Tab`.
2. Make sure that *Edit ▸ Lock Object Modes* is unchecked in the topbar.
3. Select the mesh and bring it into *Weight Paint Mode*.
4. Make sure that *Bone Selection* is checked in the 3D Viewport's header.
5. Select a bone using `Alt-LMB` (or `Shift-Ctrl-LMB`). This will activate the bone's vertex group and display its current weights on the mesh.
6. Paint weights for the bone using `LMB`.

Note

You can only select one bone at a time in this mode.

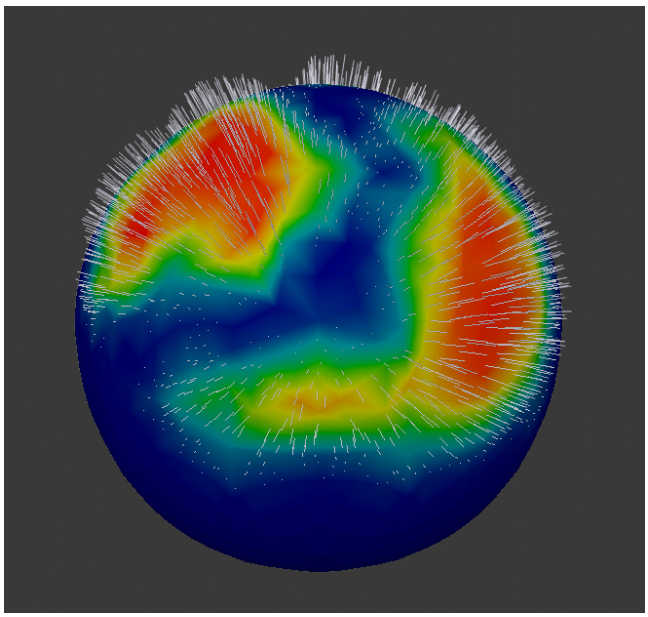
Tip

The bones are likely embedded inside the mesh, making them invisible and unselectable. To get around this, you can enable In Front for the armature.

If a bone doesn't have a vertex group yet when you start painting, Blender will create one automatically.

If you have a symmetrical mesh and a symmetrical armature, you can use Mirror Vertex Groups to automatically create vertex groups and weights for the other side.

Vertex Groups for Particles



Weight painted particle emission.

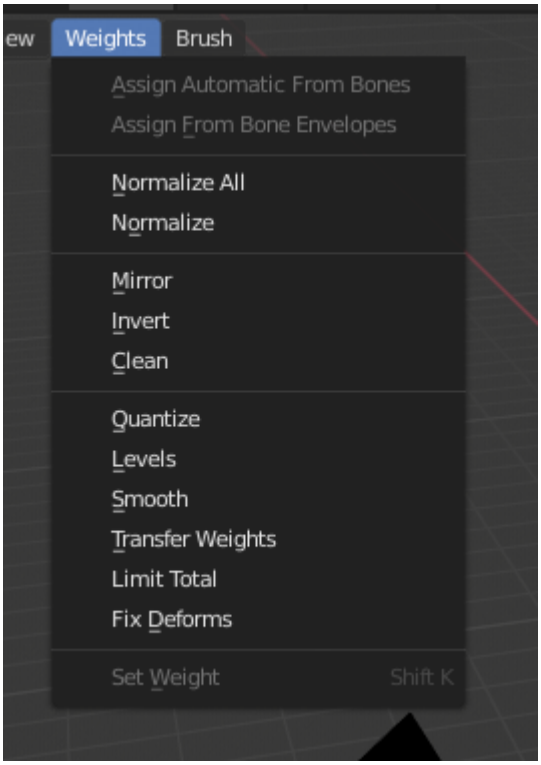
By selecting vertex groups in the Vertex Groups panel of a particle system's properties, you can have different particle densities, hair lengths etc. across different areas of the mesh.

Editing

Reference

Mode:: Edit Mode and Weight Paint Mode
Menu:: *Weights*

Blender provides a set of helper tools for Weight Painting.



Weight Paint Tools.

The Subset Option

Some of the tools also provide a Subset filter to restrict their functionality to only specific vertex groups (in the Adjust Last Operation panel, displayed after the tool is called) with following options:

- Active Group
- Selected Pose Bones
- Deform Pose Bones
- All Groups

All tools also work with Vertex Selection Masking and Face Selection Masking. In these modes the tools operate only on selected vertices or faces.

Assign from Bone Envelopes

Apply the envelope weight of the selected bone(s) to the selected vertex group.

Assign Automatic from Bone

Apply from the selected bone(s) to the vertex group the same “auto-weighting” methods as available in the Parent armature menu.

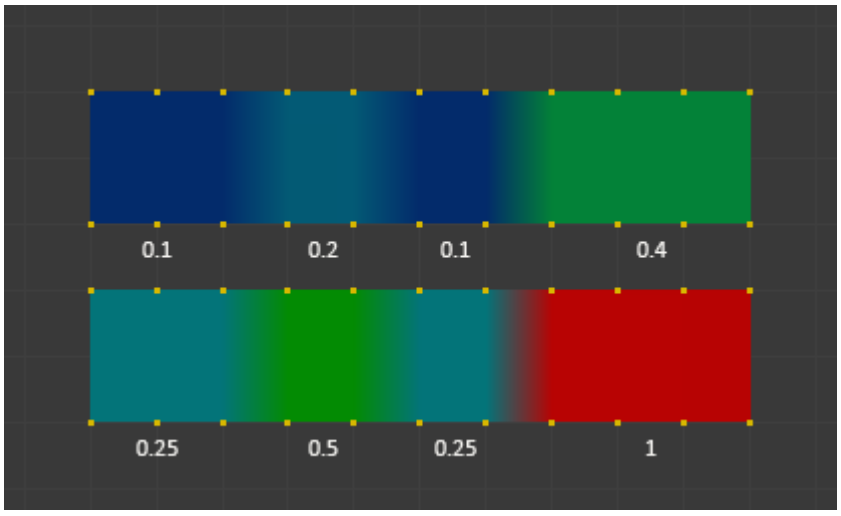
Normalize All

For each vertex, this tool makes sure that the sum of the weights across all vertex groups is equal to 1. This tool normalizes all of the vertex groups, except for locked groups, which keep their weight values untouched.

Lock Active
: Keep the values of the active group while normalizing all the others.

Normalize

This tool only works on the active vertex group. All vertices keep their relative weights, but the entire set of weights is scaled up such that the highest weight value is 1.0.



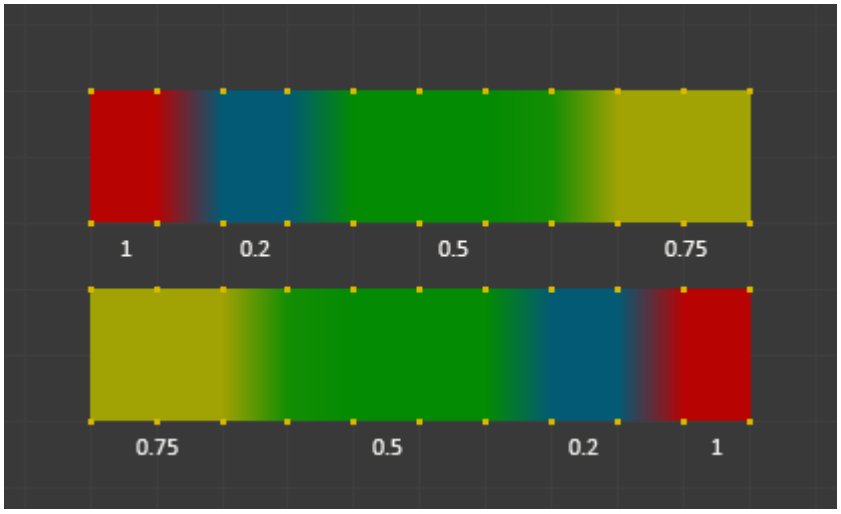
Normalize example.

Mirror

The *Mirror Vertex Group* tool mirrors the weights from one side of a perfectly symmetrical mesh to the opposite side. Those vertices that have no corresponding vertex on the other side will not be affected. But note, the weights are not transferred to the corresponding opposite bone weight group.

Note

Mirroring only works when the object's rest pose is perfectly symmetrical across the X axis.



Mirror example.

Mirror Weights

With this option checked, every selected vertex receives the weight information of its symmetrical counterpart. If both vertices are selected, it will be a weight information exchange; if only one is selected, information from the unselected will overwrite the selected one. Information on weight is passed for the active group only, unless *All Groups* is checked, in which case it is passed for all groups.

Flip Group Names

Works with selected vertices that belong to vertex groups with "symmetrical names" (with components like "L", "R", "right", "left").

All selected vertices that belong to the active group, or to the symmetrical of the active group, will have their assignation to that group replaced by an assignation to the symmetrical one; however, its weight will be preserved. If *All Groups* is checked, all assignations to these kind of groups will be replaced by the symmetrical counterpart, also keeping the old weights.

All Groups
: Operate on all vertex groups, instead of the active one.

Topology Mirror
: Mirror for meshes which are not fully symmetric (approximate mirror). See here for more information.

Tip

Mirror to Opposite Bone

If you want to create a mirrored weight group for the opposite bone (of a symmetric character), then you can do this:

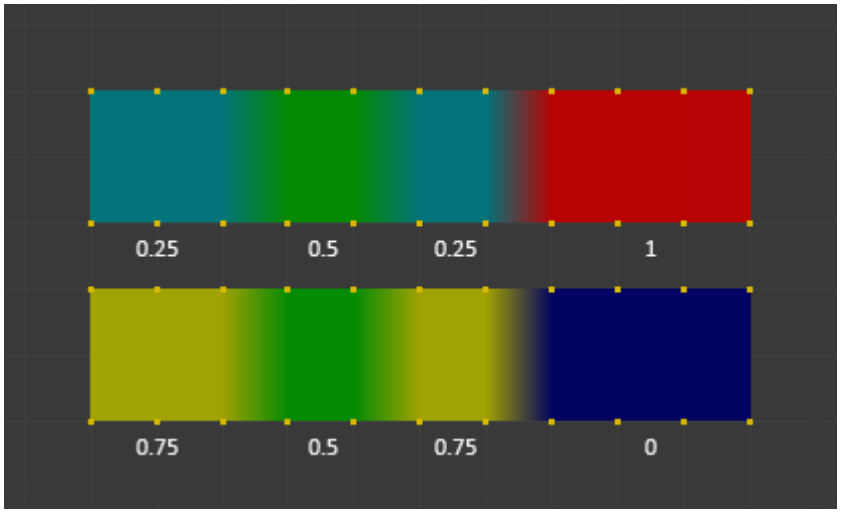
1. Delete the target vertex group (where the mirrored weights will be placed).
2. Create a copy of the source bone vertex group (the group containing the weights which you want to copy).
3. Rename the new vertex group to the name of the target vertex group (the group you deleted above).
4. Select the target vertex group and call the Mirror tool (use only the Mirror weights option and optionally *Topology Mirror* if your mesh is not symmetric).

Invert

Replaces each Weight of the selected weight group by × -1.0 weight.

Examples:

- Original 1.0 converts to 0.0
- Original 0.5 remains 0.5
- Original 0.0 converts to 1.0



Invert.

Subset
: Restrict the tool to a subset. See above The Subset Option about how subsets are defined.

Add Weights
: Add vertices that have no weight before inverting (these weights will all be set to 1.0).

Remove Weights
: Remove vertices from the vertex group if they are 0.0 after inverting.

Note

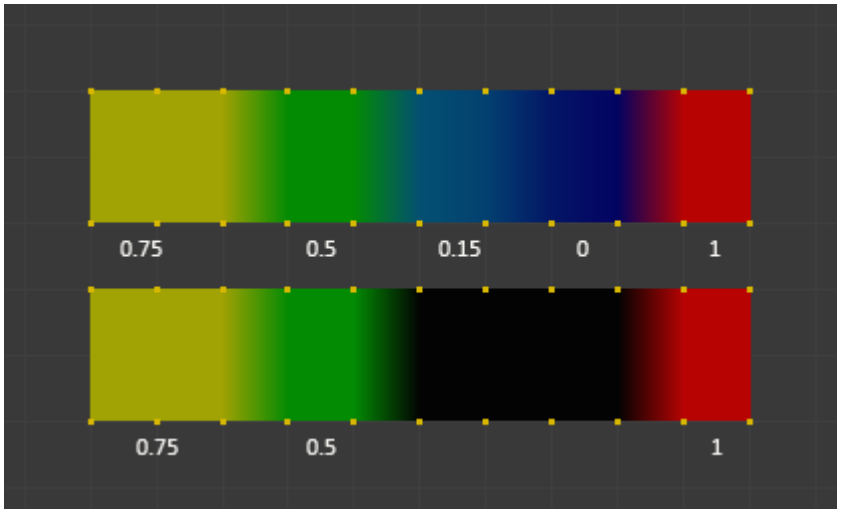
Locked vertex groups are not affected.

Clean

Clean Vertex Group Weights unassigns vertices from Vertex Groups whose weights are below the *Limit*. Removes weights below a given threshold. This tool is useful for clearing your weight groups of very low (or zero) weights.

In the example shown, a cutoff value of 0.2 is used (see operator options below) so all blue parts are cleaned out.

Note, the images use the *Show Zero weights* Active option so that unreferenced Weights are shown in Black.



Clean example.

Subset

Restrict the tool to a subset. See above The Subset Option for how subsets are defined.

Limit

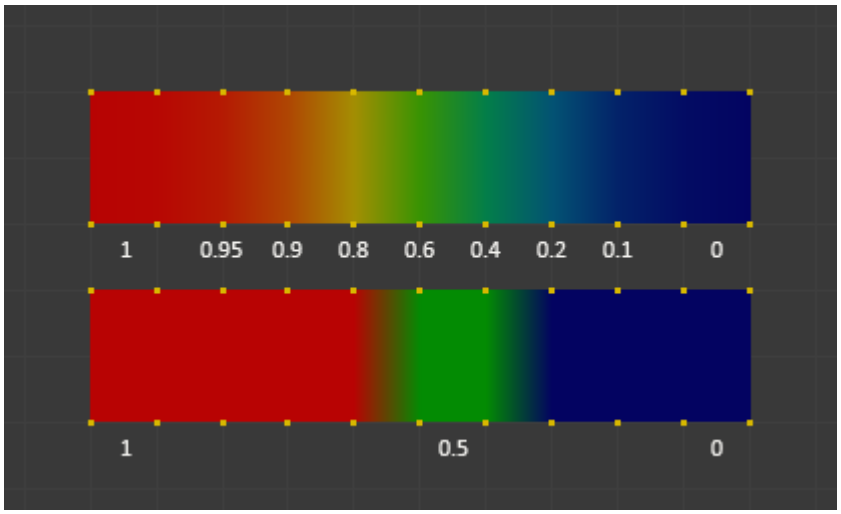
This is the minimum weight value that will be kept in the group. Weights below this value will be removed from the group.

Keep Single

Ensure that the *Clean* tool will not create completely unreferenced vertices (vertices which are not assigned to any vertex group), so each vertex will keep at least one weight, even if it is below the limit value!

Quantize

This operator uses a process known as Quantization which takes the input weights and clamps each weight to a number of steps between (0 - 1), so there is no longer a smooth gradient between values.



Quantize example (Steps = 2).

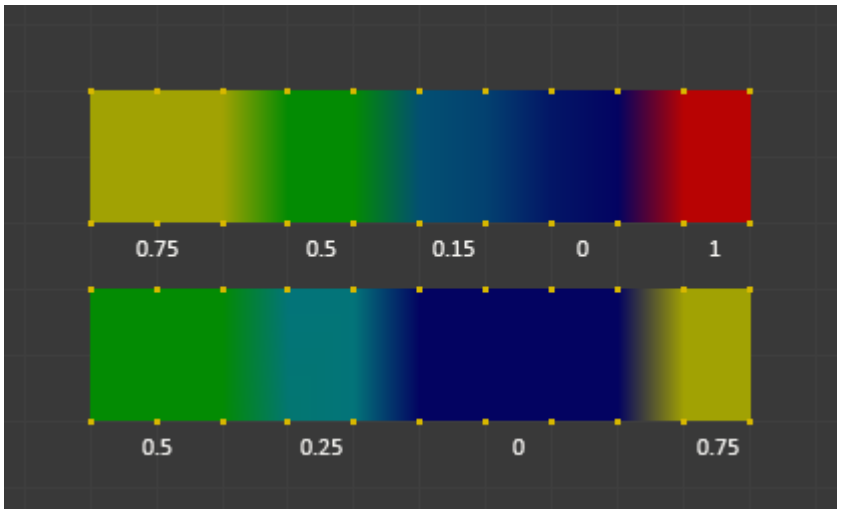
Steps
: The number of steps between 0 and 1 to quantize the weights into. For example 5 would allow the following weights `[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]`.

Levels

Adds an offset and a scale to all weights of the selected weight groups. with this tool you can raise or lower the overall “heat” of the weight group.

Note

No weight will ever be set to values above 1.0 or below 0.0 regardless of the settings.



Levels example.

Subset
: Restrict the tool to a subset. See above The Subset Option for how subsets are defined.

Offset
: A value from the range (-1.0 - 1.0) to be added to all weights in the vertex group.

Gain
: All weights in the Subset are multiplied with the gain.

Note

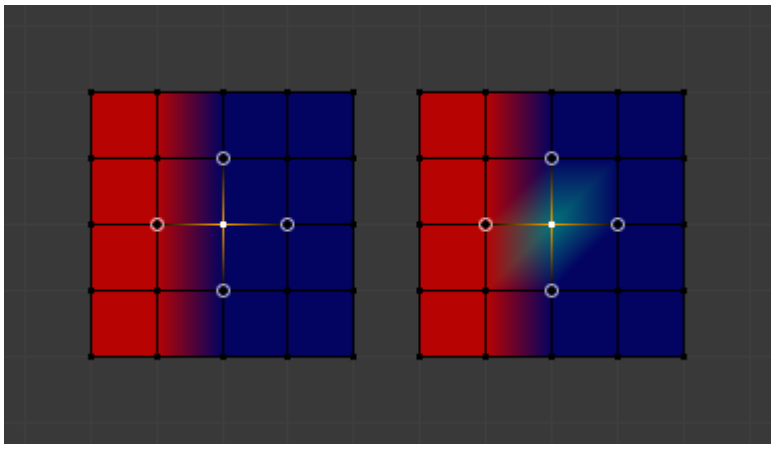
Whichever *Gain* and *Offset* you choose, in all cases the final value of each weight will be clamped to the range (0.0 - 1.0). So you will never get negative weights or overheated areas (weight > 1.0) with this tool.

Smooth

Tip

The Smooth tool only works when “Vertex selection masking for painting” is enabled. Otherwise the tool button is grayed out.

Blends the weights of selected vertices with adjacent unselected vertices. This tool only works in vertex select mode.

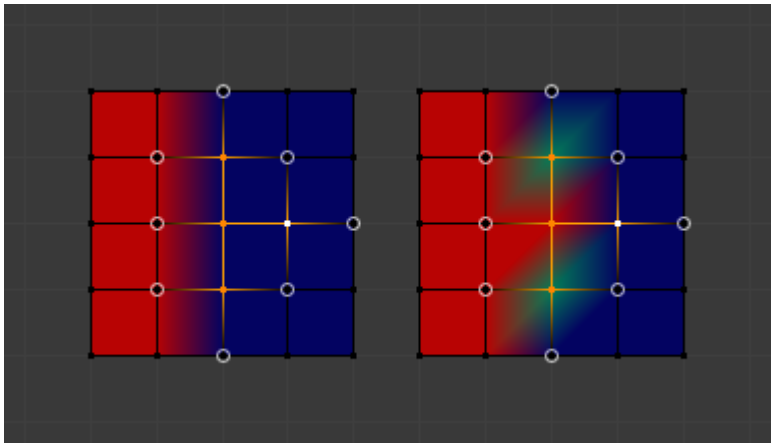


To understand what the tool really does, let us take a look at a simple example. The selected vertex is connected to four adjacent vertices (marked with a gray circle in the image). All adjacent vertices are unselected. Now the tool calculates the average weight of all connected **and** unselected vertices. In the example this is:

$(1 + 0 + 0 + 0) / 4 = 0.25$

This value is multiplied by the factor given in the Operator options (see below).

- If the factor is 0.0 then actually nothing happens at all and the vertex just keeps its value.
- If the factor is 1.0 then the calculated average weight is taken (0.25 here).
- Dragging the factor from 0 to 1 gradually changes from the old value to the calculated average.



Now let us see what happens when we select all but one of the neighbors of the selected vertex as well. Again all connected and unselected vertices are marked with a gray circle. When we call the Smooth tool now and set the Factor to 1.0, then we see different results for each of the selected vertices:

- The top-most and bottom-most selected vertices:

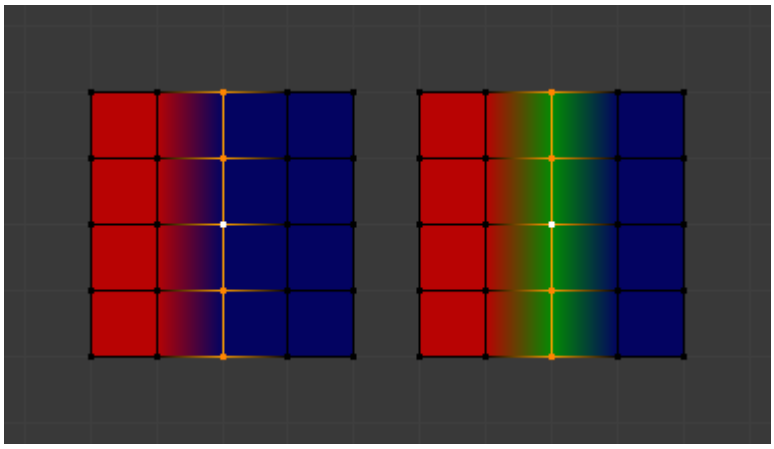
  are surrounded by three unselected vertices, with an average weight of \((1 + 0 + 0) / 3 = 0.333\) So their color has changed to light green.

- The middle vertex:

  is connected to one unselected vertex with `weight = 1`. So the average weight is 1.0 in this case, thus the selected vertex has changed to red.

- The right vertex:

  is surrounded by three unselected vertices with average weight = \((0 + 0 + 0) / 3 = 0.0\) So the average weight is 0, thus the selected vertex has not changed at all (it was already blue before Smooth was applied).



Finally let us look at a practical example. The middle edge loop has been selected and it will be used for blending the left side to the right side of the area.

- All selected vertices have two unselected adjacent vertices.
- The average weight of the unselected vertices is \((1 + 0) / 2 = 0.5\)
- Thus when the *Factor* is set to 1.0 then the edge loop turns to green and finally does blend the cold side (right) to the hot side (left).

Factor
: The effective amount of blending. When Factor is set to 0.0 then the Smooth tool does not do anything. For Factor > 0 the weights of the affected vertices gradually shift from their original value towards the average weight of all connected **and** unselected vertices (see examples above).

Iterations
: Number of times to repeat the smoothing operation.

Expand/Contract
: Positive values expand the selection to neighboring vertices while contract limits to the selection.

Source
: The vertices to mix with.

All::	Smoothing will smooth both selected and deselected vertices.
Only Selected::	Smoothing will only smooth with selected vertices.
Only Deselected::	Smoothing will only smooth with deselected vertices.

Transfer Weights

Copy weights from other objects to the vertex groups of the active object.

By default this tool copies only the active (selected) vertex group of the source object to the active vertex group of target object or creates a new one if the group does not exist. However, you can change the tool's behavior in the Adjust Last Operation panel.

For example, to transfer all existing vertex groups from the source objects to the target, change the *Source Layers Selection* option to *By Name*.

Note

This tool uses the generic "data transfer", but transfers from all selected objects to active one. Please refer to the Data Transfer docs for options details and explanations.

Prepare the Copy

You first select all source objects, and finally the target object (the target object must be the active object).

It is important that the source objects and the target object are at the same location. If they are placed side-by-side, then the weight transfer will not work. (See the *Vertex Mapping* option.) You can place the objects on different layers, but you have to ensure that all objects are visible when you call the tool.

Now ensure that the target object is in Weight Paint Mode. Open the Toolbar and call the *Transfer Weights* tool in the *Weight Tools* panel.

Adjust Last Operation Panel Confusion

You may notice that the Adjust Last Operation panel stays available after the weight transfer is done. The panel only disappears when you call another Operator that has its own Adjust Last Operation panel. This can lead to confusion when you use Transfer weights repeatedly after you changed your vertex groups. If you then use the still-visible Adjust Last Operation panel, then Blender will reset your work to its state right before you initially called the *Transfer Weights* tool.

So when you want to call the *Transfer Weights* tool again after you made some changes to your vertex groups, then **always** use the *Transfer Weights* button, even if the Adjust Last Operation panel is still available. Unless you really want to reset your changes to the initial call of the tool.

Limit Total

Reduce the number of weight groups per vertex to the specified Limit. The tool removes lowest weights first until the limit is reached.

Hint

The tool can only work reasonably when more than one weight group is selected.

Subset
: Restrict the tool to a subset. See above The Subset Option for how subsets are defined.

Limit
: Maximum number of weights allowed on each vertex.

Set Weight

Reference

Mode::	Weight Paint Mode
Menu::	*Weight ▸ Set Weight*
Shortcut::	`Ctrl-X`

Fill the active vertex group with the current paint weight.

Sample Weight

Reference

Mode::	Weight Paint Mode
Menu::	*Weight ▸ Sample Weight*
Shortcut::	`Shift-X`

Adjust the Weight of the Draw tool to the weight of the vertex under the mouse cursor.

Sample Group

Reference

Mode::	Weight Paint Mode
Menu::	*Weight ▸ Sample Group*
Shortcut::	`Shift-Ctrl-X`

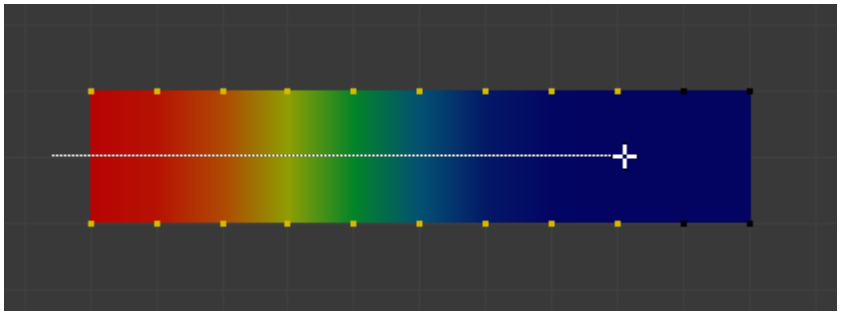
Select one of the vertex groups available under current mouse position.

Gradient (Linear)

Reference

Mode:: Weight Paint Mode
Menu:: *Weights ▸ Gradient (Linear)*
Shortcut:: `Shift-A`

Applies a linear weight gradient; this is useful at times when painting gradual changes in weight becomes difficult. Blends the weights of selected vertices with unselected vertices.



Example of the Gradient tool being used with selected vertices.

Weight
: The gradient starts at the current selected weight value, blending out to nothing.

Strength
: Lower values can be used so the gradient mixes in with the existing weights (just like with the brush).

Type
: The shape of the gradient.

Linear:: Create gradient that forms a straight line.
Radial:: Create gradient that forms a circle.

Gradient (Radial)

Reference

Mode:: Weight Paint Mode

Menu:: *Weights ▸ Gradient (Radial)*
Shortcut:: `Shift-Alt-A`

Applies a radial weight gradient; this is useful at times when painting gradual changes in weight becomes difficult. Blends the weights of selected vertices with unselected vertices.

Weight
: The gradient starts at the current selected weight value, blending out to nothing.

Strength
: Lower values can be used so the gradient mixes in with the existing weights (just like with the brush).

Type
: The shape of the gradient.

Linear:: Create gradient that forms a straight line.
Radial:: Create gradient that forms a circle.

Locks

Reference

Mode:: Edit Mode and Weight Paint Mode
Menu:: *Weights ▸ Locks*
Shortcut:: `K`

Vertex groups can be locked to prevent undesired edits to a particular vertex group.

Tip

Bones that belong to a locked vertex group are displayed in red the 3D Viewport.

Lock All
: Locks all vertex groups.

Lock Selected
: Locks selected vertex groups.

Lock Unselected
: Locks unselected vertex groups.

Lock Only Selected
: Lock selected and unlock selected vertex groups.

Lock Only Unselected
: Unlock selected and lock unselected vertex groups.

Unlock All
: Unlocks all vertex groups.

Unlock Selected
: Unlocks selected vertex groups.

Unlock Unselected
: Unlocks Unselected vertex groups.

Invert Locks
: Inverts the locks on all vertex groups.

Curves Sculpting

- Introduction
  - Curves Menu
  - Selection Modes
  - Select Menu
  - Controls
  - Display
- Brushes
  - Selection Paint
  - Add Curves
  - Delete Curves
  - Density Brush
  - Comb Curves
  - Snake Hook Curves
  - Grow / Shrink Curves
  - Pinch Curves
  - Puff Curves
  - Smooth Curves
  - Slide Curves
- Common Settings
  - Brush
  - Stroke
  - Falloff
  - Cursor

Introduction

Curves Sculpt Mode allows working with curves using various brushes. It is commonly used for hair grooming, but can be used with all kinds of curves.

The curves' surface object plays an important role in many curves sculpting brushes. Most brushes such as Add Curves require the surface to be set already.

Note

Curves Sculpt tools only use the original mesh of the surface object and don't take its modifiers into account.

Curves Menu

Snap to Deformed Surface
: Re-attach curves to a deformed surface using the existing attachment information. This only works when the topology of the surface mesh has not changed.

Snap to Nearest Surface
: Find the closest point on the surface for the root point of every curve and move the root there. This needs to be run after the surface mesh topology changed

Convert to Particle System
: Add a new hair particle system, or update an system on the surface object. The operator is used for backwards compatibility with the old hair type particle system.

Selection Modes

Reference

Mode::	Sculpt Mode
Menu::	*3D Viewport Header ▸ Select Mode*
Shortcut::	1, 2

Selection modes limits selection operators to certain curve domains. This feature is makes it easy to select whole segments at once, or to give more granular control over editing.

Control Points::	Allows selection of individual control points.
Curve::	Limits selection to whole curve segments.

Select Menu

All
: Select all control points or curves.

None
: Deselect all control points or curves.

Invert
: Invert the selection.

Random
: Randomizes inside the existing selection or create new random selection if nothing is selected already.

Endpoints
: Select endpoints of curves. Only supported in the Control Point selection mode.

  Amount Start, End
  : Number of points to select from the front or back of the curve.

Grow
: Select points or curves which are close to already selected elements.

Controls

Sculpt mode provides several properties that give advanced control of the tool's behavior. These options can be found in the right-hand side of the 3D Viewport's Header.

Mirror
: Allows tools to affect curves symmetrically according to the chosen axis.

Use Sculpt Collision
: Prevents the curve segments from passing through the Surface Object.

Display

Overlays

Cage Opacity
: Shows the original curves that are currently being edited which is useful with when procedural deformations or child curves are used.

Brushes

Brushes for Curves *Sculpt Mode* bundled in the *Essentials* library.

- Selection Paint
- Add Curves
- Delete Curves
- Density Brush
- Comb Curves
- Snake Hook Curves
- Grow / Shrink Curves
- Pinch Curves
- Puff Curves
- Smooth Curves
- Slide Curves

Selection Paint

Paint curves or control paints to use as masks for the other tools. The selection visibility can be controlled by the *Selection Opacity* option in the Viewport Overlays.

By default the selection sets a new selection. The selection paint can be extended by holding `Shift` and it can be subtracted by holding `Ctrl` while painting.

Brush Settings

Direction
: Determines whether to set a new selection or remove from it. It can be toggled holding `Ctrl` while painting.

Add Curves

Used to distribute new curves on the surface mesh. This tool requires the curve to have a surface object set.

The curves follow the surface normals. Using the interpolation options allows the brush to take the characteristics of existing curves.

Brush Settings

Count
: Number of curves added.

Note

Interpolation allows to add hair which are already combed. The new curves are created following the previously created curves which are in the vicinity.

Interpolate Length
: Use the average length of the curves in close proximity.

Interpolate Radius
: Use the average radius of the curves in close proximity. If there is no radius attribute, then the interpolation will skip.

Interpolate Shape
: Use the average shape of the curves in close proximity.

Interpolate Point Count
: Use the average amount of control points of the curves in close proximity.

Curve Length

Length of newly added curves when not interpolated.

Curve Radius
: Radius of newly added curves when not interpolated.

Points per Curve
: Number of Control Points for the new created curves when the point count is not interpolated.

Delete Curves

Remove existing curves. The tool deletes the entire curves, if any of its segments fall under the brush falloff radius.

Density Brush

Create (or remove) curves based on a target distance. It generates a high number of points and then rejects the ones that are too close to existing points.

Brush Settings

Density Mode
: Determines whether the brush adds or removes curves.

Auto::	Either add or remove curves depending on the distance between existing curve roots under the cursor.
Add::	Add new curves between existing curves, taking the minimum distance into account.
Remove::	Remove curves whose root points are too close.

Distance Min
: Goal distance between the curve roots.

  Edit Minimum Distance `R`
  : Interactively sets the *Distance Min* value by displaying a graphic inside the brush cursor, giving a representation of the density.

    The density can be adjusted by moving the mouse cursor closer or farther from the paint cursor. The *Distance Min* will be changed once the operator is confirmed.

Count Max
: The maximum amount of points that the brush tries to sample in the surface.

Comb Curves

Shape the curves by moving their control points while preserving the initial length of every curve segment.

Brush Settings

Curve Falloff
: Falloff that is applied from the tip to the root of each curve.

Snake Hook Curves

Extend existing curves in a specific direction of the brush strokes. A curve can only be extended by its tips (the control point opposite to the root).

Note

No new control points are created for the curves. Instead, existing points are sampled along the new curve shape.

Grow / Shrink Curves

Change the length of existing curves preserving the amount of control points and resampling the curve to preserve the original shape.

Brush Settings

Direction
: Determines whether to grow or shrink the curves. It can be toggled holding `Ctrl` while sculpting.

Scale Uniform
: Grow or shrink curves by changing their size uniformly instead of using trimming or extrapolation.

Minimum Length
: Avoid shrinking curves shorter than this length.

Pinch Curves

Converges adjacent curves to the location at the center of the cursor.

The pinch brush can be inverted by holding `Ctrl`.

Puff Curves

Makes the curves stand up. The brush aligns the curve with the surface normal and makes sure that points don’t move closer to the root point.

Smooth Curves

This brush makes curve segments close to one another more parallel.

Slide Curves

Slides the curves along the surface mesh. This tool requires the curve to have a Surface. Each curve is also rotated by the change in the surface normal.

Common Settings

Information on brush settings for every mode can be found in these pages:

Brush

See general and advanced Brush here.

Stroke

See the global brush settings for Stroke settings.

Falloff

See the global brush settings for Falloff settings.

Cursor

See the global brush settings for Cursor settings.

Grease Pencil

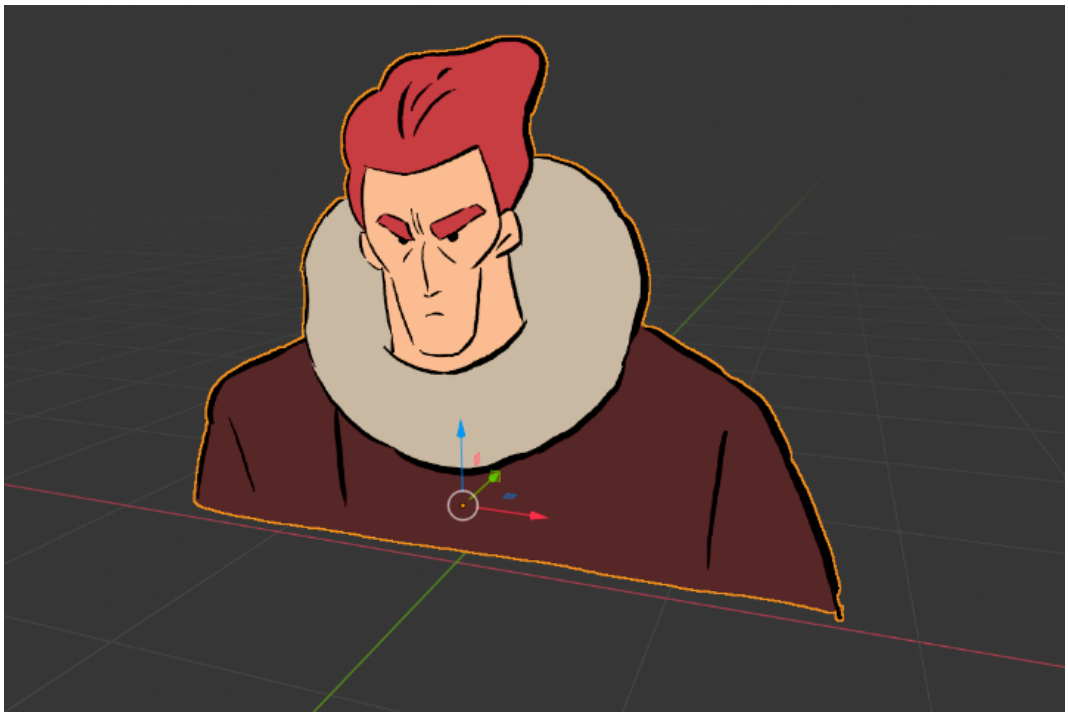
- Introduction
  - Quick Start
- Structure
  - Points
  - Edit Lines
  - Strokes
- Primitives
  - Blank
  - Stroke
  - Monkey
  - Scene Line Art
  - Collection Line Art
  - Object Line Art
- Properties
  - Object Properties
  - Data Properties
- Modifiers
  - Introduction
  - Generate
  - Deform
  - Color
  - Edit
- Visual Effects
  - Introduction
  - Types
- Materials
  - Material Shader
  - Setting Up Materials
  - Properties
- Multiframe
  - Usage
- Animation
  - Introduction

    - Interpolation
    - Animation Tools
- Object Modes
    - Draw Mode
    - Sculpt Mode
    - Edit Mode
    - Vertex Paint Mode
    - Weight Paint Mode
    - Object Mode

Introduction

Grease Pencil is a Blender object. It accepts the drawing information from a mouse or pressure-sensitive stylus and places it in 3D space as a collection of points, which are defined as a stroke.

The Grease Pencil object can be used to make traditional 2D animation, cut-out animation, motion graphics, or used it as storyboard tool, among other things.



An illustration in 3D space using the Grease Pencil object.

Strokes are created in Draw Mode, which requires a new keyframe in the animation timeline for the Grease Pencil object. Existing strokes can then be adjusted in Edit Mode and Sculpt Mode. Finally, artists can apply materials, modifiers, lighting, and visual effects to strokes.

Quick Start

Artists can add Grease Pencil to any existing Blender scene, or start with a 2D Animation template. The template offers some pre-configured options that are helpful for animation and storyboarding.

Create and Use Grease Pencil

1. From Object Mode, *Add ▸ Grease Pencil ▸ Blank*.
2. Create a new keyframe or turn on Auto Key. (See Keyframe Editing)
3. Switch to Draw Mode.
4. Click and drag across the viewport to add strokes to the Grease Pencil object.

2D Animation Template

To create a new Blender file using the "2D Animation" project template use: *File ▸ New ▸ 2D Animation*.

Note the following pre-configured setup for the 2D Animation template:

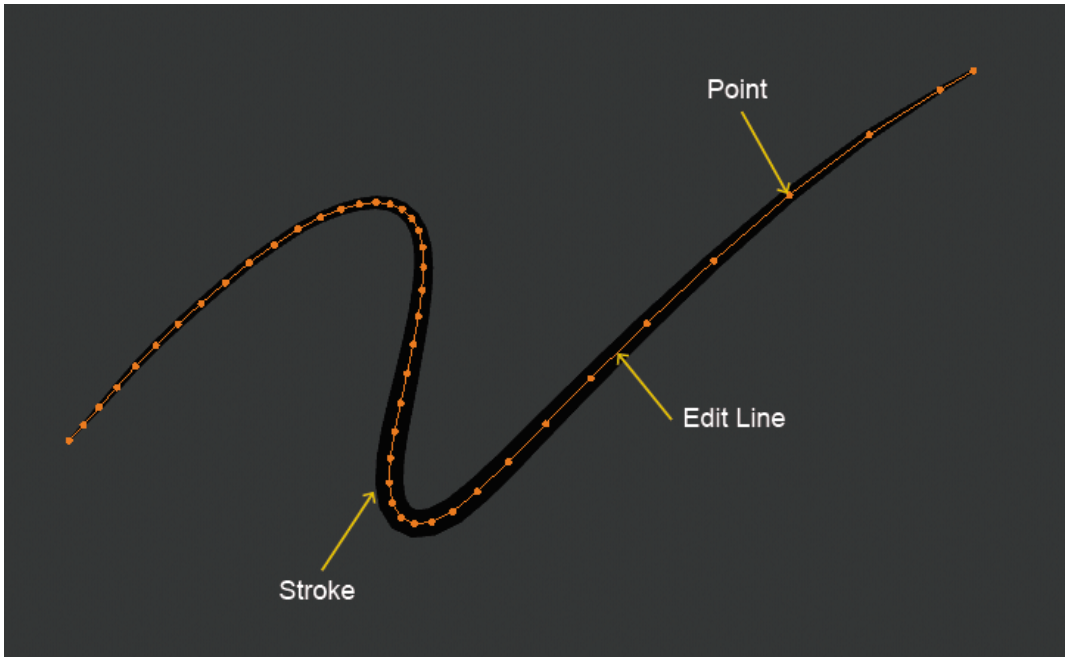
- 2D Animation is the default active workspace.
- *World Properties ▸ Surface (Background) ▸ Color* is set to white.
- *Render Properties ▸ Color Management* is set to Standard.
- The drawing plane is set to Front (X-Z).
- Line and Fill layers, along with some stroke materials, are configured for Grease Pencil.
- The animation timeline will automatically create a new keyframe when Grease Pencil is used on empty frames.

Tip

Grease Pencil can read pressure-sensitivity information from a Graphics Tablet or stylus.

Structure

Grease Pencil object has three main basic components: *points*, *edit lines* and *strokes*.



Example of Grease Pencil structure.

Points

The main element used in editing Grease Pencil objects are points. Points represent a single point in 3D space.

Each point stores all the properties that define the final appearance of the strokes as its location, thickness, alpha, weight and UV rotation for textures.

Note

Point (Grease Pencil) and Vertex (meshes) are equivalent names.

Edit Lines

Points are always connected by a straight line, which you see when you are editing in Edit Mode or when you look at a stroke in wireframe view. They are invisible on the rendered image and are used to construct the final stroke.

Strokes

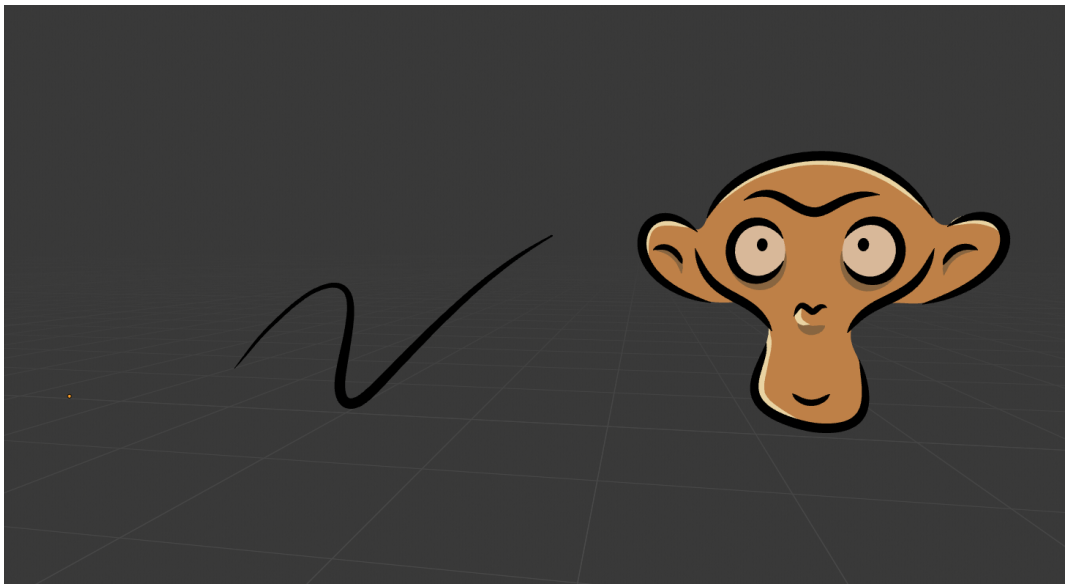
The stroke is the rendered image of the points and edit lines, using a particular Grease Pencil material. (Grease Pencil materials are linked at stroke level.)

Primitives

Reference

Mode::	Object Mode and Edit Mode
Menu::	*Add* ▸ *Grease Pencil*
Shortcut::	`Shift-A`

In Object Mode, the *Add* menu provides three different Grease Pencil primitives with preset materials and 2D layers:



Grease Pencil primitives.

Blank

Adds a Grease Pencil object without any stroke.

Stroke

Adds a Grease Pencil object with a simple stroke as a reference.

Monkey

It creates a 2D monkey head. The Monkey's name is "Suzanne" and is Blender's mascot. 2D Suzanne is very useful as a standard test.

Scene Line Art

Sets up a Line Art Modifier for the active scene by creating an "empty" Grease Pencil object with a Line Art modifier referencing each object in the scene.

Collection Line Art

Sets up a Line Art Modifier for the active collection by creating an "empty" Grease Pencil object with a Line Art modifier referencing each object in the collection.

Object Line Art

Sets up a Line Art Modifier for the active object by creating an "empty" Grease Pencil object with a Line Art modifier referencing the active object.

Properties

- Object Properties
  - Visibility
- Data Properties
  - Layers
  - Onion Skinning
  - Settings
  - Attributes
  - Vertex Groups
  - Custom Properties

Object Properties

Visibility

Reference

Panel:: *Object Properties* ▸ *Visibility*

Use Light
: Enables the Grease Pencil object to be affected by lights.

  This property affect the whole object, for more control with lights you can enable or disable the use of lights by layers. See Layers for more information.



Lights disabled (left) and enabled (right).

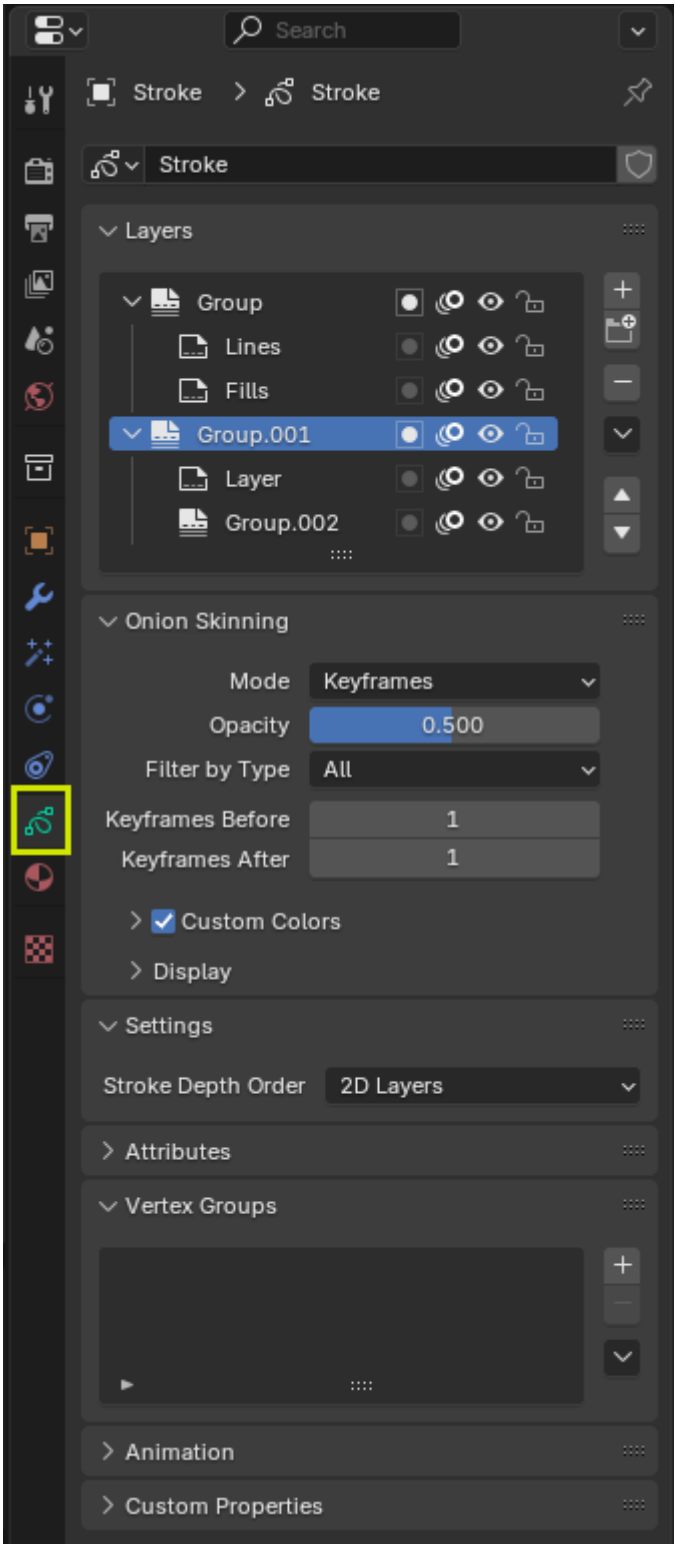
See also

There are several other general visibility properties.

Data Properties

Grease Pencil

The Grease Pencil data-block menu can be used to link the data between objects.



Grease Pencil Object Data.

Layers

Strokes can be grouped in 2D layers, a special Grease Pencil layers that help to organize the drawing order and visibility of the strokes. Layers can be organized into layer groups.

Onion Skinning

Onion skinning is used in animation to see several frames at once and make decisions or edits based on how the previous/next frames are drawn.

Settings

General settings for Grease Pencil strokes.

Attributes

Layers can store Custom Attributes. The attributes are stored on the Layer domain.

For example, the Layer Adjustments are stored as layer attributes.

Attributes

List view of all the attributes stored on the layers.

Name

Name of the layer attribute.

Data Type

The Data Type of the attribute.

Vertex Groups

Vertex groups can be used to assign a group or weighted group to some operator. An object can have several weight groups and can be assigned in

Weight Paint Mode.

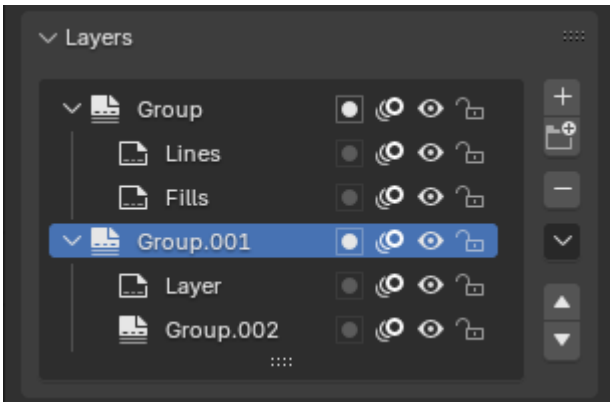
Custom Properties

Create and manage your own properties to store data in the Grease Pencil's data-block.

Layers

Reference

Mode:: All Modes
Panel:: *Object Data tab ▸ Layers*



Grease Pencil Layers panel.

Grease Pencil objects can be organized into a tree known as the layer tree for grouping and arranging strokes.

Any stroke can only belong to a single 2D layer. The selected layer is the active layer. Only one layer or group can be active at a time. When you draw, the new strokes are added to the active layer. By default the view order of the layers in the viewport is top to bottom.

Layers can be grouped using Layer Groups. A layer can only be in one group at a time. Layers can be moved into groups using drag-and-drop. Groups can be color coded with a color tag.

Every layer correspond to a channel in the Dope Sheet editor (in Grease Pencil mode). See Dope Sheet for more information.

Layers can also be used together with Modifiers to only affects part of your drawing. See Modifiers for more information.

Layers can mask other layers by enabling Use Mask (mask icon) or using the checkbox in the Masks panel header. See Masks for more information.

Tip

Sometimes the layers you are not working on can be a distraction in the 3D Viewport. Activate the Fade Inactive Layers overlay to control the opacity of the non-active layers.

Layer Tree
: Tree view of all layers and groups for the Grease Pencil object.

  Next to the layer name there are four icons buttons that control common properties of the layer:

  Use Mask (mask icon)
  : Toggle the affect of Masks on the layer.

  Onion Skinning (onion skin icon)
  : Toggle using the layer for Onion Skinning.

  Hide (eye icon)
  : Toggle layer visibility in the viewport and in render.

  Lock (padlock icon)
  : Toggle layer from being editable.

Add New Layer
: Adds a new layer to the active object.

Add New Layer Group
: Adds a new layer group to the active object. Note, layer groups cannot be added from the Dopesheet; they must be added from the Properties editor.

Remove Layer/Group
: Removes the active layer or layer group.

Layer Specials
: Operators for working with layers.

  Duplicate

Makes an exact copy of the selected layer appending a number to differentiate its name.

Duplicate Empty Keyframes
: Makes a copy of the selected layer but with empty keyframes. Useful to easily have empty keyframes preset to work on the cleanup or filling process.

Show All
: Turns on the visibility of every layer in the list.

Hide Others
: Turns off the visibility of every layer in the list except the active one.

Lock All
: Locks edition of all the layers in the list.

Unlock All
: Unlocks edition of all the layers in the list.

Autolock Inactive Layer
: Locks automatically the edition of every layer in the list except the active one. This way you avoid to make unwanted changes in other layers without the need to lock them every time.

Use Locked Materials Editing
: Avoids editing locked materials in the layer. When disabled, any material can be edited even if they are locked in the material list.

Merge Down `Shift-Ctrl-M`
: Combine the selected layer with the layer below, the new layer keeps the name of the lower layer.

Merge Group
: Combine layers in the active layer group into a single layer.

Merge All
: Combine all layers into the active layer.

Copy Layer to Selected
: Copy the active layer to the selected Grease Pencil object.

Copy All Layers to Selected
: Copy all layers to the selected Grease Pencil object.

Reorder Layer
: Moves the active layer or layer group up/down in the tree.

Below the layers list there are additional settings:

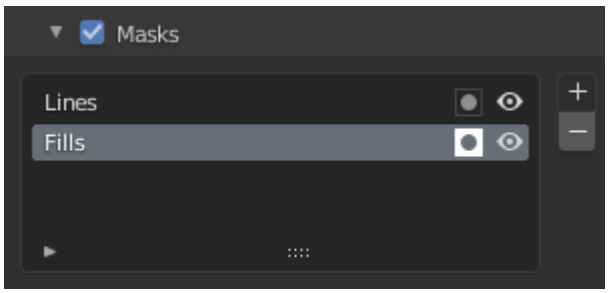
Blend Mode
: The layer blending operation to perform. See Color Blend Modes.

Opacity
: Used to set the opacity of the layer.

Lights
: When enabled, the layer is affected by lights.

Masks

In Grease Pencil there are no special mask layers, any layer can act as a mask for other layers. The mask system is flexible enough to allow top-bottom and bottom-top masking.



Masks panel.

Layers used as masks can use all the blend modes and different opacity values like any other layer.

Tip

If you want to make a full transparent masking you will have to set the mask layer's opacity to 0.

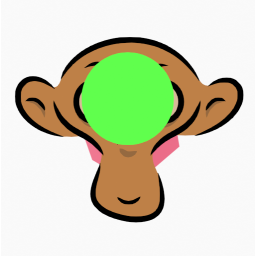
The layer/s that will act as mask of the current layer could be added to the Mask list view.

In the Masks list next to the layers name there are two icons buttons that control common properties of the layer mask:

Invert (mask icon)
Inverts the mask.

Viewport/Render Visibility (eye icon)
Toggle layer visibility in the viewport and in render.

Mask (green circle) samples.



Original image (Blend: Regular, Opacity: 1).



Blend: Hard Light, Opacity: 1.



Blend: Regular, Opacity: 1.

Transform

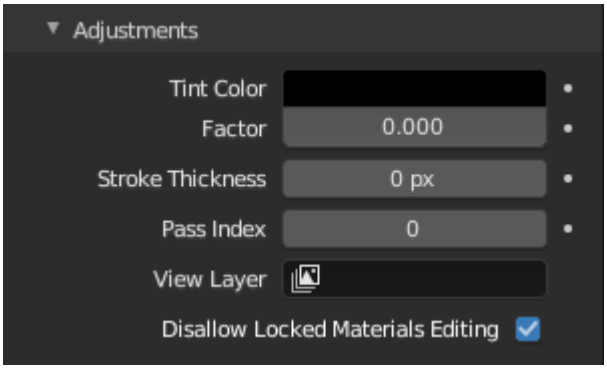
Allows per-layer location, rotation and scale transformations.

Adjustments

Tint Color
: Color that tint any material colors used in the layer.

Factor
: Controls the amount of tint color to apply.

Stroke Thickness
: Thickness value that override the strokes thickness in the layer.



Layers adjustment panel.

Relations

Parent
: Select a Parent object to manipulate the layer. The layer will inherit the transformations of the parent, this is especially useful when rigging for cut-out animation.

Pass Index
: The layer index number can be used with some modifiers to restrict changes to only certain areas.

  See Modifiers for more information.

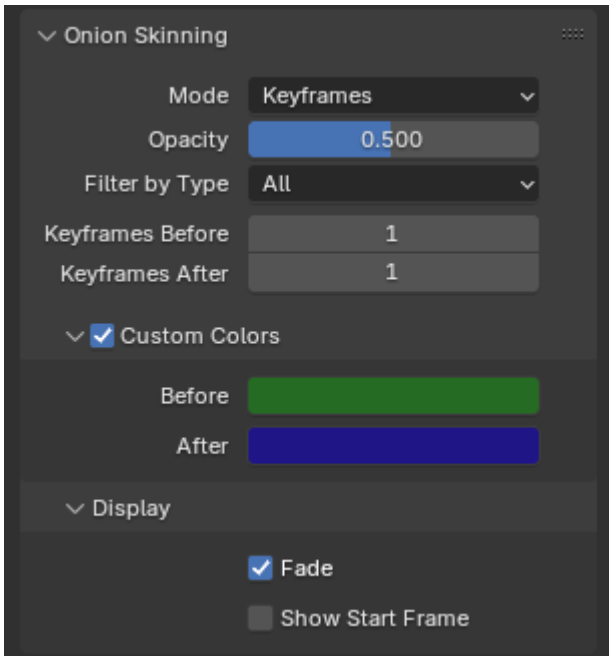
View Layer
: Defines the View Layer to use for the Grease Pencil layer. If empty, the layer will be included in all View Layers. This is useful to separate drawings parts for compositing.

Use Masks in Render
: If disabled, no masks on the layer are included in the view layer render.

Onion Skinning

Onion Skinning show ghosts of the keyframes before and after the current frame allowing animators to make decisions in the animation sequence.

The main switch to show/hide Onion Skinning is in the Viewport Overlays, but Grease Pencil Onion Skinning is per-layer and the visibility can be toggle in the layer list. See 2D Layers for more information.



Onion Skinning panel.

Options

Mode

Keyframes:: Shows Keyframes in the range determined by the *Before*/*After* settings.

Frames:: Shows Frames in the range determined by the *Before*/*After* settings.

Selected::

Shows only on the manually selected keyframes in the Dope Sheet.

Opacity
: Control the opacity of the ghost frames.

Filter by Type
: Filters what type of frames to show in the Onion Skinning range.

Keyframes Before/After
: Sets how many frames or keyframes, depending on the *Mode*, to show before and after the current frame.

Custom Colors

Before/After
: Color to use before and after the current frame on ghost frames.

Display

Fade
: Opacity of the ghosts frames decrease the further away from the current frame.

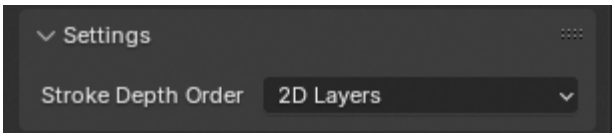
Show Start Frame
: Help working on loop animations showing the first keyframe/frame as ghost when you are on the last frame of your animation.



An example of Onion Skinning activated.

Settings

General settings for Grease Pencil strokes.



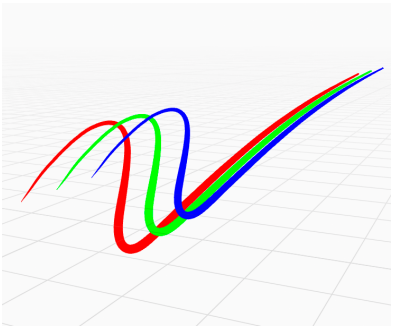
Strokes panel.

Stroke Depth Order

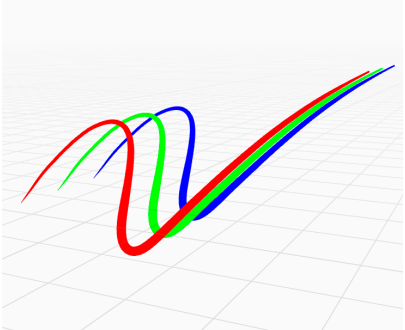
Defines how the strokes are ordered in 3D space (for objects not displayed *In Front*).

2D Layers:: The Strokes drawing order respect the order of the 2D layers list (top to bottom) and ignores the real position of the strokes in 3D space. See 2D Layers for more information.

3D Location:: The strokes drawing order is based on the stroke location in 3D space.



Blue, Green and Red strokes in three different layers using 2D Layers depth order.



Blue, Green and Red strokes in three different layers using 3D Location depth order.

Grease Pencil Modifiers

- Introduction
  - Interface
- Generate
  - Array Modifier
  - Build Modifier
  - Dot Dash Modifier
  - Envelope Modifier
  - Length Modifier
  - Line Art Modifier
  - Mirror Modifier
  - Multiple Strokes
  - Outline Modifier
  - Simplify Modifier
  - Subdivide Modifier
- Deform
  - Armature Modifier
  - Hook Modifier
  - Lattice Modifier
  - Noise Modifier
  - Offset Modifier
  - Shrinkwrap Modifier
  - Smooth Modifier
  - Thickness Modifier
- Color
  - Hue/Saturation Modifier
  - Opacity Modifier
  - Tint Modifier
- Edit
  - Texture Mapping Modifier
  - Time Offset Modifier
  - Vertex Weight Angle Modifier
  - Vertex Weight Proximity Modifier

Introduction

Reference

Panel:: *Properties* ▸ *Modifiers*

Grease Pencil has their own set of modifiers. Modifiers are automatic operations that affect an object in a non-destructive way. With modifiers, you can perform many effects automatically that would otherwise be too tedious to do manually and without affecting the base geometry of your object.

With Geometry Nodes, it is possible to create custom Grease Pencil modifiers.

They work by changing how an object is displayed and rendered, but not the geometry which you can edit directly. You can add several modifiers to a single object forming the modifier stack and Apply a modifier if you wish to make its changes permanent.

There are four types of modifiers for Grease Pencil:

Edit
: These are tools similar to the Deform ones (see below), however, they usually do not directly affect the geometry of the object, but some other data, such as vertex groups.

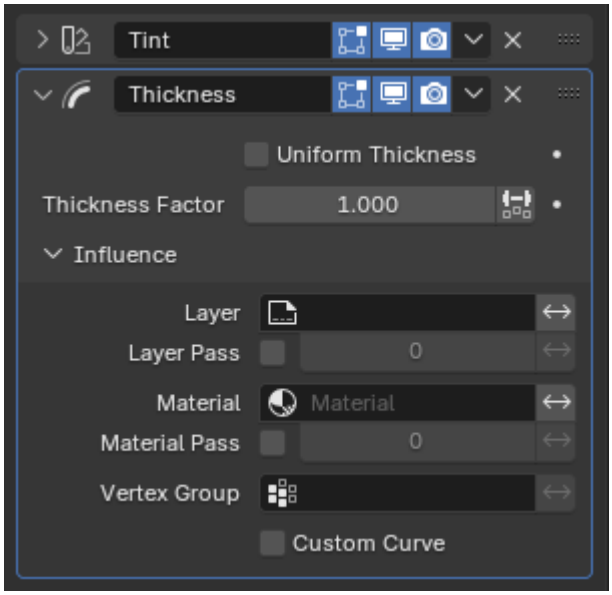
Generate
: The *Generate* group of modifiers includes constructive tools that either change the general appearance of or automatically add new geometry to an object.

Deform
: The *Deform* group of modifiers only changes the shape of an object without adding new geometry,

Color

The *Color* group of modifiers change the object color output.

Interface



Panel layout (Thickness modifier as an example).

Each modifier's interface shares the same basic components like modifiers for meshes.

See Modifiers Interface for more information.

Note

Grease Pencil strokes, unlike meshes, still can not be edited directly in the place.

Applying Modifiers

Applying a modifier makes the effects of the modifier "real"; converts the strokes to match the applied modifier's results, and deletes the modifier.

When applying a modifier to an object that shares Object Data between multiple objects, the object must first be made a Single User which can be performed by confirming the pop-up message.

Warning

Applying a modifier that is not first in the stack will ignore the stack order (it will be applied as if it was the first one), and may produce undesired results.

Reference

Panel:: *Properties* ▸ *Modifiers* ▸ *Modifier Header* ▸ *Specials*

Apply (Active Keyframe) `Ctrl-A`
: Applies the modifier for the current keyframe.

Apply (All Keyframes)
: Applies the modifier for all keyframes.

Note

With Geometry Nodes it is possible to add new layers to the geometry. When applying, this will create a single keyframe on the first frame of evaluation. Layers with duplicated names in evaluated geometry will be deduplicated.

It is also possible to have layers with empty names. When applying these get renamed to *Layer* (and *Layer.001* etc. when such a layer already exists in the original geometry).

Influence Filters

Most of the modifiers share some special properties that restrict the effect only to certain items.

Layer
: Restricts the effect only to one layer or to any layers that share the same material Pass Index.

Material
: Restricts the effect only to material that share the same material or material Pass Index.

Vertex Group
: Restricts the effect only to a vertex group.

Custom Curve
: When enabled, use a custom curve to shape the effect along the strokes from start to end points.

The Invert toggle <–> allows you to reverse the filters behavior.

Grease Pencil Generate Modifiers

- Array Modifier
- Build Modifier
- Dot Dash Modifier
- Envelope Modifier
- Length Modifier
- Line Art Modifier
- Mirror Modifier
- Multiple Strokes
- Outline Modifier
- Simplify Modifier
- Subdivide Modifier

Array Modifier

The *Array* modifier creates an array of copies of the base object, with each copy being offset from the previous one in any of a number of possible ways.

Useful for creating complex repetitive drawings.

Multiple Array modifiers may be active for an object at the same time (e.g. to create complex three-dimensional constructs).

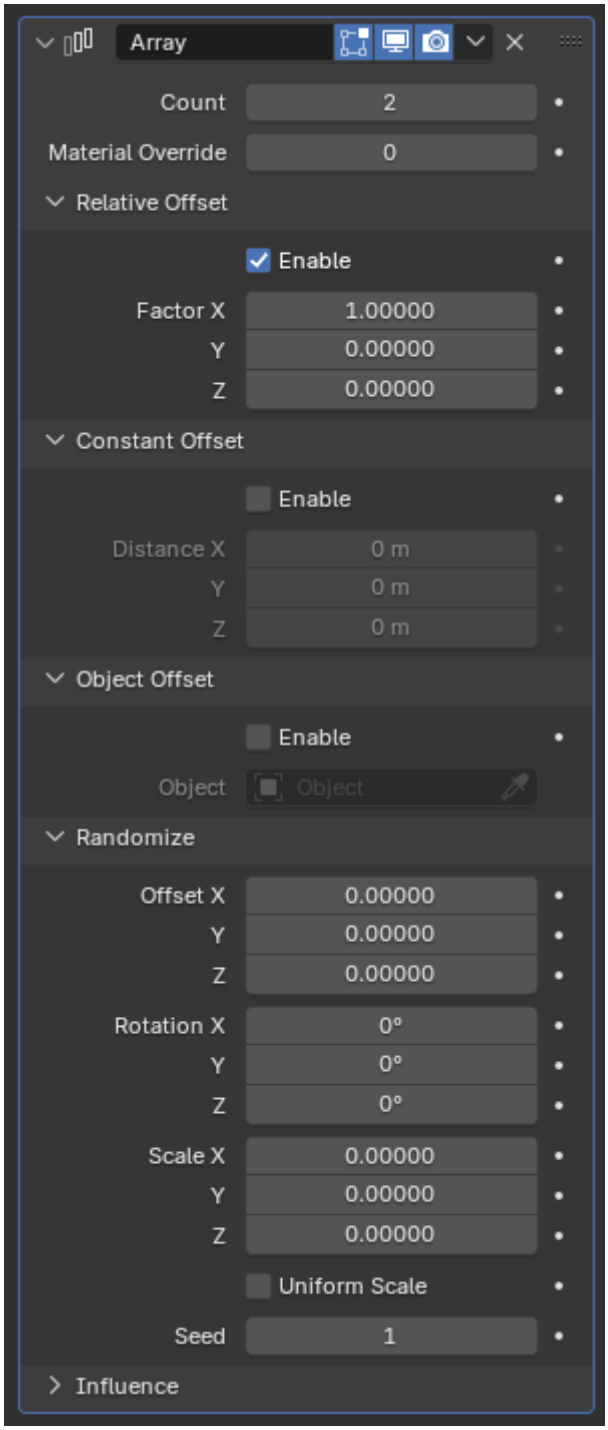
See also

This documentation refers to the Array Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Array Modifier.

Options

Count
: Total number of copies.

Material Override
: Index of the material to use on duplicated strokes (0 use strokes original materials).



The Array modifier.

Relative Offset

Factor X, Y, Z
: Adds a translation equal to the object's bounding box size along each axis, multiplied by a scaling factor, to the offset. X, Y and Z scaling factors can be specified.

Constant Offset

Factor X, Y, Z
: Adds a constant translation component to the duplicate object's offset. X, Y and Z constant components can be specified.

Object Offset

Distance X, Y, Z
: Adds a transformation taken from an object (relative to the current object) to the offset. It is good practice to use an empty object centered or near to the initial object.

Randomize

Offset X, Y, Z
: Add random offset values to the copies.

Rotation X, Y, Z
: Add random rotation values to the copies.

Scale X, Y, Z
: Add random scale values to the copies.

Uniform Scale
: Use the same random *Seed* for each scale axis in the copies for a uniform scale.

Seed
: Seed used by the pseudo-random number generator.

Note

The *Depth Order* used in the Grease Pencil object has an influence on the visualization of the strokes when using the Array modifier. See Depth Order for more information.

Influence

See Influence Filters.

Build Modifier

The *Build* modifier make strokes appear or disappear in a frame range to create the effect of animating lines being drawn or erased.

See also

This documentation refers to the Build Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Build Modifier.

Options

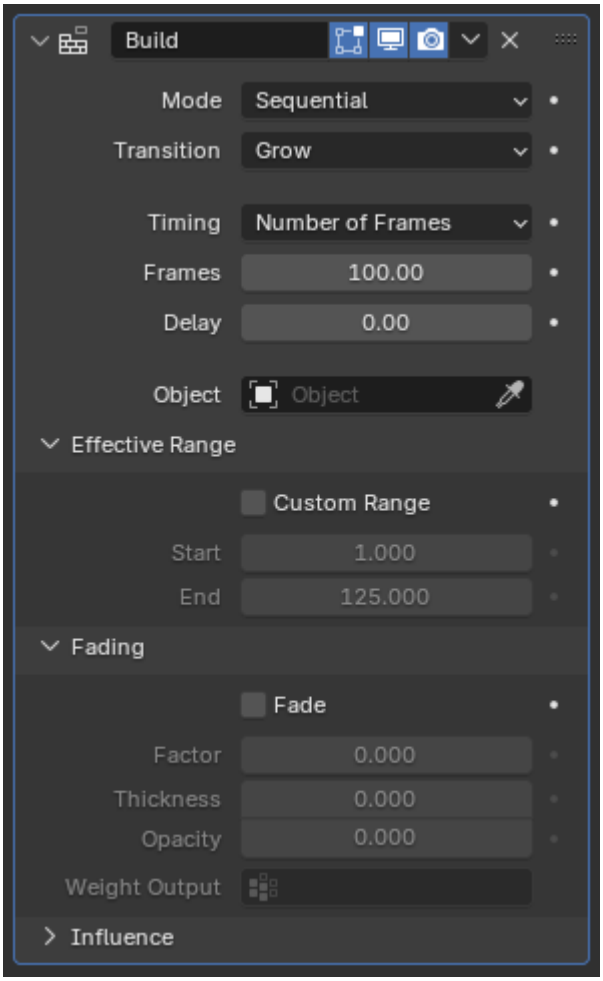
Mode
: Determines how many strokes are being animated at a time.

  Strokes appear/disappear one after the other, but only a single one changes at a time.
  Multiple stroke appear/disappear at a time.
  Builds only the strokes that are new compared to last keyframe. The assumption is Additive Drawing was used so that the shared strokes are the same.

Transition *(in Sequential and Concurrent Mode)*
: Determines the animation type to build the strokes.

  Shows points in the order they occur in each stroke, from first to last stroke. (Simulating lines being drawn.)
  Hide points from the end of each stroke to the start, from last to first stroke. (Simulating lines being erased.)

Sequential::
Concurrent::
Additive::
Grow::
Shrink::
Vanish::



The Build modifier.

Hide points in the order they occur in each stroke, from first to last stroke. (Simulating ink fading or vanishing after getting drawn.)

Timing

The way you want to time the building of the strokes.

Natural Drawing Speed::

Use the recorded speed of the stylus when the strokes were drawn. Only available in Sequential and Additive Mode.
Speed Factor

The recorded speed is multiplied by this value.

Maximum Gap
: The maximum gap between strokes in seconds.

Number of Frames::
: Set a fixed maximum number of frames for the build animation. (Unless another Grease Pencil keyframe occurs before this time has elapsed.)

Frames
: The maximum number of frames used.

Delay
: Number of frames after each Grease Pencil keyframe before the modifier has any effects.

Percentage Factor:: Manually set a percentage factor to control the amount of the strokes that are visible.

Factor
: The factor from 0 to 1.

Time Alignment:: Only available in Concurrent Mode.

Align Start
: All stroke start at the same time (i.e. shorter strokes finish earlier).

Align End
: All stroke end at the same time (i.e. shorter strokes start later).

Object
: Use the distance to an object to define the order in which strokes appear.

Custom Range

If enabled, only modify strokes during the specified frame range.

Start, End
: Determines the start and end frame for the build effect.

Fade

Factor
: Defines how much the stroke is fading in/out.

Thickness
: How much strength fading is applied to the stroke's thickness.

Opacity
: How much strength fading applies to the stroke's opacity.

Weight Output
: Assign a weight value to points that have started/finished the fade.

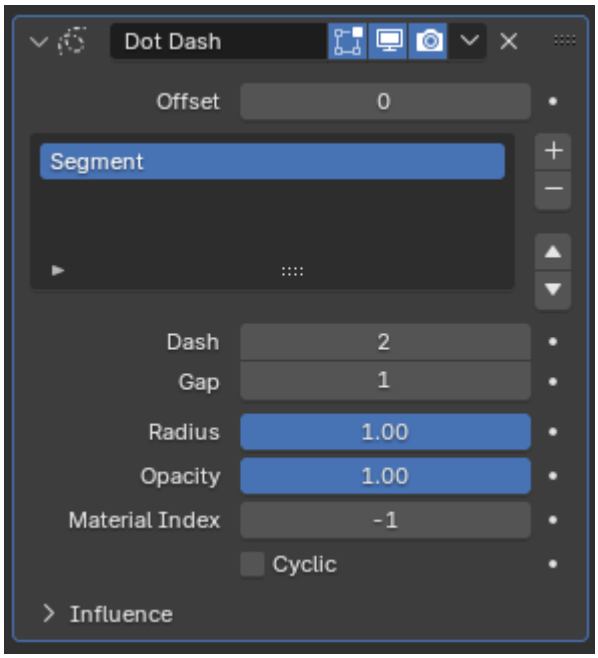
Influence Filters

See Influence Filters.

Dot Dash Modifier

The *Dot Dash* modifier generates dot dash segments from the original stroke.

Options



The Dot Dash modifier.

Offset
: Determines the starting offset of the pattern.

Segment
: Makes up individual stroke of a dot dash pattern.

  Use the plus/minus button on the side of the list to add/remove segments.

Dash
: The number of consecutive points from the original stroke to include in this segment.

Gap
: The number of points skipped after the segment ends.

Radius
: The factor to apply to the original point's radius for the new points.

Opacity
: The factor to apply to the original point's opacity for the new points.

Material Index
: Use this index on generated segment, use -1 for existing material.

Use Cyclic
: Close the segment.

Influence Filters

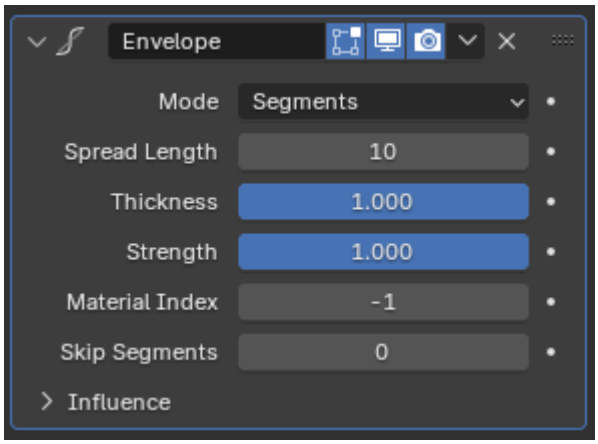
See Influence Filters.

Envelope Modifier

The *Envelope* modifier creates a shape known as envelope over the existing strokes connecting all the points that have n points between them.

Options

Mode



The Envelope modifier.

Deform:: Replaces the original stroke with the envelope shape.

Segments:: Add segments to create the envelope shape keeping the original stroke.

Fill:: Add segments to create the envelope without the original stroke.

Spread Length

The number of points to skip when creating the straight segments that define the envelope.

Thickness

The thickness of the generated stroke segments.

Strength
: The Opacity of the generated stroke segments.

Material Index
: Defines the material to use on the generated stroke segments.

Skip Segments
: The Number of generated stroke segments to skip to reduce complexity.

Influence Filters

See Influence Filters.

Length Modifier

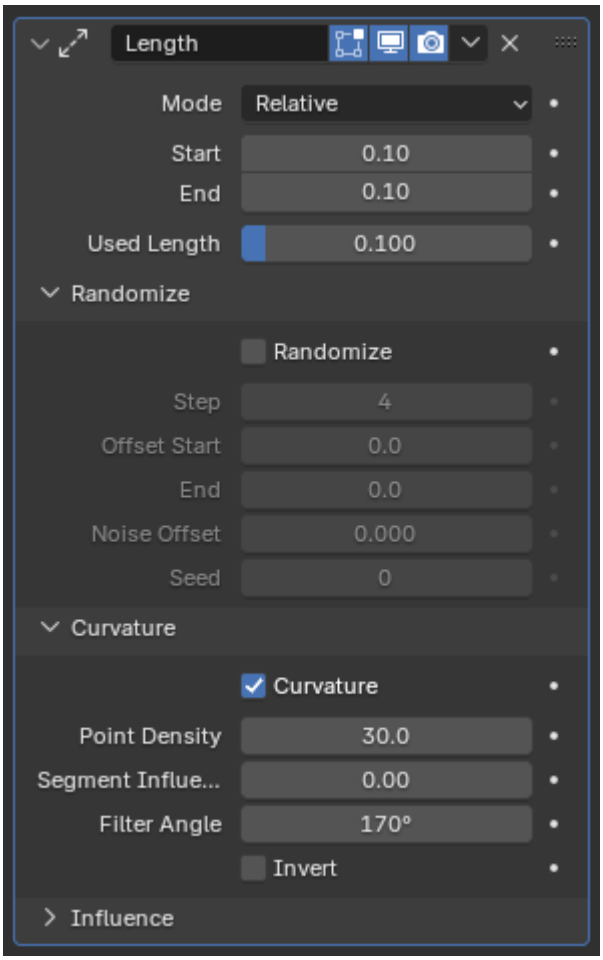
The *Length* modifier can shrink or extend strokes.

Options

Mode

Absolute:: Length is in geometry space.

Relative:: Length is in ratio to the stroke's length.



The Length modifier.

Start

Added length to the start of the stroke. Negative value will shrink the stroke.

End
: Added length to the end of the stroke. Negative value will shrink the stroke.

Used Length
: Define what portion of the stroke is used to calculate the direction of the extension.

Curvature

When enabled, the extension will follow the curvature of the stroke.

Point Density
: Multiplied by Start/End for the total point count.

Segment Influence
: Factor to determine how much the length of the individual segments should influence the final computed curvature. Higher factors makes small segments influence the overall curvature less.

Filter Angle
: Ignore points on the stroke that deviate from their neighbors by more than this angle when determining the extrapolation shape.

Invert
: Invert the curvature of the stroke's extension.

Random Offsets

Random Offset Start/End
: Size of random length added to the start/end of each stroke.

Random Noise Offset
: Smoothly offset each stroke's random value.

Seed
: Number used to generate different noise patterns.

Randomize

Re-randomizes values over time.

Step
: Number of frames before recalculate random values again.

Influence Filters

See Influence Filters.

Line Art Modifier

The *Line Art* modifier generates stylized line art from a scene, collection, or object.

In order for the effects of the modifier to be visible, the scene must have an active camera. The generated lines are only generated on portions of the object that a visible from this camera.

Note

Due to lack of global cache at the moment, each Line Art modifier will run the entire occlusion calculation for itself. So if you have multiple line art modifiers to select different parts of the scene (to apply different styles, etc.), the evaluation will take much longer. There are plans to remedy this in the future, but this is a known limitation for now.

Options

Use Cache

Optimize rendering by using cached scene data from the first line art modifier in the stack. This option has the disadvantage of certain settings becoming unavailable.

This option is only available when you have more than one Line Art modifier in the same modifier stack and the modifier is not the first Line Art modifier in the stack.

Source Type

What type of geometry source should line art be generated from.

Scene, Collection, Object

Object/Collection
: Based on the source type, collection or object can be selected as source geometry.

  Invert Collection Filtering
  : Select everything except lines from specified collection.

Note

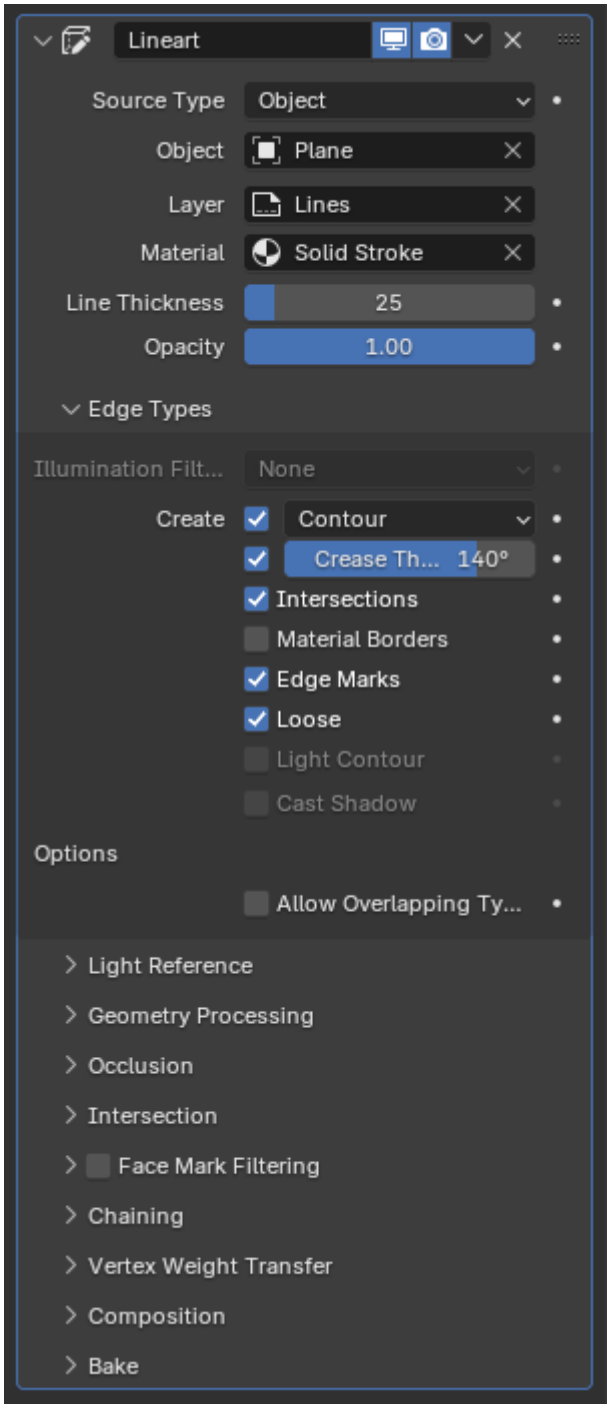
Line Art will still load and calculate the entire visible scene to produce correct occlusion result, unless specified to do otherwise in object or collection Line Art *Usage* property.

Layer
: The Grease Pencil Layers to put the result in.

Material
: The Grease Pencil Grease Pencil Materials to generate strokes with.

Line Thickness
: The strokes generated by line art are given this thickness.

Opacity
: The strokes generated by line art are given this Opacity.



The Line Art modifier.

Edge Types

Line Art can identify different edge types. Selected edge types will be included in the result.

Illumination Filtering

Select feature lines that comes from lit or shaded regions. Will not affect cast shadow and light contour since they are at the border.

None:: Not filtering any lines based on illumination region.

Illuminated:: Only selecting lines from illuminated regions.

Shaded:: Only selecting lines from shaded regions.

Illuminated (Enclosed Shapes):: Selecting lines from lit regions, and make the combination of contour, light contour and shadow lines into enclosed shapes.

Create

Contour

Generate strokes from contour lines. Where the edge becomes the separation line of front/backfacing faces. The silhouette can also be inverted by clicking the invert button.

Contour:: Generate lines from contour.

Silhouette:: Only generate lines from the silhouette of the source objects as a whole.

Individual Silhouette:: Generate lines from the individual silhouettes of the source objects.

Crease

Generate strokes where the edge angle is small enough.

Crease Threshold

Angles smaller than this will be treated as creases. Crease angle priority: object line art crease override > line art default crease.

Intersections
: Generate strokes where lines intersect between faces.

Material Borders
: Generate strokes where the edge separates faces with different materials.

Edge Marks
: Generate strokes from freestyle edge marks.

Loose
: Generate strokes for edges that do not form a Face.

Light Contour
: Generate light/shadow separation lines from a reference Light Object.

Cast Shadow
: Project contour lines using a light source object.

Options

Allow Overlapping Types
: Allow an edge to have multiple overlapping types. This will create a separate stroke for each overlapping type.

Light Reference

Light Object
: Use this light object to generate Light Contour.

Shadow Camera Size
: This value represents the “Orthographic Scale” of an ortho camera. If the camera is put at the lamps position with this scale, it will represent the coverage of the shadow “camera”.

Near
: Near clipping distance of shadow camera.

Far
: Far clipping distance of shadow camera

Geometry Processing

Custom Camera
: Use custom camera instead of the active camera for calculating strokes. Useful when baking multiple shots in different angles as well as for motion graphics effects.

Overlapping Edges as Contour
: This option allows overlapping edges (e.g. from an Edge Split modifier or imported geometry where two edges occupy the exact same space) to be drawn as contour. Enabling this option will slow down the calculation slightly but it will handle edge overlapping cases without erroneous occlusion results.

Instanced Objects
: This option enables particles and other instanced objects to be loaded for line art calculation. There will be performance impact when there are a large amount of instanced objects in the scene.

Clipping Boundaries
: When enabled, line art will generate clipping lines as contour type at the place where near or far clipping planes cut the model. Otherwise there will be no lines.

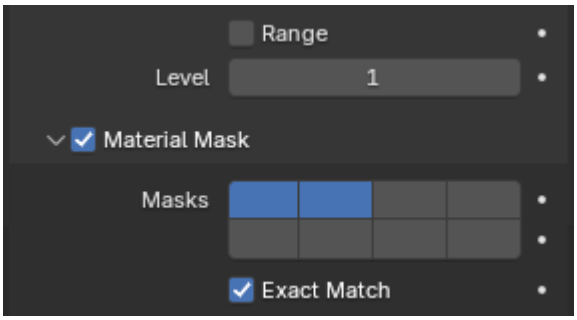
Crease on Smooth
: Allow crease edges to show inside smooth surfaces.

Crease on Sharp
: Allow creases to show on sharp edges.

Force Backface Culling

Remove all back faces to speed up calculation. Note, removing back faces will create edges in different occlusion levels than when disabled.

Occlusion



Occlusion subpanel.

Range
: If enabled, the modifier will select lines that have an occlusion level between start and end values.

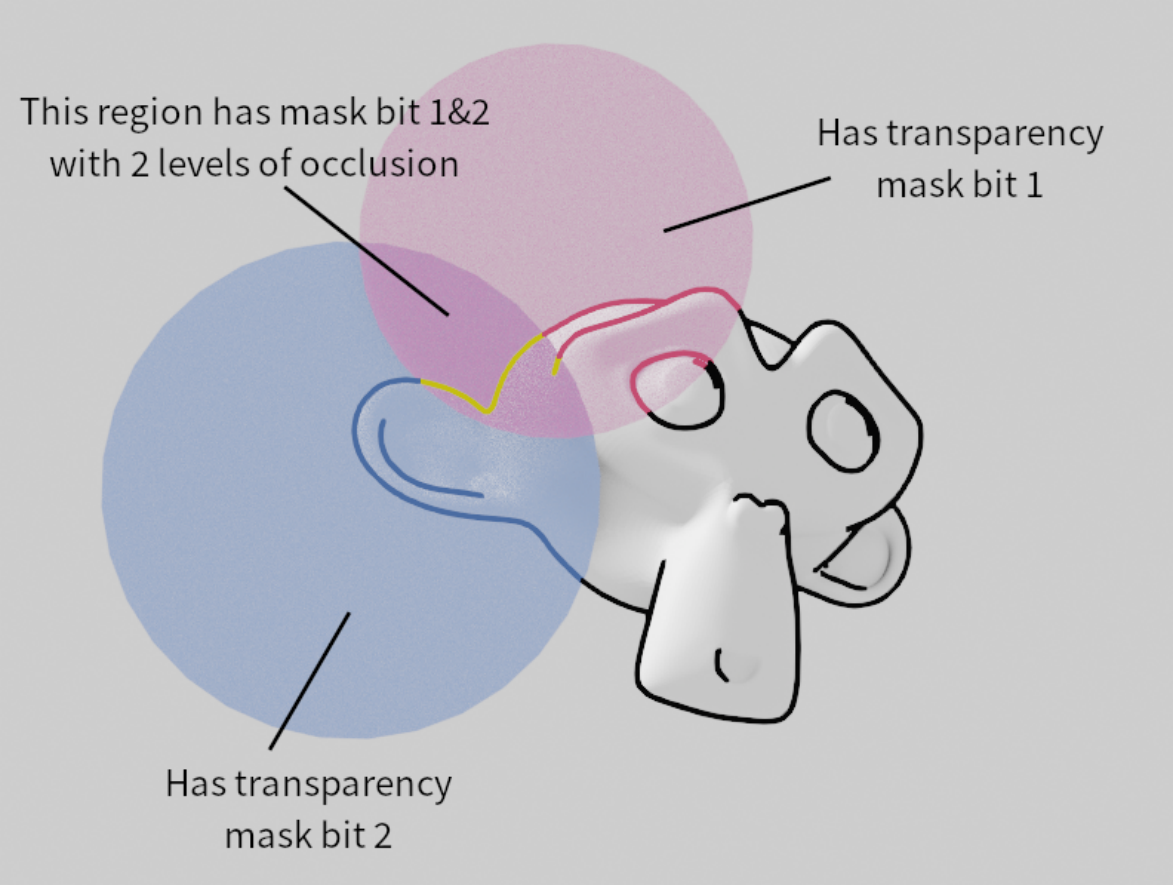
Level
: Desired occlusion level to be selected as line art result. A value of 0 means visible lines (no occlusion). A value of 1 means selecting lines that have been occluded by exactly one layer of faces.

Material Mask

If enabled, Line Art will only select lines that are occluded by certain faces whose material have specific occlusion masks set.

Masks
: To select edges that have been occluded by the selected Material Mask.

Exact Match
: If enabled, only lines that are occluded with the exact mask bit combination will be selected. Otherwise, lines that have been occluded by any one of specified material masks will be selected.



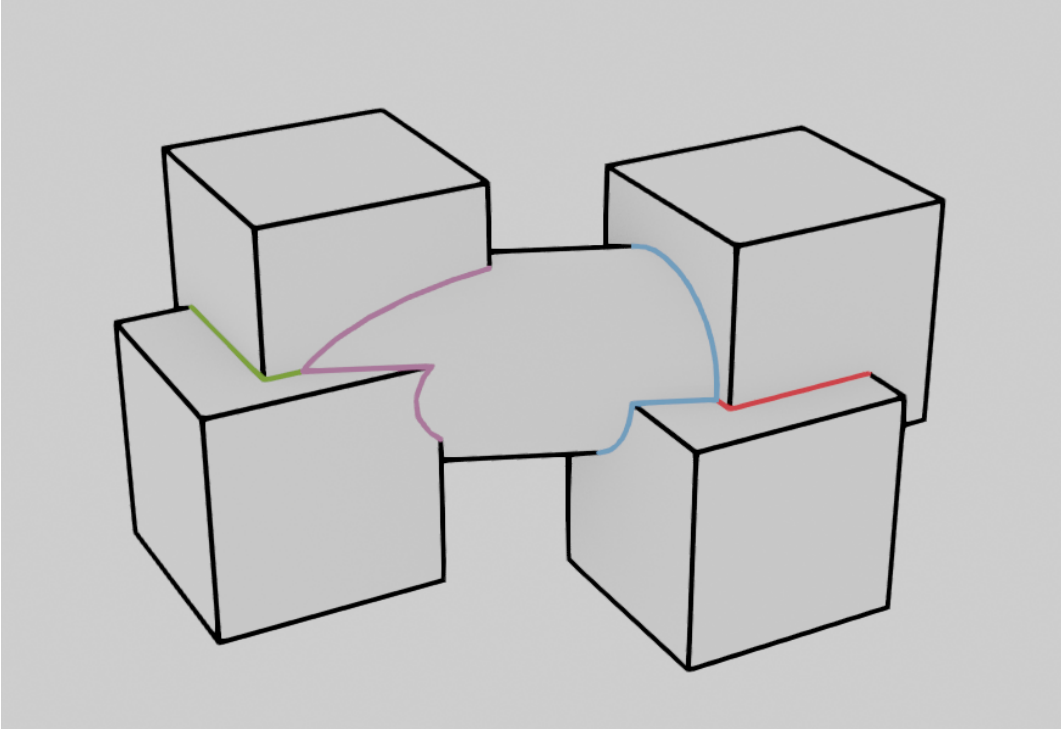
Demonstration of the usage of material masks.

Intersection

Allows you to select edges that intersect between two collections.

Collection Mask
: Mask bits to match from Collection Line Art properties.

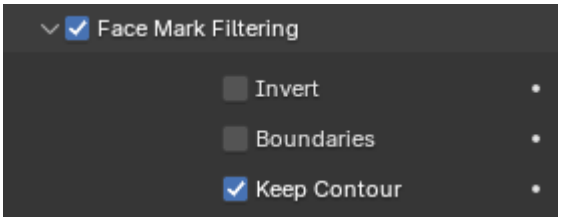
Exact Match
: Require matching all intersection masks instead of just one.



Demonstration of the usage of collection masks.

Face Mark Filtering

Face Mark Filtering can be used to have manual control over which feature edges produce strokes by using Freestyle face marks.

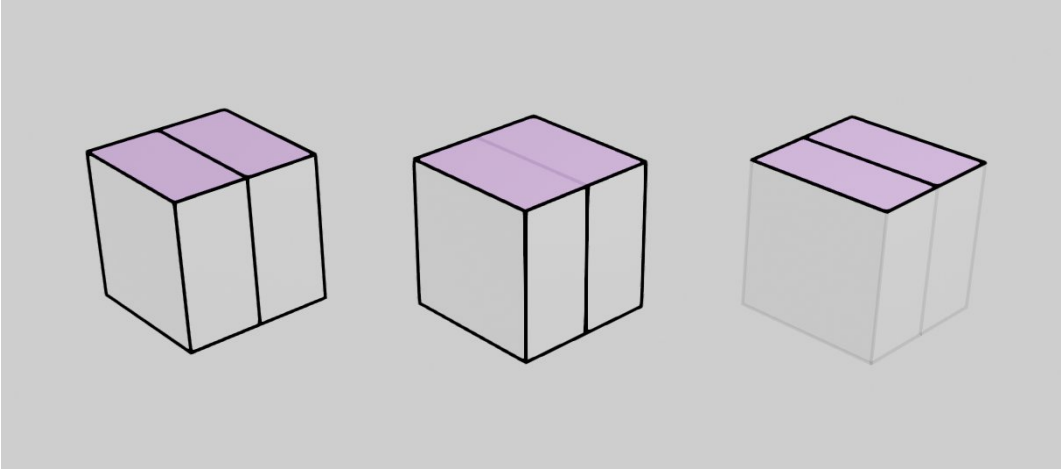


Face Mark Filtering subpanel.

Invert
: Invert face mark filtering.

Boundaries
: Filter feature lines based on face mark boundaries.

Keep Contour
: Preserve contour lines while filtering.



Chaining

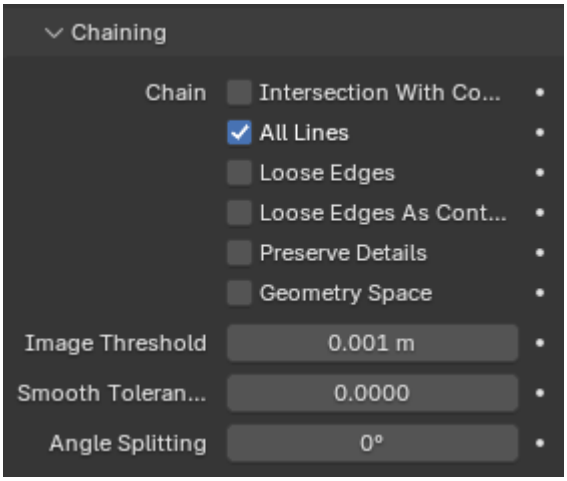
Chain

Intersection with Contour

Allows intersection lines to be chained together with contour lines.

Note

Enabling this option will lead to ambiguity in intersection edge types. Intersection lines that have not been able to chain with any nearby contour lines will remain as intersection lines.



Chaining subpanel.

All Lines

Enabling this option will cause all lines to have the type of contour and to be chained together.

Loose Edges

Allow floating Edges that do not form a face to be chained together.

Loose Edges as Contour
: Edges that do not form a face will be classified as contour lines.

Preserve Details
: Instead of splitting at each occlusion change, keep small details from the initial chain. When details are not kept, will create a much smoother result.

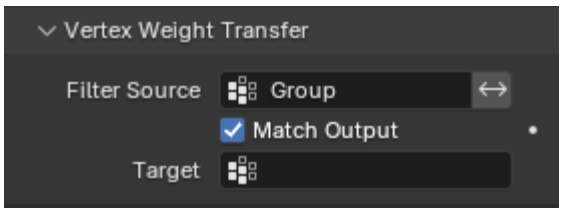
Geometry Space
: Use geometry distance for chaining instead of image space.

Image Threshold
: Allow the end point of short segments to be chained together if the 2D image space distance between them are within the specified threshold.

Smooth Tolerance
: The strength of smoothing applied on jagged chains.

Angle Splitting
: Split a chain at sharp “turning” points specified by this angle.

Vertex Weight Transfer



Vertex Weight Transfer subpanel.

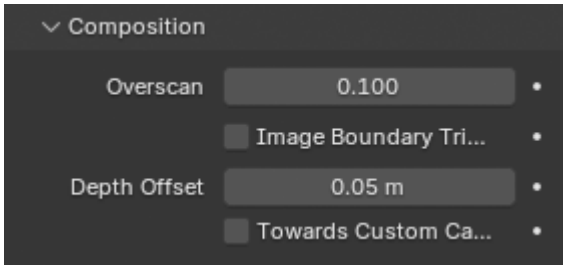
Filter Source
: If source mesh has vertex groups whose name starts with this text, then the vertex weight info will be transferred into weight groups in Grease Pencil strokes.

Match Output
: Transfer the filtered object vertex weights into Grease Pencil weight groups with the same names as the filtered ones.

Target
: If *Match Output* is off, then a target vertex group has to be specified. If there are multiple weight groups copied into target, then the highest

weight value is copied into it.

Composition



Composition subpanel.

Overscan
: To optimize rendering, Blender only renders the strokes for edges of the object that are in the camera's view. This optimization however, can result in strokes ending abruptly at the edge of the image.

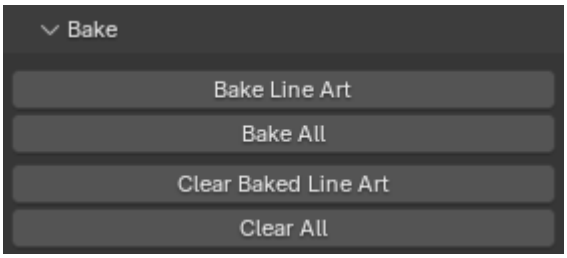
  This value prevents this error by adding a margin outside the camera's view to continue computing strokes.

Image Boundary Trimming
: Trim all stroke right at the boundary of image (including overscan region).

Depth Offset
: Move strokes slightly towards the camera to avoid clipping while preserve depth for the viewport. This option is unavailable unless Show in Front is disabled.

Towards Custom Camera
: Offset strokes towards selected camera (see *Custom Camera* above) instead of the active camera.

Bake



Bake Line Art
: Bakes Line Art strokes for active Grease Pencil object within the *start*, *end* frame range in scene. After baking, baked Line Art

modifiers will be deactivated automatically.

Bake options.

Bake All

Bakes all Grease Pencil objects that contains at least one Line Art modifier. After baking, baked Line Art modifiers will be deactivated automatically.

Clear Baked Line Art

Clears baked line art frames within the scene frame range for active Grease Pencil object.

Clear All

Clears baked line art for all Grease Pencil objects that contains at least one Line Art modifier.

Warning

If you have drawn anything manually in the frame range of where line art runs, this operation will also clear those strokes!

Continue without Clearing

Re-activate a specific Line Art modifier without clearing baked strokes. This is useful for working on multiple portions of frames separately.

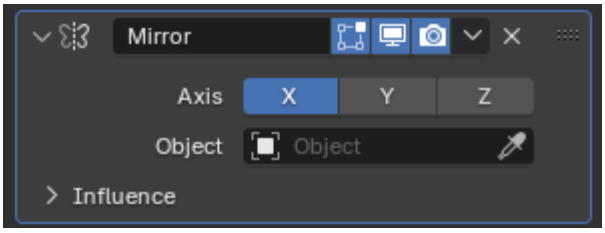
Mirror Modifier

The *Mirror* modifier mirrors the strokes along its local X, Y and/or Z axes, across the Object Origin. It can also use another object as the mirror center, then use that object's local axes instead of its own.

See also

This documentation refers to the Mirror Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Mirror Modifier.

Options



The Mirror modifier.

Axis
: The X, Y, Z axis along which to mirror, i.e. the axis perpendicular to the mirror plane of symmetry.

  To understand how the axis applies to the mirror direction, if you were to mirror on the X axis, the positive X values of the original stroke would become the negative X values on the mirrored side.

  You can select more than one of these axes. And will then get more mirrored copies. With one axis you get a single mirror, with two axes four mirrors, and with all three axes eight mirrors.

Object
: A Data ID to select an object (usually an empty), which position and rotation will be used to define mirror planes (instead of using the ones

from the modified object).

Influence

See Influence Filters.

Multiple Strokes

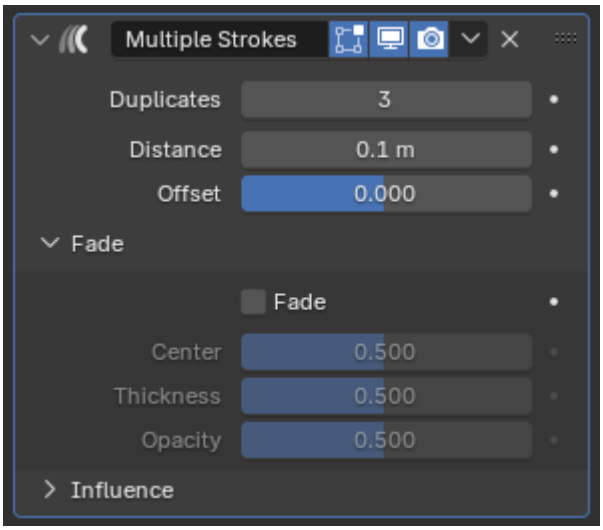
The *Multiple Strokes* modifier generate multiple parallel strokes around the original ones.

Options

Duplicates
: The number of additional strokes.

Distance
: Distance between the original and the duplicate strokes.

Offset
: Control the offset position (inner or outer) for duplicate strokes.



The Multiple Strokes modifier.

Fade

When activated, the duplicate strokes fades out using their opacity or thickness.

Center
: Control the initial position for the fading.

Thickness
: Fade influence on strokes thickness.

Opacity

Fade influence on strokes opacity.

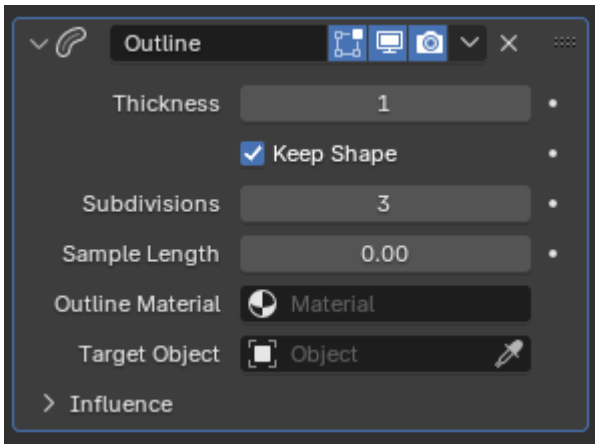
Influence

See Influence Filters.

Outline Modifier

The *Outline* modifier convert strokes to outline tracing all strokes perimeter with new strokes.

Options



The Outline modifier.

Thickness
: The thickness of the generated strokes outline.

  Keep shape
  : The perimeter strokes are maintaining inside the original stroke perimeter trying to keep the original shape.

Subdivisions
: controls the number of subdivision of the generated strokes outline.

Sample Length
: Controls the accuracy of the perimeter conversion.

Outline Material
: Defines the material to use on the generated strokes outline.

Target Object
: Controls the origin of the cyclic strokes generated.

Influence Filters

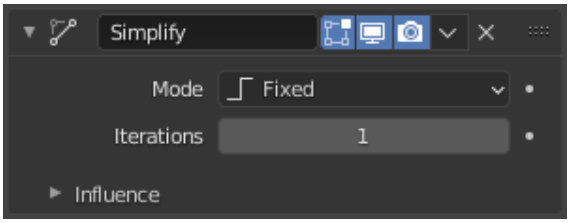
See Influence Filters.

Simplify Modifier

The *Simplify* modifier allows you to reduce the amount of points in the strokes. The goal of this modifier is reduce points while maintaining the lines shape.

Apply the modifier can help to obtain a better performance (more FPS) while animating.

Options



The Simplify modifier.

Mode
: Determines how to reduce points in the strokes.

Fixed:: Deletes alternated points in the strokes, except the start and end points.
Iterations
: Number of times to repeat the procedure.

Adaptive:: Uses the RDP algorithm (Ramer-Douglas-Peucker algorithm) for points deletion. The algorithm try to obtain a similar line shape with fewer points.
Factor
: Controls the amount of recursively simplifications applied by the algorithm.

Sample:: Recreates the stroke geometry with a predefined length between points.
Length
: The distance between points on the recreated stroke. Smaller values will require more points to

recreate the stroke, while larger values will result in fewer points needed to recreate the curve.

Sharp Threshold
: Preserve corners that have sharper angle than this threshold.

Merge:: Simplifies the strokes by merging points that are closer than a specified distance to each other.

Distance
: Sets the distance threshold for merging points.

Influence

See Influence Filters.

Example

Fixed Mode sample.



Original Model.



Iteration: 1.



Iteration: 2.



Iteration: 3.

Adaptive Mode sample.



Original Model.



Factor: 0.1.



Factor: 0.5.



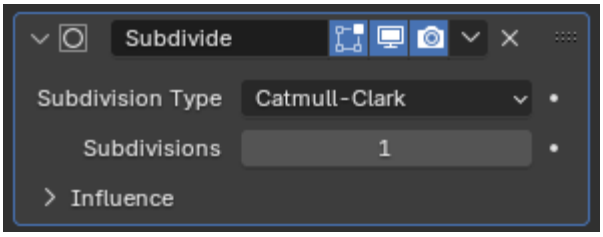
Factor: 1.

Subdivide Modifier

The *Subdivide* modifier subdivide the strokes by inserting points between other points to the lines.

Options

Subdivision Type
: **Catmull-Clark::** Subdivides and smooths the surfaces.
: **Simple::** Only subdivides the surfaces, without any smoothing.



The Subdivide modifier.

Subdivisions
: Recursively adds more points.

Influence

See Influence Filters.

Grease Pencil Deform Modifiers

- Armature Modifier
- Hook Modifier
- Lattice Modifier
- Noise Modifier
- Offset Modifier
- Shrinkwrap Modifier
- Smooth Modifier
- Thickness Modifier

Armature Modifier

The *Armature* Modifier is used for building skeletal systems for animating the poses of characters and anything else which needs to be posed.

By adding an armature to an object, this object can be deformed accurately so that geometry does not have to be animated by hand.

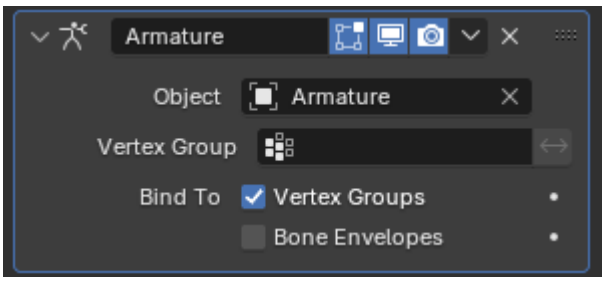
See also

For more details on armatures usage, see the armature section.

See also

This documentation refers to the Armature Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Armature Modifier.

Options



The Armature modifier.

Object
: The name of the armature object used by this modifier.

Vertex Group
: The name of a vertex group of the object, the weights of which will be used to determine the influence of this Armature Modifier's result when mixing it with the results from other *Armature* ones.

Only meaningful when having at least two of these modifiers on the same object, with *Multi Modifier* activated.

Invert <–>

Inverts the influence set by the vertex group defined in previous setting (i.e. reverses the weight values of this group).

Bind to

Vertex Groups

When enabled, bones of a given name will deform points which belong to vertex groups of the same name. E.g. a bone named "forearm", will only affect the points in the "forearm" vertex group.

The influence of one bone on a given point is controlled by the weight of this point in the relevant group. A much more precise method than *Bone Envelopes*, but also generally longer to set up.

Bone Envelopes

When enabled, bones will deform points or control points near them, defined by each bone's envelope radius and distance. Enable/Disable bone envelopes defining the deformation (i.e. bones deform points in their neighborhood).

Hook Modifier

The *Hook* Modifier is used to deform stroke points using another object (usually an empty or a bone but it can be any object).

As the hook moves, it pulls points from the strokes with it. You can think of it as animated Proportional Editing.

See also

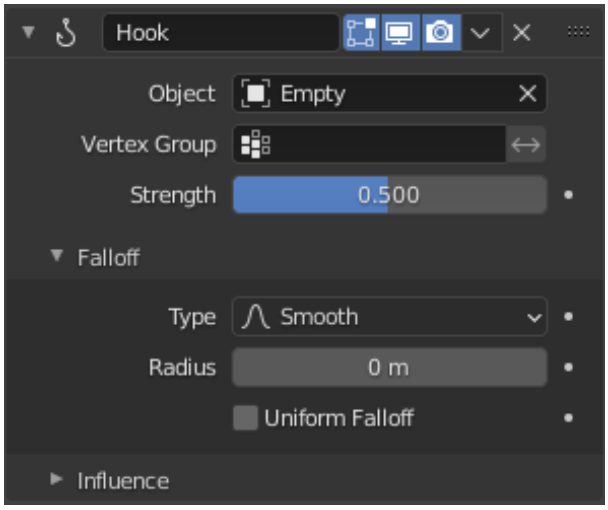
This documentation refers to the Hook Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Hook Modifier.

Options

Object
: The name of the object to hook points to.

Vertex Group
: Restricts the effect only to a vertex group.

Strength
: Adjust this hooks influence on the stroke points, were (0.0 to 1.0) (no change to fully follow the hook).



The Hook modifier.

Falloff

Type
: This can be used to adjust the type of curve for the Strength attenuation. You can also define a custom curve to get a much higher level of control.

Radius
: The size of the hooks influence.

Uniform Falloff
: This setting is useful when using hooks on scaled objects, especially in cases where non-uniform scale would stretch the result of the hook.

Influence

See Influence Filters.

Note

The Hook Modifier stores points indices from the original strokes to determine what to affect; this means that modifiers that generate geometry, like a Subdivision Surface Modifier, should always be applied **after** the Hook Modifier; otherwise the generated geometry will be left out of the hook's influence.

Example



Empty used as a hook to manipulate a vertex group (right eye of the monkey).

Lattice Modifier

The Lattice modifier deforms the base object according to the shape of a Lattice object.

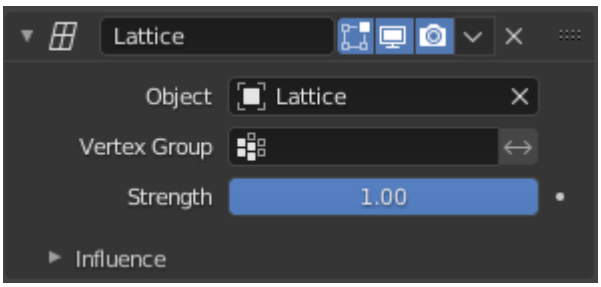
Tip

A Lattice Modifier can quickly be added to selected objects by selecting them all, then selecting the lattice object last and pressing `Ctrl-P` and choosing *Lattice Deform*. This will both add Lattice Modifiers to the selected objects and parent them to the lattice.

See also

This documentation refers to the Lattice Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Lattice Modifier.

Options



Lattice Modifier.

Object
: The Lattice object with which to deform the base object.

Vertex Group
: Restricts the effect only to a vertex group.

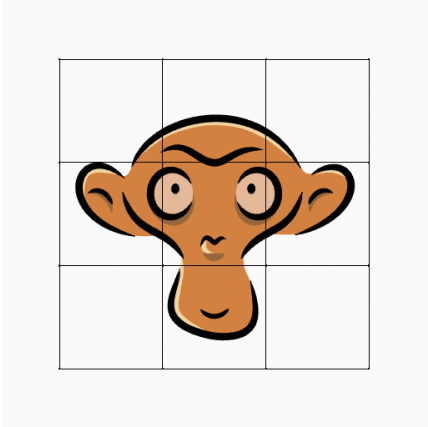
Strength
: A factor to control blending between original and deformed points positions.

Influence

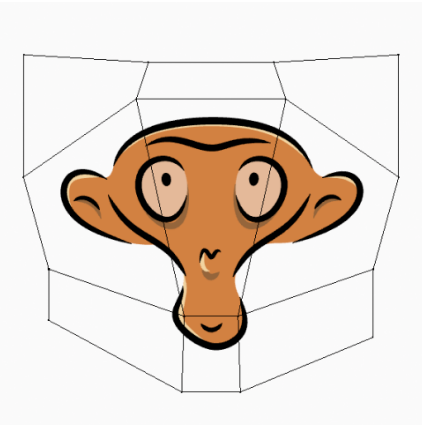
See Influence Filters.

Example

Lattice modifier example.



Original model.



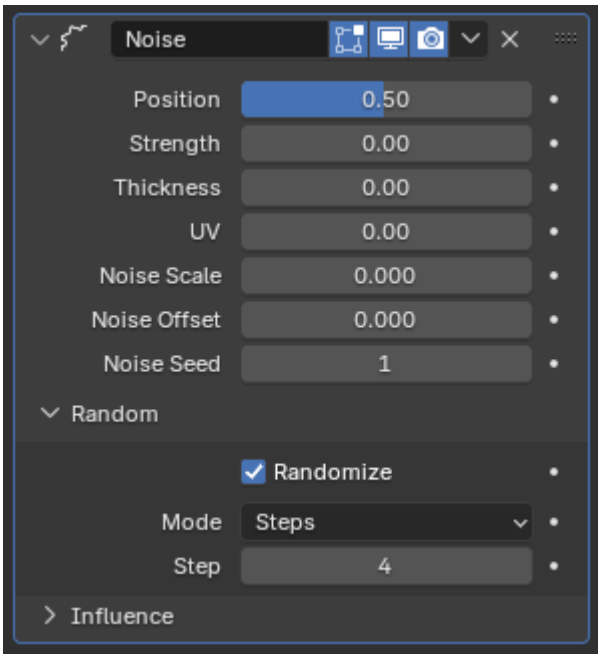
After lattice edition.

Noise Modifier

The *Noise* Modifier changes the value of one or more stroke/points properties like: location, strength, thickness or UV texture position by adding varied values that make the line unstable and noisy.

Random values can be used for the noise factor for more vivid effects.

Options



Noise Modifier.

Position
: Strength of the noise effect over the point location.

Strength
: Strength of the noise effect over the point strength (opacity).

Thickness
: Strength of the noise effect over the point thickness.

UV
: Strength of the noise effect over the point UV rotation.

Noise Scale
: Control the noise frequency scale.

Noise Offset
: Moves the noise along the strokes.

Seed
: Seed used by the pseudo-random number generator.

Randomize

When enabled, the noise uses a random value over time.

Mode

Steps:: New random value at defined steps.

Step

Number of frames before using a new random value.

Keyframes:: New random value only on keyframes.

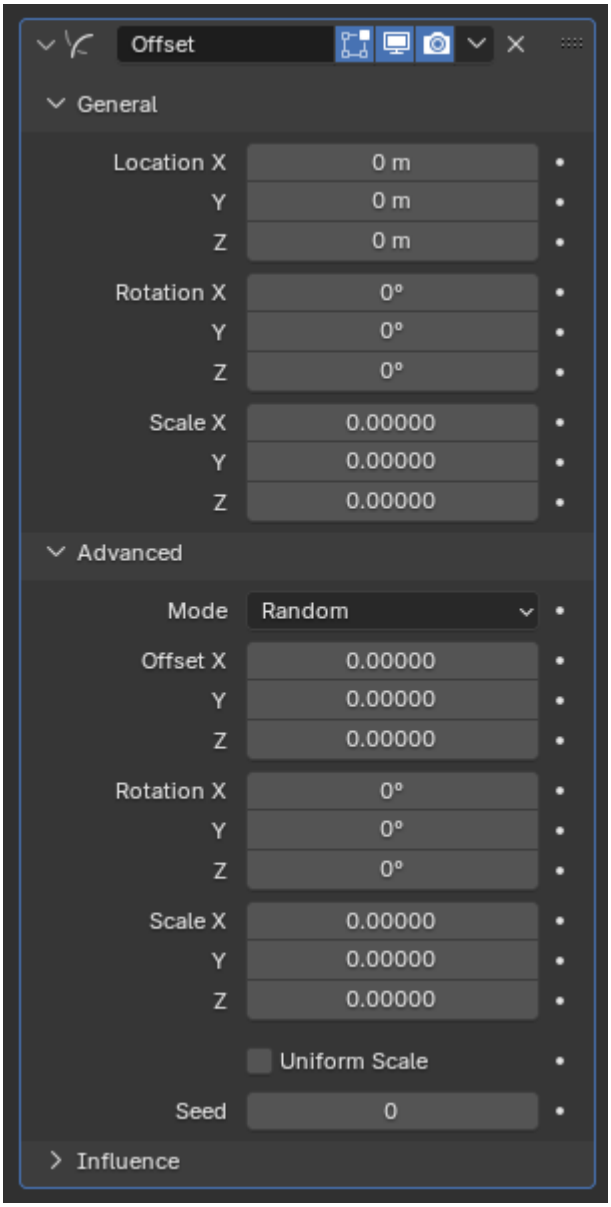
Influence

See Influence Filters.

Offset Modifier

The *Offset* Modifier changes the strokes location, rotation or scale starting from the object origin.

Options



Offset Modifier.

General

Location X, Y, Z
: Sets strokes location offset from its object origin.

Rotation X, Y, Z
: Sets strokes rotation.

Scale X, Y, Z
: Sets strokes scale.

Advanced

Mode

Random::	Add random values to the individual strokes offset.
Layer::	Offset by layers.
Stroke::	Offset by strokes (based on the stroke draw order).
Material::	Offset by Materials.

Offset X, Y, Z
: Sets individual element location offset.

Rotation X, Y, Z
: Sets individual element rotation.

Scale X, Y, Z
: Sets individual element scale.

Uniform Scale (Random mode)
: Use the same random *Seed* for each scale axis in the strokes for a uniform scale.

Seed (Random mode)

Seed used by the pseudo-random number generator.

Layer/Stroke/Material Step (For Layer, Stroke and Material mode)
The number of elements to be grouped and offset together.

Offset (For Layer, Stroke and Material mode)
Offset the element starting point.

Influence

See Influence Filters.

Shrinkwrap Modifier

The *Shrinkwrap* modifier allows a Grease Pencil object to “shrink” to the surface of another object. It moves each point of the object being modified to the closest position on the surface of the given mesh.

See also

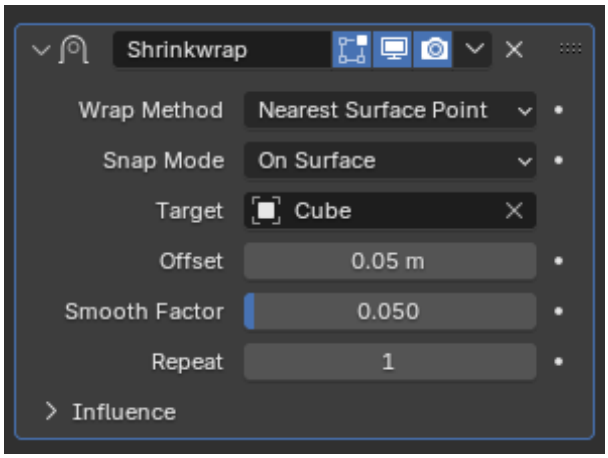
Shrinkwrap Constraint.

See also

This documentation refers to the Shrinkwrap Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Shrinkwrap Modifier.

Options

Wrap Method
: This selector specifies the method to be used to determine the nearest point on the target’s surface for each point of the modified



The Shrinkwrap modifier in Nearest Surface Point mode.

object. Some options will add some extra, specific controls to the panel. See Wrap Methods for an explanation of each method.

Snap Mode

Most modes support an additional setting to control how the point is moved to the target point selected by the methods described above. Some of the choices only differ if *Offset* is not zero.

On Surface:: The point is always moved. The offset is applied along the projection line connecting the original point and selected target point towards the original position.

Inside:: The point is not moved if it is already inside the target. Offset shrinks the allowed volume towards the inside along the projection line.

Outside:: The point is not moved if it is already outside the target. Offset expands the exclusion volume towards the outside along the projection line.

Outside Surface:: Like *On Surface*, but the offset is always applied towards the outside of the target.

Above Surface:: Like *On Surface*, but the offset is applied along the smooth normal of the target.

Note

The *Inside* and *Outside* options can be used for very crude collision detection. The inside vs outside determination is done based on the target normal and is not always stable near 90 degree and sharper angles in the target mesh.

Target

Shrink target, the mesh to shrink to/wrap around.

Offset

The distance that must be kept from the calculated target position.

Smooth Factor

Amount of smoothing to apply.

Repeat
: The number of time to apply smoothing.

Influence

See Influence Filters.

Wrap Methods

Nearest Surface Point

This will select the nearest point over the surface of the shrunk target.

Project

This will project vertices along a chosen axis until they touch the shrink target. Vertices that never touch the shrink target are left in their original position.

Limit
: This is a distance limit between original point and surface. If the distance is larger than this limit point would not be projected onto the surface.

Subdivision Levels
: This applies a (temporary) *Catmull-Clark* subdivision to the modified object's geometry, before computing the wrap.

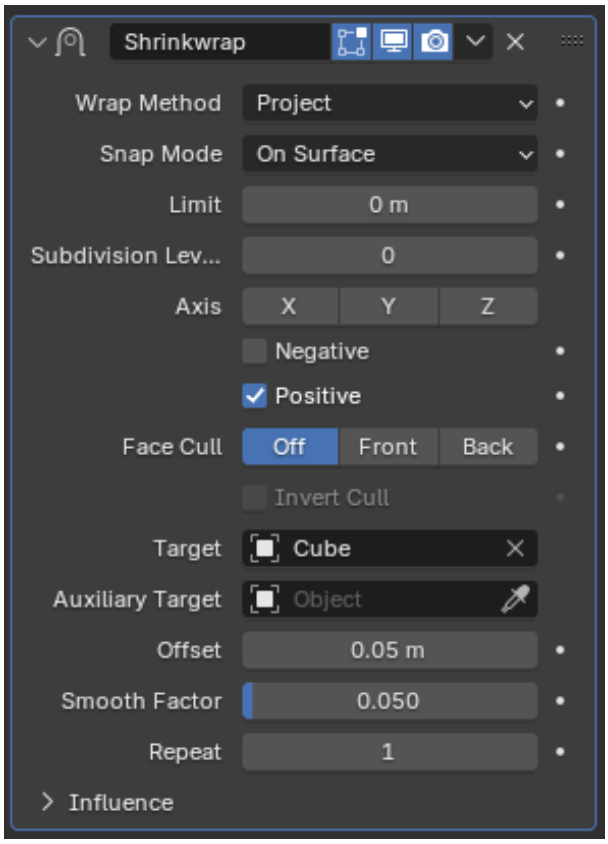
Axis
: Along which local axis of the modified object the projection is done. These options can be combined with each other, yielding a "median axis" of projection. If none are selected, the normal direction is used.

Negative/Positive
: This allows you to select the allowed direction(s) of the shrink along the selected axis. If both options are enabled, both ways are evaluated and the closest hit is selected.

Face Cull
: Allows you to prevent any projection over the “front side” (respectively the “back side”) of the target’s faces. The “side” of a face is determined by its normal (front being the side “from where” the normal “originates”).

Invert Cull
: If *Cull Faces* is enabled, and *Negative* direction along axis is allowed, this option can be used to invert the *Front* or *Back* cull choice for the *Negative* direction. This is useful when projecting in both directions.

Auxiliary Target
: An additional object to project over.



Project mode.

Nearest Vertex

This will snap vertices to the nearest vertex of the shrunk target. It adds no extra options.

This method doesn’t support the *Snap Mode* setting.

Target Normal Project

This mode is similar to *Nearest Surface Point*, but produces a much smoother projection in return for being significantly slower.

Instead of finding the closest point, it searches for the nearest point that has its interpolated smooth normal pointing towards or away from the original

point position. Non-manifold boundary edges are specially handled as infinitely thin cylinders that emit normals in all perpendicular directions; ignores flat shading.

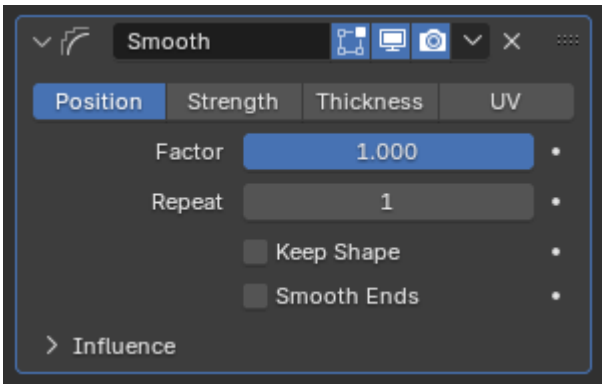
Smooth Modifier

The *Smooth* Modifier changes the value of one or more stroke/points properties like: location, strength, thickness or UV texture position trying to maintain similar values that make the line fluid and smoother.

Options

Affect
: Combination of stroke/points properties that will be affected by the smooth factor.

  Position:: Smooth affect the point location.
  Strength:: Smooth affect the point strength (opacity).
  Thickness:: Smooth affect the point thickness.
  UV:: Smooth affect the point UV rotation.



Smooth Modifier.

Factor
: Strength of the smooth effect.

Repeat
: The number of smoothing iterations, equivalent to executing the Smooth tool multiple times. High values can reduce the animation performance (FPS).

Keep Shape

When enabled, the smoothing algorithm will try to maintain as close as possible the overall shape of the original stroke.

Smooth Ends
: Smooth the ends of the strokes.

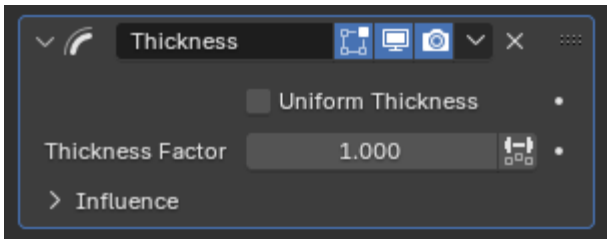
Influence

See Influence Filters.

Thickness Modifier

The *Thickness* Modifier change the stroke points thickness.

Options



Thickness Modifier.

Uniform Thickness
: When enabled, makes the thickness equal for the entire strokes.

Thickness
: Absolute Thickness for the stroke points.

Thickness Factor
: Value to add or subtract to the actual points thickness.

Influence

See Influence Filters.

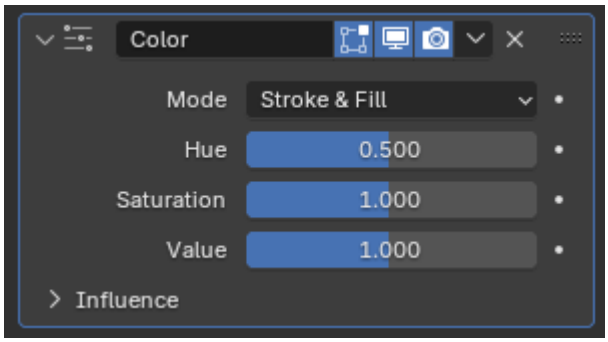
Grease Pencil Color Modifiers

- Hue/Saturation Modifier
- Opacity Modifier
- Tint Modifier

Hue/Saturation Modifier

The *Hue/Saturation* Modifier applies a color transformation to the object output color.

Options



Hue/Saturation Modifier.

Mode
: The color transformation will be applied on the stroke and/or the fill color.

  Stroke & Fill, Stroke, Fill

Hue
: Specifies the hue rotation of the image. 360° are mapped to (0 to 1). The hue shifts of 0 (-180°) and 1 (+180°) have the same result.

Saturation
: A saturation of 0 removes hues from the image, resulting in a grayscale image. A shift greater than 1.0 increases saturation.

Value
: Value is the overall brightness of the image. De/Increasing values shift an image darker/lighter.

Influence

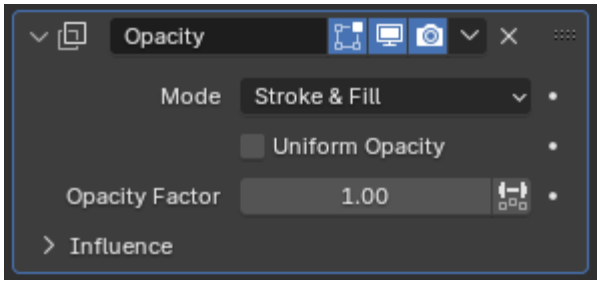
See Influence Filters.

Opacity Modifier

The *Opacity* Modifier change the opacity (alpha) value of the stroke points.

The alpha value in Grease Pencil is stored per-point. The modifier can alter these values to go from totally transparent points to totally opaque points.

Options



Opacity Modifier.

Mode
: The color transformation will be applied to the stroke/fill color or stroke Hardness. When Hardness is selected the opacity affects the stroke's transparency (alpha) from the center to the border.

  Stroke & Fill, Stroke, Fill, or Hardness

Uniform Opacity
: When enabled, makes the opacity equal for the entire strokes.

  Strength
  : Absolute opacity for the stroke points.

Opacity Factor
: Controls the opacity value of the stroke points. A value of 1.0 respect the original alpha value of the points, a shift less than 1.0 make the points more transparent than originally, and a shift greater than 1.0 make the points more opaque than originally.

  Sets value to 2.0 makes the points alpha fully opaque.

Influence

See Influence Filters.

Example

Opacity Factor samples.



Opacity Factor: 0.3.



Opacity Factor: 1.0 (original alpha).

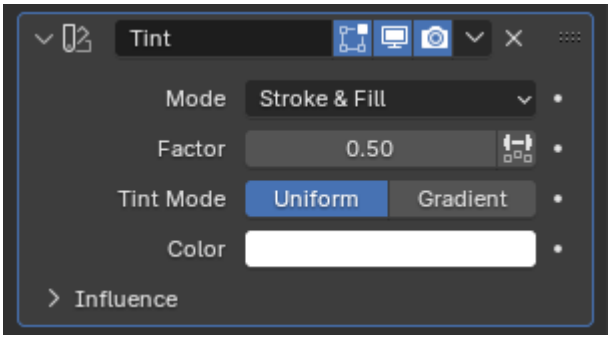


Opacity Factor: 2.0 (fully opaque).

Tint Modifier

The *Tint* Modifier colorize the original stroke or fill with a selected color.

Options



Tint Modifier.

Mode
: The color transformation will be applied on the stroke and/or the fill color.

  Stroke & Fill, Stroke, Fill

Strength
: Controls the amount for the color mixing.

  A value of 0 respect the original stroke's color, a value of 1.0 totally replace the original color with the tint color.

  A shift greater than 1.0 will make the points alpha less transparent than originally (2.0 is fully opaque).

Tint Type
: **Uniform::** Color
    Defines the tint color for mixing with the original color.

  Gradient::

Color Ramp
: Defines the tint gradient color for mixing with the original color. For controls see Color Ramp Widget.

Object
: A Data ID to select an object (usually an empty), which position and rotation will be used to define the center of the effect.

Radius
: Defines the maximum distance of the effect.

Influence

See Influence Filters.

Example

Tint uniform color sample.



Strength: 0 (original color).

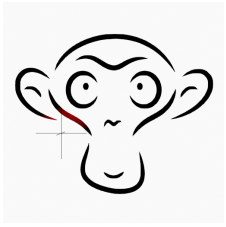


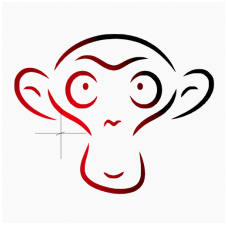
Strength: 0.5.

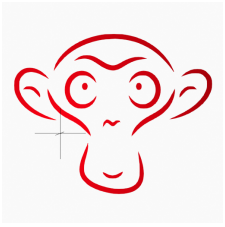


Strength: 1.0 (fully tinted).

Tint gradient color sample.







Radius: 1, Strength: 1.	Radius: 5, Strength: 1.	Radius: 10, Strength: 1.

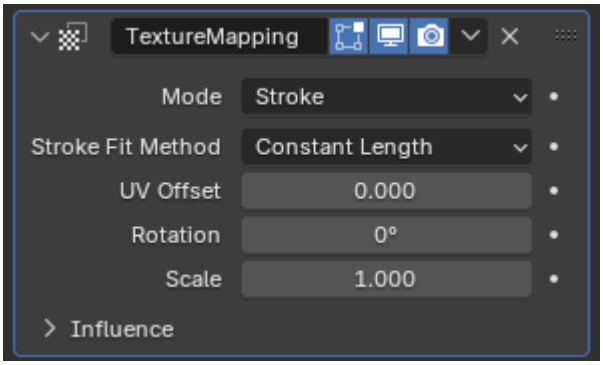
Grease Pencil Edit Modifiers

- Texture Mapping Modifier
- Time Offset Modifier
- Vertex Weight Angle Modifier
- Vertex Weight Proximity Modifier

Texture Mapping Modifier

The *Texture Mapping* Modifier change the strokes texture UV position.

Options



Texture Mapping.

Mode
: The texture transformation will be applied to the stroke/fill or stroke UVs.

Stroke::

Stroke Fit Method
: Selects the texture fitting method.

Constant Length:: The texture keep a consistent length along the strokes.

Stroke Length:: The texture is normalized to fit the stroke length.

UV Offset
: Moves the texture along the strokes.

Rotation
: Rotates the points of the strokes.

Note
The *Rotation* option is limited to a range of -90 to 90 degrees.

Fill::

Scale
: Factor for the texture scale.

Fill Rotation
: Sets the texture angle.

Offset
: Moves the texture origin.
X, Y

Scale
: Factor for the texture scale.

Influence

See Influence Filters.

Example

Opacity Factor samples.



Rotation: 0°.



Rotation: 45°.



Rotation: 90°.

Time Offset Modifier

The *Time Offset* Modifier applies a temporal offset to Grease Pencil keyframes on your timeline. If you have duplicated a Grease Pencil object you can use the Time Offset Modifier on the copies to desynchronize their animation. This can give more natural looking results.

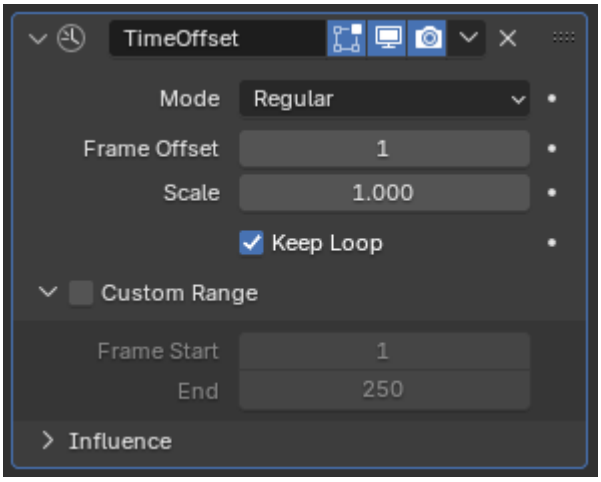
Using the Time Offset Modifier it's possible to have Grease Pencil frame ranges play back as repeating loops. Traditionally, 2D animation that uses looped drawings includes characters walking, rising smoke, and falling rain. In Fixed Frame mode the Time Offset Modifier can display drawings on your timeline entirely independently of the playhead position.

This can be handy for displaying drawings that will appear often in your animation. Think of switching between predefined mouth shapes for instance.

Options

Mode

Regular::



Time Offset Modifier.

Offsets keyframes in the default animation playback direction (playhead moving from left to right).

Reverse:: Offsets keyframes in reversed animation playback direction (playhead moving from right to left).

Fixed Frame:: The Frame parameter determines which frame is displayed. This value needs to be animated in order to have the displayed frame change during playback.

Frame
: The number of the frame to display.

Ping Pong:: Loop back and forth animation.

Chain:: It allows to combine the different Modes consecutively.

Repeat
: Number of cycle repeats

Frame Offset
: The number of frames to offset the original keyframes by.

Scale
: Controls the speed of the frames playback. 1 is equal to the actual frame rate, could be positive (faster) or negative (slower).

Keep Loop
: Moves end frame to the animation start to keep animation in a loop.

Custom Range

When enabled, the animation playback is restricted only to a frame range.

Frame Start/End
: Sets the range start and end frames.

Influence

See Influence Filters.

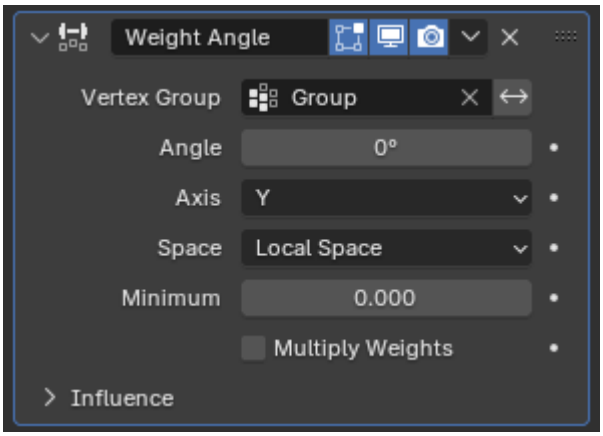
Vertex Weight Angle Modifier

This modifier sets the weights of the given vertex group, based on a predetermined angle.

Warning

This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

Options



The Vertex Weight Proximity modifier panel.

Vertex Group
: The vertex group to affect.

Invert <-->
: Inverts the influence of the selected vertex group. The setting reverses the weight values of the group.

Angle

Sets the angle for the maximum weights value.

Axis
: The axis along which the angle affects the weights.

  X, Y, Z

Space
: Coordinate space to be used.

Minimum
: Minimum value for vertex weight.

Multiply Weights
: Multiply the calculated weights with the existing values in the vertex group.

Influence

See Influence Filters.

Vertex Weight Proximity Modifier

This modifier sets the weights of the given vertex group, based on the distance between the object (or its vertices), and another target object (or its geometry).

Warning

This modifier does implicit clamping of weight values in the standard (0.0 to 1.0) range. All values below 0.0 will be set to 0.0, and all values above 1.0 will be set to 1.0.

See also

This documentation refers to the Vertex Weight Proximity Modifier specific to the Grease Pencil object. For uses with other object types refer to the general Vertex Weight Proximity Modifier.

Options

Vertex Group
: The vertex group to affect.

  Invert <-->
  : Inverts the influence of the selected vertex group. The setting reverses the weight values of the group.

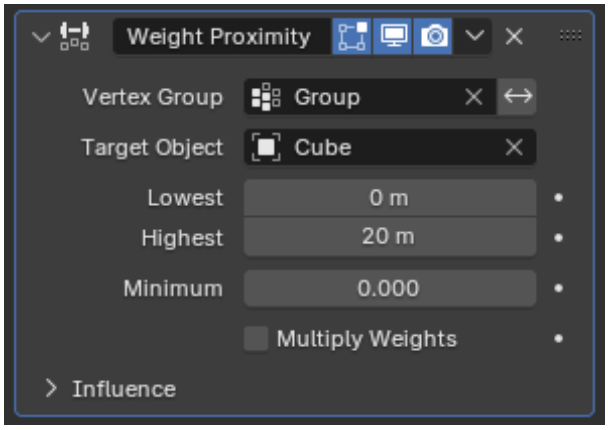
Target Object
: The object from which to compute distances.

Lowest
: Distance mapping to 0.0 weight.

Highest
: Distance mapping to 1.0 weight.

Minimum
: Minimum value for vertex weight.

Multiply Weights
: Multiply the calculated weights with the existing values in the vertex group.



The Vertex Weight Proximity modifier panel.

Influence

See Influence Filters.

Visual Effects

- Introduction
  - Interface

Types

- Blur Visual Effect
- Colorize Visual Effect
- Flip Visual Effect
- Glow Visual Effect
- Pixelate Visual Effect
- Rim Visual Effect
- Shadow Visual Effect
- Swirl Visual Effect
- Wave Distortion Visual Effect

Introduction

Reference

Panel:: *Properties ▸ Visual Effects*

Grease Pencil has a special set of viewport real-time visual effects that can be apply to the object.

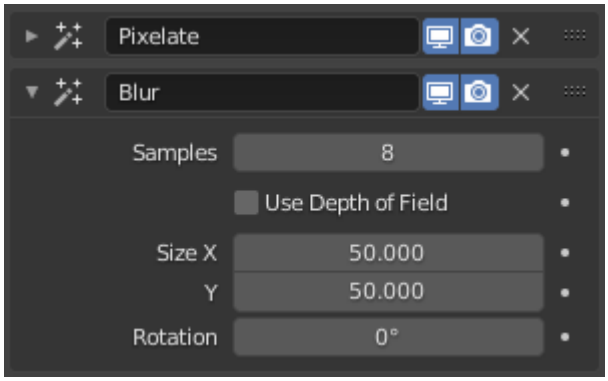
These effects treat the object as if it was just an image, for that reason they have effect on the whole object and cannot limit their influence on certain parts like layers, materials or vertex group as with modifiers. Also unlike modifiers, they can not be applied to the object.

Their main purpose is to have a quick way to apply visual effects on your drawings like blurring, pixelation, wave distortion, among others.

Note

Visual Effects best fit for quick viewport visualization. You can use it for final renders but if you want more precision with effects it is still recommended to use the Compositor.

Interface



Panel layout (Blur effect as an example).

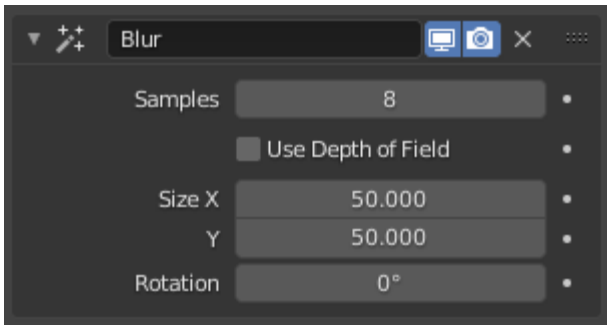
The visual effects panels and interface are similar to modifiers. Each effect shares the same basic interface components similar to modifiers for meshes.

See Modifiers Interface for more information.

Blur Visual Effect

The *Blur* Visual Effect applies a Gaussian blur to the object.

Options



Blur Visual Effect.

Samples
: Number of blur samples (0 disabled the blur effect).

Use Depth of Field
: When enabled, the blur effect uses the focal plane distance of the actual camera to calculate the object blur. Only available in camera view.

Size X, Y
: Control the blur scale in pixels on the X and Y axis.

Rotation
: Control the Rotation of the blur.

Example

Blur Effect samples (Samples: 8).



Original Model.



Factor: 10, 10.



Factor: 50, 50.

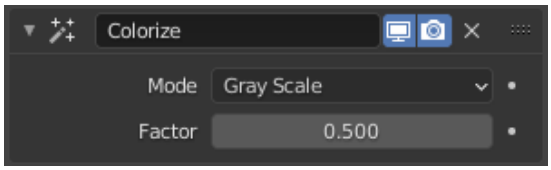


Factor: 100, 100.

Colorize Visual Effect

The *Colorize* Visual Effect applies different preset colorizing effects to the object.

Options



Blur Visual Effect.

Mode
- Grayscale
  - Converts to a grayscale image.
- Sepia
  - Converts to a sepia tone image.
- Duotone
  - Converts to a posterize image with high contrast and brightness.
    - Low Color
      - Primary color.
    - High Color
      - Secondary color.
- Transparent
  - Add color transparency.
- Custom
  - Allows to define a tint custom color.
    - Color
      - Sets the tint color.

Factor
- Control the mix value.

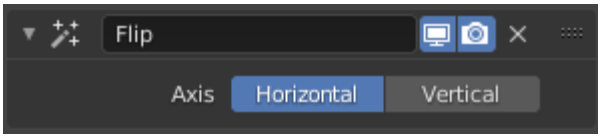
Example

Colorize Effect samples.

Flip Visual Effect

The *Flip* Visual Effect shows the object flipped horizontally and/or vertically.

Options



Flip Visual Effect.

Axis
: Which axis or axes to flip the object about.

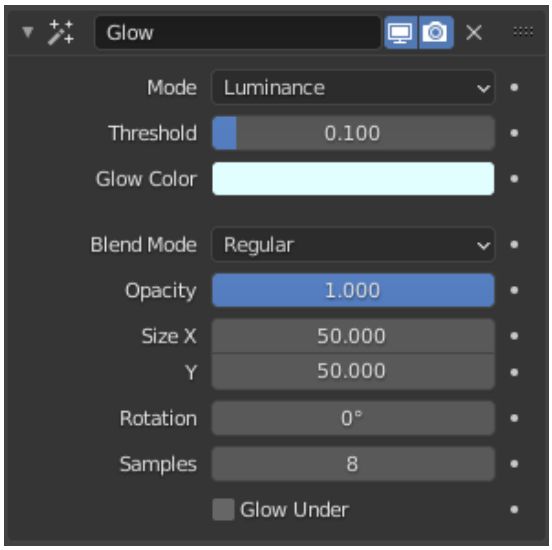
Horizontal:: When enabled, shows the object flipped horizontally.

Vertical:: When enabled, shows the object flipped vertically.

Glow Visual Effect

The *Glow* Visual Effect add a glowing rim around the object.

Options



Glow Visual Effect.

Mode
: Determines the mode for the glow effect.

Luminance:: The glow light illuminates the entire object.

Color:: The glow light only affect a single color.
Select Color
: Allows to select a single color to apply the glow light.

Threshold
: Limits the colors affected by the glow light. (A value of 1 means no colors affected.)

Glow Color
: Defines the glow color.

Blend Mode
: The mask blending operation to perform. See Color Blend Modes.

Opacity
: Control the Opacity of the glow over the object.

Size X, Y
: Control the glow scale in pixels on the X and Y axis.

Rotation
: Control the Rotation of the glow.

Samples
: Number of Blur samples (0 disabled the blur effect).

Glow Under
: When enabled, glow only affects alpha areas.

Example

Glow Effect samples.



Original image.



Mode: Luminance.



Mode: Luminance (Glow Under).



Mode: Color (Black lines).

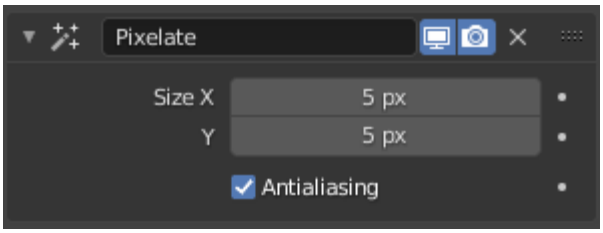
Pixelate Visual Effect

The *Pixelate* Visual Effect shows the object as a pixelated image.

Options

Size X, Y
: Horizontal and vertical size of the final pixels to apply.

Anti-aliasing
: Applies an anti-aliasing effect to the resulting pixels.



Pixelate Visual Effect.

Example

Pixelate Effect samples.



Original image.



Size: 20 px.



Size: 100 px.



Size: 200 px.

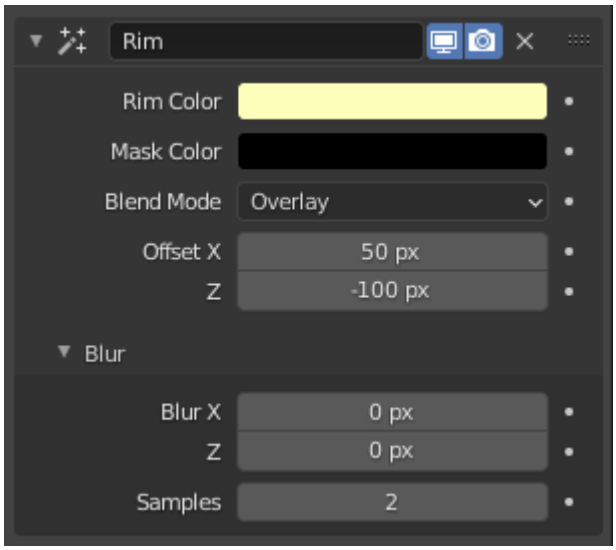
Rim Visual Effect

The *Rim* Visual Effect shows a simulated rim light on the object contour.

For simulating the rim light, a masked color silhouette of the object is displaced in horizontal and/or vertical direction.

Many blending modes can be applied to the resulting mask.

Options



Rim Visual Effect.

Rim Color
: Defines the rim light color.

Mask Color
: Defines a color to keep unaltered.

Blend Mode
: The mask blending operation to perform. See Color Blend Modes.

Offset X, Y
: Control the color mask displacement in pixels on the X and Y axis.

Blur

Blur X, Y
: Control the blur scale in pixels on the X and Y axis.

Samples
: Number of blur samples (0 disabled the blur effect).

Example

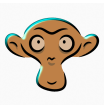
Rim Effect samples (Mode: Add).



Original image.



No blur.



Blur.



Mask color: Black.

Shadow Visual Effect

The *Shadow* Visual Effect shows a simulated shadow casting by the object.

For simulating the shadow a color silhouette of the object is displaced in horizontal and/or vertical direction on the back of the object.

Options

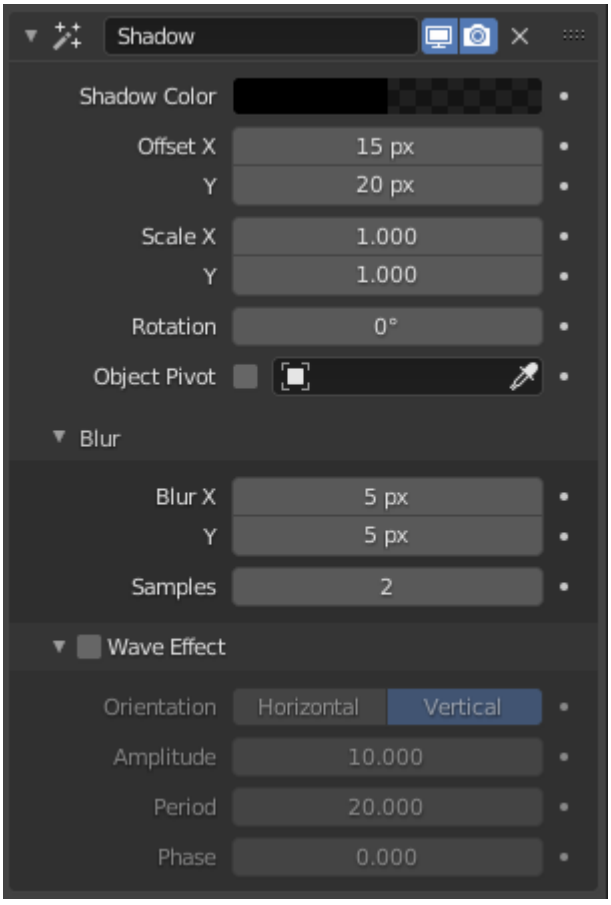
Shadow Color
: Defines the shadow color.

Offset X, Y
: Control the shadow displacement in pixels on the X and Y axis.

Scale X, Y
: Control the size of the shadow on the X and Y axis.

Rotation
: Sets the shadow rotation around the Grease Pencil object center or another object when *Use Object As Pivot* is enabled.

Object Pivot
: When enabled, an *Object* is used by the shadow as the center of rotation.



Shadow Visual Effect.

Blur

Blur X, Z
: Control the blur scale in pixels on the X and Z axis.

Samples
: Number of blur samples (0 disabled the blur effect).

Wave Effect

When enabled, apply a wave distortion to the shadow.

Orientation
: Sets horizontal or vertical direction for the waves.

Amplitude
: Controls the strength and the depth of the wave.

Period
: Controls the wave period. The time it takes to complete one cycle.

Phase
: Shifts the wave pattern over the shadow.

Example

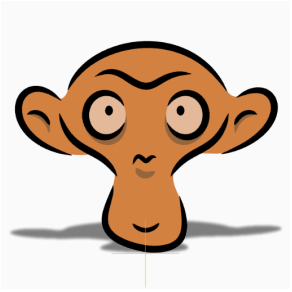
Shadow Effect samples.



Simple Shadow.



Blurred Shadow.

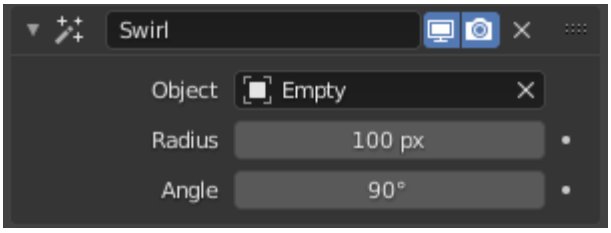


Stretched shadow with an empty as center of rotation.

Swirl Visual Effect

The *Swirl* Visual Effect applies a swirling pattern to the object. The effect use an object as the center of the swirl.

Options



Swirl Visual Effect.

Object
: Sets the object to use as the center of the swirl.

Radius
: External radius size of the swirl.

Angle
: Rotation angle of the swirl. A value of 0 shows no swirl.

Example

Swirl Effect samples (with a Radius of 100 px).



Angle: 0°.



Angle: 15°.



Angle: 45°.

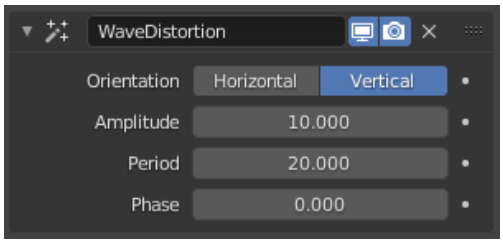


Angle: 90°.

Wave Distortion Visual Effect

The *Wave Distortion* Visual applies a wavy effects to the object.

Options



Wave Distortion Effect.

Orientation
: Sets horizontal or vertical direction for the waves.

Amplitude
: Controls the strength and the depth of the wave.

Period
: Controls the wave period. The time it takes to complete one cycle.

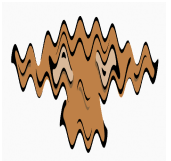
Phase
: Shifts the wave pattern over the Object.

Example

Wave Distortion Effect samples.



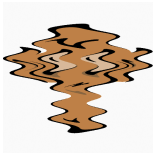
Amplitude: 10 (horizontal).



Amplitude: 30 (horizontal).



Amplitude: 10 (vertical).



Amplitude: 30 (vertical).

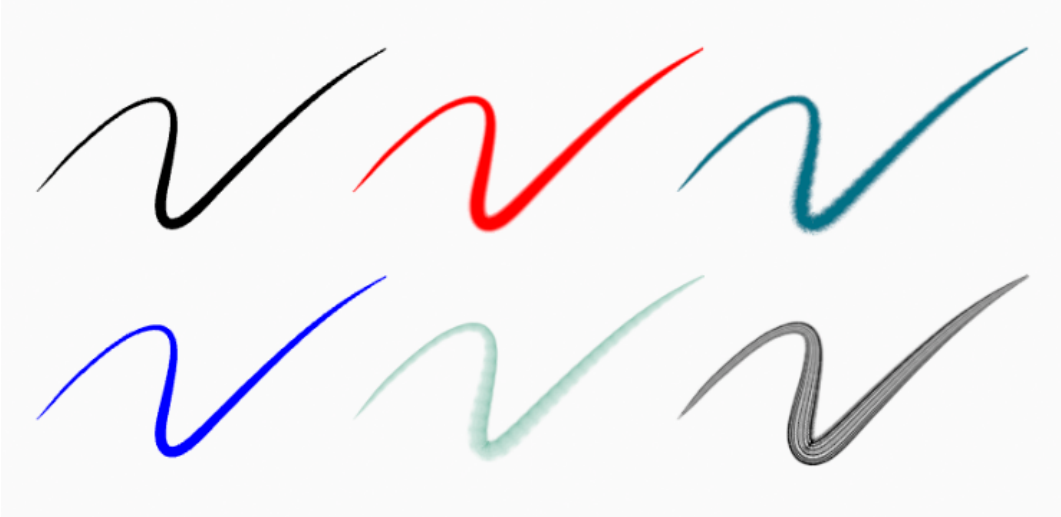
Grease Pencil Materials

Materials control the appearance of the Grease Pencil object. They define the base color and texture of the strokes and filled areas.

There is always only one active material in the list (the selected one). When you draw, the new strokes use the active material.

You can override the base material color using the tools in Vertex Mode or the Draw and Tint tool in Draw Mode.

The material always remains linked to the strokes, this means that any change in a material will change the look of already drawn strokes.



Same stroke linked to different materials.

Material Shader

Grease Pencil materials use a special shader that define the appearance of the surface of the stroke and fill.

Stroke and fill components has it own section panel and they can be enabled with a checkbox on the panel header.

Stroke only has effect on the lines and *Fill* only on the areas determined by closed lines (by connecting the lines start and end points).

Note

The shader is not a BSDF capable shader and can only be setting up on the Material Properties panel (it is not a shader node).

Setting Up Materials

Reference

Mode::	Drawing Mode
Panel::	*Material* ▸ *Material Slots*
Shortcut::	U

Grease Pencil materials can be created in the Material properties as any other materials in Blender. See Material assignment for more information.

The 3D Viewport can be set to Material Preview or Rendered shading, to interactively preview how the material looks in the scene.

Grease Pencil materials are data-blocks that can be assigned to one or more objects, and different materials can be assigned to different strokes.

In Grease Pencil the brush settings together with the material used will define the look and feel of the final strokes.

Materials slots also have some extra controls that help to work with materials while drawing or editing lines.

Properties

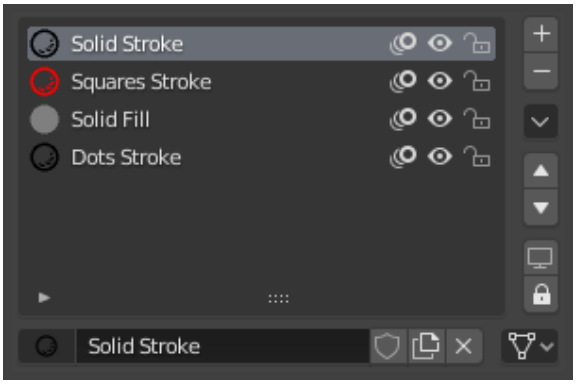
Material Slots

Surface

Settings

Grease Pencil Material Properties

Material Slots



Grease Pencil material slots panel.

Next to the material name there are three icons buttons that control common properties of the material:

/ (Show/Hide in Ghosts)
: Toggle the use of the material for Onion Skinning.

/ (Hide/Show Material)
: Toggle whether the active material is the only one that can be edited and is visible.

/ (Lock/Unlock Material)
: Toggle whether the active material is the only one that can be edited.

Specials

Show All
: Turns on the visibility of every material in the list.

Hide Others
: Turns off the visibility of every material in the list except the active one.

Lock All
: Locks edition of all the materials in the list.

Unlock All
: Unlocks edition of all the materials in the list.

Lock Unselected
: Locks all materials not used in the selected strokes.

Lock Unused
: Locks and hides all unused materials.

Copy Material to Selected
: Copy the active material to the selected Grease Pencil object.

Copy All Materials to Selected
: Copy all materials to the selected Grease Pencil object.

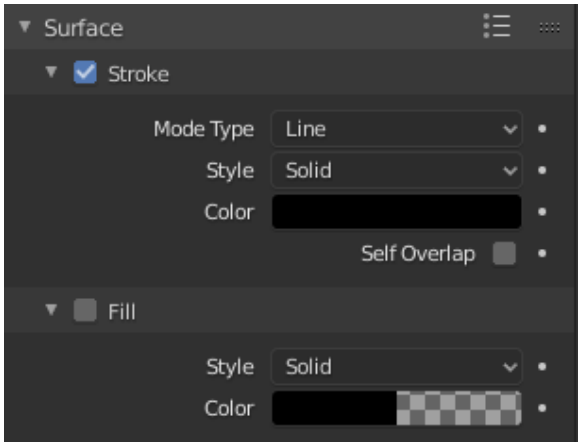
Remove Unused Slots
: Remove all unused materials.

Lock & Visibility Controls

(Isolate Material)
: Toggle whether the active material is the only one that can be edited.

(Isolate Material)
: Toggle whether the active material is the only one that can be edited and is visible.

Surface



Shader panel with only Stroke component activated.

Stroke

When enabled, the shader use the stroke component. The *Stroke* component controls how to render the edit lines.

Line Type
: Defines how to display or distribute the output material over the stroke.

Line:: Connects every points in the strokes showing a continuous line.

Dots:: Use a disk shape at each point in the stroke. The dots are not connected.

Squares:: Use a square shape at each point in the stroke. The squares are not connected.

Style
: The type of the material.

Solid:: Use a solid color.

Texture:: Use an image texture.

Image
: The image data-block used as an image source.

Blend
: Texture and Base Color mixing amount.

UV Factor
: The image size along the stroke.

Base Color
: The base color of the stroke.

Holdout
: Removes the color from strokes underneath the current by using it as a mask.

Alignment
: Defines how to align the *Dots* and *Squares* along the drawing path and with the object's rotation.

Path:: Aligns to the drawing path and the object's rotation.

Object:: Aligns to the object's rotation; ignoring the drawing path.

Fixed:: Aligns to the screen space; ignoring the drawing path and the object's rotation.

Rotation

Rotates the points of *Dot* and *Square* strokes.

Note

The *Rotation* option is limited to a range of -90 to 90 degrees.

Self Overlap

Disables stencil and overlap self-intersections with alpha materials.

Samples of different material strokes mode types and styles.



Mode Type: Line, Style: Solid.



Mode Type: Line, Style: Texture.



Mode Type: Dot, Style: Solid.



Mode Type: Dot, Style: Texture.

Fill

When enabled, the shader use the fill component. The *Fill* component control how to render the filled areas determined by closed edit lines.

Style

The type of material.

Solid:: Use solid color.

Gradient:: Use a color gradient.

Gradient Type

Linear:: Mix the colors along a single axis.

Radial:: Mix the colors radiating from a center point.

Texture:: Use an image texture.

Image
: The image data-block used as an image source.

Samples of different material fill styles.



Style: Solid.



Style: Gradient (Linear).



Style: Gradient (Radial).



Style: Texture.

Base Color
: The base color of the fill.

Secondary Color *Gradient*
: The secondary color.

Holdout
: Removes the color from strokes underneath the current by using it as a mask.

Blend *Gradient* / *Texture*
: The amount that the *Secondary Color* (for *Gradient Style*) or image texture (for Texture Style) Mixxes with the *Base Color*.

Flip Colors *Gradient*
: Flips the gradient, inverting the Base Color and *Secondary Color*.

Location X, Y *Gradient* / *Texture*
: Shifts the position of gradient or image texture.

Rotation *Gradient* / *Texture*
: Rotates the gradient or image texture.

Scale X, Y *Gradient* / *Texture*
: Scales the gradient or image texture.

Clip Image *Texture*
: When enabled, show one image instance only (do not repeat).

Settings

Pass Index
: This index can be used with some modifiers to restrict changes to only a certain material. See Modifiers for more information.

Multiframe

Multiframe allows you to draw, edit, sculpt, or weight painting on several frames at the same time. Extremely useful to avoid repeating a task one frame at a time when animating.



Multiframe pop-over.

Use Falloff

When enabled, the effects on the strokes start to falloff from the current frame as defined by a curve widget.

Usage

1. Select the desired keyframes to draw, edit or sculpt at the same time.
2. Activate the Multiframe tool in the 3D Viewport's header with the toggle button (faded lines icon).
3. Once activated you can:
    - Select the points in all the selected keyframes and make your editions.
    - Start sculpting. The sculpt brushes will affects all the strokes in the selected keyframes.
    - Start weight painting. The weight paint brush will affect all the strokes in the selected keyframes.
    - Start Drawing. The new strokes will be added in all the selected keyframes. If you are using the Fill tool then it will be applied in all the selected keyframes.

- When interpolating you can select the stroke from the different frames in the right order. Interpolate tool will use the selection order to calculate the correct stroke pairs.

Note

Not all operators support Multiframe mode.

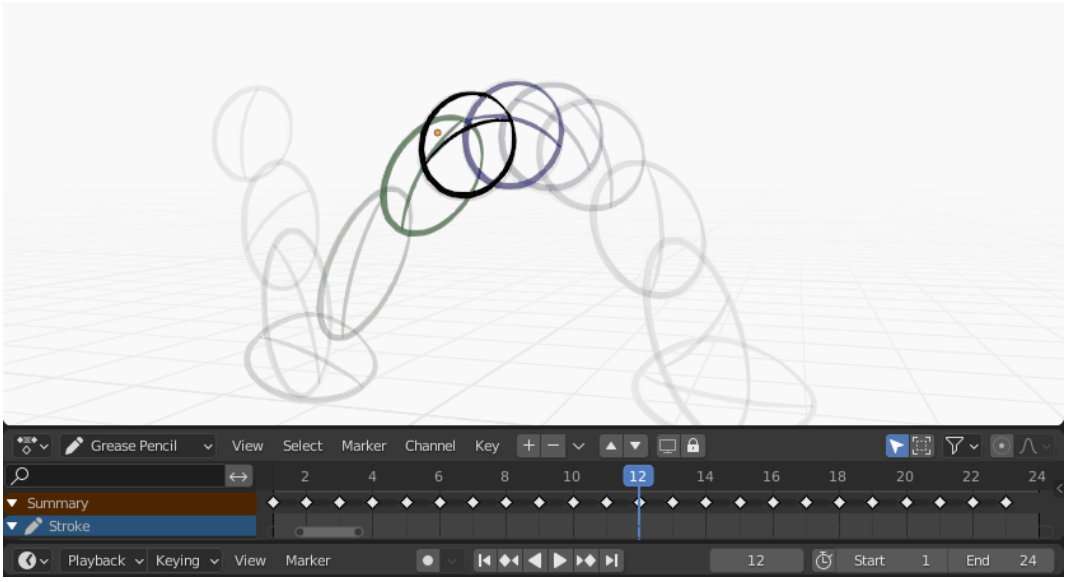
Animation

- Introduction
  - Animating with Grease Pencil
  - 2D Traditional Animation
  - Animation Options
  - Examples
- Interpolation
  - Interpolate
  - Interpolate Sequence
- Animation Tools
  - Insert Blank Keyframe (Active Layer)
  - Insert Blank Keyframe (All Layers)
  - Duplicate Active Keyframe (Active Layer)
  - Duplicate Active Keyframe (All Layers)
  - Delete Active Keyframe (Active Layer)
  - Delete Active Keyframe (All Layers)
  - Interpolate Sequence
  - Bake Mesh to Grease Pencil
  - Bake Object Transform to Grease Pencil

Introduction

Animating with Grease Pencil

The main goal of Grease Pencil is to offer a 2D animation tool full immersed in a 3D environment.



Sample animation showing Grease Pencil object keyframes in the Dope Sheet with onion skinning enabled.

In Blender, Grease Pencil objects can be animated in many ways:

Moving as a whole object
: Changing their position, orientation or size in time;

Drawing frame by frame
: Drawing one frame at a time (traditional animation).

Deforming them
: Animating their points;

Inherited animation
: Causing the object to move based on the movement of another object (e.g. its parent, hook, armature, etc.). Useful for cut-out animation for example.

For a complete overview of animation in Blender please refer to the Animation & Rigging chapter.

2D Traditional Animation

Keyframes

Traditional animation in Grease Pencil is achieved with the use of keyframes that hold the strokes information at a particular frame or frame range.

With Auto keyframe activated, every time you create a stroke in Grease Pencil object Draw Mode a new keyframe is added at the current frame on the active channel. With Auto keyframe deactivated, you will have to add manually a new keyframe or your new strokes will be added on the active keyframe.

See Keyframe Editing for more information.

Note

The channels in the Dope Sheet correspond to the active 2D layer of the Grease Pencil object.

Grease Pencil has its own mode in the Dope Sheet to work with keyframes. See Grease Pencil mode in the Dope Sheet section for more information. There are also several tools on the Stroke menu to work with keyframes and strokes. See Animation tools for more information.

Onion Skinning

One key element in traditional animation is the use of onion skinning. Grease Pencil offer a lot of flexibility and options for this tool. See Onion Skinning for more information.

Animation Options

Draw Mode

In Draw Mode there are three options related to the animation workflow that you can use.



General drawing/animation options.

Add Weight Data
: When enabled, new strokes weight data is added according to the current vertex group and weights. If there is no vertex group selected, no weight data is added.

  This is useful for example in cut-out animation for adding new drawing on the same vertex group without the need to creating it afterwards.

  See Weight Paint Mode for more information.

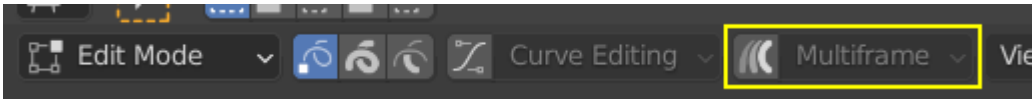
Additive Drawing
: When creating new frames, the strokes from the previous/active frame are include as a basis for the new one.

Multiframe
: If you need to add new strokes to your animation on several frames you can use multiframe drawing.

  You can activate multiframe drawing with the Multiframe button next to the modes selector (faded lines icon). See Multiframe for more information.

Edit Mode

In Edit Mode there is an option related to the animation workflow that you can use.



Multiframe edition.

Multiframe
: Sometimes you may need to modify several frames at the same time with edit tools, for example to repositioning drawings in an animation.

  You can activate multiframe edition with the Multiframe button next to the modes selector (faded lines icon). See Multiframe for more information.

Examples

Traditional Animation

This example shows you how to animate a bouncing ball with a traditional 2D animation technique and Grease Pencil.

First, go to menu *File ▸ New ▸ 2D Animation* to start with a new 2D animation template. The template is ready to quick start your animation with a Grease Pencil object already created, Onion Skinning activated, Auto Keyframe enabled and in camera view.

1. Set the range of the animation in the Timeline from 1 to 24.
2. In the 3D Viewport draw a ball on the upper left corner with the Draw Tool (extreme).
3. Move to frame 12 and draw a squashed ball in the bottom center (breakdown).
4. Move to frame 24 and draw a ball in the top right corner of the 3D Viewport (extreme).

5. Keep drawing all the in-between frames you want using the onion skinning ghost as a reference.

To test the animation, press `Spacebar` to play.

Interpolation

Interpolate

Reference

Mode::	Draw and Edit Modes
Tool::	*Toolbar* ▸ *Interpolate*
Shortcut::	`Ctrl-E`

When you are animating simple shapes you can use the interpolate tool to automatically add new breakdown keyframes.

See Interpolate tool for more details.

Interpolate Sequence

Reference

Mode::	Draw and Edit Modes
Menu::	*Header* ▸ *Interpolate*
Shortcut::	`Shift-Ctrl-E`

Interpolate strokes between the previous and next keyframe by adding *multiple* keyframes. When you are on a frame between two keyframes and click the sequence button a breakdown keyframe will be added on every frame between the previous and next keyframe.

Step
: The number of frames between generated interpolated frames.

Layer
: Restrict the interpolation to Active or All layers.

Only Selected *Edit Mode*
: When enabled, only selected strokes will be interpolated.

Exclude Breakdowns
: Exclude existing Breakdowns keyframes as interpolation extremes.

Flip Mode
: Invert strokes start and end. Automatic will try to found the right mode for every stroke.

Smooth
: Amount of smoothing to apply to interpolated strokes for reducing jitter/noise.

Iterations
: Number of time to smooth newly created strokes.

Type
: Interpolation method to use for the sequence.

Animation Tools

Insert Blank Keyframe (Active Layer)

Reference

Mode::	Draw Mode, Edit Mode, Sculpt Mode
Menu::	*Stroke ▸ Animation ▸ Insert Blank Keyframe (Active Layer)*
Shortcut::	`Shift-I`

Add a new blank keyframe to the active layer at the current frame. If there is already a keyframe at the current frame, a new blank keyframe will be added on the next frame.

All Layers
: When enabled, Blank keyframe will be created on all layers, not only the active one.

Duration
: The number of blank frames to insert.

Insert Blank Keyframe (All Layers)

Reference

Mode::	Draw Mode, Edit Mode, Sculpt Mode
Menu::	*Stroke ▸ Animation ▸ Insert Blank Keyframe (All Layers)*

Same as Insert Blank Keyframe (Active Layer) but *All Layers* is enabled by default.

Duplicate Active Keyframe (Active Layer)

Reference

Mode::	Draw Mode, Edit Mode, Sculpt Mode
Menu::	*Stroke ▸ Animation ▸ Duplicate Active Keyframe (Active Layer)*

Duplicates the strokes on the last keyframe by copying them to the current frame.

Mode
: Pick which layers to duplicate.

Active::	Duplicate only the active layer.
All::	Duplicate all the layers.

Duplicate Active Keyframe (All Layers)

Reference

Mode::	Draw Mode, Edit Mode, Sculpt Mode
Menu::	*Stroke ▸ Animation ▸ Duplicate Active Keyframe (All Layers)*

Same as Duplicate Active Keyframe (Active Layer) but the *Mode* is set to *All* by default.

Delete Active Keyframe (Active Layer)

Reference

Mode::	Draw Mode, Edit Mode, Sculpt Mode
Menu::	*Stroke ▸ Animation ▸ Delete Active Keyframe (Active Layer)*

Shortcut:: `Alt-I`

Deletes the last keyframe in the Dope Sheet or the current keyframe if you are on one.

Type
: Pick which layer to delete keyframes.

Active Frame:: Deletes current frame in the active layer.
All Active Frames:: Delete active frames for all layers.

Delete Active Keyframe (All Layers)

Reference

Mode:: Draw Mode, Edit Mode, Sculpt Mode
Menu:: *Stroke ▸ Animation ▸ Delete Active Keyframes (All Layers)*
Shortcut:: `Shift-Delete`

Same as Duplicate Active Keyframe (Active Layer) but the *Type* is set to *All Active Frames* by default.

Interpolate Sequence

Reference

Mode:: Draw Mode, Edit Mode
Menu:: *Grease Pencil ▸ Interpolate Sequence*
Shortcut:: `Shift-Ctrl-E`

Interpolate strokes between the previous and next keyframe by adding *multiple* keyframes. A breakdown keyframe will be added on every frame between the previous and next keyframe.

Step
: umber of frames between generated interpolated frames.

Layer
: Layers included in the interpolation.

Exclude Break Downs
: Exclude existing Breakdowns keyframes as interpolation extremes.

Flip Mode
: Invert destination stroke to match start and end with source stroke.

Smooth
: Amount of smoothing to apply to interpolated strokes, to reduce jitter/noise.

Iterations
: Number of times to smooth newly created strokes.

Type
: Interpolation method to use the next time *Interpolate Sequence* is run.

Bake Mesh to Grease Pencil

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	*Object* ▸ *Animation* ▸ *Bake Mesh to Grease Pencil…*

Converts each frame of a mesh animation within a selected frame range to a Grease Pencil object keyframed strokes. The *Bake Action* operator computes the final animation of the selected objects with all those modifiers, drivers, and constraints applied, and keyframes the result.

Target Object

Determines if the image empty is kept or replaced.

New Object: Creates a new grease pencil object and keeps the image empty Selected Object: Replaces the image empty with the grease pencil object.

Start Frame, End Frame
: Start/End frame for the baking process.

Step
: Frame steps for the baking process

Thickness
: Strokes thickness.

Threshold Angle
: Threshold value that determine the strokes end.

Stroke Offset
: Sets offset to separate strokes from filled strokes.

Only Seam Edges
: Convert only edges marked as seam.

Export Faces
: Convert faces as filled strokes.

Only Selected Keyframes
: Convert only the selected keyframes.

Target Frame
: Target destination frame for the baked animation.

Projection Type
: Sets the projection type to use for the converted strokes.

Bake Object Transform to Grease Pencil

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	*Object ▸ Animation ▸ Bake Object Transform to Grease Pencil*

Applies all transform animation at Object level within a selected frame range to Grease Pencil object keyframes.

Start Frame, End Frame
: Start/End frame for the baking process.

Step
: Frame steps for the baking process.

Only Selected Keyframes
: Convert only the selected keyframes.

Target Frame
: Target destination frame for the baked animation.

Projection Type
: Sets the projection type to use for the converted strokes.

Grease Pencil Object Modes

Object Modes allow editing different aspects of Grease Pencil objects. These modes are specifically tailored to the Grease Pencil object, unlike the more general modes, which work for other object types (with the exception of object mode which is the same for all objects).

- Draw Mode

Draw Mode is where new strokes are created. Strokes are directly sketched on a canvas, using different tools and brushes.

- Sculpt Mode

Sculpt Mode can be used to deform and shape existing strokes more organically. Strokes can be smoothed, deformed, or reshaped, adding fluidity and dynamism to drawings.

- Edit Mode

Edit Mode allows modifying individual strokes and points of Grease Pencil objects. This mode is ideal for fine-tuning linework, adjusting shapes, and refining details.

- Vertex Paint Mode

Vertex Paint Mode allows adding color the vertices of strokes directly. This mode is useful for adding shading, gradients, or detailed color effects providing finer control over the drawing's appearance.

- Weight Paint Mode

Weight Paint Mode allows assigning vertex weights to strokes. This is crucial for rigging and animating characters, ensuring smooth, and precise deformations based on the painted weights.

- Object Mode

This mode allows working with the entire Grease Pencil object as a whole. It's used for overall transformations and managing the placement of the object within the scene.

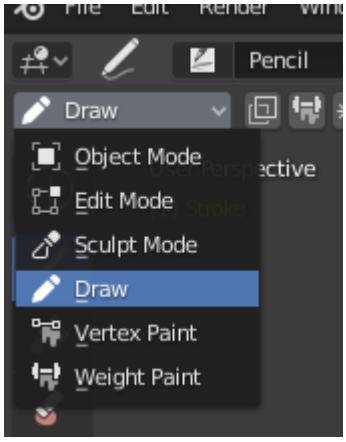
Draw Mode

- Introduction
  - Strokes Location & Orientation Controls
  - Drawing Options
- Brushes
  - Draw Brushes
  - Fill Brushes
  - Erase Brushes
- Drawing Tools
- Tools
  - Brush Tool
  - Erase Tool
  - Fill Tool
  - Trim Tool
  - Eyedropper
  - Line Tool
  - Polyline Tool
  - Arc Tool
  - Curve Tool
  - Box Tool
  - Circle Tool
  - Interpolate
- Tools Settings
  - Brush Asset
  - Brush Settings
  - Color
- Stroke Placement
- Drawing Plane
- Drawing Operations
  - Active Layer
  - Animation
  - Interpolate Sequence

Introduction

Draw Mode is the mode in Grease Pencil that allows you to draw in the 3D Viewport. This mode is actually the only one in which new strokes can be created.

Already made strokes can not be selected in Draw Mode, for editing strokes you must use the Edit Mode or Sculpt Mode.



3D Viewport Mode selector: Draw Mode.

Draw Mode is selected with the *Mode* menu in the 3D Viewport header. Once Draw Mode is activated, the Toolbar of the 3D Viewport will change to Draw Mode specific panels. Also a circle with the same color as the active material will appear and follow the location of the cursor in the 3D Viewport.

To create new strokes you have to select one of the drawing tools in the Toolbar. The most common one is the Draw tool for free-hand drawings but there are many other tools for drawing, filling areas and erasing strokes. There are also some tools to create primitives shapes like lines, arcs, curves, boxes and circles.

See Toolbar for more details.

Strokes Location & Orientation Controls

Drawing in a 3D space is not the same as drawing on a flat canvas. When drawing with Grease Pencil you have to define the location and orientation of the new strokes in the 3D space.



3D Viewport header Controls for strokes.

Stroke Placement

The Stroke Placement selector defines the new strokes location in 3D space.

See Stroke Placement for more information.

Drawing Planes

The Drawing Planes selector defines the plane (orientation) to which the new strokes will be restricted.

See Drawing Planes for more information.

Drawing Options



General drawing options.

Multiframe
: Allows to draw on several frames at the same time.

  See Multiframe for more information.

Additive Drawing

When creating new frames adding strokes with drawing tools, the strokes from the previous/active frame are include as a basis for the new one. When erasing existing strokes using Additive Drawing a new keyframe will be added.

AutoMerge
: Joins new strokes with the beginning or end of previously drawn strokes in the active layer.

Add Weight Data
: When enabled, weight data is added to new strokes according to the current vertex group and weight. If there is no vertex group selected, no weight data is added.

  Useful for example in cut-out animation for adding new drawing on the same vertex group without the need to creating it afterwards.

  See Weight Paint Mode for more information.

Draw on Back
: When enabled, new strokes are drawn below of all strokes in the layer. For example when you want to paint with a fill material below line strokes on a character and they are on the same layer.

Brushes

There are a number of brushes for draw mode bundled in the *Essentials* asset library. This is an overview of all of them.

Draw Brushes

Draw brushes are the special type of brushes that uses Grease Pencil for drawing tools. The brush can be changed in the Tool Settings. The different draw brushes (pencil, Ink, marker, etc.) are settings variations of the same *Draw Brush*. You can create many brushes, each with unique settings to get different artistic result while drawing.

Fill Brushes

Fill brushes are the special type of brushes that uses Grease Pencil for the *Fill* tools. The brush can be changed in the Tool Settings. The different fill brushes are settings variations of the same *Fill Brush*. You can create many brushes, each with unique settings to get different result when filling areas.

Erase Brushes

Erase brushes are the special types of brushes that uses Grease Pencil for *Erase* tools. The brush can be changed in the Tool Settings. Soft and hard eraser brushes are settings variations of the same *Erase Brush*. You can create many brushes, each with unique settings to get different effects while erasing. The *Erase Brush* has also other two special eraser types: point and stroke.

Draw Brushes

Reference

Mode::	Draw Mode
Brush::	*Asset Shelf* ▸ *Draw*

The Draw brush allows you to draw free-hand strokes.

Brush Settings

Material
: Data-block selector for the material.

Radius
: The radius of the brush in pixels.

  `F` allows you to change the brush size interactively by dragging the pointer or by typing a number then confirm.

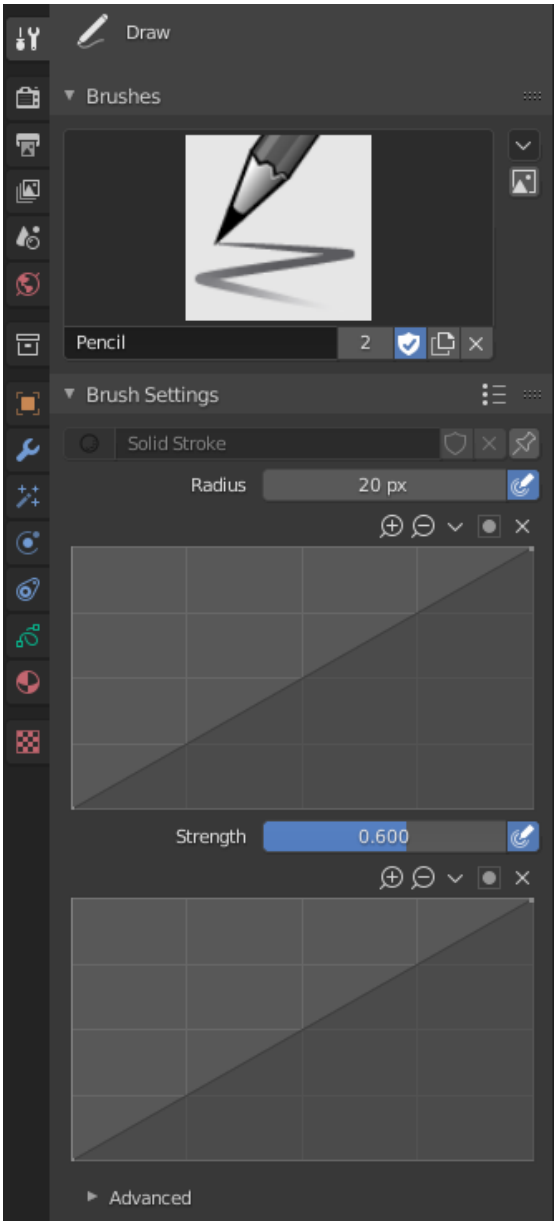
  (Size Pressure)
  : Adjusts the radius based on the stylus pressure when using a Graphics Tablet. The gradient of the pressure can be customized using the curve widget.

Strength
: Control the stroke transparency (alpha). From fully transparent (0.0) to fully opaque (1.0).

  You can change the brush strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the pointer and then `LMB`. You can also enter the size numerically.

  (Strength Pressure)

Adjusts the strength based on the stylus pressure when using a Graphics Tablet. The gradient of the pressure can be customized using the curve widget.



Caps Type

The shape of the start and end of the stroke.

Round:: Strokes start and stop with a curved shape.

Flat:: Strokes start and stop with a straight cutoff.

Advanced

Input Samples

Controls how often the input device is read to generate points on the stroke. Higher values give a higher precision (more points) but produce an irregular stroke, while lower values give a lower precision (fewer

points) but produce a soften stroke. (0 disabled extra input device samples.)

You have to set up this value according to your input device to obtain the right balance between accuracy and softness for your strokes. See Input Device for more information.

Active Smooth
: The number of smoothing iterations to apply to the stroke while drawing.

Angle
: Direction of the input device that gives the maximum thickness to the stroke (0° for horizontal).

Factor
: Amount of thickness reduction when the stroke is perpendicular to the *Angle* value.

Hardness
: Amount of transparency (alpha) to apply from the border of the point to the center. Works only when the brush is using stroke materials of *Dot* or *Box* style.

Aspect X, Y
: Controls the width and height of the alpha gradient.

Stroke

Post-Processing

Post-processing methods that are executed on the strokes when you finished drawing, right after releasing the `LMB` or `Pen` tip. You can toggle the use of post-processing using the checkbox in the section panel header.

Smooth
: Strength of smoothing process on the points location along the stroke.

Iterations
: The number of smoothing iterations to apply to the stroke.

Subdivision Steps
: Number of subdivisions to apply to newly created strokes.

Simplify
: Reduces final points numbers in the stroke with an adaptive algorithm.

Trim Strokes End
: Automatically trim intersection strokes ends.

Outline
: Activate the conversion of the newly created stroke to its outline.

  Material
  : Material used for outline stroke.

  Thickness
  : Thickness used for outline stroke.

Randomize

Adds randomness to the position of the points along the stroke. You can toggle the use of Randomize using the checkbox in the section panel header.

Radius
: The amount of randomness to apply using the pressure of the input device.

Strength
: The amount of randomness to apply to the stroke strength value (alpha).

UV
: The amount of randomness to apply to the UV rotation.

Hue, Saturation, Value
: Randomizes the hue, saturation, and value of the stroke's Color.

Jitter
: The amount of jittering to add to the stroke.

Common Options

Stroke Random (stroke icon)
: Use randomness only at stroke level.

(Use Pressure)
: Uses the stylus pressure to control how strong the effect is. The gradient of the pressure can be customized using the curve widget.

Stabilize Stroke

Stabilize Stroke helps to reduce jitter of the strokes while drawing by delaying and correcting the location of points. You can toggle the use of *Stabilize Stroke* using the checkbox in the section panel header.

Radius
: Minimum distance from the last point before the stroke continues.

Factor
: A smooth factor, where higher values result in smoother strokes but the drawing sensation feels like as if you were pulling the stroke.

Cursor

The cursor can be disabled by toggling the checkbox in the *Cursor* header.

Show Fill Color while Drawing
: Shows the brush linked material color in the viewport.

Tint Brush

Reference

Mode::	Draw Mode
Brush::	*Brush ▸ Tint*

The Tint brush allows you to paint onto strokes point mixing the material base color with a selected color.

Brush Settings

Mode
: Defines how Color Attributes affect to the strokes.

Stroke and Fill::	Color Attributes affects both the Stroke and Fill materials.
Stroke::	Color Attributes affects the Stroke material only.
Fill::	Color Attributes affects the Fill material only.

Usage

Selecting a Brush, Color & Mode

In the Tool Settings select the brush, color and mode to use with the tool.

You can configure the brush main settings included in the Tool Settings for convenience. For the vertex paint brushes configuration and settings see Vertex Paint Brush.

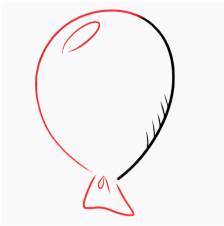
`Ctrl-LMB` erase the Color Attribute.

Painting

Click and hold LMB or use the pen tip to paint onto the stroke points.

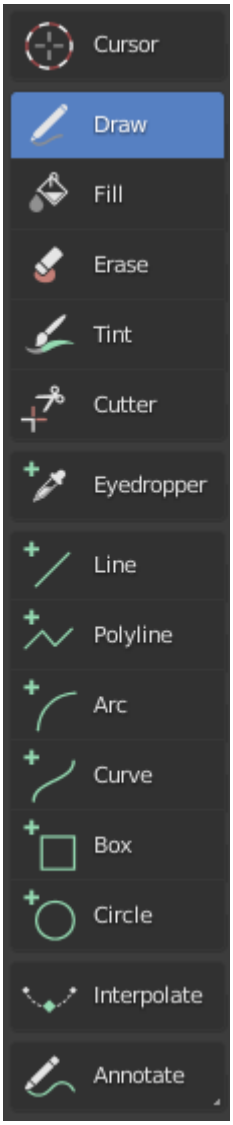
Vertex painting stroke points.







Drawing Tools



Cursor
: Change the location of the 3D Cursor.

Brush
: Tool to use for any of the drawing brushes.

Fill
: Automatic fill closed strokes areas.

Erase
: Erase strokes.

Box
: Draw rectangular shapes.

  Circle
  : Draw oval shapes.

  Line
  : Draw straight lines.

  Polyline
  : Draw straight multiple lines.

  Arc
  : Draw simple arcs.

  Curve
  : Draw complex Bézier style curves.

Trim
: Cut strokes in between others.

Eyedropper

: Eyedropper to create new materials or palette color based on sampled colors in the 3D Viewport.

Interpolate `Ctrl-E`

: Automatically create a breakdown keyframe between two normal keyframes.

Tools

- Brush Tool
  - Tool Settings
  - Usage
- Erase Tool
  - Tool Settings
  - Usage
- Fill Tool
  - Tool Settings
  - Usage
- Trim Tool
  - Tool Settings
  - Usage
- Eyedropper
  - Tool Settings
  - Usage
- Line Tool
  - Tool Settings
  - Usage
- Polyline Tool
  - Tool Settings
  - Usage
- Arc Tool
  - Tool Settings
  - Usage
- Curve Tool
  - Tool Settings
  - Usage
- Box Tool
  - Tool Settings
  - Usage
- Circle Tool
  - Tool Settings
  - Usage

- Interpolate
  - Usage
  - Tool Settings

Brush Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Brush*

Tool to free form draw Grease Pencil strokes using any of the *Draw* type brushes.

Tip

Activating a brush asset from an asset shelf or brush selector will also activate this tool for convenience.

Tool Settings

Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Eraser

Default Eraser Brush
: Select a brush to use as eraser for quickly alternating with the main brush using `Ctrl-LMB`.

Color

Settings to determine the color of strokes.

See Color

Usage

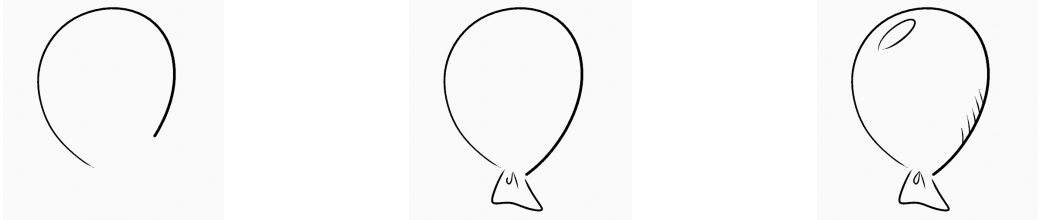
Selecting a Brush and Material

In the Tool Settings select the brush, material and color type to use with the tool. The Draw tool uses *Draw Brush* types. See Brush Settings for more information.

Free-hand Drawing

Click and hold `LMB` or use the pen tip to make free-hand drawing on the viewport.

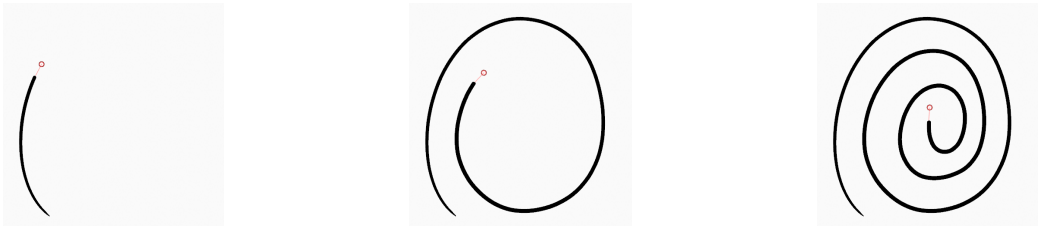
Drawing free-hand strokes.



Stabilize Stroke

`Shift-LMB` toggle the use of Stabilize Stroke on the brush to have more control while drawing and get smoother lines.

Drawing strokes using *Stabilize Stroke*.



Straight Lines

`Alt–LMB` Constrains the drawing of the strokes to horizontal or vertical straight lines.

Switching to the Erase Tool

`Ctrl–LMB` changes temporally to the active Erase tool. See Erase Tool for more information.

You can also use `B` to delete all the points in the selected drawing area.

Erase Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ▸ *Brush*

The Erase tool erases already drawn strokes.

The Erase tool uses any of the Grease Pencil *Erase* draw mode brushes. Activating a brush from an asset shelf or brush selector will also activate this tool for convenience.

Tool Settings

Brush Asset

The asset selector can be used to open a pop-up asset browser to select the active brush asset for the tool.

See Asset Operators for more information.

Brush Settings

Radius
: The radius of the brush in pixels.

  F allows you to change the brush size interactively by dragging the pointer or by typing a number then confirm.

  (Size Pressure)
  : Adjusts the radius based on the stylus pressure when using a Graphics Tablet.

Strength
: Control how much will affect the eraser has on the stroke transparency (alpha).

  You can change the brush strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the pointer and then `LMB`. You can also enter the size numerically.

  (Strength Pressure)
  : Adjusts the strength based on the stylus pressure when using a Graphics Tablet.

Mode
: Determines how the erase tool behaves.

  Dissolve:: To simulate a raster type eraser, this eraser type affects the strength and thickness of the strokes before actually delete a point.
  Point:: Delete one point at a time.
  Stroke:: Delete an entire stroke.

Cursor

The cursor can be disabled by toggling the checkbox in the *Cursor* pop-over menu.

Usage

Selecting a Brush

In the Tool Settings select the brush to use with the tool. The Erase tool uses Erase Brush types (soft, point and stroke).

Dissolve Erasing

- Select an erase brush of type Soft/Hard.

- Adjust brush settings.
- Click and hold `LMB` or use the `Pen` tip to delete strokes on the viewport.



Original drawing.



The eraser affect the transparency of the strokes.



Final result.

Point Erasing

- Select an erase brush of type Point.
- Adjust brush settings.
- Click and hold `LMB` or use the `Pen` tip to delete strokes on the viewport.



Original drawing.



The eraser delete one point at a time.



Final result.

Stroke Erasing

- Select an erase brush of type Stroke.
- Adjust brush settings.
- Click and hold `LMB` or use the `Pen` tip to delete strokes on the viewport.



Original drawing.



The eraser delete one stroke at a time.



Final result.

Fill Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ▸ *Fill*

The Fill tool is used to automatically fill closed strokes areas.

The Fill tool uses any of the Grease Pencil *Fill* draw mode brushes. Activating a brush from an asset shelf or brush selector will also activate this tool for convenience.

Tool Settings

Brush Asset

The asset selector can be used to open a pop-up asset browser to select the active brush asset for the tool.

See Asset Operators for more information.

Brush Settings

You can also configure the brush main settings exposed on the Tool Settings for convenience.

Direction `Ctrl`
: The portion of area to fill.

Normal::	Fills the area inside the shape under the cursor.
Inverted::	Fills the area outside the shape under the cursor.

Precision

Multiplier for fill boundary accuracy. Higher values are more accurate but slower.

Dilate/Contract
: Size in pixels to expand or shrink the fill area from the strokes boundary.

Thickness
: The thickness radius of the boundary stroke in pixels.

Advanced

Boundary
: Sets the type of fill boundary limits calculation to perform.

All::	Use the thickness of the strokes and the editing lines together.
Strokes::	Use only the thickness of the strokes (ignore edit lines).
Edit Lines::	Use only the edit lines (ignore strokes).

Show Lines (eye icon)
: Toggle show auxiliary lines to see the fill boundary.

Layers
: Determines which Layers are used for boundary strokes.

Visible::	Calculates boundaries based on all visible layers.
Active::	Calculates boundaries based on the active layer.
Layer Above::	Calculates boundaries based on the layer above the active layer.
Layer Below::	Calculates boundaries based on the layer below the active layer.
All Above::	Calculates boundaries based on all layers above the active layer.
All Below::	Calculates boundaries based on all layers below the active layer.

Simplify
: Number of simplify steps to apply to the boundary line. Higher values reduce the accuracy of the final filled area.

Ignore Transparent
: When enabled, strokes with transparency does not take into account on fill boundary calculations.

  The value slider controls the threshold to consider a material transparent.

Limit to Viewport
: When enabled, fill only visible areas in the viewport.

Gap Closure

Gap closure lines are automatic temporarily lines that help to close gaps on the strokes.

Size
: Control the Size of the line extension or the circumference to use to calculate the lines that will close the gaps.

Mode `S`
: Sets the type of Gap closure method to use.

Radius::	Uses the Radius of circumference of opened nearest points to calculate a line that close the gap.
Extend::	Extends the opened strokes to close gaps.

Visual Aids
: Toggle show closure lines helper.

Strokes Collision `D`
: Check if extend lines collide with strokes, stopping the extension if a collision is detected.

Only Collide Lines
    Use for closing gaps only if the extend strokes collide.

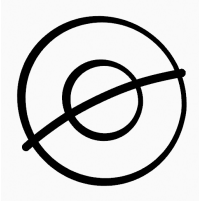
Usage

Selecting a Brush and Material

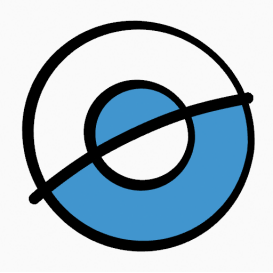
In the Tool Settings select the brush, material and color type to use with the tool. The Fill tool uses *Fill Brush* types. See Brush Settings for more information.

Filling Areas

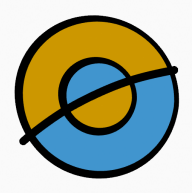
Click `LMB` in a closed stroke area. The tool will automatically calculate the boundary and create a new closed stroke filled with the material selected.



Original Drawing.



Use the fill tool to leak materials on closed areas.



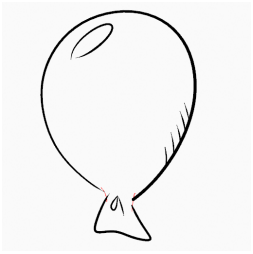
Final filled drawing.

Boundary Strokes

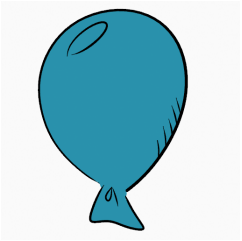
If you have a large gap in an area that you want fill, you can add boundary strokes manually, a temporary auxiliary lines for closing open shapes. To create a boundary stroke use `Alt-LMB` and draw a line to close the desired area.



Original drawing.



Add boundary strokes to close open areas (red lines).



Use the Fill tool to leak material on the new closed area.

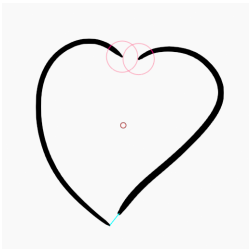
When you are satisfied with the fill result you can delete the boundary strokes using the *Clean Up* tool in the Grease Pencil Menu in Edit Mode.

Automatic Gap Closure

A more automatic way to close gaps in an area that you want fill is using temporarily helper lines. There are two method to use “Radius” or “Extend”

Radius use temporary auxiliary lines calculated from the radius of nearby open points to close open shapes. Set the size more than zero to control the circle size over opened points (the circle will disappear when the line close the gap). Click over the area you want to be filled and change the length of the strokes using `PageUp` `PageDown` or `Wheel`. When you are satisfied with the length and you are sure the temporarily strokes cross each other, click again to fill the area.







Original Drawing.

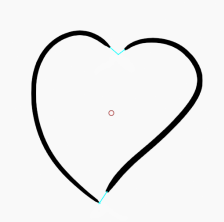
Use Radius mode to close open areas (Red circles and cyan lines).

Use Fill Tool to leak material on the new closed area.

Extend use temporary auxiliary lines extending the actual strokes ends for closing open shapes. Set the size more than zero to use the extended lines, click over the area you want to be filled and change the length of the strokes using `PageUp`/`PageDown`, `Wheel` or a pen's `MMB`. When you are satisfied with the length and you are sure the temporarily strokes cross each other, click again to fill the area.



Original Drawing.



Use Extend mode to close open areas (cyan lines).



Use Fill Tool to leak material on the new closed area.

Trim Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Trim*

The Trim tool delete points in between intersecting strokes.

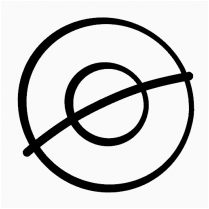
Tool Settings

Flat Caps
: Mark newly created End Caps as *Flat*.

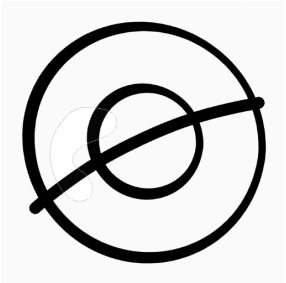
Threshold
: Determine the threshold for stroke intersections.

Usage

Draw a dotted line around the strokes you want to trim. After releasing the mouse button all the points on the selected strokes will be deleted until another intersecting stroke is found.



Original drawing.



Lasso Selecting the strokes to be



Final result.

trimmed.

Eyedropper

Reference

Mode:: Draw Mode
Tool:: *Toolbar ▸ Eyedropper*

The Eyedropper tool is used to create materials or palette color based on sampled colors in the 3D Viewport.

Tool Settings

Material:: Create a new material with the Stroke Base Color to be the sampled color.
Material Mode
The color transformation will be applied on the stroke and/or the fill color.
Stroke:: Only paint over strokes.
Fill:: Only paint over fill areas.
Both:: Paint over strokes and fill areas

Palette:: Add a new color to the color palette based on the sampled color.

Brush:: Sets the brush color to the sampled color.

Usage

- `LMB` Create a stroke material.
- `Shift-LMB` Create a fill material.
- `Shift-Ctrl-LMB` Create both a stroke and fill material.
- Holding `LMB` and dragging accumulates the average color under the mouse cursor.

Line Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Line*

The Line tool create straight lines using any of the *Draw* type brushes.

Tool Settings

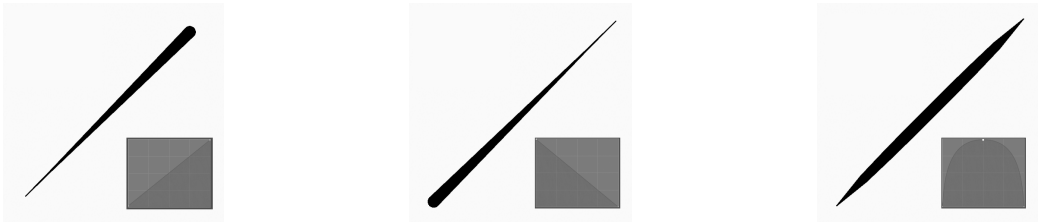
You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the line.

Different thickness profile samples.



Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Color

Settings to determine the color of strokes.

See Color

Usage

Selecting a Brush and Material

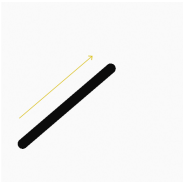
In the Tool Settings select the brush, material and color type to use with the tool. The Line tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Lines

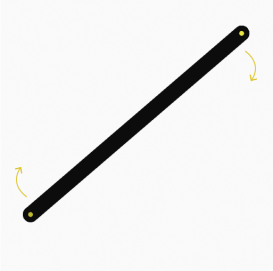
1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. After releasing you can move the start and end point by clicking and dragging on the yellow manipulators.
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can use `Shift` to snapping the line to horizontal, vertical or 45° angle or use `Alt` to create the line from a center point.

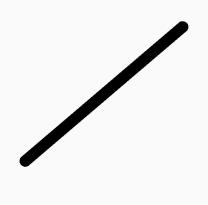
`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final line.



click and dragging the start point.



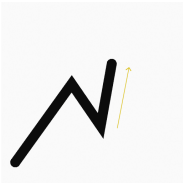
Moving start and end points with manipulators.



The line after confirming.

Extruding

Before confirming you can use `E` to extrude the end point of the line to generate multiple connected lines.



End point extruding.



Moving the end point of the last line with the manipulator.



The connected lines after confirming.

Polyline Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Polyline*

The Polyline tool creates multiple straight lines using any of the *Draw* type brushes.

Tool Settings

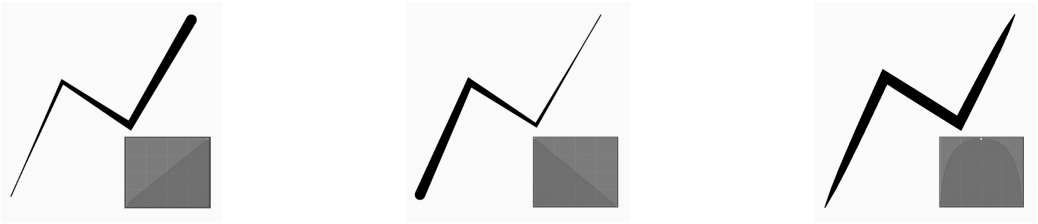
You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the line.

Different thickness profile samples.



Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Color

Settings to determine the color of strokes.

See Color

Usage

Selecting a Brush and Material

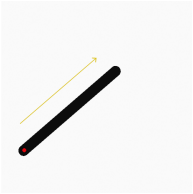
In the Tool Settings select the brush, material and color type to use with the tool. The Line tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Polylines

1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. Click multiple times on different locations to create multiple connected lines.
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can use `Shift` to snapping the line to horizontal, vertical or 45° angle.

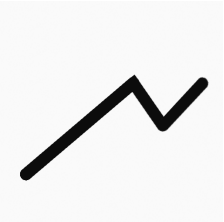
`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final line.



click and dragging the start point.



Click multiple times to create multiple connected lines.



The polyline after confirming.

Arc Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Arc*

The Arc tool create simple arcs using any of the *Draw* type brushes.

Tool Settings

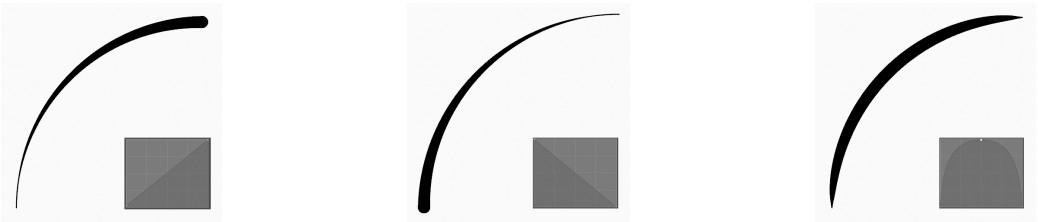
You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the arc.

Different thickness profile samples.



Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Color

Settings to determine the color of strokes.

See Color

Usage

Selecting a Brush and Material

In the Tool Settings select the brush, material and color type to use with the tool. The Arc tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Arcs

1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. After releasing you can tweak the arc using a single cyan manipulator (hand icon).
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can use `Shift` to make a perfect arc, use `Alt` to create the arc from a center point or `M` to flip.

`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final arc.



click and dragging the start point.



Tweaking arc with the manipulator.



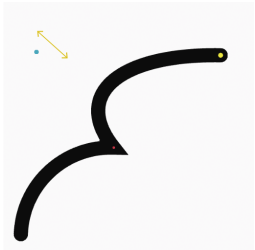
The arc after confirming.

Extruding

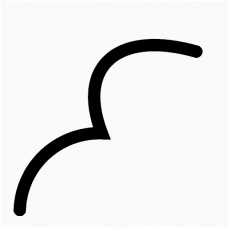
Before confirming you can use `E` to extrude the end point of the arc to generate multiple connected arcs.



End point extruding.



Tweaking the last arc with the manipulator.



The connected arcs after confirming.

Curve Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar ▸ Curve*

The Curve tool create complex Bézier style curves using any of the *Draw* type brushes..

Tool Settings

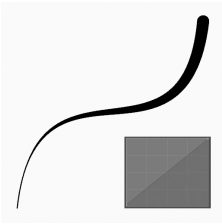
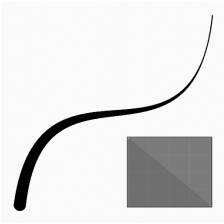
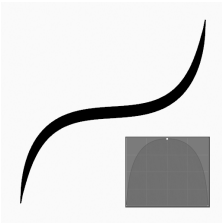
You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the curve.

Different thickness profile samples.

Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Color

Settings to determine the color of strokes.

See Color

Usage

Selecting a Brush and Material

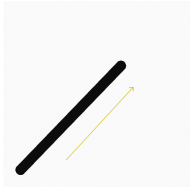
In the Tool Settings select the brush, material and color type to use with the tool. The Curve tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Curves

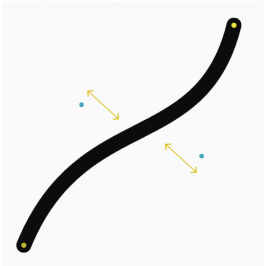
1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. After releasing you can tweak the curve using two cyan Bézier like manipulators.
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can hold `Shift` to use only one manipulator to tweak the curve (like the Arc tool), use `Alt` to create the arc from a center point.

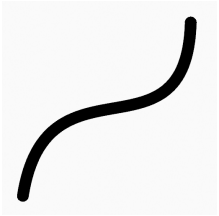
`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final curve.



click and dragging the start point.



Tweaking curve with the manipulators.



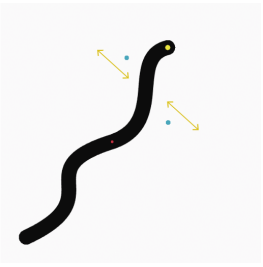
The curve after confirming.

Extruding

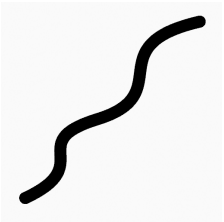
Before confirming you can use `E` to extrude the end point of the curve to generate multiple connected curves.



End point extruding.



Tweaking the last curve with the manipulators.



The connected curves after confirming.

Box Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ▸ *Box*

The Box tool create rectangular shapes.

Tool Settings

You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the line.

Usage

Selecting a Brush and Material

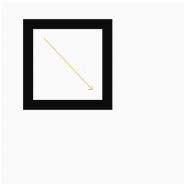
In the Tool Settings select the brush, material and color type to use with the tool. The Box tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Boxes

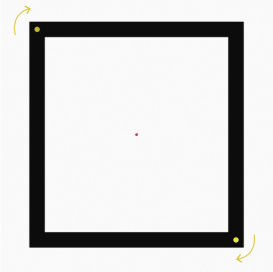
1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. After releasing you can move the start and end point by clicking and dragging on the yellow manipulators.
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can use `Shift` to make a perfect square or use `Alt` to create the box from a center point.

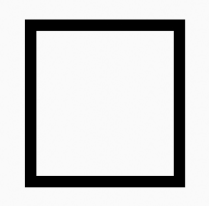
`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final box.



click and dragging the start point.



Moving start and end points with manipulators.



The box after confirming.

Circle Tool

Reference

Mode::	Draw Mode
Tool::	*Toolbar* ‣ *Circle*

The Circle tool create oval shapes using any of the *Draw* type brushes.

Tool Settings

You can configure the brush main settings exposed on the Tool Settings for convenience. For the draw brushes configuration and settings see: Draw Brush.

Subdivisions
: The number of stroke points between each stroke edge.

Thickness Profile
: Use a curve widget to define the stroke thickness from the start (left) to end (right) of the stroke.

  Use Curve
  : When enabled, the stroke use a curve profile to control the thickness along the line.

Brush Asset

Picks the brush asset used by the tool.

See Brush Asset for more information.

See Draw Brushes for a detailed list of all draw brushes and their options.

Brush Settings

Parameters to control to look of the stroke.

See Draw Brushes for details.

Color

Settings to determine the color of strokes.

See Color

Usage

Selecting a Brush and Material

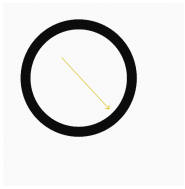
In the Tool Settings select the brush, material and color type to use with the tool. The Circle tool uses *Draw Brush* types. See Brush Settings for more information.

Creating Circles

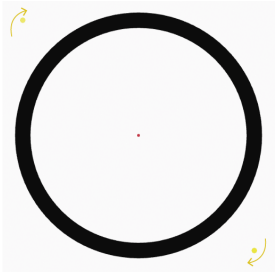
1. Click (`LMB` or the `Pen` tip) and drag the start point.
2. Release on the desired end point.
3. After releasing you can move the start and end point by clicking and dragging on the yellow manipulators.
4. Then confirm (`Return`/`MMB`) or cancel (`Esc`/`RMB`).

While dragging you can use `Shift` to make a perfect circle or use `Alt` to create the circle from a center point.

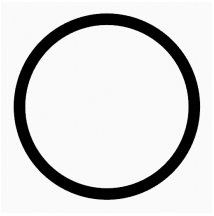
`NumpadPlus` and `NumpadMinus` or using the mouse `Wheel` will increase or decrease the amount of points in the final circle.



Click and dragging the start point.



Moving start and end points with manipulators.



The circle after confirming.

Interpolate

Reference

Mode::	Draw Mode
Tool::	*Toolbar ▸ Interpolate*

The Interpolate tool interpolates strokes between the previous and next keyframe by adding a *single* keyframe. When you are on a frame between two keyframes and click and drag a new breakdown keyframe will be added. This way you define the final interpolation for the new stroke.

Usage

Set the Playhead on the Timeline between the two keyframes you want to interpolate. Click and drag from left to right to set the desired interpolation percentage and release to confirm, a new breakdown keyframe will be added.

Tool Settings

Layer
: Restrict the interpolation to Active or All layers.

Only Selected *Edit Mode*
: When enabled, only selected strokes will be interpolated.

Exclude Breakdowns
: Exclude existing Breakdowns keyframes as interpolation extremes.

Flip Mode
: Invert strokes start and end. Automatic will try to found the right mode for every stroke.

Smooth

Amount of smoothing to apply to interpolated strokes for reducing jitter/noise.

Iterations

Number of time to smooth newly created strokes.

Tools Settings

- Brush Asset
  - Brush Types
- Brush Settings
- Color
  - Palette

Brush Asset

Brush
: The Data-Block Menu to select a preset brush type or a custom brush.

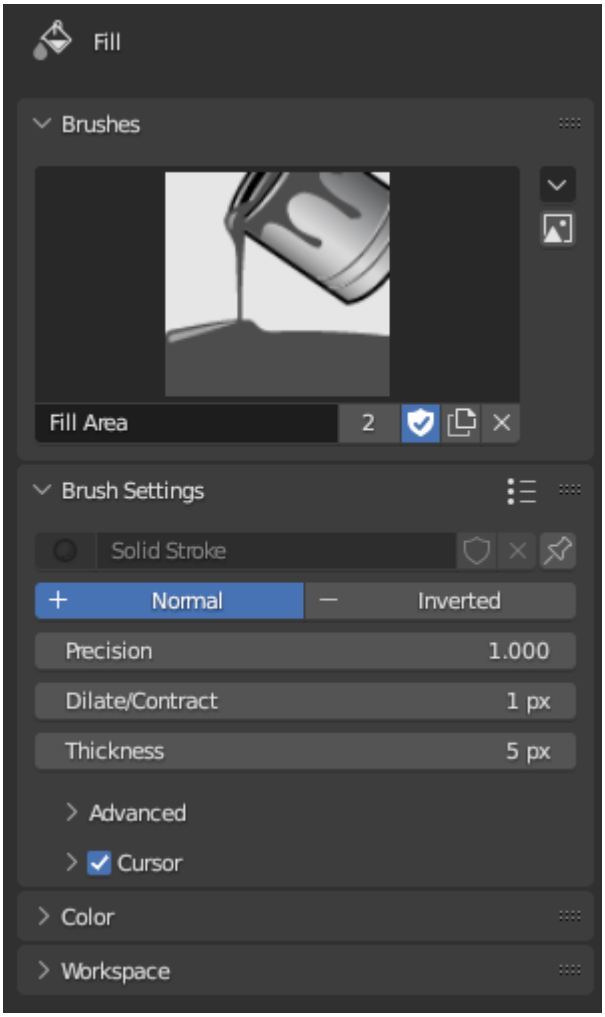
  Add Brush
  : When you add a brush, the new brush is a clone of the current one.

  Brush Specials
  : Reset Brush
    : Reset the current brush to its default settings.

    Reset All Brushes
    : Reset all brushes to their default settings.

    Custom Icon
    : Allows definition of a custom brush icon.

      Image Path
      : Defines the path to the image to use as custom icon.



Brush data-block panel.

Note

In order to save a custom brush in a blend-user, enable *Fake User*.

Brush Types

Draw Brushes

Draw brushes are the special type of brushes that uses Grease Pencil for drawing tools. The brush can be changed in the Tool Settings. The different draw brushes (pencil, Ink, marker, etc.) are settings variations of the same *Draw Brush*. You can create many brushes, each with unique settings to get different artistic result while drawing.

Fill Brushes

Fill brushes are the special type of brushes that uses Grease Pencil for the *Fill* tools. The brush can be changed in the Tool Settings. The different fill brushes are settings variations of the same *Fill Brush*. You can create many brushes, each with unique settings to get different result when filling areas.

Erase Brushes

Erase brushes are the special types of brushes that uses Grease Pencil for *Erase* tools. The brush can be changed in the Tool Settings. Soft and hard eraser brushes are settings variations of the same *Erase Brush*. You can create many brushes, each with unique settings to get different effects while erasing. The *Erase Brush* has also other two special eraser types: point and stroke.

Brush Settings

Material
: Data-block selector for the material. Except for the *Erase* tool of course.

Pin Material (pin icon)
: Pin the material to the brush.

  The final appearance of the strokes is a combination of the brush and material used, binding the material to the brush gives more control and avoids a lack of coordination between the two.

Color

Paint Mode
: Controls the source of the stroke color. The mode can be pinned to the brush by enabling the Pin icon in the Tool Settings header.

Material::	Use the stroke/fill base color material.
Color Attribute::	Use Color Attribute.

Color Picker
: Sets the primary brush color.

Color, Secondary Color
: The color of the brush. See Color Picker.

Mode
: The color transformation will be applied on the stroke and/or the fill color.

Stroke::	Only paint over strokes.
Fill::	Only paint over fill areas.
Stroke & Fill::	Paint over strokes and fill areas.

Mix Factor
: Mixing factor between the selected color and the base material color.

Palette

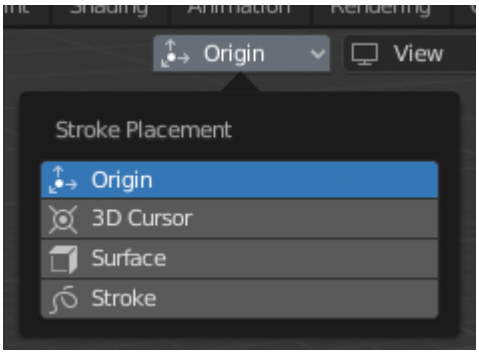
Active Color Palette. See Color Palette.

Stroke Placement

Reference

Mode:: Draw Mode
Header:: *Stroke Placement*

The Stroke Placement selector helps to select the location in which strokes are drawn.



Stroke Placement pop-over.

Note

The Stroke Placement selection only affects new strokes and does not affect the existing ones.

Origin:: Strokes are placed at Grease Pencil object origin.

3D Cursor:: Strokes are placed at 3D cursor.

Surface:: Strokes will stick on mesh surfaces.

Offset
: Distance from the mesh surface to place the new strokes.

Project Onto Selected
: Only project the strokes onto selected objects.

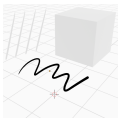
Stroke:: Strokes will stick on other strokes.

Target
: **All Points::** All the points of the new stroke sticks to other strokes.

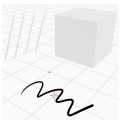
: **End Points::**

	Only the start and end points of the new stroke sticks to other strokes.
First Point::	Only the start point of the new stroke sticks to other strokes.

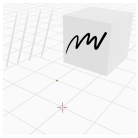
Stroke using different Stroke Placements.



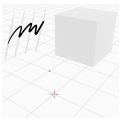
Origin.



3D Cursor.



Surface.



Stroke.

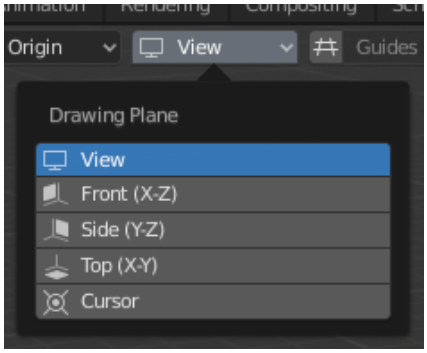
Drawing Plane

Reference

Mode:: Draw Mode and Sculpt Mode
Header:: *Drawing Plane*

The Drawing Planes selector helps to select the plane in which strokes are drawn.

To see which plane you are using when drawing strokes, you can enable *Canvas* in Viewport Overlays. See Viewport Display to know more about Canvas settings.



Drawing Planes pop-over.

Note

The Drawing Plane only affects new strokes and does not affect existing strokes.

View:: Strokes are drawn with the current 3D Viewport orientation.
Front (X-Z):: Strokes are drawn on the plane determined by the XZ axes (front view).
Side (Y-Z):: Strokes are drawn on the plane determined by the YZ axes (side view).
Top (X-Y):: Strokes are drawn on the plane determined by the XY axes (top view).
Cursor:: Strokes are drawn with the current 3D cursor orientation.

Stroke using different Drawing Planes with Canvas overlay activated.



Front.



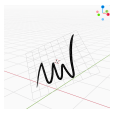
Side.



Top.



View.



Cursor.

Drawing Operations

Active Layer

Reference

Mode::	Draw Mode
Menu::	*Draw ▸ Active Layer*
Shortcut::	`Y`

Select the active layer.

Animation

Reference

Mode::	Draw Mode
Menu::	*Draw ▸ Animation*
Shortcut::	`I`

The stroke animation operations are described in the Animation section.

Interpolate Sequence

Reference

Mode::	Draw Mode
Menu::	*Draw ▸ Interpolate Sequence*

See Interpolate Sequence.

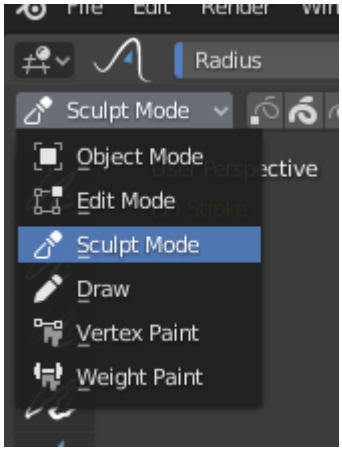
Sculpt Mode

- Introduction
  - Sculpt Mode
  - Sculpting Options
  - Keyboard Shortcuts
- Sculpting Brushes
- Sculpting Tools
- Brush Settings
  - Advanced
  - Cursor

Introduction

Sculpt Mode is similar to *Edit Mode* in that it is used to alter the shape of a drawing, but Sculpt Mode uses a very different workflow: Instead of dealing with individual elements (points and edit lines), an area of the model is altered using a brush. In other words, instead of selecting a group of points, Sculpt Mode manipulates the drawing in the brush region of influence.

Sculpt Mode



3D Viewport Mode selector: Sculpt Mode.

Sculpt Mode is selected from the *Mode* menu in the 3D Viewport header. Once Sculpt Mode is activated, the Toolbar of the 3D Viewport will change to Sculpt Mode specific panels. A red circle will appear and follow the location of the cursor in the 3D Viewport.

Sculpting Options



General sculpting options.

Selection Mask
: Sculpt Mode in Grease Pencil allows you to select points or strokes to restrict the effect of the sculpting tools to only a certain areas of your drawing.

  You can use the selection tools in the Toolbar for a quick selection. You can restrict sculpting only on the selected points or strokes with the Selection mode buttons. The three modes can be toggled with `1`, `2`, or `3` respectively.

Multiframe
: Sometimes you may need to modify several frames at the same time with the sculpting tools.

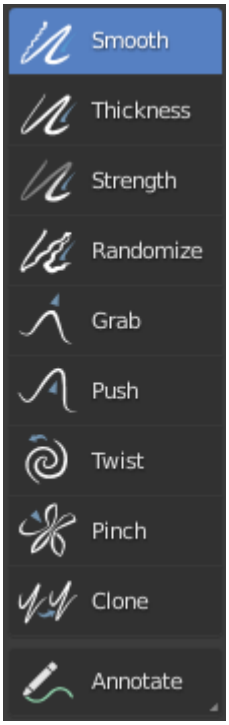
  You can activate multiframe edition with the Multiframe button next to the modes selector (faded lines icon). See Multiframe for more information.

Keyboard Shortcuts

- Invert stroke toggle `Ctrl`
- Change active material `U`

Sculpting Brushes

Brushes for Grease Pencil *Sculpt* mode bundled in the *Essentials* library. See Brush for more information.



Smooth Stroke
: Eliminates irregularities in the area of the drawing within the brush's influence by smoothing the positions of the points.

Thickness Stroke
: Increase or decrease the points thickness in the area of the drawing within the brush's influence.

Strength Stroke
: Increase or decrease the points transparency (alpha) in the area of the drawing within the brush's influence.

Randomize Stroke
: Add noise to the strokes in the area of the drawing within the brush's influence by moving points location in a random way.

Grab Stroke
: Used to drag a group of points around. Unlike the other brushes, Grab does not modify different points as the brush is dragged across the model. Instead, Grab selects a group of points on mouse-down, and pulls them to follow the mouse. The effect is similar to moving a group of points in Edit Mode with Proportional Editing enabled.

Pull Stroke
: Moves points in the direction of the brush stroke.

Twist Stroke
: Twist the points in counter-clockwise (CCW) or Clockwise (CW) rotation.

Pinch Stroke

Pulls points towards the center of the brush. The inverse setting is Inflate, in which points are pushed away from the center of the brush.

Clone Stroke

Adds copies of the strokes in the clipboard in the center of the brush. You have to copy the selected strokes you want into the clipboard with `Ctrl-C` before using the tool.

Sculpting Tools

Brush
: Tool to use for any of the sculpting brushes.

Annotate
: Draw free-hand annotation.

  Annotate Line
  : Draw straight line annotation.

  Annotate Polygon
  : Draw a polygon annotation.

  Annotate Eraser
  : Erase previous drawn annotations.

Brush Settings

Reference

Mode::	Sculpt Mode
Panel::	*Sidebar* ▸ *Tool* ▸ *Brush Settings*

Radius
: This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB` (the texture of the brush should be visible inside the circle). Typing a number then enter while using `F` allows you to enter the size numerically.

  (Size Pressure)
  : Adjusts the radius based on the stylus pressure when using a Graphics Tablet.

Strength
: Controls how much each application of the brush affects the model. For example, higher values cause the *Randomize* brush to add noise to the strokes more quickly, and cause the *Smooth* brush to soften the strokes more quickly.

  You can change the brush strength interactively by pressing `Shift-F` in the 3D Viewport and then moving the brush and then `LMB`. You can enter the size numerically also while in `Shift-F` sizing.

  (Strength Pressure)
  : Adjusts the strength based on the stylus pressure when using a Graphics Tablet.

Use Falloff
: When enabled, use Strength falloff for the brush. Brush Strength decays with the distance from the center of the brush.

Sculpt Strokes
: Filters the effect of the brush over the strokes. Applies to *Smooth* and *Randomize* brushes only.

Note

If there is no filter selected *Affect Position* is the default behavior.

Affect Position
: Toggles the brush effect on the position of the stroke points.

Affect Strength
: Toggles the brush effect on the strength (alpha) of the stroke points.

Affect Thickness
: Toggles the brush effect on the thickness of the stroke points.

Affect UV
: Toggles the brush effect on the UV rotation of the stroke points.

Direction
: The influence direction of the brush. This can be Add or Subtract.

Advanced

Auto-Masking
: Strokes
  : Affect only strokes under the cursor.

  Material
  : Affect only the active material.

  Layer
  : Affect only the active layer.

Cursor

The cursor can be disabled by toggling the checkbox in the *Cursor* header.

Color
: Set the color of the brush ring. Depending on the current mode there will be options to set a single Color or a Color for Adding/Subtracting.

Edit Mode

- Introduction
  - Accessing Stroke Editing Tools
- Selecting
  - Select Mode
  - Select All/None/Invert
  - Select Random
  - Select Alternated
  - Select More/Less
  - Select Similar
  - Select Linked
  - Select First/Last
- Editing Tools
- Grease Pencil Menu
  - Transform
  - Mirror
  - Snap
  - Active Layer
  - Animation
  - Interpolate Sequence
  - Duplicate
  - Copy
  - Paste
  - Weights
  - Show/Hide
  - Separate
  - Clean Up
  - Delete
- Stroke Menu
  - Subdivide
  - Subdivide and Smooth
  - Simplify
  - Trim
  - Join

    - Move to Layer
    - Assign Material
    - Set as Active Material
    - Arrange
    - Close
    - Toggle Cyclic
    - Set Caps
    - Switch Direction
    - Set Uniform Thickness
    - Set Uniform Opacity
    - Scale Thickness
    - Set Curve Type
    - Set Curve Resolution
    - Reset UVs
- Point Menu
    - Extrude
    - Smooth
    - Vertex Groups
    - Set Handle Type

Introduction

Blender provides a variety of tools for editing Grease Pencil strokes. These are tools used to add, duplicate, move and delete elements.

Note

Editing multiple Grease Pencil objects at once is currently not supported.

Accessing Stroke Editing Tools

These are available through the different tools in the Toolbar, the Stroke menu in the 3D Viewport header, and context menus in the 3D Viewport, as well as individual shortcut keys.

Toolbar

When you select a stroke and `Tab` into Edit Mode, the Toolbar changes from *Object Tools* to *Stroke editing Tools*. These are only some of the stroke editing tools.

Menus

The Stroke Menu is located in the header.

Context Menu

`RMB` brings up the complete Stroke Menu.

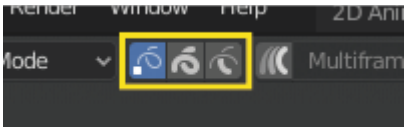
Selecting

Select Mode

Reference

Mode:: Edit Mode
Menu:: *3D Viewport Header ▸ Select Mode*
Shortcut:: 1, 2, 3

In Edit Mode there are three different selection modes. You can enter the different modes by selecting one of the three buttons in the header.



Edit Mode selection buttons.

Points:: To select individual points.
Strokes:: To select an entire stroke.
Segments:: To select all points that are between other strokes.



Points, stroke and in between stroke selection sample.

Select All/None/Invert

All these options have the same meaning and behavior as in Object Mode.

Select Random

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Random*

Randomly selects unselected points or strokes.

Ratio
: The likelihood of an unselected elements being selected. Note that, this is not the percentage amount of elements that will be selected.

Random Seed
: Seed used by the pseudo-random number generator.

Action
: Selection or deselection of elements.

Select Alternated

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *Select Alternated*

Selects alternate points in the selected strokes.

Select More/Less

Reference

Mode::	Edit Mode
Menu::	*Select* ▸ *More/Less*
Shortcut::	`Ctrl-NumpadPlus`, `Ctrl-NumpadMinus`

The purpose of these operators is to reduce or enlarge the current selection within a stroke (i.e. they will never "go outside" of a stroke or "jump" to another stroke in the same object).

More
: For each selected point, select *all* its linked points (i.e. one or two…).

Less
: For each selected point, if *all* points linked to this point are selected, keep this one selected. Otherwise, deselect it.

Hint

When *all* points of a stroke are selected, nothing will happen (as for *Less*, all linked points are always selected, and of course, *More* cannot add any). Conversely, the same goes when no points are selected.

Select Similar

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Similar*
Shortcut::	`Shift-G`

Select all strokes with similar characteristics.

Mode
: The characteristics to compare.

Layer::	Selects all the points/strokes with a similar layer index.
Material::	Selects all the points/strokes with a similar material index.
Vertex Color::	

Selects all the points/strokes with a similar vertex color.

Radius:: Selects all the points/strokes with a similar stroke radius.

Opacity:: Selects all the points/strokes with a similar layer opacity

Threshold
: How similar the selection must be.

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Linked*
Shortcut::	`L`, `Ctrl-L`

`L` (or `Ctrl-L` for all) will add to the selection the cursor's nearest control point, and all the linked ones, i.e. all points belonging to the same stroke.

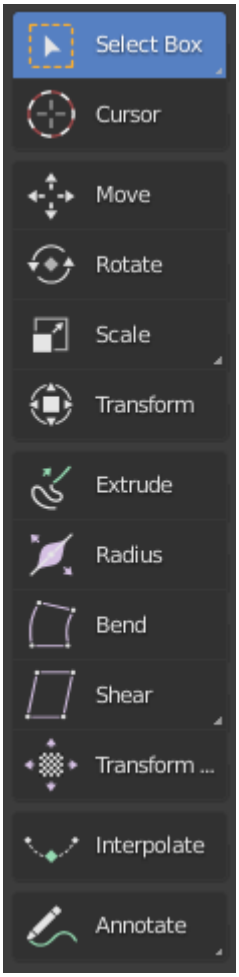
Select First/Last

Reference

Mode::	Edit Mode
Menu::	*Select ▸ First/Last*

These operators will toggle the selection of the first or last point(s) of the stroke(s) in the object. This is useful to quickly find the start of a stroke.

Editing Tools



Select
: Select or moved.

  Select Box
  : Select geometry by dragging a box.

  Select Circle
  : Select geometry by painting on it.

  Select Lasso
  : Select geometry by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform
: Tool to adjust the objects translation, rotations and scale.

Extrude E
: Extrusion tools duplicate points, while keeping the new geometry connected with the original points.

Radius `Alt-S`
: Expand or contract the thickness radius of the selected points.

Bend `Shift-W`
: Bend selected points between the 3D cursor and the pointer.

Shear `Shift-Ctrl-Alt-S`
: Shear selected points along the horizontal or vertical screen axis.

  To Sphere `Shift-Alt-S`
  : Move selected points outward in a spherical shape around the selected strokes' center.

Interpolate `Ctrl-E`
: Automatically create a breakdown keyframe between two normal keyframes.

Gradient
: Draw a line to set the fill material gradient for the selected strokes.

Annotate
: Draw free-hand annotation.

  Annotate Line
  : Draw straight line annotation.

  Annotate Polygon
  : Draw a polygon annotation.

  Annotate Eraser
  : Erase previous drawn annotations.

Grease Pencil Menu

Transform

Strokes can be edited by transforming the locations of points.

Move, Rotate & Scale

Reference

Mode::	Edit Mode
Tool::	*Toolbar* ▸ *Move, Rotate, Scale*
Menu::	*Grease Pencil* ▸ *Transform* ▸ *Move, Rotate, Scale*
Shortcut::	`G`, `R`, `S`

Like other elements in Blender, points and strokes can be moved `G`, rotated `R` or scaled `S` as described in the Basic Transformations section. When in *Edit Mode*, Proportional Editing is also available for the transformation actions.

Transform Snapping

Basic move, rotate and scale transformations for selected points/strokes. See Move, Rotate, Scale Basics for more information.

Tools

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil* ▸ *Transform*
Tool::	*Toolbar* ▸ *Bend/Shear*

The *Bend*, *Shear*, *To Sphere*, *Extrude* and *Shrink Fatten* transform tools are described in the Editing tools section.

Mirror

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Mirror*
Shortcut:: `Ctrl-M`

The *Mirror* tool is also available, behaving exactly the same as with mesh vertices.

Snap

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Snap*
Shortcut:: `Shift-S`

Mesh snapping also works with Grease Pencil components.

Active Layer

Reference

Mode:: Edit Mode, Draw Mode
Menu:: *Grease Pencil ▸ Active Layer*
Shortcut:: `Y`

Select the active layer.

Animation

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Animation*
Shortcut:: `I`

The stroke animation operations are described in the Animation section.

Interpolate Sequence

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Interpolate Sequence*

See Interpolate Sequence.

Duplicate

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Duplicate*
Shortcut:: `Shift-D`

Duplicates the selected elements, without creating any connections with the rest of the strokes (unlike *Extrude*, for example), and places the duplicate at the location of the original elements.

Copy

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Copy*
Shortcut:: `Ctrl-C`

Copy the selected points/strokes to the clipboard.

Paste

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Paste*
Shortcut:: `Ctrl-V`

Paste Grease Pencil points or strokes from the internal clipboard to the active layer.

Paste on Back `Shift-Ctrl-V`
: Add pasted strokes behind all strokes.

Keep World Transform

Keep the world transform of strokes from the clipboard unchanged.

Weights

Reference

Mode::	Edit Mode, Draw Mode
Menu::	*Grease Pencil* ‣ *Weights*

The stroke weight operations are described in the Weights Menu section.

Show/Hide

Contains operators to adjust the visibility of points and strokes in the viewport.

Show All Layers

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil* ‣ *Show/Hide* ‣ *Show All Layers*
Shortcut::	`Alt-H`

Shows all Grease Pencil layers.

Hide Active Layer

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil* ‣ *Show/Hide* ‣ *Hide Active Layer*
Shortcut::	`H`

Hides the active Grease Pencil layers.

Hide Inactive Layers

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil* ‣ *Show/Hide* ‣ *Hide Active Layer*

Shortcut:: `Shift-H`

Hides the all Grease Pencil layers except the active layer.

Separate

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ‣ Separate*
Shortcut:: `P`

Separate the selected geometry into a new Grease Pencil object.

Selected Points
: Separate the selected geometry into a new object.

Material
: Separates the geometry by each material.

Active Layer
: Separate all the strokes in the active layer into a new object. See 2D Layers for more information.

Clean Up

These tools help to cleanup degenerate geometry on the strokes.

Clean Loose Points

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ‣ Clean Up ‣ Delete Loose Points*

Removes strokes with only a few points.

Limit
: The number of points to consider a stroke as loose.

Delete Duplicate Frames

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Clean Up ▸ Delete Duplicate Frames*

Removes any duplicate keyframes.

Merge by Distance

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Clean Up ▸ Merge by Distance*

Simplifies a stroke by merging the selected points that are closer than a specified distance to each other. Note, unless using *Unselected*, selected points must be contiguous, else they will not be merged.

Merge Distance
: Sets the distance threshold for merging points.

Unselected
: Allows points in selection to be merged with unselected points. When disabled, selected points will only be merged with other selected ones.

Reproject

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Clean Up ▸ Reproject*

Sometimes you may have drawn strokes unintentionally in different locations in the 3D space but they look right from a certain plane or from the camera view. You can use Reproject to flatten all the selected strokes from a certain viewpoint.

Reprojected Type

Front:: Reproject selected strokes onto the front plane (XZ).

Side:: Reproject selected strokes onto the side plane (YZ).

Top:: Reproject selected strokes onto the top plane (XY).

View:: Reproject selected strokes onto the current view.

Surface:: Reproject selected strokes onto the mesh surfaces.
Surface Offset
When Surface Mode is activated controls the stroke offset from the object.

Cursor:: Reproject selected strokes onto 3D cursor rotation.

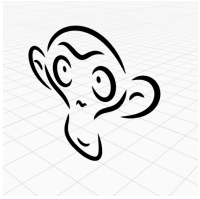
Keep Original
Maintains the original strokes after applying the tool.



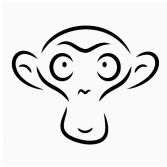
Original drawing from the front view.



Original drawing in the 3D Viewport.



Strokes reprojected onto the front plane to fix strokes misalignment.



Drawing after reprojection operation from the front view.

Delete

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Delete*
Shortcut:: `X`, `Delete`

Opens a pop-up menu with operators to remove geometry from the grease pencil object.

Frames
Deletes all the strokes at the current frame and in the current layer/channel.

Delete

Reference

Mode:: Edit Mode
Menu:: *Grease Pencil ▸ Delete ▸ Delete*

Deletes the selected points. When only one point remains, there is no more visible stroke, and when all points are deleted, the stroke itself is deleted.

Dissolve

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil ▸ Delete ▸ Dissolve*
Shortcut::	`Ctrl-X`

Dissolving removes points between other points and connect the remaining points.

`Ctrl-X` Opens a pop-up to choose the dissolve type.

Dissolve
: Deletes the selected points without splitting the stroke. The remaining points in the strokes stay connected.

Dissolve Between
: Deletes all the points between the selected points without splitting the stroke. The remaining points in the strokes stay connected.

Dissolve Unselect
: Deletes all the points that are not selected in the stroke without splitting the stroke. The remaining points in the strokes stay connected.

Delete Active Keyframe (Active Layer)

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil ▸ Delete ▸ Delete Active Keyframe (Active Layer)*

Deletes all the strokes at the current frame in the active layer.

Delete Active Keyframes (All Layers)

Reference

Mode::	Edit Mode
Menu::	*Grease Pencil ▸ Delete ▸ Delete Active Keyframes (All Layers)*
Shortcut::	`Shift-Delete`

Deletes all the strokes at the current frame in all layer.

Stroke Menu

This page covers many of the tools in the *Strokes* menu. These are tools that work primarily on strokes, however, some also work with point selections.

Subdivide

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Subdivide*

Subdivides the strokes by inserting points between the selected points.

Number of Cuts
: The number of subdivisions to perform.

Selected Points
: When enabled, limits the effect to only the selected points within the stroke.

Subdivide and Smooth

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Subdivide and Smooth*

Subdivides and smooths the strokes by inserting points between the selected points.

Number of Cuts
: The number of subdivisions to perform.

Selected Points
: When enabled, limits the effect to only the selected points within the stroke.

Iterations
: Number of times to repeat the procedure.

Factor
: The amount of the smoothness on subdivided points.

Smooth Endpoints
: Smooths the stroke's endpoints.

Keep Shape
: Preserves the strokes shape.

Position
: When enabled, the operator affect the points location.

Radius
: When enabled, the operator affect the points thickness.

Opacity
: When enabled, the operator affect the points strength (alpha).

Simplify

Reference

Mode::	Edit Mode
Menu::	*Stroke ‣ Simplify*

Uses the RDP algorithm (Ramer-Douglas-Peucker algorithm) for points deletion. The algorithm tries to obtain a similar line shape with fewer points.

Factor

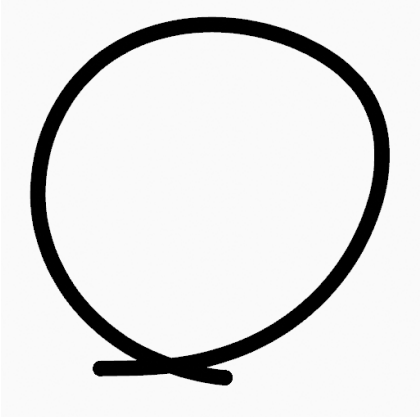
Controls the amount of recursively simplifications applied by the algorithm.

Trim

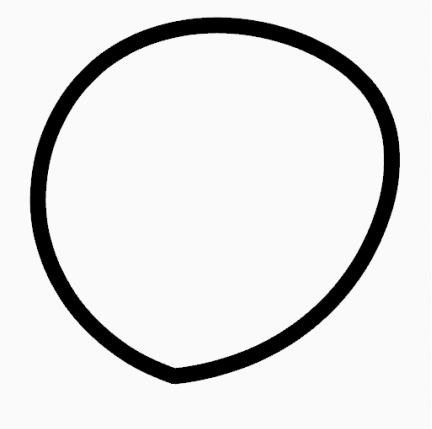
Reference

Mode::	Edit Mode
Menu::	*Stroke ▸ Trim*

Trims selected stroke to first loop or intersection.



Original stroke.



Result of trim operation.

Join

Join

Reference

Mode::	Edit Mode
Menu::	*Stroke ▸ Join ▸ Join*,

Join two or more strokes into a single one.

Type
: **Join::** `Ctrl-J` Join selected strokes by connecting points.
: **Join and Copy::** Join selected strokes by connecting points in a new stroke.

Leave Gaps
: When enabled, do not use geometry to connect the strokes.

Join and Copy

Reference

Mode::	Edit Mode
Menu::	*Stroke* ▸ *Join* ▸ *Join and Copy*
Shortcut::	`Shift-Ctrl-J`

Same as Join but *Type* defaults to *Join and Copy*.

Move to Layer

Reference

Mode::	Edit Mode
Menu::	*Stroke* ▸ *Move to Layer*
Shortcut::	`M`

A pop-up menu to move the stroke to a different layer. You can choose the layer to move the selected strokes to from a list of layers of the current Grease Pencil object. You can also add a new layer to move the selected stroke to. When creating a new layer, there is another pop-up to type in the name of the new layer.

Assign Material

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Assign Material*

Changes the material linked to the selected stroke. You can choose the name of the material to be used by the selected stroke from a list of materials of the current Grease Pencil object.

Set as Active Material

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Set as Active Material*

Sets the active object material based on the selected stroke material.

Arrange

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Arrange*

Change the drawing order of the strokes in the 2D layer.

Bring to Front
: Moves to the top the selected points/strokes.

Bring Forward
: Moves the selected points/strokes upper the next one in the drawing order.

Send Backward

Moves the selected points/strokes below the previous one in the drawing order.

Send to Back
: Moves to the bottom the selected points/strokes.

Close

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Close*
Shortcut:: F

Close or open strokes by connecting the last and first point.

Type
: **Close All::** Close all open selected strokes.
: **Open All::** Open all closed selected strokes.
: **Toggle::** Close or Open selected strokes as required.

Toggle Cyclic

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Toggle Cyclic*

Toggles between an open stroke and closed stroke (cyclic).

Type
: **Close All::** Close all open selected strokes.
: **Open All::** Open all closed selected strokes.
: **Toggle::** Close or Open selected strokes as required.

Set Caps

Reference

Mode::	Edit Mode
Menu::	*Stroke ▸ Set Caps*

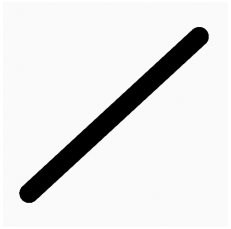
Toggle ending cap styles of the stroke.

Rounded
: Sets stroke start and end points to rounded (default).

Flat
: Toggle stroke start and end points caps to flat or rounded.

Toggle Start
: Toggle stroke start point cap to flat or rounded.

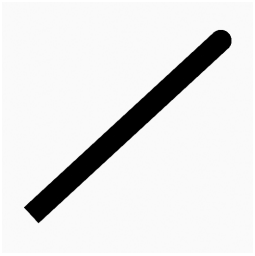
Toggle End
: Toggle stroke end point cap to flat or rounded.



Stroke ending with rounded caps.



Stroke ending with flat caps.



Stroke ending with combined caps.

Switch Direction

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Switch Direction*

Reverse the direction of the points in the selected strokes (i.e. the start point will become the end one, and vice versa).

Set Uniform Thickness

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Set Uniform Thickness*

Makes the thickness equal for the entire stroke.

Thickness
: Thickness value to use on all points of the stroke.

Set Uniform Opacity

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Set Uniform Opacity*

Makes the opacity equal for the entire stroke.

Opacity
: Opacity value to use on all points of the stroke.

Scale Thickness

Reference

Mode:: Edit Mode

Menu:: *Stroke ▸ Scale Thickness*

When enabled, scales the stroke thickness during scale transformations.

Set Curve Type

Reference

Mode:: Edit Mode
Menu:: *Stroke ▸ Set Curve Type*

Sets the spline type for the splines in the stroke component that are in the selection.

Type
: The type to convert the splines in the selection to. Read the Spline Types page for more details on the different spline types.

Bézier:: Convert to a Bézier spline. A spline converted from a poly spline gets vector handles, while one converted from NURBS or Catmull Rom spline gets auto handles.

> **Note**
> When converting from a NURBS spline to a Bézier spline, at least six points are needed. When the number of points is not a multiple of three a full conversion is not possible and the spline has to be truncated.

NURBS:: Convert to a NURBS spline.

Poly:: Convert to a poly spline.

Catmull Rom:: Convert to a Catmull Rom spline.

Handles
: Take handle information into account in the conversion

Set Curve Resolution

Reference

Mode::	Edit Mode
Menu::	*Stroke ‣ Set Curve Resolution*

Sets the number of points generated along each curve segment (between two handles).

Reset UVs

Reference

Mode::	Edit Mode
Menu::	*Stroke ‣ Set Curve Resolution*

Reset UV transformation to default values.

Point Menu

Extrude

Reference

Mode::	Edit Mode
Menu::	*Point ▸ Extrude*
Tool::	*Toolbar ▸ Extrude*
Shortcut::	E

Extrudes points by duplicating the selected points, which then can be moved. The new points stay connected with the original points of the edit line.

Note

Since Grease Pencil strokes can only have one start an end point, a new stroke will be created when extrude intermediate points in the strokes.

Smooth

Reference

Mode::	Edit Mode
Menu::	*Point ▸ Smooth*

Softens strokes by reducing the differences in the locations of the points along the line, while trying to maintain similar values that make the line fluid and smoother.

Iterations

The number of times to repeat the procedure.

Factor
: The amount of the smoothness to apply.

Smooth Endpoints
: Smooths the stroke's endpoints.

Keep Shape
: Preserves the strokes shape.

Position
: When enabled, the operator affect the points location.

Radius
: When enabled, the operator affect the points thickness.

Opacity
: When enabled, the operator affect the points strength (alpha).

Vertex Groups

Operators for working with vertex groups.

Set Handle Type

Reference

Mode::	Edit Mode
Menu::	*Point ▸ Set Handle Type*
Shortcut::	V

Sets the handle type for the points on the Bézier curve that are in the selection.

Type
: The handle type to switch to.

Free:: The handles are independent of each other.

Auto:: This handle has a completely automatic length and direction which is set by Blender to ensure the smoothest result. These handles convert to *Aligned* handles when moved.

Vector:: Both parts of a handle always point to the previous handle or the next handle which allows you to create curves or sections thereof made of straight lines or with sharp corners. Vector handles convert to *Free* handles when moved.

Aligned:: These handles always lie in a straight line, and give a continuous curve without sharp angles.

Vertex Paint Mode

- Introduction
  - Vertex Paint Mode
  - Vertex Paint Options
- Vertex Paint Brushes
- Vertex Paint Tools
- Brush Settings
  - Color Picker
  - Color Palette
  - Falloff
- Editing
  - Set Color Attribute
  - Reset Vertex Color
  - Invert
  - Levels
  - Hue/Saturation/Value
  - Brightness/Contrast

Introduction

Vertex Painting is a simple way of painting color onto a Grease Pencil object, by directly manipulating the color of points/vertices, rather than use only the materials base color.



Stroke with original base material color (left) and with vertex painting (right).

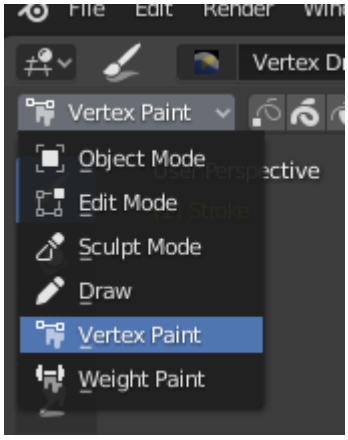
When a point is painted, the color of the points is mixing with the base material color according to the settings of the brush.

Note

A vertex in Grease Pencil is called point. Point and vertex names are equivalent.

Vertex Paint Mode

Vertex Paint Mode is selected from the *Mode* menu in the 3D Viewport header. Once Vertex Paint Mode is activated, the Toolbar of the 3D Viewport will change to Vertex Paint Mode specific panels.



3D Viewport Mode selector set to Vertex Paint Mode.

Vertex Paint Options



General Vertex Paint options.

Selection Mask

Vertex Paint Mode in Grease Pencil allows you to select points or strokes to restrict the effect of the painting tools to only a certain areas of your drawing.

You can use the selection tools in the Toolbar for a quick selections.

You can restrict painting only on the selected points or strokes with the Selection mode toggle. The three modes can be toggled with `1`, `2`, or `3` respectively.

Multiframe

Sometimes you may need to modify several frames at the same time with the painting tools.

You can activate multiframe edition with the Multiframe button next to the modes selector (faded lines icon). See Multiframe for more information.

Vertex Paint Brushes

Brushes for Grease Pencil *Vertex Paint* mode bundled in the *Essentials* library.

Paint Point Color
: Paints a specified color over the object.

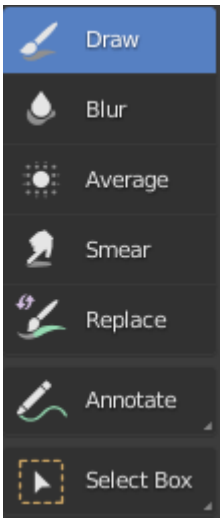
Blur Point Color
: Smooths out the colors of adjacent vertices. In this mode the Color Value is ignored. The strength defines how much the colors are blurred.

Average Point Color
: Smooths color by painting the average resulting color from all colors under the brush.

Smear Point Color
: Smudges colors by grabbing the colors under the brush and “dragging” them. This can be imagined as a finger painting tool.

Replace Point Color
: Change the color only to the stroke points that already have a color applied.

Vertex Paint Tools



Brush
: Tool to use for any of the vertex paint brushes.

Annotate
: Draw free-hand annotation.

Annotate Line
: Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Brush Settings

Painting needs paint brushes and Blender provides a Brush panel within the Toolbar when in *Vertex Paint Mode*.

Brush
: In the Data-Block menu you find predefined Brush presets. And you can create your own custom presets as needed.

Radius
: This option controls the radius of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse and then `LMB` (the texture of the brush should be visible inside the circle). Typing a number then enter while using `F` allows you to enter the size numerically.

  (Size Pressure)
  : Adjusts the brush radius based on the stylus pressure when using a Graphics Tablet.

Strength
: How powerful the brush is when applied.

  (Strength Pressure)
  : Adjusts the brush strength based on the stylus pressure when using a Graphics Tablet.

Mode

Stroke::	Only paint over strokes.
Fill::	Only paint over fill areas.
Stroke & Fill::	Paint over strokes and fill areas.

Cursor
: See the global brush settings for Cursor settings.

Color Picker

The color of the brush. See Color Picker.

Note

Note that Vertex Paint works in sRGB space and the RGB representation of the same colors will be different between the paint tools and the materials that are in linear space.

Color Palette

The active Color Palette. See Color Palette.

Falloff

See the global brush settings for Falloff settings.

Editing

Set Color Attribute

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ‣ Set Color Attribute*

Sets the active color to all selected vertices.

Reset Vertex Color

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ‣ Reset Vertex Color*

Removes the Color Attribute information of the active strokes, if no strokes are selected, all strokes are reset.

Invert

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ‣ Invert*

Invert RGB values.

Levels

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Levels*

Adjust the levels of Color Attributes.

Hue/Saturation/Value

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Hue/Saturation/Value*

Adjust the color's HSV values.

Brightness/Contrast

Reference

Mode::	Vertex Paint Mode
Menu::	*Paint ▸ Brightness/Contrast*

Adjust the color's brightness/contrast.

Weight Paint Mode

- Introduction
  - Weight Paint
  - Weight Options
- Weight Paint Brushes
- Weight Paint Tools
- Tool Settings
  - Brush Settings
  - Options
- Weights Menu
  - Normalize All
  - Normalize
  - Invert
  - Smooth
  - Sample Weight

Introduction

Assigning weight to the points is primarily used for rigging strokes in cut-out animation, where the vertex groups are used to define the relative bone influences on the strokes. See Using Vertex Group for more information.

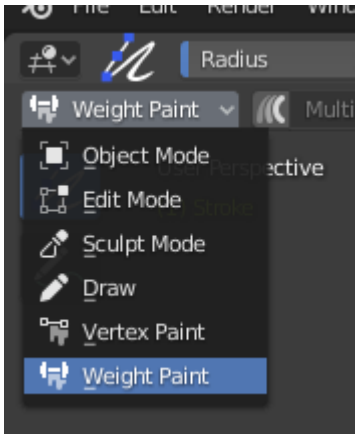
Note

A vertex in Grease Pencil is called point. Point and vertex names are equivalent.

Weight Painting is a method to maintain large amounts of weight information in an intuitive way. The selected Grease Pencil object is displayed slightly shaded with a rainbow color spectrum. The color visualizes the weights associated to each point in the active vertex group. By default blue means unweighted and red means fully weighted.

You assign weights to the points of the object by painting on it with weight brushes. Starting to paint on a strokes automatically adds weights to the active vertex group (a new vertex group is created if needed).

Weight Paint



3D Viewport Mode selector: Weight Paint Mode.

Weight Paint Mode is selected from the *Mode* menu in the 3D Viewport header. Once Weight Paint Mode is activated, the Toolbar of the 3D Viewport will change to Weight Paint Mode specific panels. A red circle will appear and follow the location of the cursor in the 3D Viewport.

Weight Options

Multiframe

Sometimes you may need to assign weight to several frames at the same time with the Weight Paint tools.

You can activate multiframe edition with the Multiframe button next to the modes selector (faded lines icon). See Multiframe for more information.

Weight Paint Brushes

Brushes for Grease Pencil *Weight Paint* mode bundled in the *Essentials* library.

Paint Point Weight
: Paints a specified weight over the strokes.

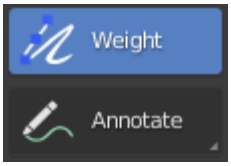
Blur Point Weight
: Smooths out the weighting of adjacent points. In this mode the Weight Value is ignored. The strength defines how much the smoothing is applied.

Average Point Weight
: Smooths weights by painting the average resulting weight from all weights under the brush.

Smear Point Weight
: Smudges weights by grabbing the weights under the brush and “dragging” them. This can be imagined as a finger painting tool.

Weight Paint Tools

For Grease Pencil Weight Paint modes each brush type is exposed as a tool, the brush can be changed in the Tool Settings. See Brush for more information.



Brush
: Tool to use for any of the weight paint brushes.

Annotate
: Draw free-hand annotation.

  Annotate Line
  : Draw straight line annotation.

  Annotate Polygon
  : Draw a polygon annotation.

  Annotate Eraser
  : Erase previous drawn annotations.

Tool Settings

- Brush Settings
- Options

Brush Settings

Painting needs paint brushes and Blender provides a Brush panel within the Toolbar when in *Weight Paint Mode*.

Brush
: In the Data-Block menu you find predefined Brush presets. And you can create your own custom presets as needed.

Radius `F`
: The radius defines the area of influence of the brush.

  (Size Pressure)
  : Adjusts the radius based on the stylus pressure when using a Graphics Tablet.

Strength
: This is the amount of paint to be applied per brush stroke.

  (Strength Pressure)
  : Adjusts the strength based on the stylus pressure when using a Graphics Tablet.

Use Falloff
: When enabled, use Strength falloff for the brush. Brush Strength decays with the distance from the center of the brush.

Weight `Ctrl-F`
: The weight (visualized as a color) to be used by the brush.

  Using `Ctrl-RMB` you can set the weight to the value thats under the cursor.

Direction `D`

Brush direction toggle, *Add* adds weight value while *Subtract* removes weight value. This setting can be toggled with `D`.

Options

Auto Normalize

Ensures that all deforming vertex groups add up to one while painting. When this option is turned off, then all weights of a point can have any value between 0 and 1. However, when vertex groups are used as deform groups for character animation then Blender always interprets the weight values relative to each other. That is, Blender always does a normalization over all deform bones. Hence in practice it is not necessary to maintain a strict normalization and further normalizing weights should not affect animation at all.

This option works most intuitively when used to maintain normalization while painting on top of weights that are already normalized with another tool.

Weights Menu

Reference

Mode:: Edit Mode
Menu:: *Weights*

This page covers many of the tools in the *Weights* menu.

Normalize All

Reference

Mode:: Edit Mode
Menu:: *Weights ▸ Normalize All*

For each point, this tool makes sure that the sum of the weights across all vertex groups is equal to 1. It normalizes all of the vertex groups, except for locked groups, which keep their weight values untouched.

Lock Active
: Keep the values of the active group while normalizing all the others.

Normalize

Reference

Mode:: Edit Mode
Menu:: *Weights ▸ Normalize*

This tool only works on the active vertex group. All points keep their relative weights, but the entire set of weights is scaled up such that the highest weight value is 1.0.

Invert

Reference

Mode::	Edit Mode
Menu::	*Weights ▸ Invert*

Replaces each weight of the selected vertex group by × -1.0 weight.

Examples:

- Original 1.0 converts to 0.0
- Original 0.5 remains 0.5
- Original 0.0 converts to 1.0

Smooth

Reference

Mode::	Edit Mode
Menu::	*Weights ▸ Smooth*

Smooths the weights of the active vertex group.

Sample Weight

Reference

Mode::	Edit Mode
Menu::	*Weights ▸ Sample Weight*
Shortcut::	`Shift-X`

Adjust the Weight of the Draw tool to the weight of the vertex under the mouse cursor.

Object Mode

- Trace Image to Grease Pencil
  - Usage
  - Options

Trace Image to Grease Pencil

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Convert* ▸ *Trace Image to Grease Pencil*

The *Trace Image to Grease Pencil* tool traces a black and white image and generates Grease Pencil strokes. If the image is not black and white, it will be internally converted. For better results, convert the images manually to black and white. Also try to keep the resolution of the image small; high resolutions can produce very dense strokes.

Usage

1. Add an Image Empty to the scene.
2. Run *Trace Image to Grease Pencil*.

Options

Target Object
: Determines if the image empty is kept or replaced.

  New Object: Creates a new grease pencil object and keeps the image empty Selected Object: Replaces the image empty with the grease pencil object.

Radius
: The thickness of the generated Grease Pencil strokes.

Color Threshold
: Determine the Luminance threshold above which strokes are generated.

Turn Policy

Determines how to resolve ambiguities during decomposition of an image into paths.

Black::	Prioritizes to connect black (foreground) components.
White::	Prioritizes to connect white (background) components.
Left::	Always take a left turn.
Right::	Always take a right turn.
Minority::	Prioritizes to connect the color (black or white) that occurs least frequently in the local neighborhood of the current position.
Majority::	Prioritizes to connect the color (black or white) that occurs most frequently in the local neighborhood of the current position.
Random::	Choose pseudo-randomly.

Mode

Determines if the image being traced is a single image or image sequence.

Single::	The image empty is a single image or the current frame of an image sequence.
Sequence::	The image empty is an Image Sequence.

Start at Current Frame

When enabled, start the tracing process at the current image frame.

Trace Frame

Used to trace only one frame of the image sequence, set to zero to trace all.

Animation & Rigging

- Introduction
  - Animation
  - Rigging
- Keyframes
  - Introduction
  - Editing
  - Keying Sets
- Armatures
  - Introduction
  - Bones
  - Properties
  - Structure
  - Skinning
  - Posing
- Lattice
  - Editing
  - Properties
  - Usage
- Constraints
  - Introduction
  - Interface
  - Constraint Types
- Actions
  - Working with Actions
  - Properties
- Drivers
  - Introduction
  - Usage
  - Drivers Panel
  - Workflow & Examples
  - Troubleshooting
- Markers
  - Types

    - Visualization
    - Add Marker
    - Selecting
    - Editing
    - Bind Camera to Markers
- Shape Keys
    - Introduction
    - Shape Keys Panel
    - Workflow
- Motion Paths
    - Options
    - Example

Introduction

Animation

Animation is making an object move or change shape over time. Objects can be animated in many ways:

Moving as a whole object
: Changing their position, orientation or size in time;

Deforming them
: Animating their vertices or control points;

Inherited animation
: Causing the object to move based on the movement of another object (e.g. its parent, hook, armature, etc.).

In this chapter, we will cover the first two, but the basics given here are actually vital for understanding the following chapters as well.

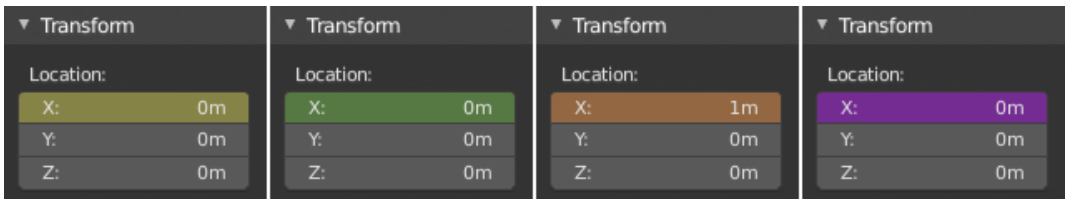
Animation is typically achieved with the use of keyframes.

See also

Related Sections

- Physical Simulation
- Motion Tracking

State Colors



State colors of properties.

Properties have different colors and menu items for different states.

Gray	Not animated
Yellow	Keyframed on the current frame
Green	Keyframed on a different frame
Orange	Changed from the keyframed value
Purple	Controlled by a driver

The changed value highlight currently doesn't work with NLA.

Rigging

Rigging is a general term used for adding controls to objects, typically for the purpose of animation.

Rigging often involves using one or more of the following features:

Armatures
: This allows mesh objects to have flexible joints and is often used for skeletal animation.

Constraints
: To control the kinds of motions that make sense and add functionality to the rig.

Object Modifiers
: Mesh deformation can be quite involved, there are multiple modifiers that help control this.

Shape Keys
: To support different target shapes *(such as facial expressions)* to be controlled.

Drivers
: So your rig can control many different values at once, as well as making some properties automatically update based on changes elsewhere.

Rigging can be as advanced as your project requires, rigs are effectively defining own user interface for the animator to use, without having to be concerned the underlying mechanisms.

Examples

- An armature is often used with a modifier to deform a mesh for character animation.
- A camera rig can be used instead of animating the camera object directly to simulate real-world camera rigs (with a boom arm, mounted on a rotating pedestal for example, effects such as camera jitter can be added too).

See also

The content of this chapter is simply a reference to how rigging is accomplished in Blender. It should be paired with additional resources such as Nathan Vegdahl's excellent introduction to the fundamental concepts of character rigging, Humane Rigging.

Keyframes

- Introduction
  - Visualization
  - Interpolation
  - Keyframe Types
  - Handles & Interpolation Mode Display
- Editing
  - Insert Keyframe
  - Insert Keyframe with Keying Set
  - Delete Keyframes
  - Clear Keyframes
  - Editing Keyframes
  - Examples
- Keying Sets
  - Keying Set Panel
  - Adding Properties to a Keying Set
  - Set Active Keying Set
  - Whole Character Keying Set

Introduction

A *Keyframe* is simply a marker of time which stores the value of a property.

For example, a Keyframe might define that the horizontal position of a cube is at 3 m on frame 1.

The purpose of a Keyframe is to allow for interpolated animation, meaning, for example, that the user could then add another key on frame 10, specifying the cube's horizontal position at 20 m, and Blender will automatically determine the correct position of the cube for all the frames between frame 1 and 10 depending on the chosen interpolation method (e.g. Linear, Bézier, Quadratic, etc.).

An overview of existing keyframes can be seen via the Dope Sheet editor.

Visualization

There are some important visualization features in the 3D Viewport that can help animation.

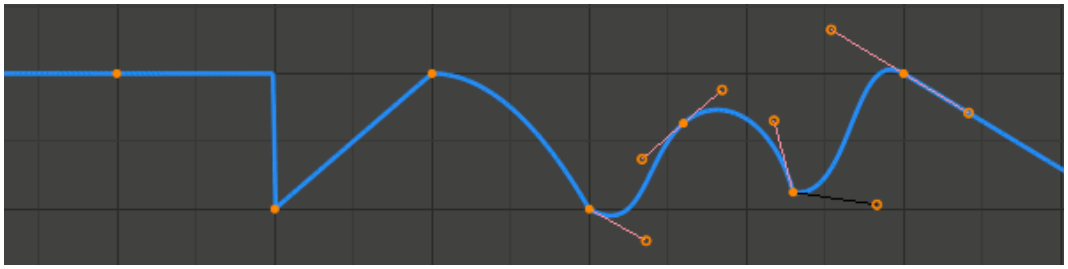
When the current frame is a keyframe for the current active object, the name of this object (shown in the upper left corner of the 3D Viewport) turns yellow.



Top: Current frame is a keyframe for Cube. Bottom: Current frame isn't a keyframe.

Interpolation

Keyframe interpolation is represented and controlled by animation curves, also known as F-Curves. These curves can be viewed and modified via the Graph Editor.



Constant, Linear, Quadratic and Bézier interpolation, with Linear extrapolation.

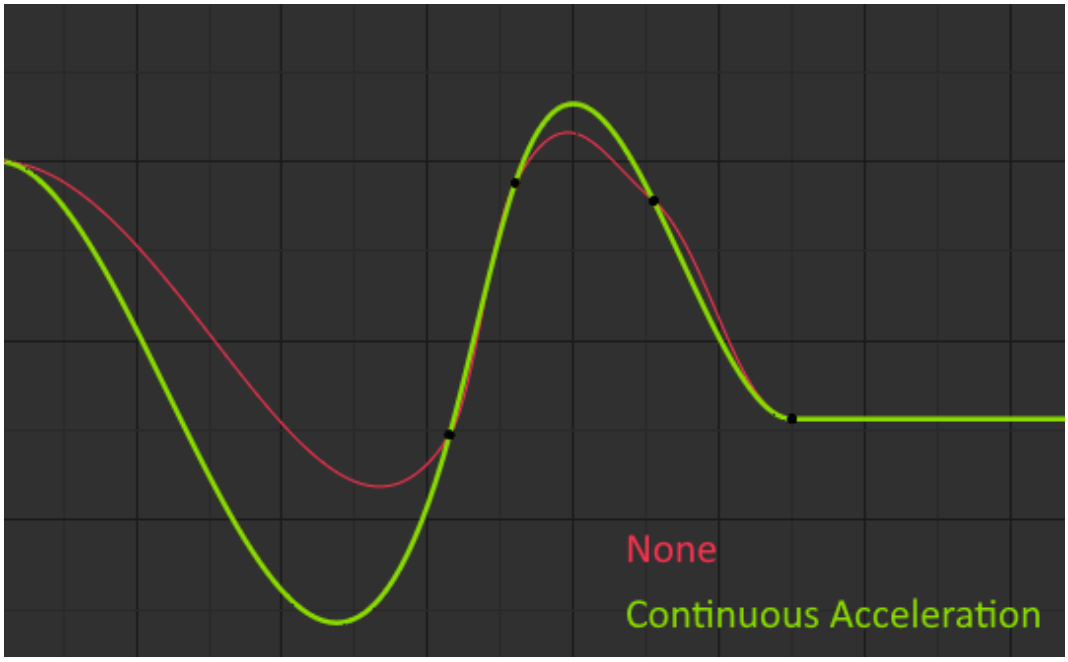
The X axis of the curve corresponds to time, while Y represents the value of the property. Keyframes themselves define points of the curve, while interpolation is controlled by additional parameters.

The Interpolation Mode is the main setting that specifies for each keyframe how the curve is interpolated from that key to the next one. There are a number of modes with fixed shapes, e.g. *Constant*, *Linear*, *Quadratic* etc, and a free form *Bézier* mode.

Extrapolation specifies how the curve extends before the first, and after the last keyframe. The main available choices are *Constant* and *Linear*; it is also possible to configure the curve to loop.

Bézier interpolation is controlled by handles, which have a handle type and position. The position of *Free* and *Aligned* handles must be set manually from the Graph editor, while *Vector*, *Automatic* and *Auto Clamped* handles are computed automatically from keyframe values.

Interpolation, Extrapolation and Handle Type can also be changed from the Dope Sheet editor.



Handle smoothing modes. Yellow: *None*, Cyan: *Continuous Acceleration*.

The method how the three automatic handle types are computed is controlled by the per-curve Auto Handle Smoothing setting. The *None* mode resembles how most other software works and only considers the values of the immediately adjacent keys. The *Continuous Acceleration* mode considers the shape of the whole curve, which produces smoother results out of the box, but means that changes in one key affect interpolation over a larger section of the curve; it also tends to overshoot more with *Automatic* handles.

Keyframe Types

For visually distinguishing regular keyframes from different animation events or states (extremes, breakdowns, or other in-betweens) there is the possibility of applying different colors on them for visualization.

Keyframe (white / yellow diamond)
: Normal keyframe.



Breakdown (small cyan diamond)

Left: not selected; Right: selected.

Breakdown state. e.g. for transitions between key poses.

Moving Hold (dark gray / orange diamond)

A keyframe that adds a small amount of motion around a holding pose. In the Dope Sheet it will also display a bar between them.

Extreme (big pink diamond)

An ‘extreme’ state, or some other purpose as needed.

Jitter (tiny green diamond)

A filler or baked keyframe for keying on ones, or some other purpose as needed.

Generated (dark diamond)

A key generated by some tool, for example Copy Global Transform: Fix to Camera. This keyframe type indicates to Blender and add-ons that it is safe to remove and re-generate them, so be careful when manually marking your hand-made animation with this type.

Handles & Interpolation Mode Display

Dope Sheet can display the Bézier handle type associated with the keyframe, and mark segments with non-Bézier interpolation. This facilitates basic editing of interpolation without the use of the Graph Editor.

The icon shape represents the type of the Bézier Handles belonging to the keyframe.

Circle	Auto Clamped (default)
Circle With Dot	Automatic



From top: summary, Bézier, linear.

Square	Vector
Clipped Diamond	Aligned
Diamond	Free

If the handles of a keyframe have different types, or in case of summary rows representing multiple curves, out of the available choices the icon that is furthest down the list is used. This means that if a grouped row uses a circle icon, it is guaranteed that none of the grouped channels have a non-auto key.

Horizontal green lines mark the use of non-Bézier Interpolation. The line is dimmed in summary rows if not all grouped channels have the same interpolation.

Display of this information can be disabled via the *Show Handles and Interpolation* option of the Dope Sheet's View Menu.

Editing

Insert Keyframe

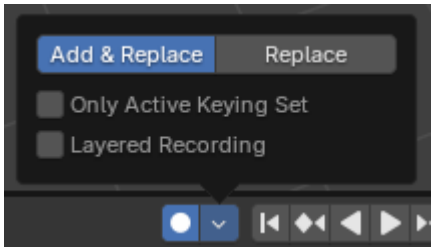
Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Animation* ▸ *Insert Keyframe*
Shortcut::	`I`

There are several methods of adding new keys. Namely:

- In the 3D Viewport, pressing `I` will key properties based on the Default Key Channels User Preferences.
- When a Keying Set is active, it is used instead of reading the User Preferences.
- Hovering over a property and pressing `I` or with the context menu by `RMB` a property and choose *Insert Keyframe* from the menu.
- With the User Preference "Pie Menu on Drag" enabled, holding down `I` and moving the cursor will bring up a pie menu to insert one of Location, Rotation, Scale, and Available.

Auto Keyframe



Timeline Auto Keyframe.

Auto Keyframe is the record button in the *Timeline* header. Auto Keyframe adds keyframes automatically to the set frame if the value for transform

type properties changes.

See Timeline Keyframe Control for more info.

Insert Keyframe with Keying Set

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Animation* ▸ *Insert Keyframe with Keying Set*
Shortcut::	`K`

Insert Keyframes for specified Keying Set, with menu of available Keying Sets.

Delete Keyframes

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Animation* ▸ *Delete Keyframes…*
Shortcut::	`Alt-I`

There are several methods of removing keyframes:

- In the 3D Viewport press `Alt-I` to remove keys from selected objects on the current frame.
- When the mouse is over a value, press `Alt-I`.
- `RMB` a value and choose *Delete Keyframe* from the menu.

Clear Keyframes

Reference

Mode::	Object Mode
Menu::	*Object* ▸ *Animation* ▸ *Clear Keyframes…*
Shortcut::	`Shift-Alt-I`

Removes all keyframes from the selected object.

Editing Keyframes

Keyframes can be edited in two editors. To do so go to either the Graph Editor or the Dope Sheet.

Examples

Keyframe Animation

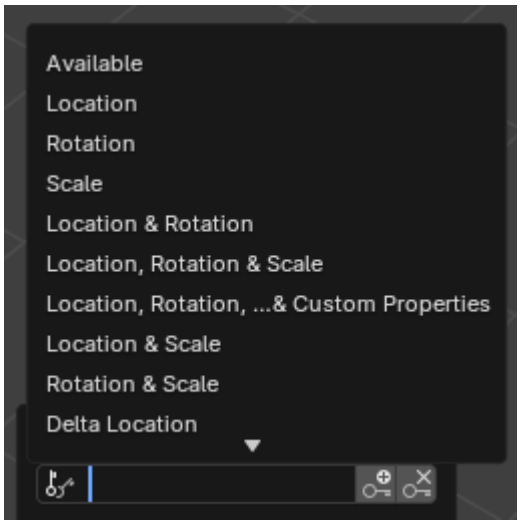
This example shows you how to animate a cube's location, rotation, and scale.

1. First, in the Timeline, or other animation editors, set the frame to 1.
2. With the cube selected in Object Mode, press `I` in the 3D Viewport. This will record the location, rotation, and scale, for the cube on frame 1.
3. Set the frame to 100.
4. Use Move `G`, Rotate `R`, Scale `S`, to transform the cube.
5. Press `I` in the 3D Viewport.

To test the animation, press `Spacebar` to play.

Keying Sets

Keying Sets are a collection of animated properties that are used to animate and keyframe multiple properties at the same time. For example, pressing K in the 3D Viewport will bring up the available Keying Sets. Blender will then add keyframes for whichever Keying Set is chosen. There are some built-in Keying Sets and also custom Keying Sets called “Absolute Keying Sets”.



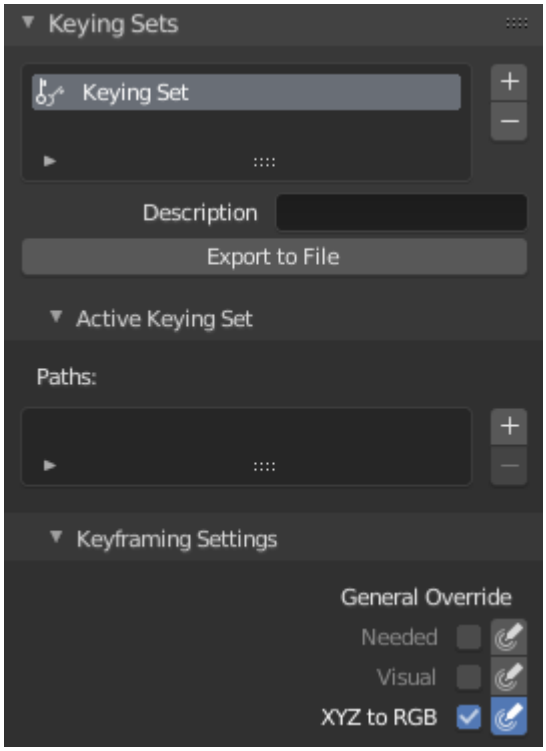
The Active Keying Sets data ID in the Timeline.

Keying Set Panel

Reference

Editor:: Properties
Panel:: *Scene ▸ Keying Set*

This panel is used to add, select, manage “Absolute Keying Sets”.



The Keying Set panel.

Active Keying Set
: A List View of Keying Sets in the active scene. Selecting a keying set makes it active

  Add +
  : Adds an empty Keying Set.

  Remove –
  : Removes the active keying set.

Description
: A short description of the Keying Set.

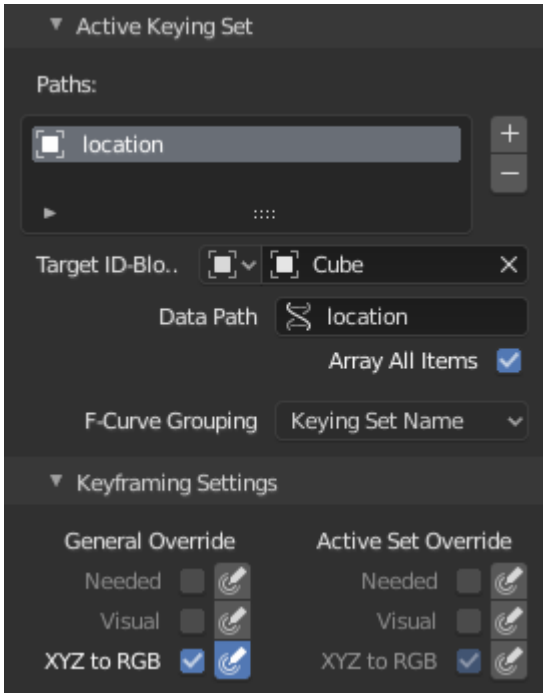
Export to File
: Export Keying Set to a Python script `File.py`. To re-add the Keying Set from the `File.py`, open then run the `File.py` from the Text Editor.

Active Keying Set Panel

Reference

Editor:: Properties
Panel:: *Scene ▸ Active Keying Set*

This panel is used to add properties to the active Keying Set.



The Active Keying Set panel.

Paths
A collection of paths in a List View each with a *Data Path* to a property to add to the active Keying Set.

Add +
Adds an empty path.

Remove –
Removes the selected path.

Target ID-Block
Set the ID Type and the *Object IDs* data path for the property.

Data Path
: Set the rest of the Data Path for the property.

Array All Items
: Use *All Items* from the Data Path or select the array index for a specific property.

F-Curve Grouping
: This controls what group to add the channels to.

  Keying Set Name, None, Named Group

Keyframing Settings

General Override
: These options control all properties in the Keying Set. Note that the same settings in *Preferences* override these settings if enabled.

Active Set Override
: These options control individual properties in the Keying Set.

Common Settings
: Needed
  : Only insert keyframes where they are needed in the relevant F-Curves.

  Visual
  : Insert keyframes based on the visual transformation.

Adding Properties to a Keying Set

Reference

Menu::	*Context menu* ‣ *Add All/Single to Keying Set*
Shortcut::	K

Some ways to add properties to Keying Sets.

RMB the property in the *User Interface*, then select *Add Single to Keying Set* or *Add All to Keying Set*. This will add the properties to the active Keying Set, or to a new Keying Set if none exist.

Hover the mouse over the properties, then press K, to add *Add All to Keying Set*.

Set Active Keying Set

Reference

Shortcut:: Shift-K

There are several ways to designate the active keying set:

- Press Shift-K in the 3D Viewport.
- Select a keying set in the Keying Set panel.
- Select a keying set in the Keying popover in the Timeline header,

Whole Character Keying Set

The built-in *Whole Character* Keying Set is made to keyframe all properties that are likely to get animated in a character rig. It was also implicitly used by the Old Pose Library system.

This keying set ignores bones whose name starts with one of the following prefixes, as it assumes these are technical bones that are not meant to be animated directly. The built-in Rigify addon generates such bones, for example.

- COR (Corrective)
- DEF (Deformation)
- GEO (Geometry)
- MCH (Mechanism)
- ORG (Original from meta rig)
- VIS (Visualization)

Armatures

- Introduction
  - Your First Armature
  - The Armature Object
- Bones
  - Introduction
  - Bone Collections
  - Structure
  - Tools
  - Selecting
  - Editing
  - Properties
- Properties
  - Introduction
  - Bone Collections
  - Selection Sets
  - Viewport Display
- Structure
  - Chains of Bones
- Skinning
  - Introduction
  - Armature Deform Parent
- Posing
  - Introduction
  - Selecting
  - Editing
  - Tool Settings
  - Bone Constraints

Introduction

An armature in Blender can be thought of as similar to the armature of a real skeleton, and just like a real skeleton an armature can consist of many bones. These bones can be moved around and anything that they are attached to or associated with will move and deform in a similar way.

An “armature” is a type of object used for rigging. A rig is the controls and strings that move a marionette (puppet). Armature object borrows many ideas from real-world skeletons.

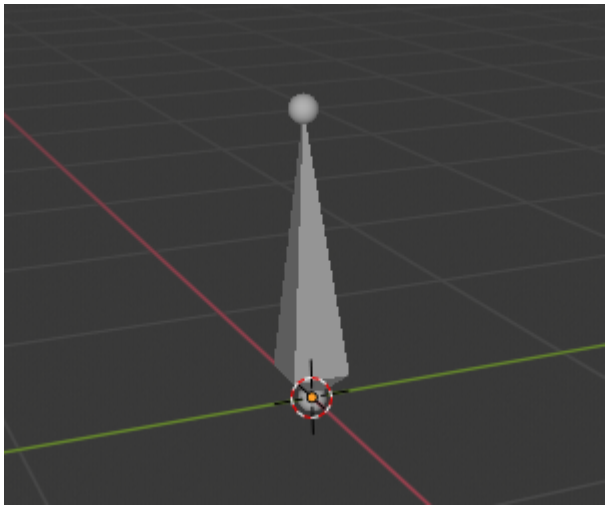
Your First Armature

In order to see what we are talking about, let us try to add the default armature in Blender.

(Note that armature editing details are explained in the armatures editing section.)

Open a default scene, then:

1. Delete all objects in the scene.
2. Make sure the cursor is in the world origin with `Shift-C`.
3. Press `Numpad1` to see the world in Front view.
4. Add a *Single Bone* (*Add ▸ Armature*).
5. Press `NumpadPeriod` to see the armature at maximum zoom.



The default armature.

The Armature Object

As you can see, an armature is like any other object type in Blender:

- It has an origin, a position, a rotation and a scale factor.
- It has an Object Data data-block, that can be edited in *Edit Mode*.
- It can be linked to other scenes, and the same armature data can be reused on multiple objects.
- All animation you do in *Object Mode* is only working on the whole object, not the armature's bones (use the *Pose Mode* to do this).

As armatures are designed to be posed, either for a static or animated scene, they have a specific state, called "rest position". This is the armature's default "shape", the default position/rotation/scale of its bones, as set in *Edit Mode*.

In *Edit Mode*, you will always see your armature in rest position, whereas in *Object Mode* and *Pose Mode*, you usually get the current "pose" of the armature (unless you enable the *Rest Position* button of the *Armature* panel).

Bones

- Introduction
  - Classification
- Bone Collections
  - Visibility
  - Library Overrides
  - Some history
- Structure
  - Roll
  - Bones Influence
- Tools
  - Toolbar
  - Tool Settings
- Selecting
  - Selecting Bone Joints
  - Selecting Bones
  - Select Mirror
  - More/Less
  - Select Linked
  - Select Similar
  - Select Pattern
  - Parent/Child
  - Extend Parent/Child
- Editing
  - Introduction
  - Transform
  - Bone Roll
  - Extrude
  - Duplicate
  - Fill Between Joints
  - Split
  - Separate Bones
  - Subdivide
  - Switch Direction

    - Symmetrize
    - Naming
    - Parenting
    - Properties
    - Delete
- Properties
    - Transform
    - Bendy Bones
    - Relations
    - Inverse Kinematics
    - Deform
    - Viewport Display
    - Custom Properties

Introduction

Bones are the base elements of armatures. The visualization of bones can be set in the Armatures Viewport Display.

Classification

Bones in an Armature can be generally classified into two different types:

1. Deforming Bones
2. Control Bones

Deforming Bones

Are bones which when transformed will result in vertices associated with them also transforming in a similar way. Deforming Bones are directly involved in altering the positions of vertices associated with their bones.

Control Bones

Are Bones which act in a similar way to switches, in that, they control how other bones or objects react when they are transformed. A Control Bone could for example act as a sliding switch control when the bone is in one position to the left, it could indicate to other bones that they react in a particular way when transformed, and when the Control Bone is positioned to the right, transforming other bones or objects could do something completely different. Control Bones are not directly used to alter the positions of vertices; in fact, Control Bones often have no vertices directly associated with themselves.

Bone Collections

Bone Collections group the bones of an Armature into named collections. The armature is the owner of these collections, so they are available in all modes. Bone Collections are identified by their name, which are unique within the Armature. Bone Collections can be nested inside other Bone Collections to create an organized hierarchy for complex rigs.

In the text below, "collection" is understood to refer to "bone collection"; Scene Collections are not described here.

Bone Collections can be managed via the Armature and Bone property panels.

Visibility

Bone Collections can be shown & hidden via the list in the Armature properties, as well as via the list in the Bone properties. Bone visibility is determined by the visibility of its collections, its own 'solo' and 'hidden' properties:

- If the bone itself is marked as 'hidden', it is invisible regardless of the bone collections.
- If a parent collection is hidden, child collections will also be hidden; same is true for soloed collections.
- A bone is visible when it is contained in any visible collection.
- If a collection is soloed, it will be visible regardless of the collection's 'hidden' property.
- A bone that is not assigned to any bone collection is visible; otherwise it would be impossible to select it & assign it to a collection.

Library Overrides

Bone collections can be added using library overrides. For this to work, both the armature Object and the Armature itself need to be overridden.

Limitations

There are a few limitations when it comes to bone collections & overrides:

- Only bone collections that are local to the current blend file can be edited.
- Bone collections that already existed on the linked-in Armature are read-only, and only their visibility can be toggled. Those visibility changes won't be saved, though.
- Custom properties of overridden bone collections cannot be edited in the properties panel. Python access is fine; this is just a current limitation of Blender's UI code.

How It Works

Bone collections added via overrides are 'anchored' to the preceding collection, by name. Here is an example. The *italic* collections are defined on the linked Armature in `armature.blend`. The **bold** ones are added by overrides in `armature_shot_47.blend`.

- *FK Controls*
- *IK Controls*
- **Left Pinky** (anchored to "IK Controls")
- **Right Pinky** (anchored to "Left Pinky")

Now if the Armature in `armature.blend` gets updated with two more collections it might look like this:

- *FK Controls*
- *IK Controls*
- *Face Controls*
- *Face Detail Controls*

After reloading `armature_shot_47.blend`, it will look like this:

- *FK Controls*
- *IK Controls*
- **Left Pinky** (still anchored to “IK Controls”)
- **Right Pinky** (still anchored to “Left Pinky”)
- *Face Controls*
- *Face Detail Controls*

Some history

Bone Collections were introduced in Blender 4.0, as a replacement for armature layers and bone groups. Bone Collections are owned by the Armature, so they are available in all modes. To contrast, bone groups were stored on the object’s pose, and thus were not available in armature edit mode.

Structure

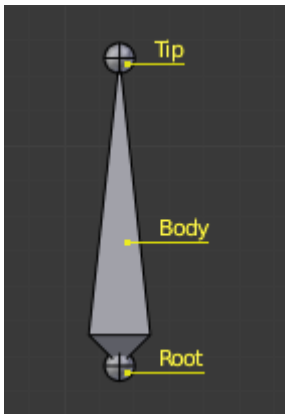
They have three elements:

1. The “start joint” named *root* or *head*.
2. The “body” itself.
3. And the “end joint” named *tip* or *tail*.

With the default armature in Edit Mode, you can select the root and the tip, and move them as you do with mesh vertices.

Both root and tip (the “joints”) define the bone by their respective position.

They also have a radius property, only useful for the envelope deformation method (see below).



The elements of a bone.

Roll

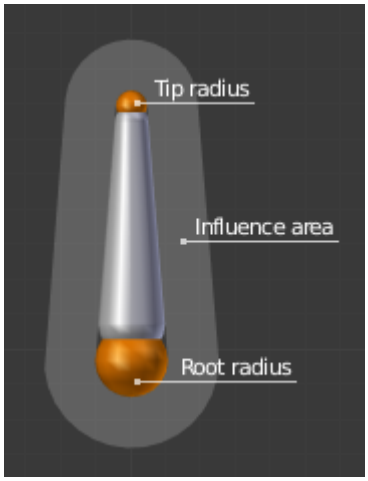
Activating the Axes checkbox will show local axes for each bone’s tip. The Y axis is always aligned along the bone, oriented from root to tip, this is the “roll” axis of the bones.

Bones Influence

Basically, a bone controls a geometry when vertices “follow” the bone. This is like how the muscles and skin of your finger follow your finger-bone when you move a finger.

To do this, you have to define the strength of *influences* a bone has on a certain vertex.

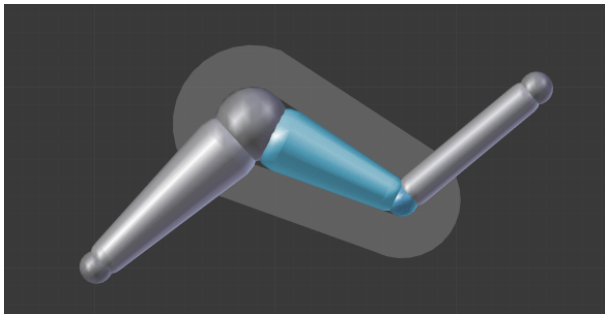
The simplest way is to have each bone affecting those parts of the geometry that are within a given range from it. This is called the *envelope technique*, because each bone can control only the geometry "enveloped" by its own influence area.



A bone in Envelope visualization, in Edit Mode.

If a bone is visualized as *Envelope*, in *Edit Mode* and in *Pose Mode* you can see the area of influence, which depends on:

- The *distance* property and
- the root's radius and the tip's radius.



Our armature in Envelope visualization, in Pose Mode.

All these influence parameters are further detailed in the skinning pages.

Tools

- Toolbar
- Tool Settings
  - Options

Toolbar

Mesh Edit Mode tools:

Select
: Select or move.

  Select Box
  : Select geometry by dragging a box.

  Select Circle
  : Select geometry by dragging a circle.

  Select Lasso
  : Select geometry by drawing a lasso.

Cursor
: Change the location of the 3D Cursor.

Move
: Translation tool.

Rotate
: Rotation tool.

Scale
: Scale tool.

  Scale Cage
  : Change the scale of an object by controlling its cage.

Transform
: Tool to adjust the objects translation, rotations and scale.

Annotate

Draw free-hand annotation.

Annotate Line
: Draw straight line annotation.

Annotate Polygon
: Draw a polygon annotation.

Annotate Eraser
: Erase previous drawn annotations.

Measure
: Measure distances in the scene.

Roll
: Rotates a bone around its local Y axis.

Bone Size
: Todo.

Bone Envelope
: Todo.

Extrude
: Creates a new bone connected to the last selected joint.

Extrude to Cursor
: Creates a new bone between the last selected joint and the mouse position.

Shear
: Todo.

Tool Settings

Options

X-Axis Mirror

See X-Axis Mirror Pose Mode.

Selecting

You can select and edit bones of armatures in *Edit Mode* and in *Pose Mode*. Here, we will see how to select bones in *Edit Mode*. Selecting bones in *Pose Mode* is similar to selecting in *Edit Mode* with a few specific differences that will be detailed in the posing part.

Similar to vertex/edge selection in meshes, there are two ways to select whole bones in *Edit Mode*:

1. Directly, by selecting the bone's body.
2. Selecting both of its joints (root and tip).

This is an important point to understand, because selecting bones' joints only might lead to non-obvious behavior, with respect to which bone you actually select.

Note that unlike the mesh display type, the armature display type has no effect on selection behavior. In other words, you can select a bone's joint or body the same way regardless of the bone visualization chosen.

Selecting Bone Joints

To select bones' joints you have the standard selection methods.

Inverse Selection

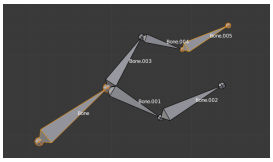
As stated above, you have to remember that these selection tools are for bones' joints only, not the bones' bodies.

For example, the *Inverse* selection option `Ctrl-I` inverts the selection of bones' joints, not of bones (see *Inverse selection*).

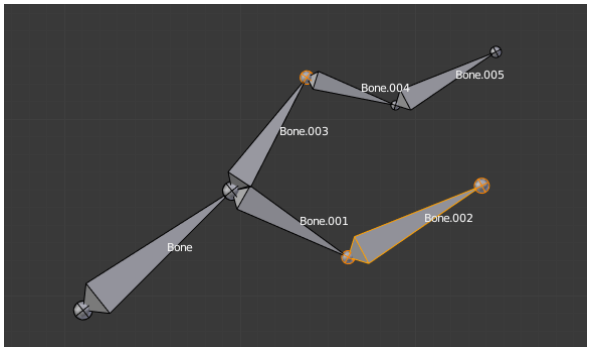
Remember that a bone is selected only if both its joints are selected. So, when the selection status of bones' joints is inverted, a new set of bones is

selected.

Inverse selection.



Two bones selected.



The result of the inverse selection `Ctrl-I`: The bones joints selection has been inverted, and not the bones selection.

Selecting Connected Bone Joints

Another example is: when you select the root of a bone connected to its parent, you also implicitly select the tip of its parent (and vice versa).

Note

Remember that when selecting bones' joints, the tip of the parent bone is the "same thing" as the root of its children bones.

Selecting Bones

By clicking on a bone's body, you will select it (and hence you will implicitly select its root and tip).

Using `Shift`-click, you can add to/remove from the selection.

You also have some *advanced selection* options, based on their relations.

Pick Shortest Path `Ctrl`-click
: Selects the path from the active bone to the bone under the mouse.

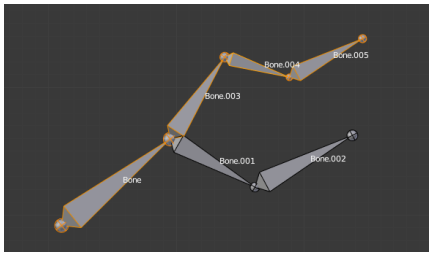
Deselecting Connected Bones

There is a subtlety regarding connected bones.

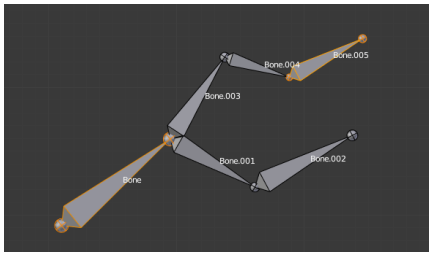
When you have several connected bones selected, if you deselect one bone, its tip will be deselected, but not its root, if it is also the tip of another selected bone.

To understand this, look at Fig. Bone deselection in a selected chain..

Bone deselection in a selected chain.



A selected chain.



Two selected bones.

After `Shift`-clicking “Bone.003”:

- “Bone.003” ‘s tip (which is same as “Bone.004” ‘s root) is deselected.
- “Bone” is “Bone.003” ‘s parent. Therefore, “Bone.003” ‘s root is the same as the tip of “Bone”. Since “Bone” is still selected, its tip is selected. Thus the root of “Bone.003” remains selected.

Select Mirror

Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select Mirror*
Shortcut:: `Shift-Ctrl-M`

Flip the selection from one side to another.

More/Less

Reference

Mode:: Edit Mode
Menu:: *Select ▸ More/Less*

More `Ctrl-NumpadPlus`
: Expand the current selection to the connected bones.

Less `Ctrl-NumpadMinus`
: Contrast the selection, deselect bones at the boundaries of each selection region.

Select Linked

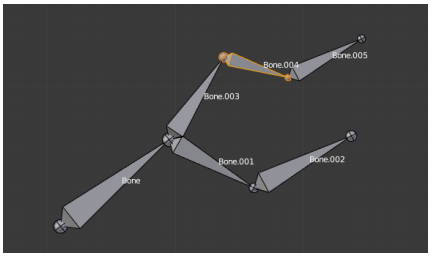
Reference

Mode:: Edit Mode
Menu:: *Select ▸ Select Linked*
Shortcut:: `Ctrl-L`

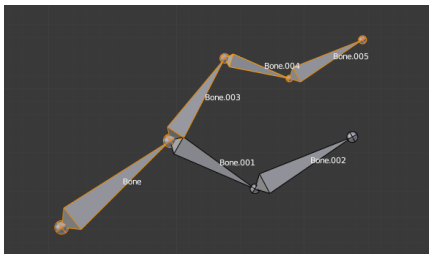
Selects all the bones in the chain which the active (last selected) bone belongs to.

All Forks
: Selects all bones connected to the active bone even if the branch off from the current bone.

Linked bones selection.



A single selected bone.



Its whole chain selected with Linked.

Select Similar

Reference

Mode::	Edit Mode
Menu::	*Select ▸ Select Similar*
Shortcut::	`Shift-G`

Children
: Extends the selection to all hierarchical descendant bones.

Immediate Children
: Extends the selection to all direct child bones.

Siblings
: Selects bones that have the same parent as the active bone.

Length
: Selects bones with a similar bone length under the specified *Threshold*.

Direction (Y axis)
: Select bones aligned on the Y axis (along the bone's length).

Prefix

Select bones with matching name prefix (separated by `.`).

Suffix
: Select bones with matching name suffix (separated by `.`).

Bone Collection
: Select bones that share one or more bone collections with the active bone.

Color
: Select bones that have the same color as the active bone.

Shape
: Select bones using the same shape object (in Pose Mode).

Select Pattern

Reference

Mode::	Pose & Armature Edit Modes
Menu::	*Select* ‣ *Select Pattern…*

Select bones by names, see Object Select Pattern for details.

Parent/Child

Parent `[`, Child `]`
: You can deselect the active bone and select its immediate parent or one of its children.

Extend Parent/Child

Extend Parent `Shift-[`, Extend Child `Shift-]`
: Similar to *Parent*/*Child* but it keeps the active bone in the selection.

Editing

- Introduction
  - Add Menu
  - Locking Bones
- Transform
  - Scale Radius
  - Scale Envelope Distance
  - Align Bones
- Bone Roll
  - Recalculate Roll
  - Set Roll
- Extrude
  - Mouse Clicks
- Duplicate
- Fill Between Joints
- Split
- Separate Bones
- Subdivide
- Switch Direction
- Symmetrize
- Naming
  - Naming Conventions
  - Auto-Name
  - Flip Names
- Parenting
  - Bone Collections
- Properties
- Delete
  - Bones
  - Dissolve

Introduction

As with any other object, you edit your armature in *Edit Mode* `Tab`.

The set of bone editing tools is quite similar to the one for mesh editing.

Important

One important thing to understand about armature editing is that you edit the *rest position* of your armature, i.e. its "default state". An armature in its *rest position* has all bones with *no* rotation and scaled to 1.0 in their own local space.

The different poses you might create afterwards are based on this rest position. So if you modify it in *Edit Mode*, all the poses already existing will also be modified. Thus you should in general be sure that your armature is definitive before starting to skin and pose it!

Note

Please note that some tools work on bones' joints, while others work on bones themselves. Be careful not to get confused.

Add Menu

Reference

Mode::	Edit Mode
Menu::	*Add*
Shortcut::	`Shift-A`

In the 3D Viewport, `Shift-A` to add a new bone to your armature.

This bone will be:

- Of one unit of length.
- Oriented towards the global Z axis.
- With its root placed at the 3D cursor position.
- With no relationship with any other bone of the armature.

Locking Bones

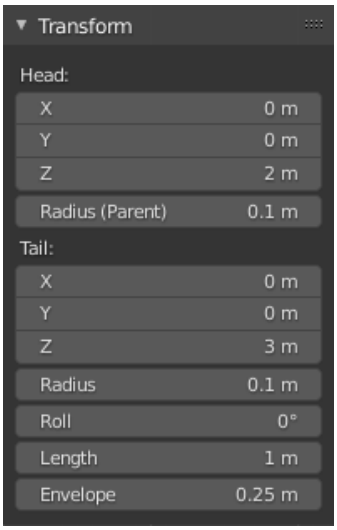
You can prevent a bone from being transformed in *Edit Mode* in several ways:

- All bones can be locked clicking on the *Lock* checkbox of their Transform panel in the *Bones* tab;
- Press `Shift-W` *Toggle Bone Options* ▸ *Locked*
- Select *Armature* ▸ *Bone Settings* ▸ *Toggle a Setting*.

If the root of a locked bone is connected to the tip of an unlocked bone, it will not be locked, i.e. you will be able to move it to your liking. This means that in a chain of connected bones, when you lock one bone, you only really lock its tip. With unconnected bones, the locking is effective on both joints of the bone.

Transform

We will not detail here the various transformations of bones, nor things like axis locking, pivot points, and so on, as they are common to most object editing, and already described in the mesh section. The same goes for mirroring, as it is nearly the same as with mesh editing. Just keep in mind that bones' roots and tips behave more or less like meshes' vertices, and bones themselves act like edges in a mesh.



The Transform panel for armatures in Edit Mode.

As you know, bones can have two types of relationships: They can be parented, and in addition connected. Parented bones behave in *Edit Mode* exactly as if they had no relations. They can be moved, rotated, scaled, etc. without affecting their descendants. However, connected bones must always have parent's tips connected to child's roots, so by transforming a bone, you will affect all its connected parent/children/siblings.

While with other transform tools, the "local axes" means the object's axes, here they are the bone's own axes (when you lock to a local axis, by pressing the relevant key twice, the constraint is applied along the selected bone's local axis, not the armature object's axis).

Finally, you can edit in the *Transform* panel in the Sidebar region the positions and radius of both joints of the active selected bone, as well as its roll rotation.

Scale Radius

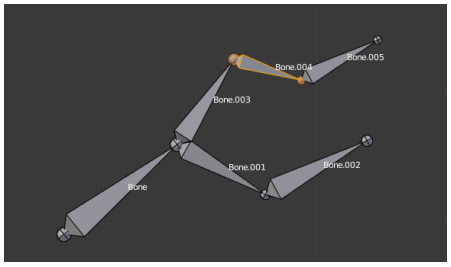
Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Transform* ▸ *Scale Radius*
Shortcut::	`Alt-S`

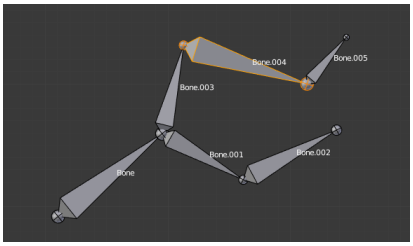
You can alter the radius that a bone has by selecting the head, body or tail of a bone, and then press `Alt-S` and move the mouse left or right. If the body is selected the mean radius will be scaled. And as usual, with connected bones, you scale at the same time the radius of the parent's tip and of the children's roots.

You can also alter the bone radius by selecting the tail or head of the bone you wish to alter, then navigate to *Properties* ▸ *Bone* ▸ *Deform* ▸ *Radius Section* and entering new values for the *Tail* and *Head* number fields.

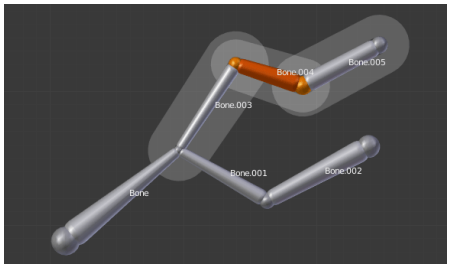
Bone Scale and Scale Radius comparison.



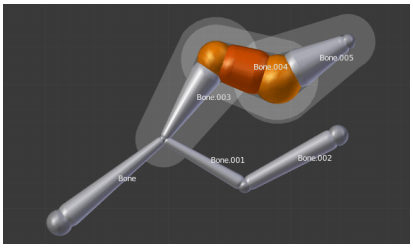
A single selected bone in Octahedron visualization.



After normal scale.



A single selected bone in Envelope visualization.



After Scaled Radius. Its length remains the same, but its joints' radius are bigger.

Note that, when you resize a bone (either by directly scaling it, or by moving one of its joints), Blender automatically adjusts the end-radii of its envelope proportionally to the size of the modification. Therefore, it is advisable to place all the bones first, and only then edit their properties.

Scale Envelope Distance

Reference

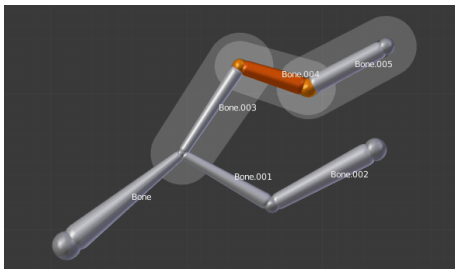
Mode::	Edit Mode and Pose Mode
Menu::	*Armature* ▸ *Transform* ▸ *Scale Envelope Distance*
Shortcut::	`Ctrl-Alt-S`

You can alter the size of the Bone Envelope volume by clicking on the body of the bone you want to alter, `Ctrl-Alt-S` then drag your mouse left or right and the Bone Envelope volume will alter accordingly.

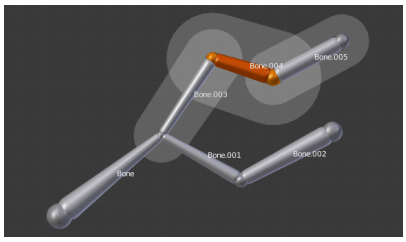
You can also alter the Bone Envelope volume by selecting the Bone you wish to alter and then navigate to *Properties* ▸ *Bone* ▸ *Deform* ▸ *Envelope* ▸ *Distance* then enter a new value into it.

Altering the Bone Envelope volume does not alter the size of the bone just the range within which it can influence vertices of child objects.

Envelope scaling example.

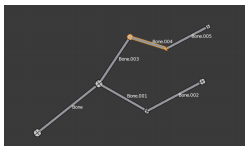


A single bone selected in Envelope visualization.

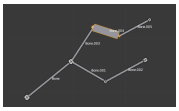


Its envelope distance scaled.

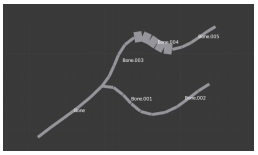
“Bone size” scaling example.



A single “default size” bone selected in B-Bone visualization.



Its envelope distance scaled.



The same armature in Object Mode and B-Bone visualization, with Bone.004’s size scaled up.

Align Bones

Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Transform* ▸ *Align Bones*
Shortcut::	`Ctrl-Alt-A`

Rotates the selected bones to achieve the same orientation as the active one.

Bone Roll

In *Edit Mode*, you can control the bone roll (i.e. the rotation around the Y axis of the bone).

However, after editing the armature, or when using Euler Rotation, you may want to set the bone roll.

Recalculate Roll

Reference

Mode::	Edit Mode
Menu::	*Armature* ‣ *Bone Roll* ‣ *Recalculate Roll*
Shortcut::	`Shift-N`

Axis Orientation
: Local Tangent
  : Align roll relative to the axis defined by the bone and its parent.

    X, Z

: Global Axis
  : Align roll to global X, Y, Z axis.

    X, Y, Z

: Active Bone
  : Follow the rotation of the active bone.

: View Axis
  : Set the roll to align with the viewport.

: Cursor

Set the roll towards the 3D cursor.

Flip Axis
: Reverse the axis direction.

Shortest Rotation
: Avoids rolling the bone over 90 degrees from its current value.

Set Roll

Reference

Mode::	Edit Mode
Menu::	*Armature* ‣ *Bone Roll* ‣ *Set Roll*
Shortcut::	`Ctrl-R`

This is a transform mode where you can edit the roll of all selected bones.

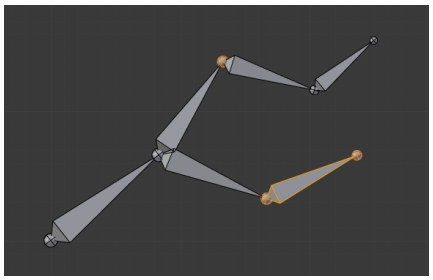
Extrude

Reference

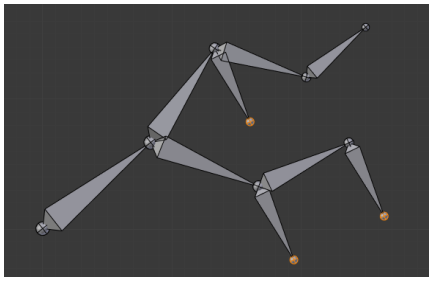
Mode::	Edit Mode
Menu::	*Armature* ▸ *Extrude*
Shortcut::	`E`, `Shift-E`

When you press `E`, for each selected tip (either explicitly or implicitly), a new bone is created. This bone will be the child of "its" tip owner, and connected to it. As usual, once extrusion is done, only the new bones' tips are selected, and in select mode, so you can place them to your liking. See Fig. Extrusion example..

Extrusion example.



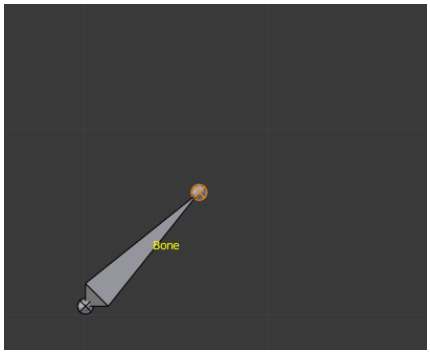
An armature with three selected tips.



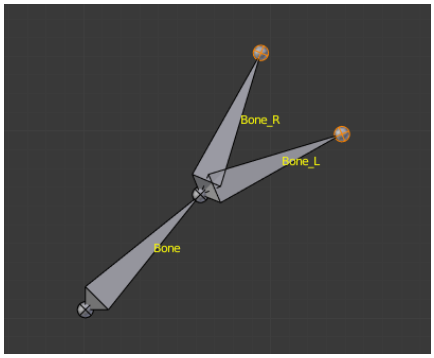
The three extruded bones.

You also can use the rotating/scaling extrusions, as with meshes, by pressing respectively `E R` and `E S` – as well as locked extrusion along a global or local axis.

Mirror extrusion example.



A single selected bone's tip.



The two mirror-extruded bones.

Bones have an extra "mirror extruding" tool, called by pressing `Shift-E`. By default, it behaves exactly like the standard extrusion. But once you have enabled X-Axis Mirror editing option, each extruded tip will produce *two new bones*, having the same name except for the "_L"/ "_R" suffix (for left/right, see the naming conventions). The "_L" bone behaves like the single one produced by the default extrusion – you can move, rotate or scale it exactly the same way. The "_R" bone is its mirror counterpart (along the armature's local X axis), see Fig. Mirror extrusion example..

Important

Canceling the extrude action causes the newly created bones to snap back to the source position, (creating zero length bones). These will be removed when exiting Edit Mode, however, they can cause confusion and it's unlikely you want to keep them. If you realize the problem immediately, undo the extrude action.

In case you are wondering, you cannot just press `X` to solve this as you would in mesh editing, because extrusion selects the newly created tips, and as explained below the Delete tool ignores bones' joints. To get rid of these extruded bones without undoing, you would have to move the tips, then select the bones and delete them.

Mouse Clicks

Reference

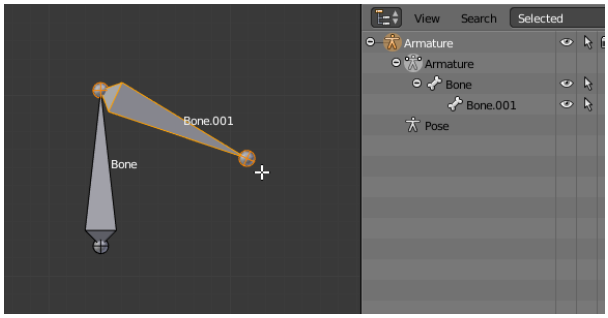
Mode::	Edit Mode
Shortcut::	`Ctrl-RMB`

If at least one bone is selected, `Ctrl-RMB`-clicking adds a new bone.

About the new bone's tip:

After you `Ctrl-RMB`-clicked it becomes the active element in the armature, it appears to be right where you clicked, but (as in mesh editing) it will be on the plane parallel to the view and passing through the 3D cursor.

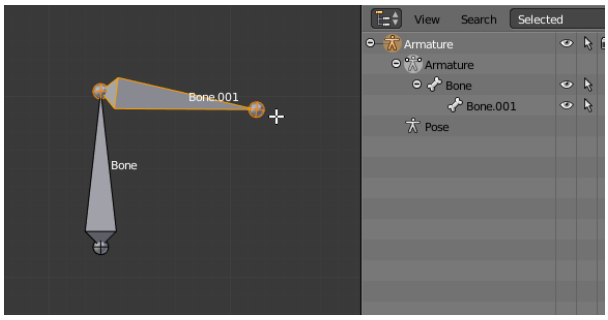
The position of the root and the parenting of the new bone depends on the active element:



`Ctrl`-clicking when the active element is a bone.

If the active element is a *bone*:

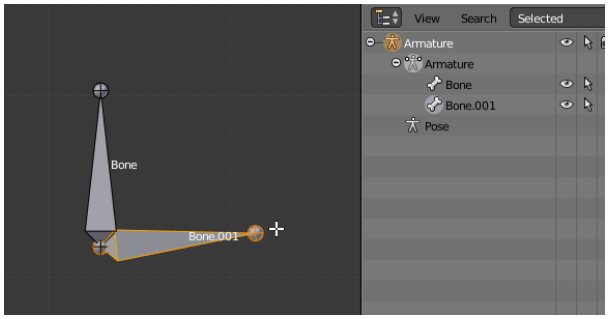
- The new bone's root is placed on the active bone's tip.
- The new bone is parented and connected to the active bone (check the Outliner in Fig. Ctrl-clicking when the active element is a tip.).



`Ctrl`-clicking when the active element is a tip.

If the active element is a *tip*:

- The new bone's root is placed on the active tip.
- The new bone is parented and connected to the bone owning the active tip (check the Outliner in Fig. Ctrl-clicking when the active element is a tip.).



`Ctrl`-clicking when the active element is a disconnected root.

If the active element is a *disconnected root*:

- The new bone's root is placed on the active root.
- The new bone is **not** parented to the bone owning the active root (check the Outliner in Fig. Ctrl-clicking when the active element is a disconnected root.).

And hence the new bone will **not** be connected to any bone.

Duplicate

Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Duplicate*
Shortcut::	`Shift-D`

Note

This tool works on selected bones; selected joints are ignored.

As in mesh editing, by pressing `Shift-D` the selected bones will be duplicated. The duplicates become the selected elements and they are placed in select mode, so you can move them wherever you like.

If you select part of a chain, by duplicating it you will get a copy of the selected chain, so the copied bones are interconnected exactly like the original ones.

The duplicate of a bone which is parented to another bone will also be parented to the same bone, even if the root bone is not selected for the duplication. Be aware, though, that if a bone is parented **and** connected to an unselected bone, its copy will be parented, but **not** connected to the unselected bone (see Fig. Duplication example.).

Duplication example.

Fill Between Joints

Reference

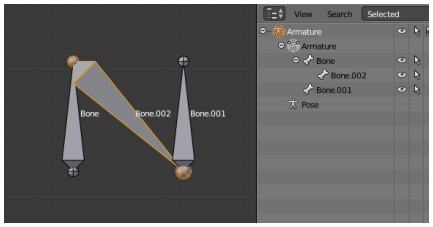
Mode::	Edit Mode
Menu::	*Armature ▸ Fill Between Joints*
Shortcut::	F

The main use of this tool is to create one bone between two selected joints by pressing F, similar to how in mesh editing you can “create edges/faces”.

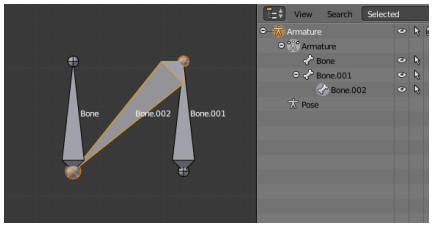
If you have one root and one tip selected, the new bone:

- Will have the root placed on the selected tip.
- Will have the tip placed on the selected root.
- Will be parented and connected to the bone owning the selected tip.

Fill between a tip and a root.



Active tip on the left.

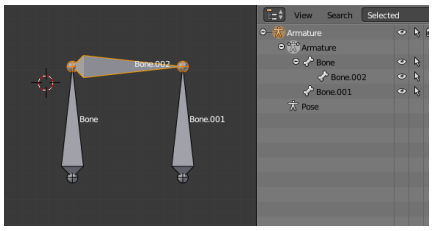


Active tip on the right.

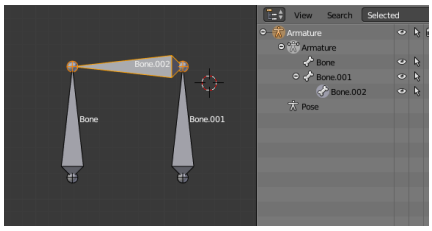
If you have two tips selected, the new bone:

- Will have the root placed on the selected tip closest to the 3D cursor.
- Will have the tip placed on the other selected tip.
- Will be parented and connected to the bone owning the tip used as the new bone’s root.

Fill between tips.



3D cursor on the left.



3D cursor on the right.

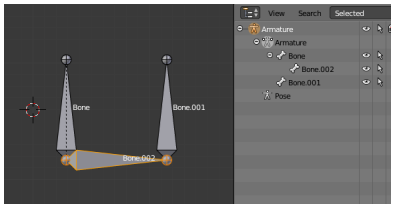
If you have two roots selected, you will face a small problem due to the event system in Blender not updating the interface in real-time.

When clicking `F`, similar to the previous case, you will see a new bone:

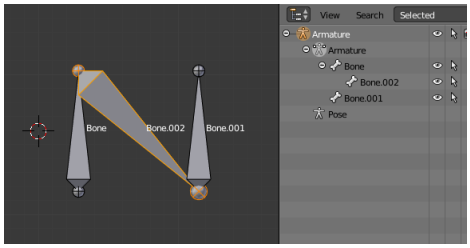
- With the root placed on the selected root closest to the 3D cursor.
- With the tip placed on the other selected root.
- Parented and connected to the bone owning the root used as the new bone's root.

If you try to move the new bone, Blender will update the interface and you will see that the new bone's root moves to the tip of the parent bone.

Fill between roots.



Before UI update (3D cursor on the left).

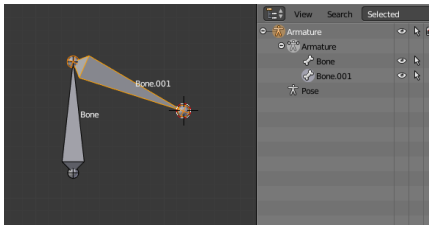


After UI update, correct visualization.

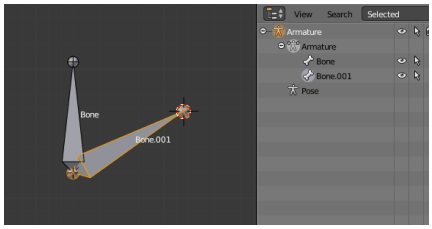
Clicking `F` with only one bone joint selected will create a bone from the selected joint to the 3D cursor position, and it will not parent it to any bone

in the armature.

Fill with only one bone joint selected.



Fill with only one tip selected.



Fill with only one root selected.

You will get an error when:

- Trying to fill two joints of the same bone.
- Trying to fill more than two bone joints.

Split

Reference

Mode::	Edit Mode
Menu::	*Armature* ‣ *Split*
Shortcut::	Y

Disconnects the selection and clears the parent at the start and end. ToDo <2.8 add.

Separate Bones

Reference

Mode::	Edit Mode
Menu::	*Armature ▸ Separate Bones*
Shortcut::	`P`

You can, as with meshes, separate the selected bones in a new armature object *Armature ▸ Separate*, `Ctrl-Alt-P` and of course, in *Object Mode*, you can join all selected armatures in one *Object ▸ Join Objects*, `Ctrl-J`.

Subdivide

Reference

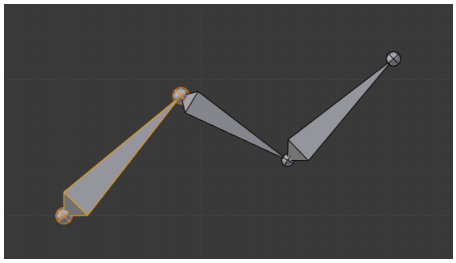
Mode::	Edit Mode
Menu::	*Armature* ‣ *Subdivide*

You can subdivide bones, to get two or more bones where there was just one bone. The tool will subdivide all selected bones, preserving the existing relationships: the bones created from a subdivision always form a connected chain of bones.

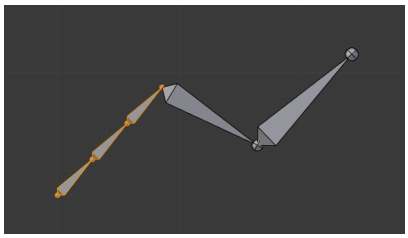
To create an arbitrary number of bones from each selected bone in the Subdivide Multi Adjust Last Operation panel.

Number of Cuts
: Specifies the number of cuts. As in mesh editing, if you set *n* cuts, you will get *n* + 1 bones for each selected bone.

Subdivision example.



An armature with one selected bone, just before multi-subdivision.



The selected bone has been “cut” two times, giving three sub-bones.

Switch Direction

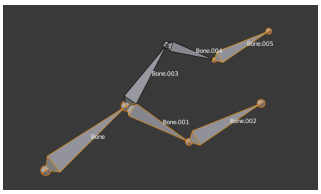
Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Switch Direction*
Shortcut::	`Alt-F`

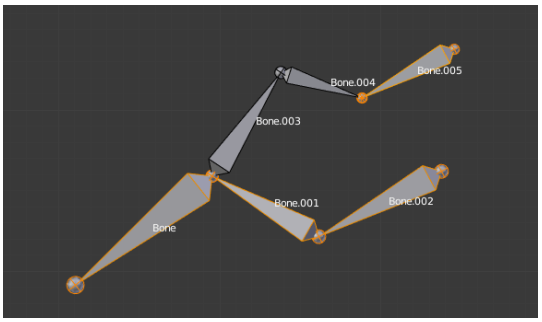
This tool allows you to switch the direction of the selected bones (i.e. their root will become their tip, and vice versa).

Switching the direction of a bone will generally break the chain(s) it belongs to. However, if you switch a whole (part of a) chain, the switched bones will still be parented/connected, but in "reversed order". See the Fig. Switching example..

Switching example.



An armature with one selected bone, and one selected chain of three bones, just before switching.



The selected bones have been switched. Bone.005 is no more connected nor parented to anything. The chain of switched bones still exists, but reversed (now Bone.002 is its root, and Bone is its tip). Bone.003 is now a free bone.

Symmetrize

Reference

Mode:: Edit Mode
Menu:: *Armature ▸ Symmetrize*

This operator will mirror the selected bones along the X axis based on Blender's bone naming convention for symmetrical armatures, either from left to right or right to left, depending on the selection.

Note

If the side of the bone cannot be determined, it will be ignored.

Bones with the opposite names that don't yet exist will be created, and already existing ones will be overwritten. If matching bones are selected on both sides, mirroring will happen from right to left.

Symmetrized bone and constraint properties will have the necessary changes to mirror their behaviors. When symmetrizing bones with Action Constraints, the necessary keyframes will be added to the target Action to result in symmetrical movement when the Action is activated.

Note

Note that bone or constraint drivers will not be created or affected in any way.

Naming

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Bone Properties*

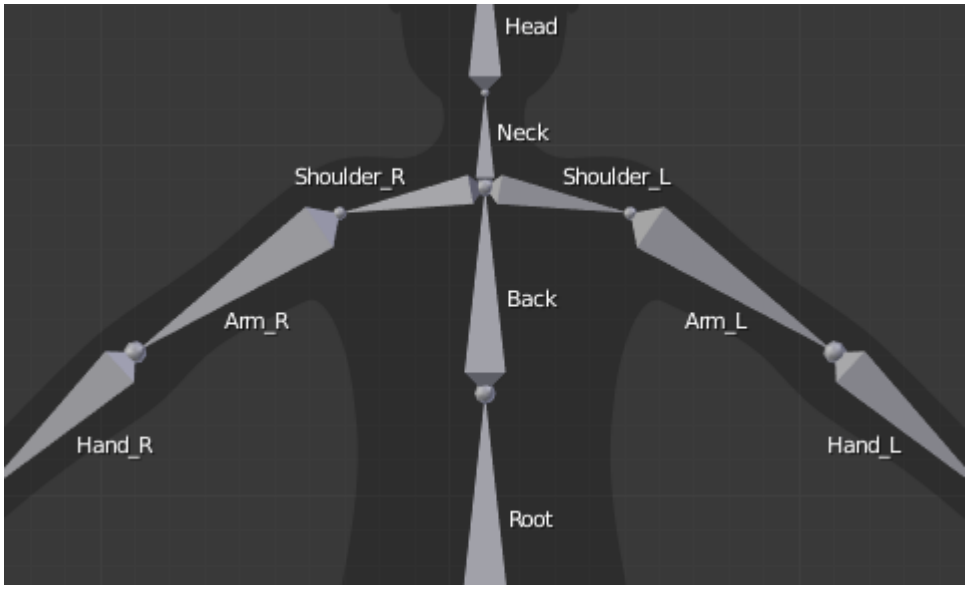
You can rename your bones, either using the *Name* field in the *Bones Properties*. It is also possible to rename by double-clicking bones in the Outliner.

Blender also provides you some tools that take advantage of bones named in a left/right symmetry fashion, and others that automatically name the bones of an armature.

Naming Conventions

Naming conventions in Blender are not only useful for you in finding the right bone, but also to tell Blender when any two of them are counterparts.

In case your armature can be mirrored in half (i.e. it is bilaterally symmetrical), it is worthwhile to stick to a left/right naming convention. This will enable you to use some tools that will probably save you time and effort (like the *X-Axis Mirror* editing tool).



An example of left/right bone naming in a simple rig.

1. First you should give your bones meaningful base-names, like “leg”, “arm”, “finger”, “back”, “foot”, etc.

2. If you have a bone that has a copy on the other side (a pair), like an arm, give it one of the following separators:

    - Left/right separators can be either the second position “L_calfbone” or last-but-one “calfbone**.**R”.
    - If there is a lower or upper case “L”, “R”, “left” or “right”, Blender handles the counterpart correctly. See below for a list of valid separators. Pick one and stick to it as close as possible when rigging; it will pay off.

Examples of valid separators:

- (nothing): handLeft –> handRight
- “_” (underscore): hand_L –> hand_R
- “.” (dot): hand**.**l –> hand**.**r
- “-” (dash): hand**-**l –> hand**-**r
- “ “ (space): hand LEFT –> hand RIGHT

Note

Note that all examples above are also valid with the left/right part placed before the name. You can only use the short “L”/ “R” code if you use a separator (e.g. “handL”/ “handR” will not work!).

3. Before Blender handles an armature for mirroring or flipping, it first removes the number extension, e.g. “.001”.

4. You can copy a bone named “bla.L” and flip it over using Flip Names. Blender will name the copy “bla.L.001” and flipping the name will give you “bla.R”.

Auto-Name

Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Names* ▸ *Auto-Name Left/Right, Front/Back, Top/Bottom*

The three *AutoName* entries of the *Armature* ▸ *Names* menu allow you to automatically add a suffix to all selected bones, based on the position of their root relative to the armature’s origin and its local coordinates:

AutoName Left/Right
: Will add the “.L” suffix to all bones with a *positive* X coordinate root, and the “.R” suffix to all bones with a *negative* X coordinate root. If the root is exactly at 0.0 on the X axis, the X coordinate of the tip is used. If both joints are at 0.0 on the X axis, the bone will just get a period suffix, with no “L”/ “R” (as Blender cannot decide whether it is a left or right bone…).

AutoName Front/Back
: Will add the “.Bk” suffix to all bones with a *positive* Y coordinate root, and the “.Fr” suffix to all bones with a *negative* Y coordinate root. The same as with *AutoName Left-Right* goes for 0.0 Y coordinate bones…

AutoName Top/Bottom
: Will add the ".Top" suffix to all bones with a *positive* Z coordinate root, and the ".Bot" suffix to all bones with a *negative* Z coordinate root. The same as with *AutoName Left-Right* goes for 0.0 Z coordinate bones…

Flip Names

Reference

Mode::	Edit Mode
Menu::	*Armature ▸ Names ▸ Flip Names*

You can flip left/right markers (see above) in selected bone names. This can be useful if you have constructed half of a symmetrical rig (marked for a left or right side) and duplicated and mirrored it, and want to update the names for the new side. Blender will swap text in bone names according to the above naming conventions, and remove number extensions if possible.

Parenting

Reference

Mode::	Edit Mode
Menu::	*Armature ▸ Parent*
Panel::	*Properties ▸ Bones ▸ Relations*
Shortcut::	`Ctrl-P`, `Alt-P`

You can edit the relationships between bones (and hence create/modify the chains of bones) both from the 3D Viewport and the Properties. Whatever method you prefer, it's always a matter of deciding, for each bone, if it has to be parented to another one, and if so, if it should be connected to it.

To parent and/or connect bones, you can:

In the *3D Viewport*, select the bone and *then* its future parent, and press `Ctrl-P` (or *Armature ▸ Parent ▸ Make Parent…*). In the small *Make Parent* menu that pops up, choose *Connected* if you want the child to be connected to its parent, else click on *Keep Offset*. If you have selected more than two bones, they will all be parented to the last selected one. If you only select one already-parented bone, or all selected bones are already parented to the last selected one, your only choice is to connect them, if not already done. If you select only one non-parented bone, you will get the *Need selected bone(s)* error message…

Note

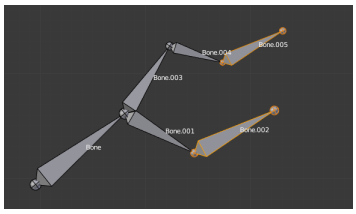
With this method, the newly-children bones will not be scaled nor rotated – they will just be moved if you choose to connect them to their parent's tip.

In the *Properties*, Bones tab, for each selected bone, you can select its parent in the *Parent* data ID to the upper right corner of its Relations panel. If you want them to be connected, just enable the checkbox to the right of the list.

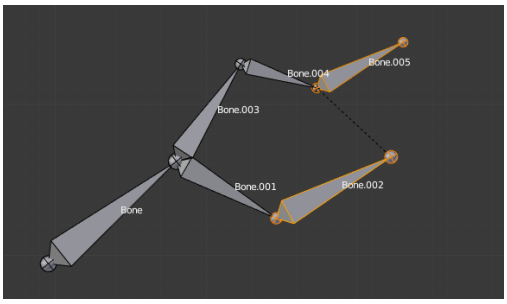
Note

With this method, the tip of the child bone will never be moved – so if *Connected* is enabled, the child bone will be completely transformed by the operation.

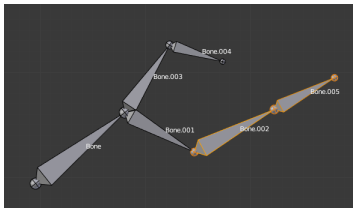
Parenting example.



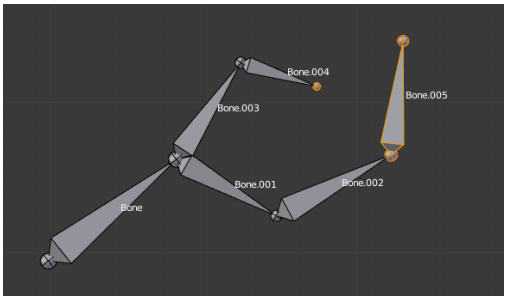
The starting armature, with Bone.005 parented and connected to Bone.004.



Bone.005 re-parented to Bone.002, but not connected to it (same result, using either `Ctrl-P 2` in 3D Viewport, or the Bones tab settings).



Bone.005 parented and connected to Bone.002, using



Bone.005 parented and connected to Bone.002.

`Ctrl-P 1` in 3D Viewport.	Using the Parent data ID of Bone.005 Relations panel.

To disconnect and/or free bones, you can:

- In a 3D Viewport, select the desired bones, and press `Alt-P` (or *Armature* ‣ *Parent* ‣ *Clear Parent*…). In the small *Clear Parent* menu that pops up, choose *Clear Parent* to completely free all selected bones, or *Disconnect Bone* if you just want to break their connections.
- In the Properties, *Bones* tab, for each selected bone, you can select no parent in the *Parent* data ID of its Relations panel, to free it completely. If you just want to disconnect it from its parent, disable the *Connected* checkbox.

Note that relationships with non-selected children are never modified.

Bone Collections

Reference

Mode::	Edit Mode, Pose Mode
Menu::	*Armature* ‣ *Bone Collections*, *Pose* ‣ *Bone Collections*

Manages the Bone Collections the bone is assigned to.

Move to Collection `M`
: Move bones to a collection.

Assign to Collection `Shift-M`
: Assign all selected bones to a collection, or unassign them, depending on whether the active bone is already assigned or not.

Show All `Ctrl-AccentGrave`

Unhides any hidden bone collections.

Assign to new Collection

Assigns the selected bones to a new collection named “New Collection”. This collection can be renamed in the Bone Collections panel of the Armature properties.

Properties

Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Bone Settings* ▸ …
Shortcut::	`Shift-W`, `Shift-Ctrl-W`, `Alt-W`

Most bones' properties (except the transform ones) are regrouped in each bone's panels, in the *Bones* tab in *Edit Mode*. Let us detail them.

Note that some of them are also available in the 3D Viewport, through the three pop-up menus within the same entry:

- *Toggle Setting*: `Shift-W` or *Armature* ▸ *Bone Settings* ▸ *Toggle a Setting*
- *Enable Setting*: `Shift-Ctrl-W` or *Armature* ▸ *Bone Settings* ▸ *Enable a Setting*
- *Disable Setting*: `Alt-W` or *Armature* ▸ *Bone Settings* ▸ *Disable a Setting*

Display Wire
: Always display the bone as wireframe.

Deform
: (also `Shift-W` ▸ *(Deform, …)*).

Multiply Vertex Group by Envelope
: (also `Shift-W` ▸ *(Multiply Vertex Group by Envelope, …)*).

  These settings control how the bone influences its geometry, along with the bones' joints radius. This will be detailed in the skinning part.

Inherit Rotation

The bone automatically rotates together with its parent in *Pose Mode*. For more details, see the relations page.

Lock

(also `Shift-W` ▸ *(Locked, …)*) This will prevent all editing of the bone in *Edit Mode*; see bone locking.

Delete

Bones

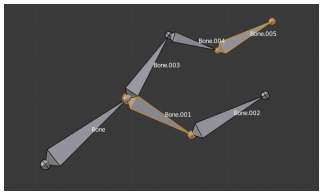
Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Delete* ▸ *Bones*
Shortcut::	X

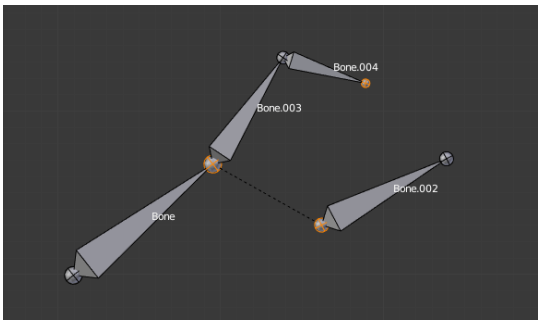
This tool delete selected bones, selected *joints* are ignored.

If you delete a bone in a chain, its child(ren) will be automatically re-parented to its own parent, but **not** connected, to avoid deforming the whole armature.

Deletion example.



An armature with two selected bones, just before deletion.



The two bones have been deleted. Note that Bone.002, previously connected to the deleted Bone.001, is now parented but not connected to Bone.

Dissolve

Reference

Mode::	Edit Mode
Menu::	*Armature* ▸ *Delete* ▸ *Dissolve*
Shortcut::	`Ctrl-X`

Todo 2.76.

Properties

Reference

Mode::	Object Mode, Edit Mode and Pose Mode
Panel::	*Properties* ▸ *Bone*

When bones are selected (hence in *Edit Mode* and *Pose Mode*), their properties are shown in the *Bone* tab of the Properties. This shows different panels used to control features of each selected bone; the panels change depending on which mode you are working in.

- Transform
- Bendy Bones
  - Technical Details
  - Display
  - Rest Pose
  - Example
  - Options
- Relations
  - Parenting
  - Bone Collections
- Inverse Kinematics
- Deform
  - Envelope
- Viewport Display
  - General
  - Bone Colors
  - Custom Shape
- Custom Properties

Transform

Reference

Mode::	Edit Mode and Pose Mode
Panel::	*Bone ▸ Transform*

When in *Edit Mode* you can use this panel to control position and roll of individual bones. Whereas in *Pose Mode* you can only set location for the main bone, and you can now set rotation and scale.

In addition, in *Pose Mode* it is possible to restrict changes in position, rotation and scale by axis on each bone in the armature.

Head X, Y, Z
: Location of head end of the bone.

Tail X, Y, Z
: Location of tail end of the bone.

Roll
: Bone rotation around head-tail axis.

Length
: The distance from the bone's head to it's tail. Changing the length moves the tail end.

Lock
: Bone is not able to be transformed when in Edit Mode.

Bendy Bones

Reference

Mode::	All Modes
Panel::	*Bone* ▸ *Bendy Bones*

Bendy Bones (B-Bones) are an easy way to replace long chains of many small rigid bones. A common use case for curved bones is to model spine columns or facial bones.

Technical Details

Blender treats the bone as a section of a Bézier curve passing through the bones' joints. Each of the *Segments* will bend and roll to follow this invisible curve representing a tessellated point of the Bézier curve. The control points at each end of the curve are the endpoints of the bone. The shape of the B-Bones can be controlled using a series of properties or indirectly through the neighboring bones (i.e. first child and parent). The properties construct handles on either end of the bone to control the curvature.

When using the B-bone as a constraint target Data ID offers an option to follow the curvature.

Note

However, if the bone is used as a target rather than to deform geometry, only *Armature* and *Copy Transforms* constraints will use the full transformation including roll and scale.

Display

You can see these segments only if bones are visualized as *B-bones*.

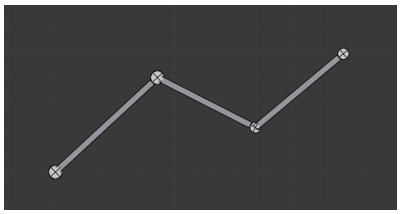
When not visualized as *B-Bone*s, bones are always shown as rigid sticks, even though the bone segments are still present and effective. This means that even in e.g. *Octahedron* visualization, if some bones in a chain have several segments, they will nonetheless smoothly deform their geometry.

Rest Pose

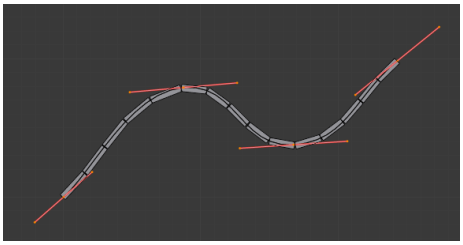
The initial shape of a B-Bone can be defined in Edit Mode as a rest pose of that bone. This is useful for curved facial features like curved eyebrows or mouths.

B-Bones have two sets of the Bendy Bone properties – one for Edit Mode (i.e. the Rest Pose/Base Rig) and another for Pose Mode – adding or multiplying together their values to get the final transforms.

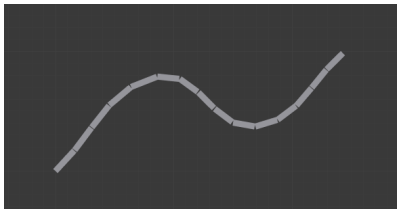
Example



Bones with just one segment in Edit Mode.



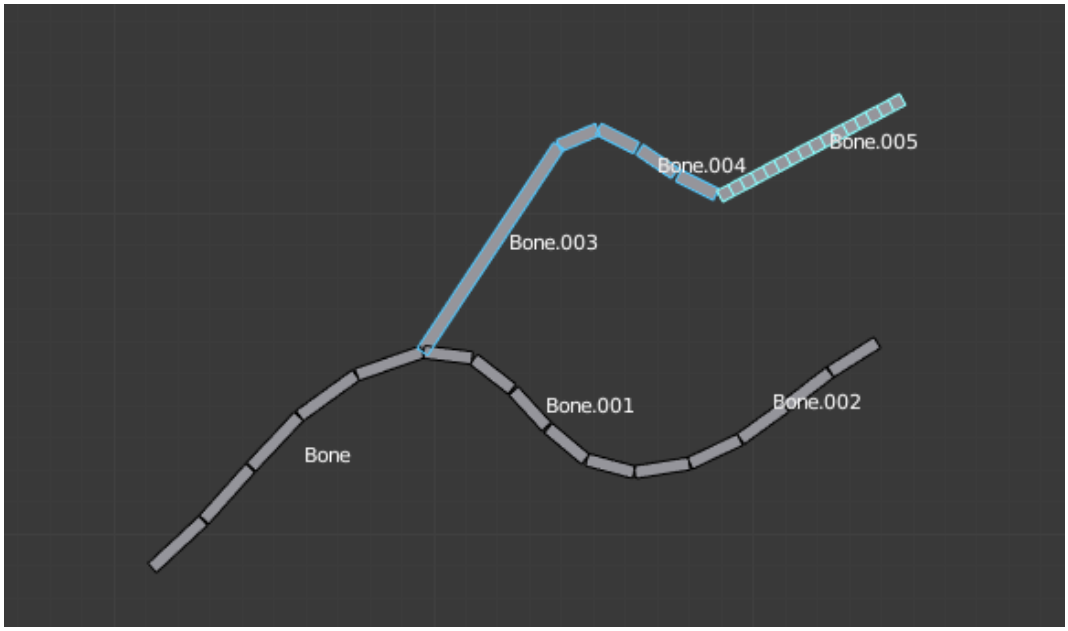
The Bézier curve superposed to the chain, with its handles placed at bones' joints.



The same armature in Object Mode.

In Fig. Bones with just one segment in Edit Mode. we connected three bones, each one made of five segments.

Look at Fig. The same armature in Object Mode., we can see how the bones' segments smoothly "blend" into each other, even for roll.



An armature in Pose Mode, B-Bone visualization: Bone.003 has one segment, Bone.004 has four, and Bone.005 has sixteen.

Options

Segments
: The number of segments, which the given bone is subdivided into. Segments are small, rigid linked child bones that interpolate between the root and the tip. The higher this setting, the smoother "bends" the bone, but the heavier the pose calculations.

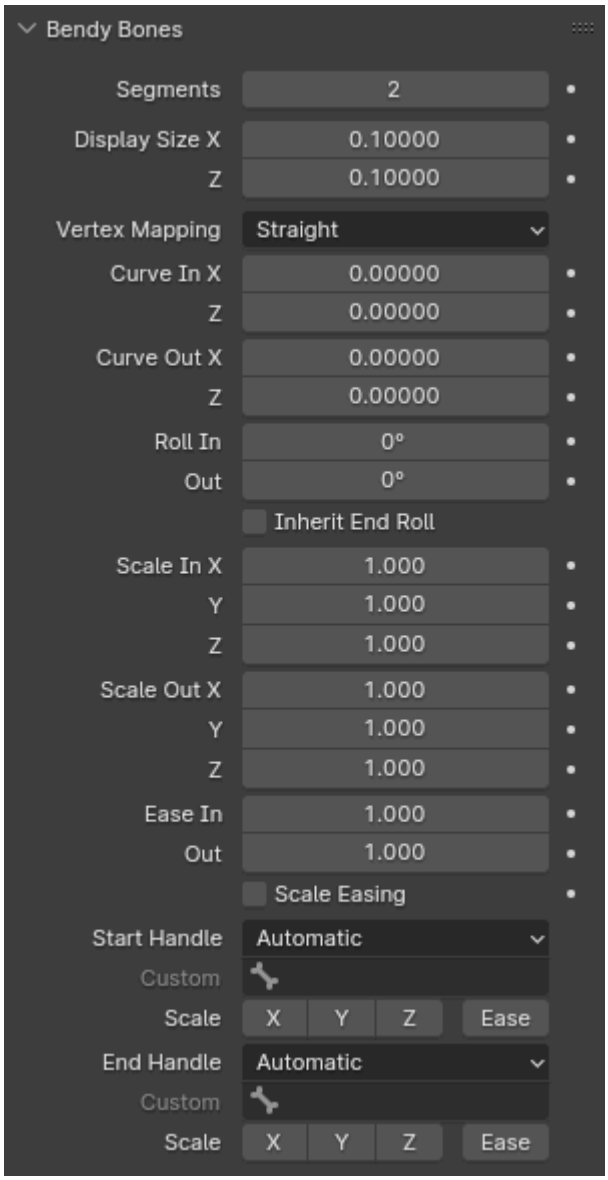
Display Size X, Z

Controls the visible thickness of the bone segments when the armature is rendered in the *B-Bones* mode.

Vertex Mapping

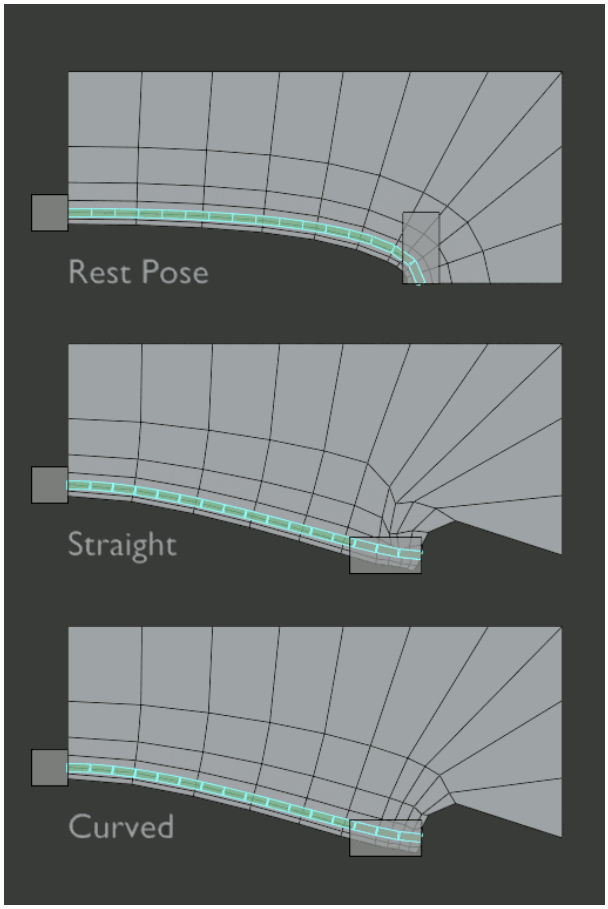
Controls how vertices are weighted to the individual segments of a B-Bone for deformations:

Straight::



Bendy Bones panel.

Curved:: A slower mapping that improves deformations for B-Bones with a strongly curved rest pose. This should be used selectively when needed.



Straight vs Curved vertex mapping on a B-Bone with a strongly curved rest pose.

Curve In/Out X, Y, Z

Applies offsets to the curve handle positions on the plane perpendicular to the bone's primary (Y) axis. As a result, the handle moves per axis (XZ) further from its original location, causing the curve to bend.

Roll In, Out

The roll value (or twisting around the main Y axis of the bone) is interpolated per segment, between the start and end roll values. It is applied as a rotational offset on top of the rotation defined by the handle bones.

Inherit End Roll

If enabled, the *Roll Out* value of the *Start Handle* bone (connected parent by default) will be implicitly added to the *Roll In* setting of the current bone.

Scale In/Out X, Y, Z

Scaling factors that adjust the thickness of each segment for the X and Z axes, or introduce non-uniform spacing along the Y axis. Similar to *Roll* it is interpolated per segment.

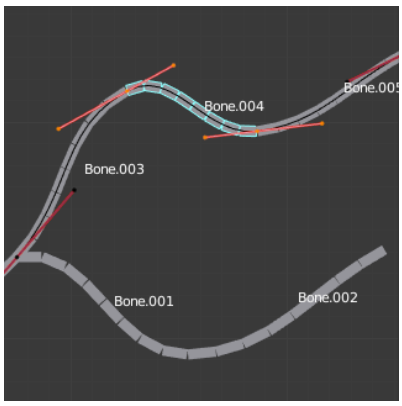
Since all segments are still uniformly scaled in the Y direction to fit the actual length of the curve, only the ratio between *Scale In Y* and *Scale Out Y* actually matters.

Ease In, Out

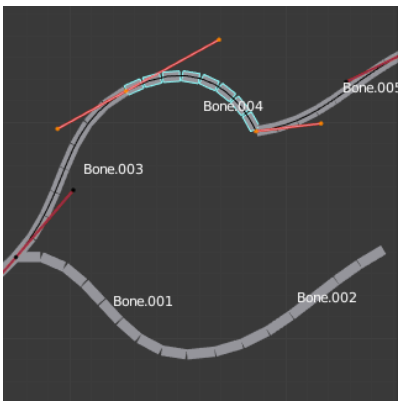
The *Ease In/Out* number fields, change the "length" of the "auto" Bézier handle to control the "root handle" and "tip handle" of the bone, respectively. These values are proportional to the default length, which of course automatically varies depending on bone length, angle with the reference handle, and so on.

Although easing is a scale-like value, the Edit Mode and Pose Mode versions of the values are added, so they get corresponding start values of 1 and 0 by default.

Ease In/Out settings example, with a materialized Bézier curve.



Bone.004 with default In and Out (1.0).



Bone.004 with In at 2.0, and Out at 0.0.

Scale Easing

If enabled, the final easing values are implicitly multiplied by the corresponding *Scale Y* values.

Custom Handles

B-Bones can use custom bones as their reference bone handles, instead of only using the connected parent/child bones.

Start/End Handle

Specifies the type of the handle from the following choices:

Automatic:: The connected parent (or first connected child) of the bone is chosen as the handle. Calculations are done according to the *Absolute* handle type below.

Absolute:: The Bézier handle is controlled by the **position** of the head (tail) of the handle bone relative to the head (tail) of the current bone. Note that for this to work, there must be a nonzero distance between these bones. If the handle is also a B-Bone, additional processing is applied to further smooth the transition, assuming that the bones in effect form a chain.

Relative:: The Bézier handle is controlled by the **offset** of the head (tail) of the handle bone from its rest pose. The use of this type is not recommended due to numerical stability issues near zero offset.

Tangent:: The Bézier handle is controlled by the **orientation** of the handle bone, independent of its location.

Custom Handle

For types other than *Automatic*, a bone to use as handle has to be manually selected. Switching to a custom handle type without selecting a bone can be used to effectively disable the handle.

It is valid for two bones to refer to each other as handles – this correlation is applied in connected chains with *Automatic* handles.

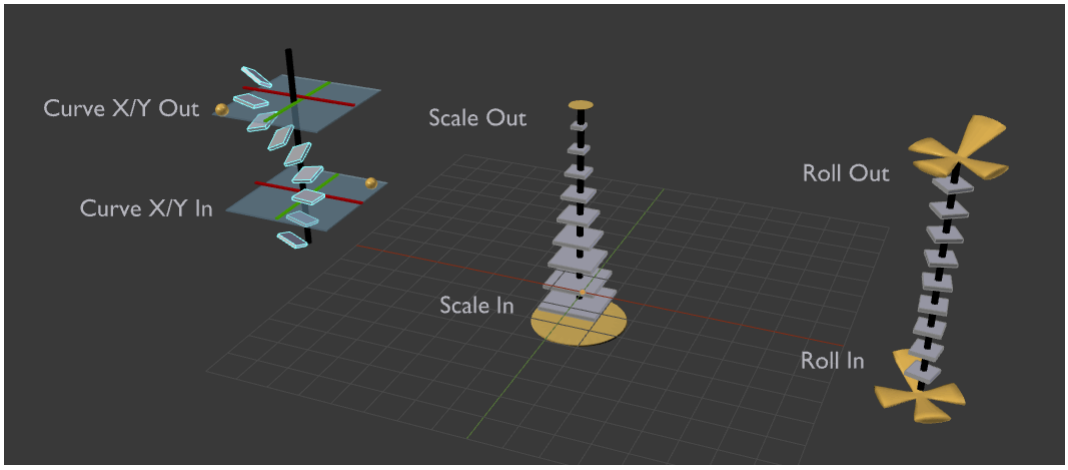
Scale X/Y/Z/Ease

If enabled, the final Scale and/or Ease values are multiplied by the corresponding local scale channels of the handle bone. This step is applied independently of *Scale Easing* and doesn't interact with it, i.e. enabling *Y* and *Scale Easing* doesn't replace the *Ease* toggle. These toggles are a more efficient replacement for up to eight trivial drivers passing segment scale data from the handle bones into the B-Bone option properties.

Tip

Keying Set

The "BBone Shape" Keying Set includes all Bendy Bones properties.



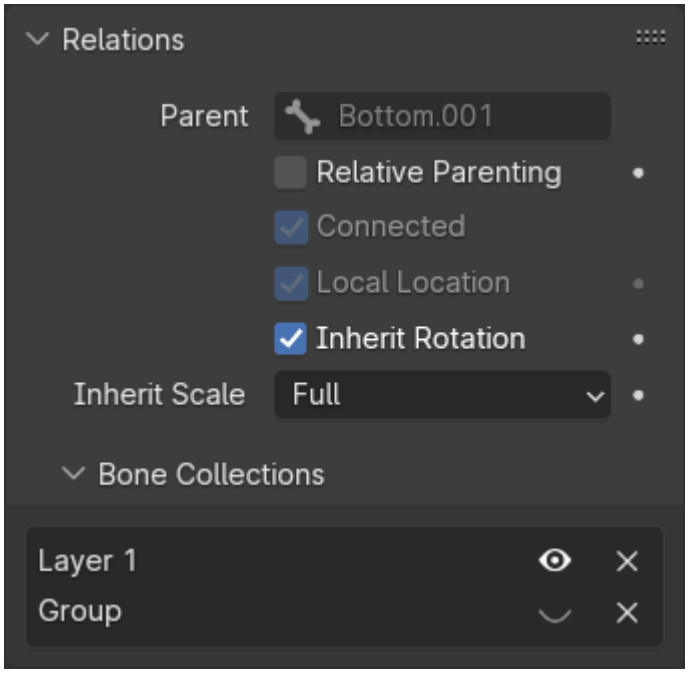
Visualization of the Bendy Bones properties.

From Left: 1) Curve X/Y offsets, 2) Scale In/Out, 3) Roll In/Out

Relations

Reference

Mode:: All Modes
Panel:: *Bone ▸ Relations*



Bone Relations panel.

In this panel you can manage the relationship of this bone with its parent bone. It also shows the bone collections the bone is assigned to.

Parenting

Parent
A Data ID to select the bone to set as a parent.

Relative Parenting *Pose Mode Only*

Changes how transformation of the bone is applied to its child Objects.

Connected
: The *Connected* checkbox set the head of the bone to be connected with its parent root.

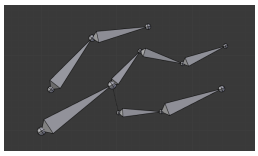
Transformations

Bones relationships have effects on transformations behavior.

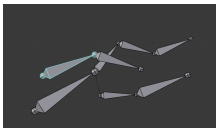
By default, children bones inherit:

- Their parent position, with their own offset of course.
- Their parent rotation (i.e. they keep a constant rotation relatively to their parent).
- Their parent scale, here again with their own offset.

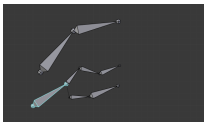
Examples of transforming parented/connected bones.



The armature in its rest position.



Rotation of a root bone.



Scaling of a root bone.

Exactly like standard children objects. You can modify this behavior on a per-bone basis, using the Relations panel in the *Bones* tab:

Local Location
: When disabled, the location transform property is evaluated in the parent bone's local space, rather than using the bone's own *rest pose* local space orientation.

Inherit Rotation

When disabled, this will “break” the rotation relationship to the bone’s parent. This means that the child will keep its rotation in the armature object space when its parent is rotated.

Inherit Scale

Specifies which effects of parent scaling the bone inherits:

These inheriting behaviors propagate along the bones’ hierarchy. So when you scale down a bone, all its descendants are by default scaled down accordingly. However, if you disable one bone’s *Inherit Scale* or *Inherit Rotation* property in this “family”, this will break the scaling propagation, i.e. this bone *and all its descendants* will no longer be affected when you scale one of its ancestors.

Full::	The bone inherits all effects of parent scaling and shear.
Fix Shear::	Full parent effects are applied to the rest state of the child, after which any shear is removed in a way that preserves the bone direction, length and volume, and minimally affects roll on average. The result is combined with the local transformation of the child. If the inherited scale is non-uniform, this does not prevent shear from reappearing due to local rotation of the child bone, or of its children.
Aligned::	Parent scaling is inherited as if the child was oriented the same as the parent, always applying parent X scale over child X scale, and so on.
Average::	Inherits a uniform scaling factor that is the total change in the volume of the parent.
None::	Ignores all scaling and shear of the parent.
None (Legacy)::	Ignores all scaling, provided the parent is not sheared. If it is, there are no guarantees. This choice replicates the behavior of the old Inherit Scale checkbox, and may be removed in a

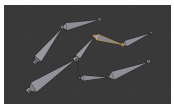
future release.

Tip

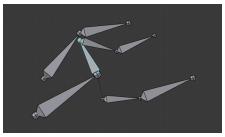
The various *Inherit Scale* options are provided as tools in avoiding shear that is caused by non-uniform scaling combined with parenting and rotation. There is no obvious best way to achieve that, so different options are useful for different situations.

- **None** – Useful for gaining full control over the scaling of the child in order to e.g. manually overwrite it with constraints.
- **Average** – Useful to block squash and stretch propagation between sub-rigs, while allowing uniform changes in the size and volume to pass through.
- **Aligned** – Can be used within bone chains, e.g. tentacles, in order to propagate lengthwise scaling as lengthwise, and sideways as sideways, no matter how the tentacle bends. Similar to using *None* with Copy Scale from parent.
- **Fix Shear** – May be useful at the base of an appendage in order to reallocate squash and stretch between axes based on the difference in rest pose orientations of the parent and child. It behaves closest to *Full* while suppressing shear.

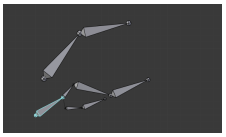
Examples of transforming parented/connected bones with Inherit Rotation disabled.



The yellow outlined Inherit Rotation disabled bone in the armature.



Rotation of a bone with an Inherit Rotation disabled bone among its descendants.

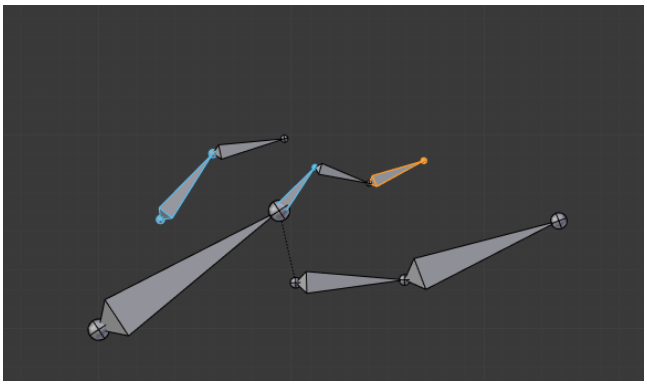


Scaling of a bone with an Inherit Rotation disabled bone among its descendants.

Connected bones have another specificity: they cannot be moved. Indeed, as their root must be at their parent's tip, if you do not move the parent, you cannot move the child's root, but only its tip, which leads to a child rotation. This is exactly what happens, when you press `G` with a connected bone selected, Blender automatically switches to rotation operation.

Bones relationships also have important consequences on how selections of multiple bones behave when transformed. There are many different situations which may not be included on this list, however, this should give a good idea of the problem:

- Non-related selected bones are transformed independently, as usual.
- When several bones of the same "family" are selected, *only* the "most parent" ones are really transformed – the descendants are just handled through the parent relationship process, as if they were not selected (see Fig. Scaling bones, some of them related. the third tip bone, outlined in yellow, was only scaled down through the parent relationship, exactly as the unselected ones, even though it is selected and active. Otherwise, it should have been twice smaller!)



Scaling bones, some of them related.

- When connected and unconnected bones are selected, and you start a move operation, only the unconnected bones are affected.
- When a child connected hinge bone is in the selection, and the "most parent" selected one is connected, when you press `G`, nothing happens,

because Blender remains in move operation, which of course has no effect on a connected bone.

So, when posing a chain of bones, you should always edit its elements from the root bone to the tip bone. This process is known as Forward Kinematics (FK). We will see in a later page that Blender features another pose method, called Inverse Kinematics (IK), which allows you to pose a whole chain just by moving its tip.

Note

This feature is somewhat extended/completed by the pose library.

Bone Collections

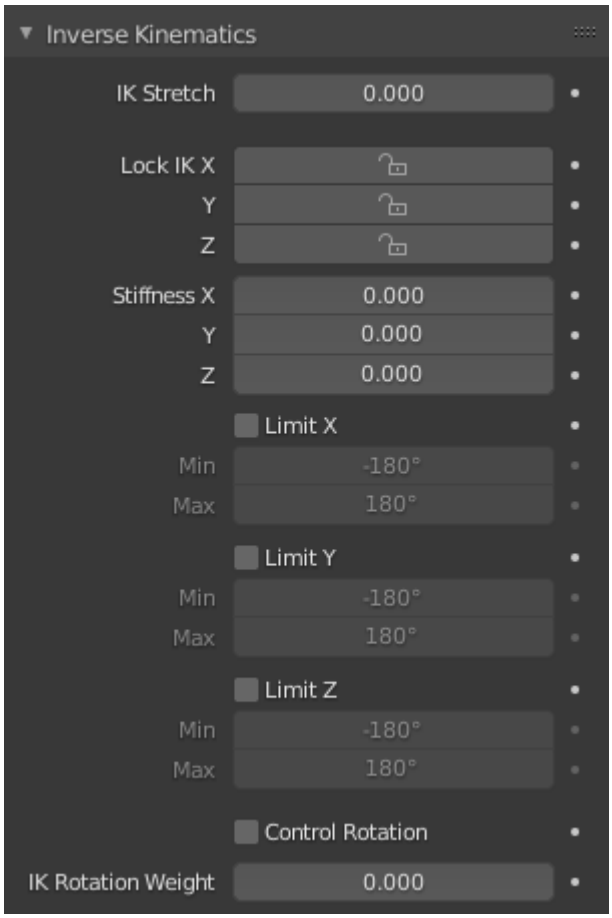
This list shows the bone collections the bone is assigned to. Press the eye icon to show or hide the entire bone collection. Press the star icon to show only this bone collection, and others also marked as ‘solo’ Press the X icon to remove the bone from that particular collection.

To assign the bone to other bone collections, either use the `M` or `Shift-M` shortcuts (see Moving Bones Between Collections) or go to the Armature properties panel.

Inverse Kinematics

Reference

Mode:: Pose Mode
Panel:: *Bone ▸ Inverse Kinematics*



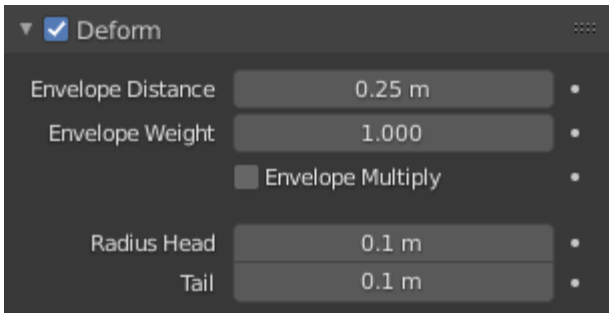
The Inverse Kinematics panel.

This panel controls the way a bone or set of bones behave when linked in an inverse kinematic chain.

Deform

Reference

Mode:: All Modes
Panel:: *Bone ▸ Deform*



The Deform panel.

In this panel you can set deformation options for each bone.

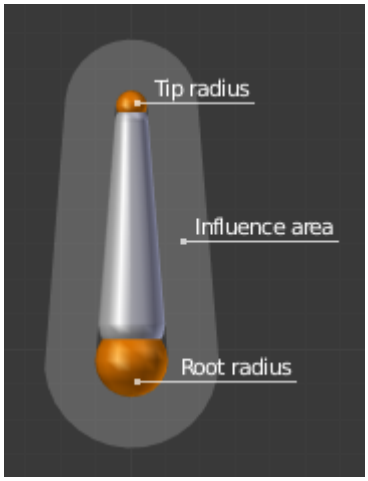
Toggling the checkbox in the panel header off, prevents the bone from deforming the geometry at all, overriding any weights that it might have been assigned before; it mutes its influence.

It also excludes the active bone in the automatic weight calculation when the mesh is parented to the armature using the *Armature Deform* tool with the *With Automatic Weights* option.

Envelope

Envelopes is the most general skinning method. It works with all available object types for skinning (meshes, lattices, curves, surfaces and texts). It is based on proximity between bones and their geometry, each bone having two different areas of influence, shown in the *Envelope* visualization:

- The inside area, materialized by the “solid” part of the bone, and controlled by both root and tip radius.
- The outside area, materialized by the lighter part around the bone, and controlled by the *Distance* setting.



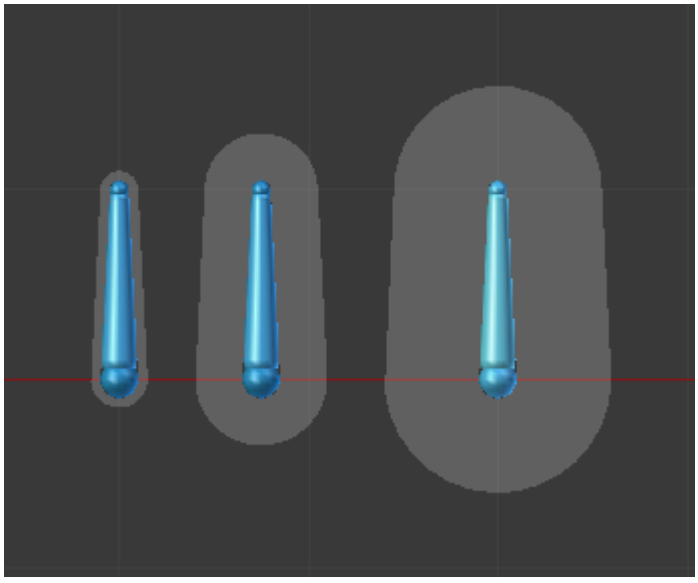
Bone influence areas for envelopes method.

See also

The editing pages for how to edit these properties.

Envelope Distance

The Distance defines a volume which is the range within the bone has an influence on vertices of the deformed object. The geometry is less and less affected by the bone as it goes away by following a quadratic decay.



Single bone with various envelope sizes.

Envelope Weight

A bone property, that controls the global influence of the bone over the deformed object, when using the envelopes method.

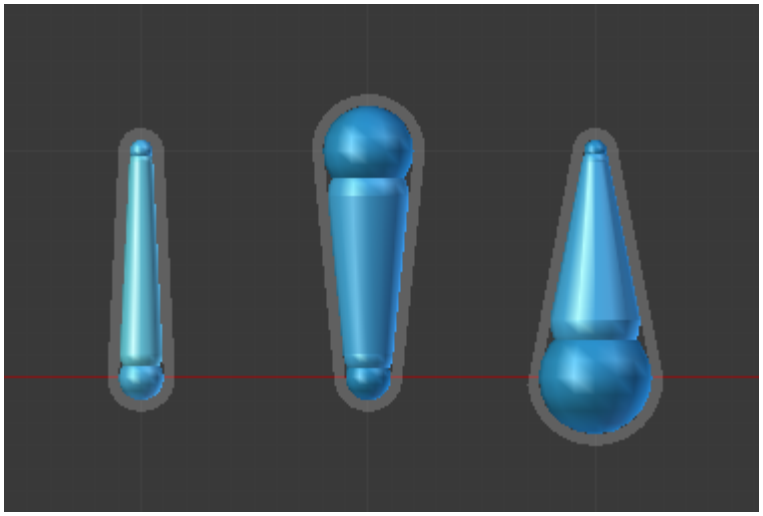
It is only useful for the parts of geometry that are “shared”, influenced by more than one bone (generally, at the joints…) – a bone with a high weight will have more influence on the result than one with a low weight… Note that when set to 0.0, it has the same effect as disabling the *Deform* option.

Envelope Multiply

This option controls how the two deforming methods interact, when they are both enabled. By default, when they are both active, all vertices belonging to at least one vertex group are only deformed through the vertex groups method. The other “orphan” vertices being handled by the envelopes one. When you enable this option, the “deformation influence” that this bone would have on a vertex (based from its envelope settings) is multiplied with this vertex’s weight in the corresponding vertex group. In other words, the vertex groups method is further “weighted” by the envelopes method.

Radius Head, Tail

Set the radius for the head and the tail of envelope bones. Inside this volume, the geometry if fully affected by the bone.



Three Armature Bones all using Envelope Weight.

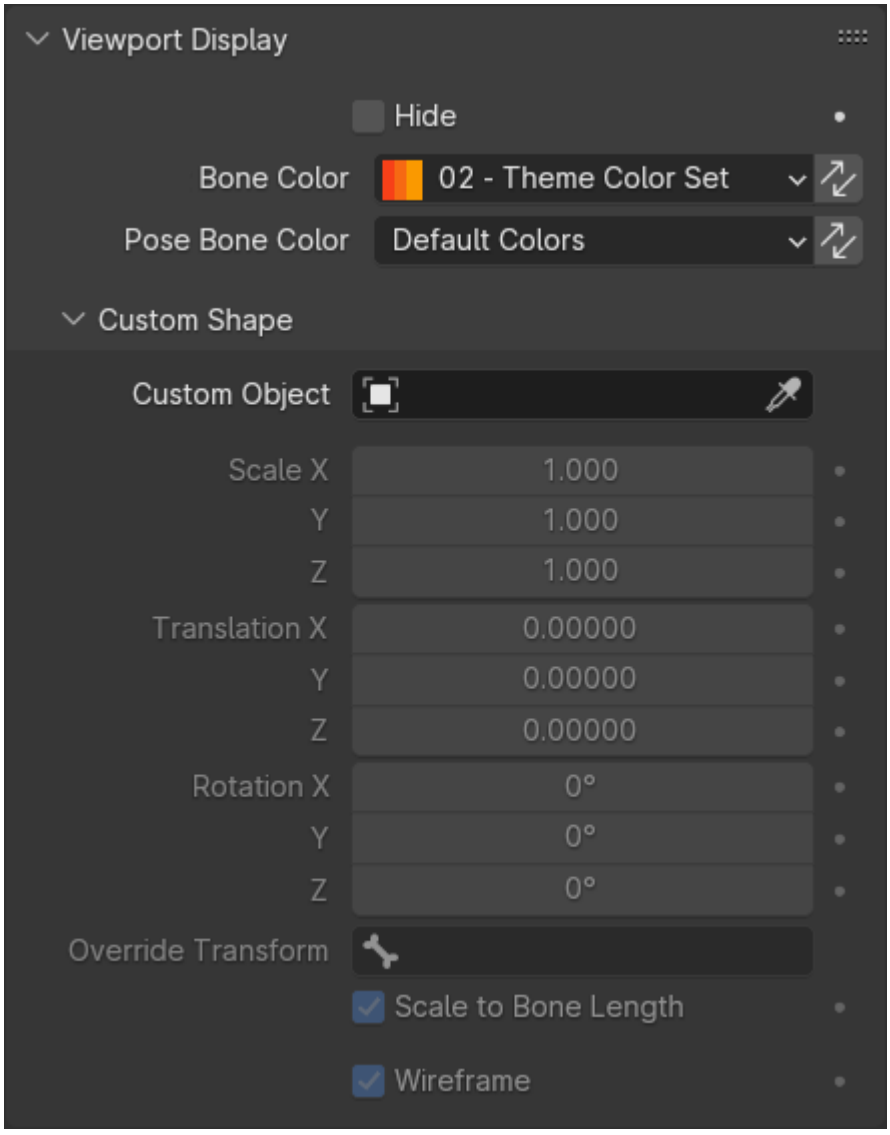
The 1st with a default radius value, the two others with differing Tail and Head radius values.

Viewport Display

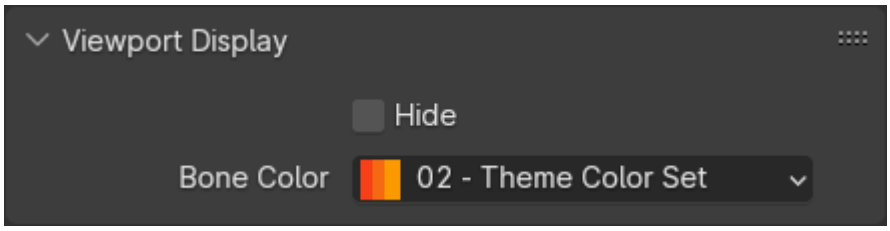
Reference

Mode:: Object, Pose, and Edit Mode
Panel:: *Bone ▸ Viewport Display*

This panel lets you customize the look of your bones.



Viewport Display panel in Object/Pose mode.



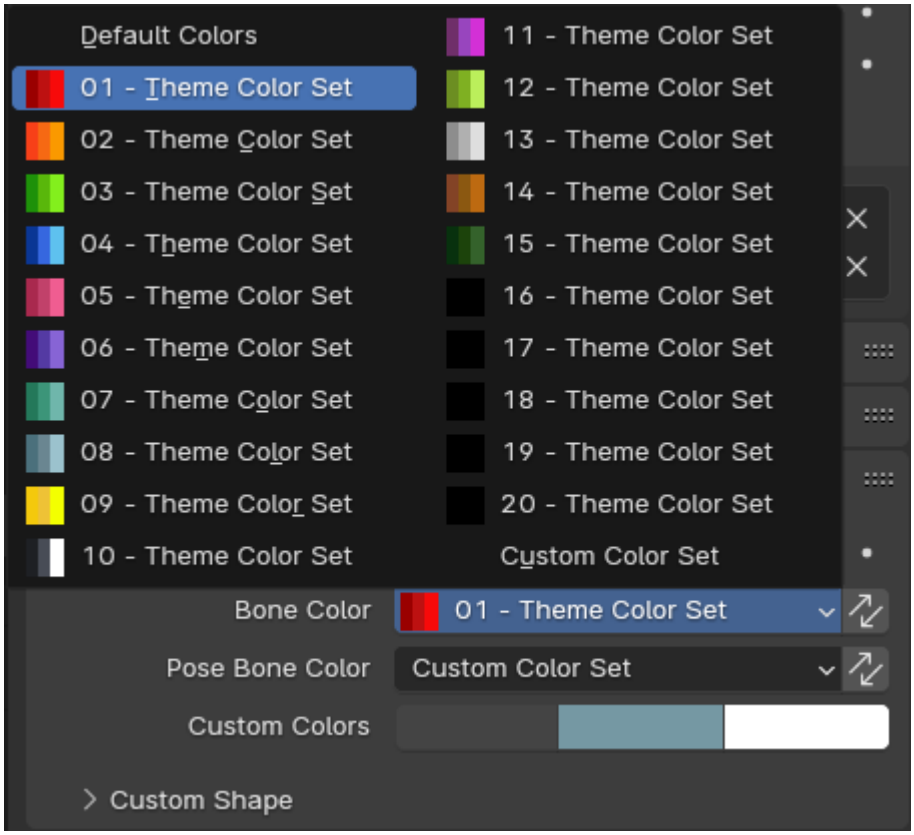
Viewport Display panel in Edit mode.

General

Hide
: Hides the bone in the 3D Viewport. When this is unchecked, the bone's visibility is determined by the visibility of its bone collections.

Bone Colors

Bones can be individually colored. You can either choose a color set from the predefined theme list or define a custom one.



When selecting *Custom Color Set*, you need to define three colors: Regular (for when the bone is not selected), Selected, and Active.

You can temporarily disable all the color assignments by unchecking Bone Colors in the armature's Viewport Display panel.

Bone Color
: The bone's primary color, affecting both Edit Mode and Pose Mode.

  This color is stored on the armature data-block, so that if you have multiple armature objects that share this data-block, they will all use the same color.

  Copy Bone Color to Selected
  : Copy the bone color of the Active bone to all selected bones.

Pose Bone Color *Pose Mode*

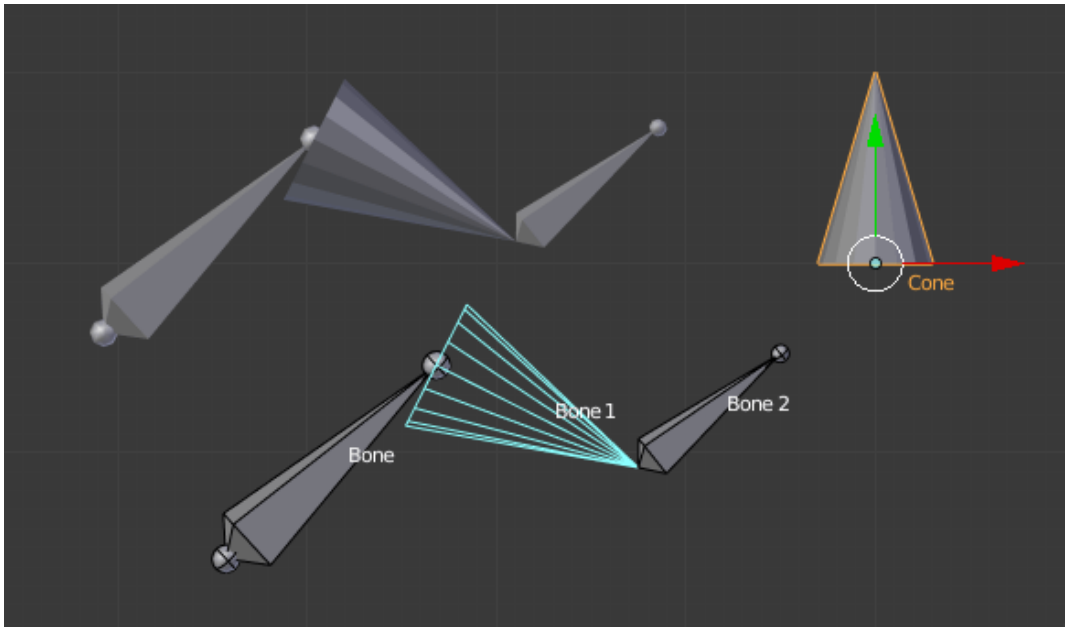
Lets you optionally override the above *Bone Color* in Pose Mode (by setting it to something else than *Default Colors*).

This color is stored on the Pose Bone, meaning it can be different in every armature object – even ones that reference the same data-block.

Copy Bone Color to Selected
: Copy the bone color of the Active bone to all selected bones.

Custom Shape

Apart from custom colors, bones can also have custom shapes (in *Object Mode* and *Pose Mode*), using another object as a "template."



A bone referencing a cone as its Custom Shape.

You can temporarily disable these shapes by unchecking *Shapes* in the armature's Viewport Display panel.

Custom Object
: Object that defines the custom shape of the selected bone.

Scale X, Y, Z

Additional scaling factor to apply to the custom shape.

Translation X, Y, Z
: Additional translation to apply to the custom shape.

Rotation X, Y, Z
: Additional rotation to apply to the custom shape.

Override Transform
: Bone that defines the display transform of the custom shape.

Scale to Bone Length
: Whether the custom shape should be scaled by a factor equal to the bone's length.

Wireframe
: When enabled, the bone is displayed in wireframe mode regardless of the viewport's shading mode.

Wire Width
: The line thickness of the wireframe for the custom shape.

Note

- Custom shapes will never be rendered. Like regular bones, they are only visible in the 3D Viewport.
- The transforms of the template object are ignored. Moving, rotating, or scaling it will have no effect on its appearance in the armature.
- The origin of each instanced shape object is at the root of the bone.
- The rotation of each shape object is such that its Y axis lies along the direction of the bone.
- For best results when Scale to Bone Length is enabled, make sure the template object is 1 unit in size along its Y axis. This will make it perfectly match the size of each bone.

Custom Properties

Reference

Mode::	Pose Mode
Panel::	*Bone* ‣ *Custom Properties*

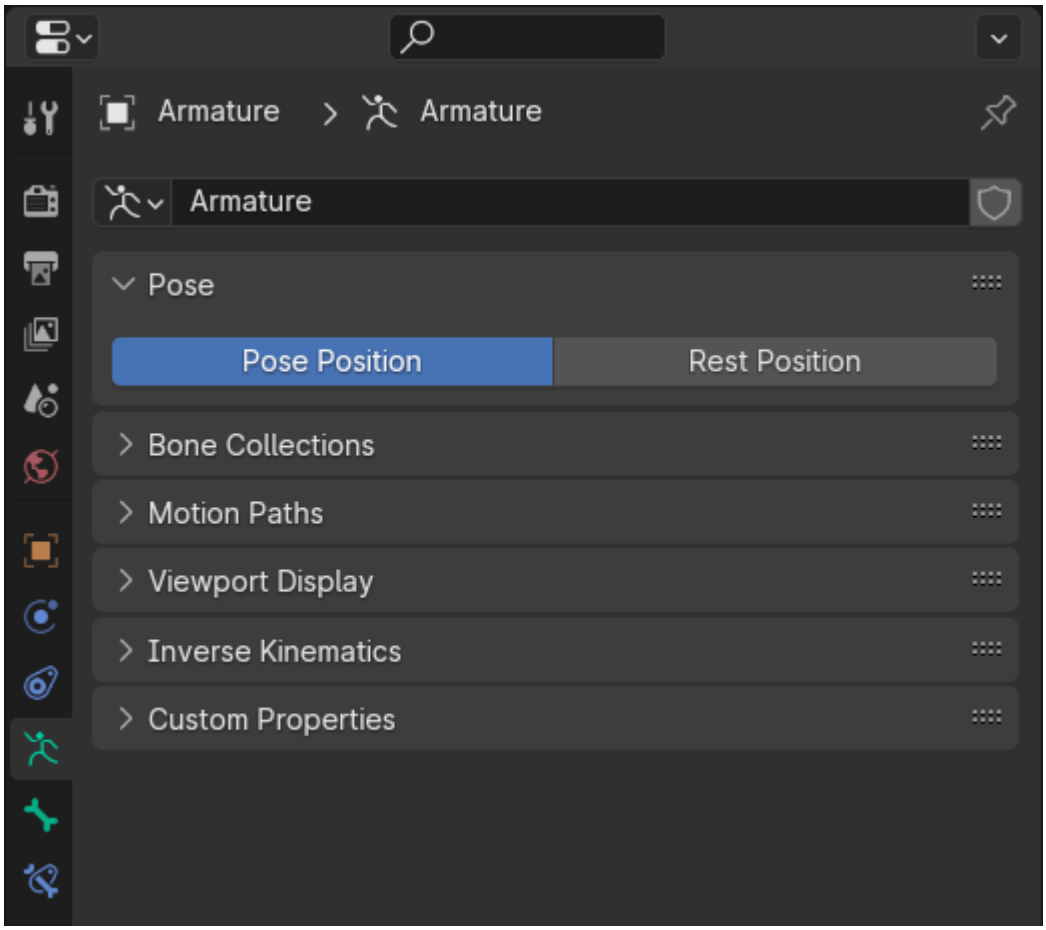
See the Custom Properties page for more information.

Properties

- Introduction
  - Pose
  - Bone Collections
  - Motion Paths
  - Inverse Kinematics
  - Custom Properties
- Bone Collections
  - Specials
  - Assign & Select
  - Moving Bones between Collections
  - Custom Properties
- Selection Sets
- Viewport Display

Introduction

The *Armature* tab in Properties contains various panels gathering the armature settings.



The Armature tab in the Properties.

Pose

Reference

Mode:: All Modes
Panel:: *Armature* ▸ *Pose*

Pose Position
: A radio button to switch between Pose Position and Rest Position.

  In *Edit Mode*, you always see armatures in their rest position, in *Object Mode* and *Pose Mode*, by default, you see them in *Pose Position* (i.e. as it was transformed in the *Pose Mode*). If you want to see it in the rest position in all modes, select *Rest Position*.

Bone Collections

See Bone Collections.

Motion Paths

Reference

Mode::	All Modes
Panel::	*Armature* ‣ *Motion Paths*

In the Motion Paths panel you can enable visualization of the motion path your skeleton leaves when animated.

Inverse Kinematics

Reference

Mode::	All Modes
Panel::	*Armature* ‣ *Inverse Kinematics*

Defines the type of IK solver used in your animation.

Custom Properties

Reference

Mode::	All Modes
Panel::	*Armature* ▸ *Custom Properties*

See the Custom Properties page for more information.

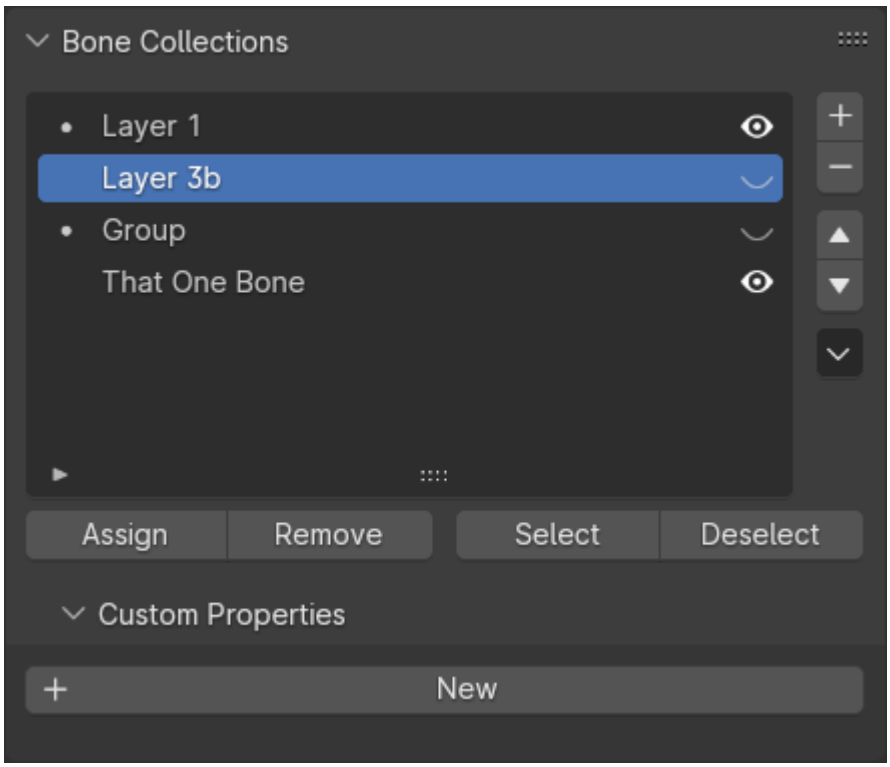
Bone Collections

Note

Bone Collections were introduced in Blender 4.0 as replacement of Armature Layers and Bone Groups. Bone colors are now managed directly on the bone.

Reference

Mode:: Pose & Armature Edit Modes
Panel:: *Properties ▸ Armature ▸ Bone Collections*
Menu:: *Pose ▸ Bone Collections ▸ …*



The Bone Collections panel in the Armature properties.

This panel contains a Tree View to manage Bone Collection From this panel, Bone Collections can be created, deleted, re-arranged, and more.

Collections can be renamed by double clicking on the name, or right clinking and selecting *Rename*. To nest a collection inside an existing collection, click and drag the name onto another collection's name. Child collection can also be made by `RMB` and selecting "Add Child Collection".

To the right of the name gives a few controls of the collection:

Visible (Eye)
: Bones in this collection will be visible in the 3D Viewport.

Solo (Star)
: Show only this bone collection, and others also marked as "solo".

Further more, collection that are not empty will have a dot to indicate the collection has bones assigned to it.

Tip

The Bone Properties panel gives a slightly different view on the bone's collections. See Bone Relations.

Specials

Show All
: Unhides any hidden bone collections.

Un-Solo All
: Clear the 'solo' setting on all bone collections

Remove Unused
: Remove all bone collections that have neither bones nor children. This is done recursively, so bone collections that only have unused children are also removed.

Assign & Select

Assign
: Assigns the selected bones to the active bone collection.

Remove
: Removes the selected bones from the active bone collection.

Select
: Selects the bones in the active bone collection.

Deselect
: Deselects the bones in the active bone collection.

Note

Individual bones can als be unassigned from their collections via the Bone Relations panel.

Tip

For setting up custom selection sets of bones, take a look at the *Selection Sets* add-on. It is bundled with Blender.

Moving Bones between Collections

Blender should be in *Edit Mode* or *Pose Mode* to move bones between collections. Note that as with objects, bones can be assigned to in several collections at once.

Move to Bone Collection
: Shows a list of the Armature's *editable* bone collections. Choosing a bone collection unassign the selected bones from all other bone collections, then assigns them to the chosen one.

Available as *Pose* ▸ *Move to Collection* (*Pose Mode*) *Armature* ▸ *Move to Collection* (*Edit Mode*), and `M` (either mode).

Bone Collections
: Shows a list of the Armature's *editable* bone collections. The collections that the active bone is assigned to are prefixed with a `-`, and choosing those will unassign all selected bones from that collection. Similarly, choosing a bone collection prefixed with a `+` will assign all selected bones to that collection.

  Available as *Pose* ▸ *Bone Collections* (*Pose Mode*) *Armature* ▸ *Bone Collections* (*Edit Mode*), and `Shift-M` (either mode).

Note

The above operators will only show the *editable* bone collections. When the Armature is linked, its bone collections will be *read-only*. New bone collections can still be added via library overrides; only those will be editable.

See Library Overrides of Bone Collections.

Custom Properties

Create and manage your own properties to store data in the Bone Collection's data-block. See the Custom Properties page for more information.

Selection Sets

Reference

Mode::	Pose Mode
Panel::	*Armature* ‣ *Selection Sets*

Selection Sets are a feature that allows the definition of sets of bones for easy selection while animating. The sets can be created in local and linked armature overrides.

Selection Set
: A List View listing all selection sets for the selected armature. Here, selection sets can be renamed by double clicking on the name.

  To the right of the name is a check box to include that selection set when copying to the clipboard.

Specials

Delete All Sets
: Removes all selection sets from the list.

Remove Selected Bones from All Sets
: Removes the selected bones from all selection sets.

Copy Selected Set(s)
: Copies the selected set to Blender's clipboard.

Paste Selected Set(s)
: Pastes a selection set from Blender's clipboard.

Assign
: Assigns the selected bones to the active selection set.

Remove
: Removes the selected bones to the active selection set.

Select
: Selects all the bones in the active selection set.

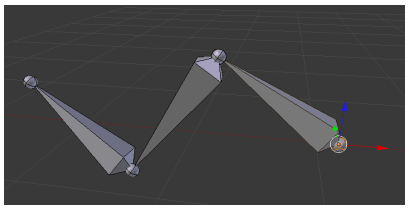
Deselect
: Deselects all the bones in the active selection set.

Viewport Display

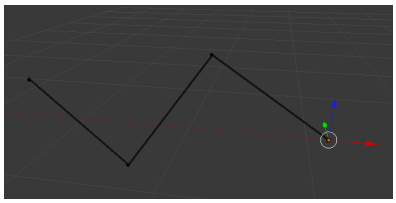
Reference

Mode:: All Modes
Panel:: *Armature ▸ Viewport Display*

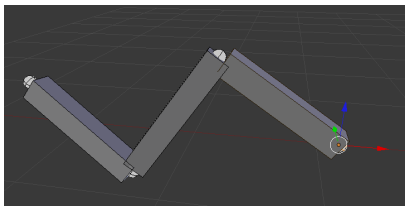
Display As
: This controls the way the bones appear in the 3D Viewport.



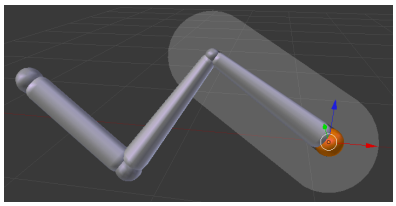
Octahedral bone display.



Stick bone display.



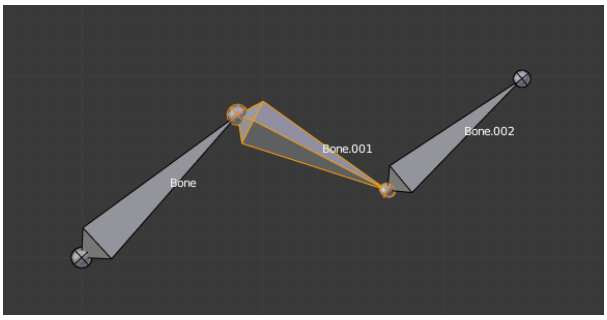
B-Bone bone display.



Envelope bone display.

Octahedral
: This is the default visualization, well suited for most of editing tasks. It materializes:

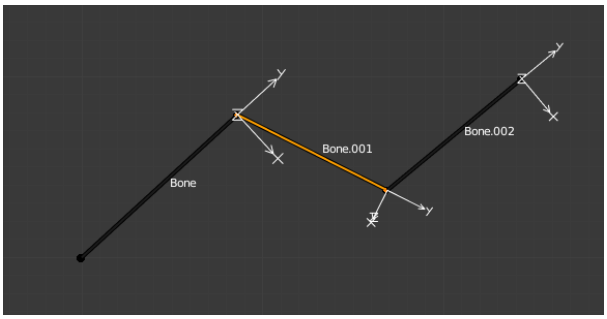
- The bone root (“big” joint) and tip (“small” joint).
- The bone “size” (its thickness is proportional to its length).
- The bone roll (as it has a square section).



Note the 40° rolled Bone.001 bone.

Stick

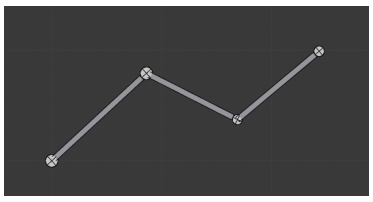
This is the simplest and most non-intrusive visualization. It just materializes bones by sticks of constant (and small) thickness, so it gives you no information about root and tip, nor bone size or roll angle.



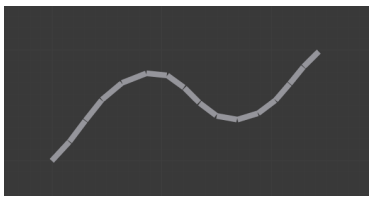
Note that Bone.001 roll angle is not visible (except by its XZ axes).

B-Bone

This visualization shows the curves of “smooth” multi-segmented bones; see the Bendy Bones for details.

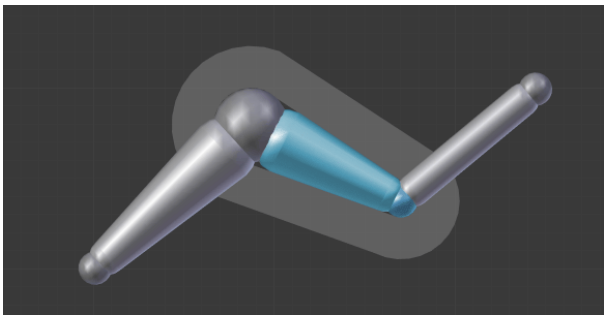


An armature of B-Bones, in Edit Mode.

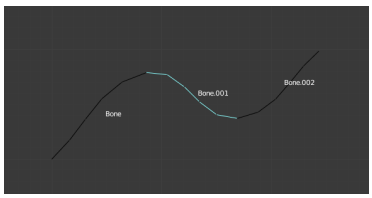


The same armature in Object Mode.

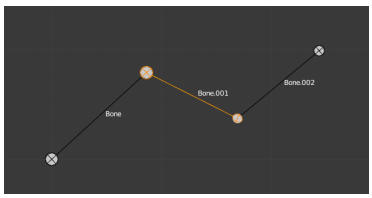
Envelope
: This visualization materializes the bone deformation influence. More on this in the bone page.



Wire
: This simplest visualization shows the curves of “smooth” multi-segmented bones.



An armature of Wire, in Pose Mode.



The same armature in Edit Mode.

Show

Names
: Displays the name of each bone.

Shapes
: When enabled, the default standard bone shape is replaced, in *Object Mode* and *Pose Mode*, by the shape of a chosen object (see Shaped Bones for details).

Bone Colors
: Draws bones in their configured colors. Disable to always draw bones in the default color. For more details see Bone Colors.

In Front

When enabled, the bones of the armature will always be shown on top of the solid objects (meshes, surfaces, …). I.e. they will always be visible and selectable (this is the same option as the one found in the *Display* panel of the *Object data* tab). Very useful when not in *Wireframe* mode.

Axis

When enabled, the (local) axes of each bone are displayed (only relevant for *Edit Mode* and *Pose Mode*).

Position

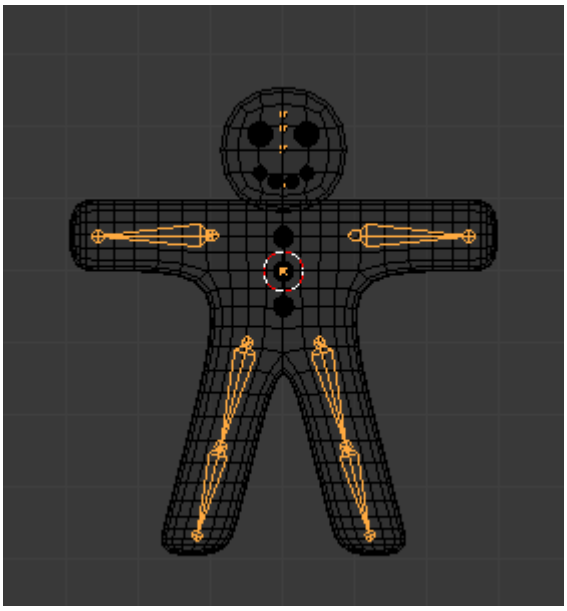
The position for the axes display on the bone. Increasing the value moves it closer to the tip; decreasing moves it closer to the root.

Relations

Whether the Relationship Lines overlay should be drawn from each parent's tail or head. The lines are always drawn towards the childrens' heads.

Structure

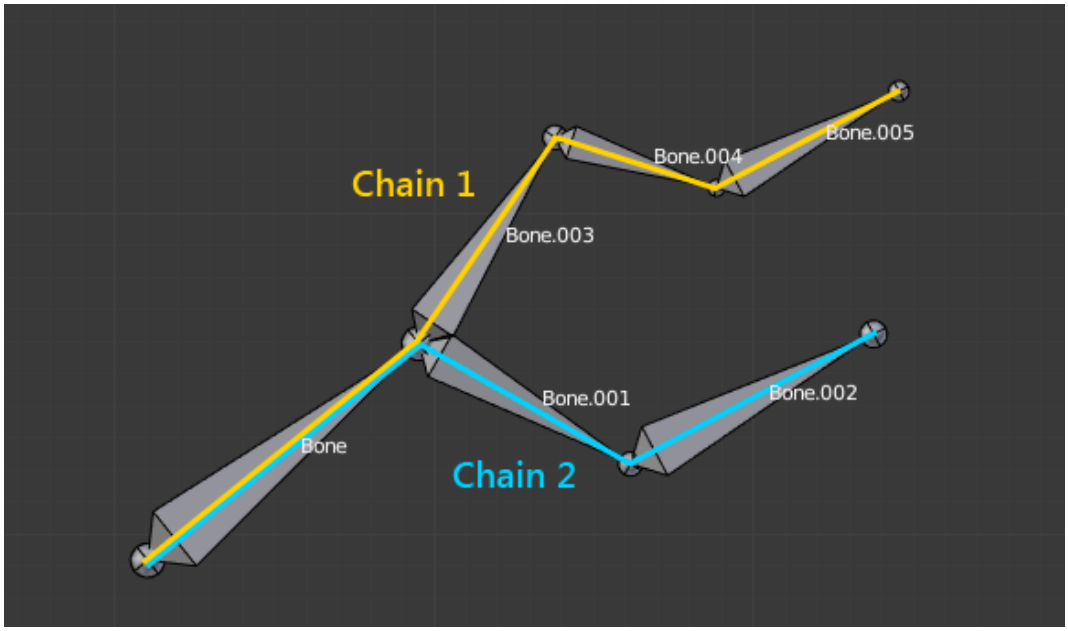
Armatures mimic real skeletons. They are made out of bones, which are (by default) rigid elements. But you have more possibilities than with real skeletons: In addition to the “natural” rotation of bones, you can also move and even scale them! And your bones do not have to be connected to each other; they can be completely free if you want. However, the most natural and useful setups imply that some bones are related to others, forming so-called “chains of bones”, which create some sort of “limbs” in your armature, as detailed in Chains of Bones.



Example of a very basic armature.

Chains of Bones

The bones inside an armature can be completely independent from each other (i.e. the modification of one bone does not affect the others). But this is not often a useful set up: To create a leg, all bones “after” the thigh bone should move “with” it in a well-coordinated manner. This is exactly what happens in armatures by parenting a bone to the next one in the limb, you create a “chains of bones”. These chains can be ramified. For example, five fingers attached to a single “hand” bone.



An armature with two chains of bones.

Bones are chained by linking the tip of the parent to the root of the child. Root and tip can be *connected*, i.e. they are always exactly at the same point; or they can be *free*, like in a standard parent-child object relationship.

A given bone can be the parent of several children, and hence be part of several chains at the same time.

The bone at the beginning of a chain is called its *root bone*, and the last bone of a chain is the *tip bone* (do not confuse them with similar names of bones' joints!).

Chains of bones are a particularly important topic in posing (especially with the standard *forward* kinematics versus "automatic" *inverse* kinematics posing techniques). You create/edit them in *Edit Mode*, but except in case of connected bones, their relationships have no effect on bone transformations in this mode (i.e. transforming a parent bone will not affect its children).

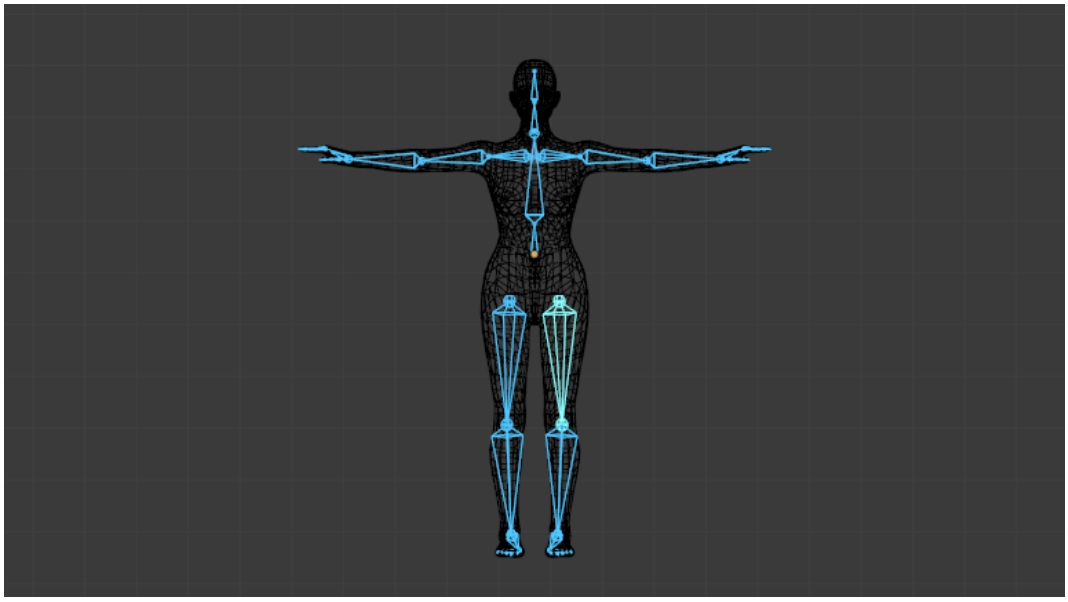
The easiest way to manage bones relationships is to use the Relations panel in the Bone tab.

Skinning

- Introduction
- Armature Deform Parent
  - With Empty Groups
  - With Automatic Weights
  - With Envelope Weights

Introduction

We have seen in previous pages how to design an armature, create chains of bones, etc. Now, having a good rig is not the final goal, unless you want to produce a "Dance Macabre" animation, you will likely want to put some flesh on your skeletons! Surprisingly, "linking" an armature to the object(s) it should transform and/or deform is called the "skinning" process…



The human mesh skinned on its armature.

In Blender, you have two main skinning types:

1. You can Parent/Constrain Objects to Bones – then, when you transform the bones in *Pose Mode*, their "children" objects are also transformed, exactly as with a standard parent/children relationship… The "children" are **never** deformed when using this method.
2. You can Use the Armature Modifier on entire Mesh, and then, some parts of this object to some bones inside this armature. This is the more complex and powerful method, and the only way to really deform the geometry of the object, i.e. to modify its vertices/control points relative positions.

Hint

Retargeting

Retargeting, which is a way to apply motion-capture data (acquired from real world) to a rig, is available through add-ons and importers.

Armature Deform Parent

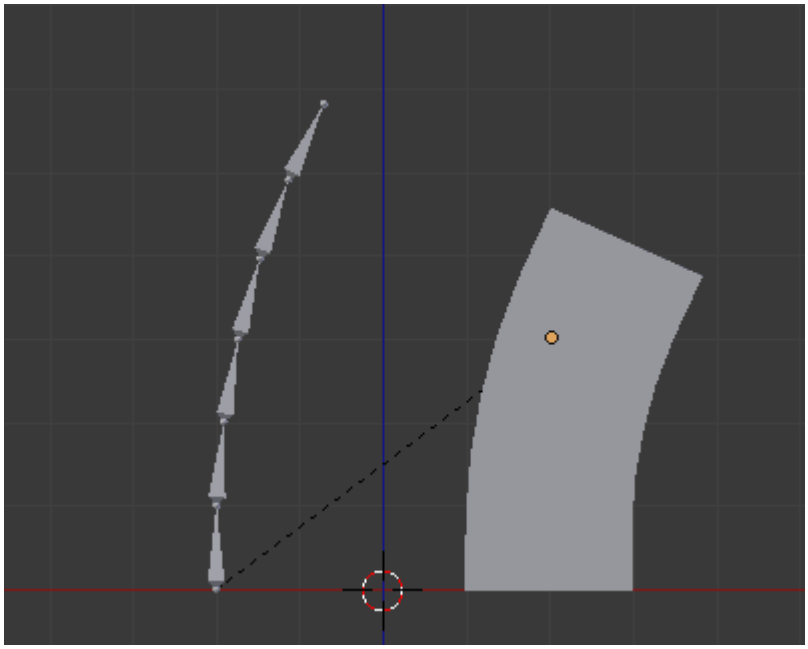
Reference

Mode::	Object Mode and Pose Mode
Menu::	*Object/Pose* ▸ *Parent* ▸ *Armature Deform*
Shortcut::	`Ctrl-P`

Armature Deform Parenting is a way of creating and setting up an Armature Modifier.

To use *Armature Deform Parenting* you must first select all the child objects that will be influenced by the armature and then lastly, select the armature object itself. Once all the child objects and the armature are selected, press `Ctrl-P` and select *Armature Deform* in the *Set Parent To* pop-up menu.

The armature will be the parent object of all the other child objects and each child object will have an Armature Modifier with the armature associated (*Object* field).



Bone associated with Mesh Object.

With Empty Groups

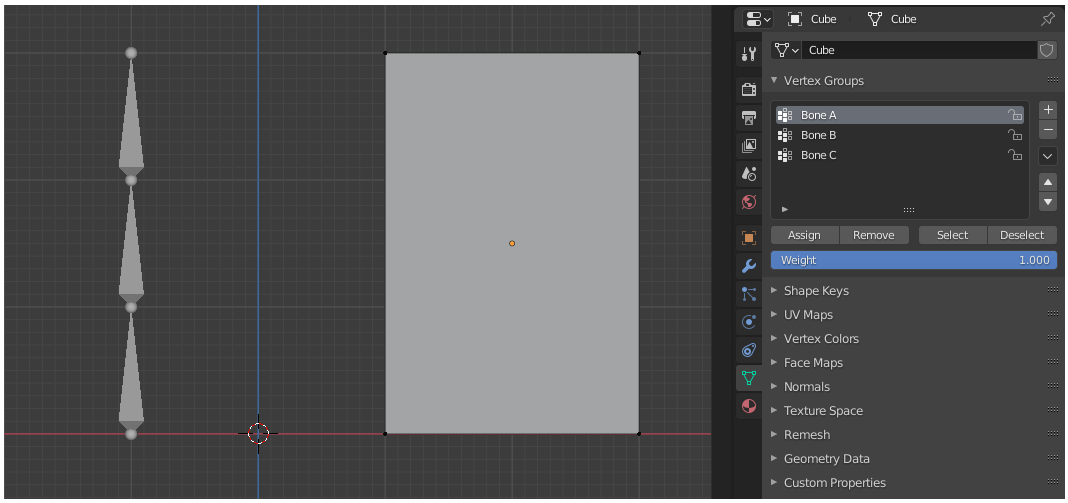
When parenting it will create empty vertex groups on the child objects (if they do not already exist) for and named after each deforming bone in the armature. The newly created vertex groups will be empty. This means they will not have any weights assigned. Vertex groups will only be created for bones which are setup as deforming (*Properties ▸ Bone ▸ Deform Panel*).

You can then manually select the vertices and assign them to a particular vertex group of your choosing to have bones in the armature influence them.

Choose this option if you have already created (and weighted) all the vertex groups the mesh requires.

Example

For example, if you have an armature which consists of three bones named "BoneA", "BoneB" and "BoneC" and cube mesh called "Cube". If you parent the cube to the armature, the cube will get three new vertex groups created on it called "BoneA", "BoneB" and "BoneC". Notice that each vertex group is empty.



Cube in Edit Mode using Armature Deform with empty groups.

With Automatic Weights

With Automatic Weights parenting works similar to *With Empty Groups*, but it will not leave the vertex groups empty. It calculates how much influence a particular bone would have on vertices based on the distance from those vertices to a particular bone ("bone heat" algorithm). This influence will be assigned as weights in the vertex groups.

This method of parenting is certainly easier to setup, but it can often lead to armatures which do not deform child objects in ways you would want. Overlaps can occur when it comes to determining which bones should influence certain vertices when calculating influences for more complex armatures and child objects. Symptoms of this confusion are that when transforming the armature in *Pose Mode*, parts of the child objects do not deform as you expect; If Blender does not give you the results you require, you will have to manually alter the weights of vertices in relation to the vertex groups they belong to and have influence in.

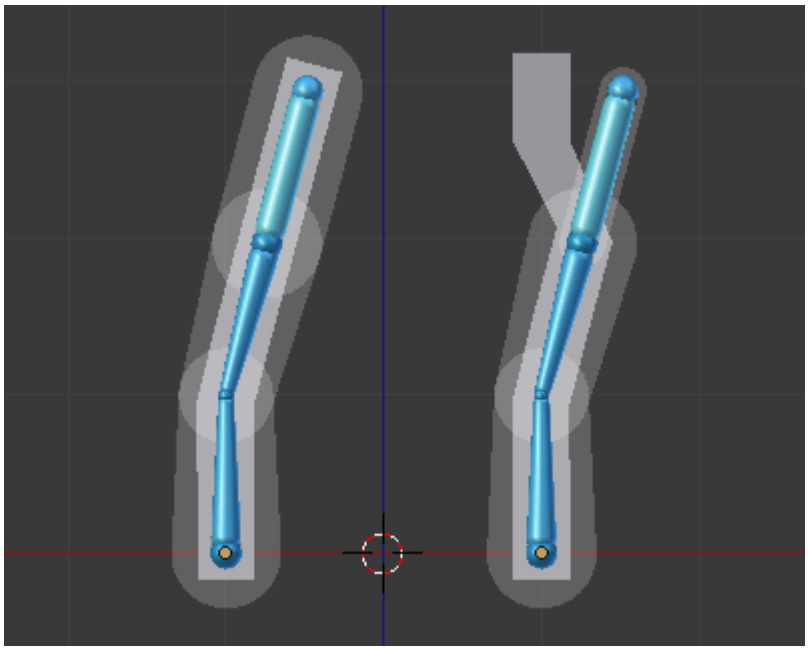
With Envelope Weights

Works in a similar way to *With Automatic Weights*. The difference is that the influences are calculated based on the Bone Envelopes settings. It will assign a weight to each vertex group the vertices that is inside its bone's influence volume, depending on their distance to this bone.

This means newly included/excluded vertices or new envelope settings will not be taken into account. You will have to apply Armature Deform With Envelope Weights parenting again.

Tip

If you want the envelope setting to be used instantly, bind the Armature Modifier to *Bone Envelopes*.



Two sets of armatures, each with three bones.

Warning

If you had defined vertex groups using same names as skinned bones, their content will be completely overridden by both *Automatic* and *Envelope Weights*. In this case *With Empty Groups* could be used instead.

See also

Vertex Groups for Bones.

Posing

- Introduction
  - Visualization
- Selecting
  - All
  - None
  - Invert
  - Box Select
  - Circle Select
  - Lasso Select
  - Select Mirror
  - Select More/Less
  - Select Grouped
  - Select Linked
  - Select Pattern
  - Constraint Target
- Editing
  - Introduction
  - Clear Transform
  - Apply
  - In-Betweens
  - Propagate
  - Copy/Paste Pose
  - Pose Library
  - Flip Quats
  - Show/Hide
- Tool Settings
  - Pose Options
  - Known Limitations
- Bone Constraints
  - Introduction
  - Inverse Kinematics

Introduction

Once an armature is skinned by the needed object(s), you need a way to configure the armature into positions known as poses. Basically, by transforming the bones, you deform or transform the skinned object(s). However, you will notice that you cannot do this in *Edit Mode* – remember that *Edit Mode* is used to edit the default, base, or "rest" position of an armature. You may also notice that you cannot use *Object Mode* either, as here you can only transform whole objects.

So, armatures have a third mode dedicated to the process of posing known as *Pose Mode*. In rest position (as edited in *Edit Mode*), each bone has its own position/rotation/scale to neutral values (i.e. 0.0 for position and rotation, and 1.0 for scale). Hence, when you edit a bone in *Pose Mode*, you create an offset in the transform properties, from its rest position. This may seem quite similar if you have worked with relative shape keys or Delta Transformations.

Even though it might be used for completely static purposes, posing is heavily connected with animation features and techniques. So if you are not familiar at all with animation in Blender, it might be a good idea to read the animation chapter first, and then come back here.

Visualization

Bone State Colors

The color of the bones are based on their state. There are six different color codes, ordered here by precedence (i.e. the bone will be of the color of the bottom-most valid state):

- Gray: Default.
- Blue wire-frame: in Pose Mode.
- Green: with Constraint.

- Yellow: with IK Solver constraint.
- Orange: with Targetless Solver constraint.

Note

When bone colors are enabled, the state colors will be overridden.

Selecting

Selection in *Pose Mode* is very similar to the one in Edit Mode, with a few deviations: You can only select *whole bones* in *Pose Mode*, not roots/tips…

All

Reference

Mode::	Pose Mode
Menu::	*Select ▸ All*
Shortcut::	`A`

Select all selectable bones.

None

Reference

Mode::	Pose Mode
Menu::	*Select ▸ None*
Shortcut::	`Alt-A`

Deselect all bones, but the active bone stays the same.

Invert

Reference

Mode::	Pose Mode
Menu::	*Select ▸ Invert*
Shortcut::	`Ctrl-I`

Toggle the selection state of all visible bones.

Box Select

Reference

Mode::	Pose Mode
Menu::	*Select* ▸ *Box Select*
Shortcut::	`B`

Interactive box selection.

Circle Select

Reference

Mode::	Pose Mode
Menu::	*Select* ▸ *Circle Select*
Shortcut::	`C`

Interactive circle selection.

Lasso Select

Reference

Mode::	Pose Mode
Menu::	*Select* ▸ *Lasso Select*
Shortcut::	`Ctrl-Alt-LMB`

See Select Lasso.

Select Mirror

Reference

Mode::	Pose Mode
Menu::	*Select ▸ Select Mirror*
Shortcut::	`Shift-Ctrl-M`

Flip the selection from one side to another.

Select More/Less

Reference

Mode::	Pose Mode
Menu::	*Select ▸ More/Less*

Parent `[`, Child `]`
: You can deselect the active bone and select its immediate parent or one of its children.

Extend Parent `Shift-[`, Extend Child `Shift-]`
: Similar to *Parent*/*Child* but it keeps the active bone in the selection.

Select Grouped

Reference

Mode::	Pose Mode
Menu::	*Select ▸ Select Grouped*
Shortcut::	`Shift-G`

You can select bones, based on various properties, through the *Select Grouped* pop-up menu `Shift-G`:

Collection

Selects all bones that are share at least one bone collection with the active bone.

Color
: Selects all bones that have the same color as the active bone.

Keying Set
: All bones affected by active Keying Set

Select Linked

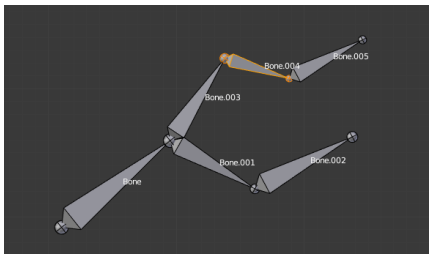
Reference

Mode::	Pose Mode
Menu::	*Select ▸ Select Linked*
Shortcut::	`Ctrl-L`

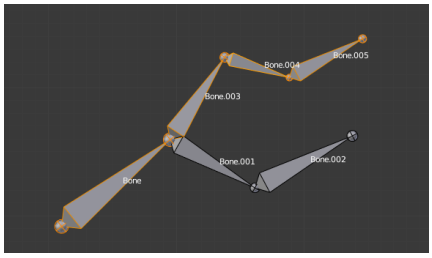
Selects all the bones in the chain which the active (last selected) bone belongs to.

All Forks
: Selects all bones connected to the active bone even if the branch off from the current bone.

Linked bones selection.



A single selected bone.



Its whole chain selected with Linked.

Select Pattern

Reference

Mode::	Pose Mode
Menu::	*Select ▸ Select Pattern…*

Selects all bones whose name matches a given pattern. Supported wild-cards: * matches everything, ? matches any single character, [abc] matches characters in “abc”, and [!abc] match any character not in “abc”. As an example *house* matches any name that contains “house”, while floor* matches any name starting with “floor”.

Case Sensitive
: The matching can be chosen to be case sensitive or not.

Extend
: When *Extend* checkbox is checked the selection is extended instead of generating a new one.

Constraint Target

Reference

Mode::	Pose Mode
Menu::	*Select ▸ Constraint Target*

Select bones used as targets for the currently selected bones

Editing

- Introduction
  - Basic Posing
- Clear Transform
- Apply
- In-Betweens
  - Push Pose from Rest Pose
  - Relax Pose to Rest Pose
  - Push Pose from Breakdown
  - Relax Pose to Breakdown
  - Pose Breakdowner
  - Blend to Neighbor
- Propagate
- Copy/Paste Pose
- Pose Library
  - What is a Pose Asset?
  - Creating a Pose Library
  - Using the Pose Library
  - Old Pose Library
- Flip Quats
- Show/Hide

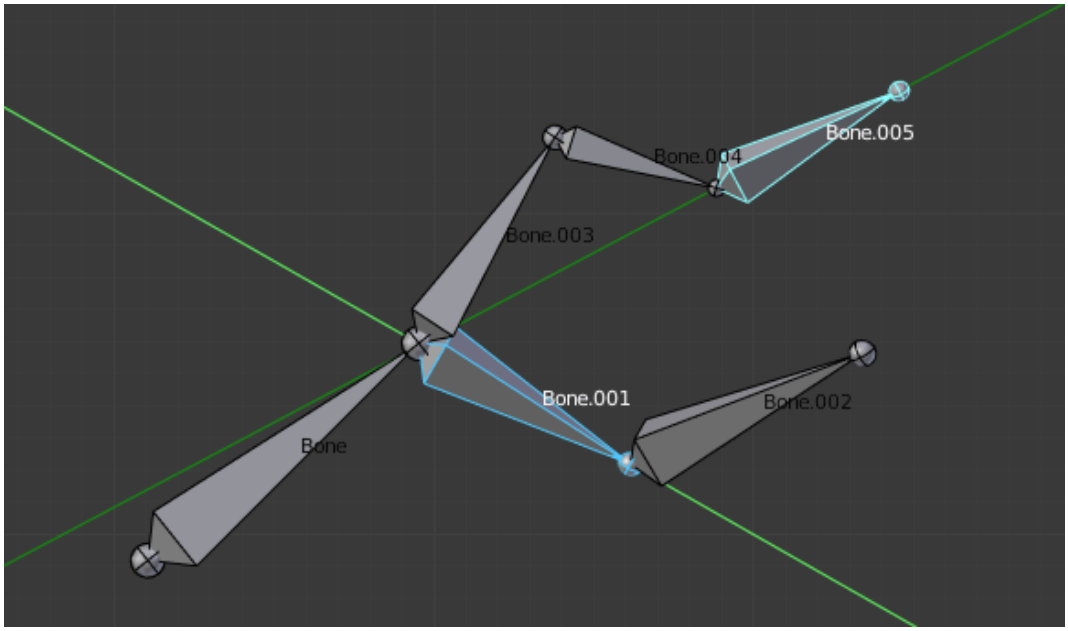
Introduction

In *Pose Mode*, bones behave like objects. So the transform actions (move, rotate, scale, etc.) are very similar to the same ones in Object Mode (all available ones are regrouped in the *Pose* ▸ *Transform* submenu). However, there are some important specifics:

- Bones' relationships are crucial (see Bone Parenting).
- The "transform center" of a given bone (i.e. its default pivot point, when it is the only selected one) is *its root*. Note by the way that some pivot point options seem to not work properly. In fact, except for the *3D Cursor* one, all others appear to always use the median point of the selection (and not e.g. the active bone's root when *Active Object* is selected, etc.).

Basic Posing

As previously noted, bones' transformations are performed based on the *Rest Position* of the armature, which is its state as defined in *Edit Mode*. This means that in rest position, in *Pose Mode*, each bone has a scale of 1.0, and null rotation and position (as you can see it in the *Transform* panel, in the 3D Viewport's Sidebar).



An example of a rotation locked to the local Y axis, with two bones selected.

Note that the two green lines materializing the axes are centered on the armature's center, and not each bone's root…

Moreover, the local space for these actions is the bone's own one (visible when you enable the *Axes* option of the *Armature* panel). This is especially important when using axis locking, for example, there is no specific "bone roll" tool in *Pose Mode*, as you can rotate around the bone's main axis just by locking on the local Y axis `R Y Y`… This also works with several bones selected; each one is locked to its own local axis!

When you pose your armature, you are supposed to have one or more objects skinned on it! And obviously, when you transform a bone in *Pose Mode*, its related objects or object's shape is moved/deformed accordingly, in real-time. Unfortunately, if you have a complex rig set-up and/or a heavy skin object, this might produce lag during interactive editing. If you experience such troubles, try enabling the *Delay Deform* button in the *Armature* panel the skin objects will only be updated once you confirm the transform operation.

Clear Transform

Reference

Mode::	Pose Mode
Menu::	*Pose ▸ Clear Transform*

Once you have transformed some bones, if you want to return to their rest position, just clear their transformations.

All
: Resets location, rotation, and scaling of selected bones to their default values.

Location, Rotation, Scale `Alt-G`, `Alt-R`, `Alt-S`
: Clears individual transforms.

  Note that in *Envelope* visualization, `Alt-S` does not clear the scale, but rather scales the *Distance* influence area of the selected bones. (This is also available through the *Pose ▸ Scale Envelope Distance* menu entry, which is only effective in *Envelope* visualization, even though it is always available…)

Reset Unkeyed
: Clears the transforms to their keyframe state.

  Only Selected
  : Operate on just the selected or all bones.

Apply

Reference

Mode::	Pose Mode
Menu::	*Pose* ▸ *Apply*
Shortcut::	`Ctrl-A`

Pose as Rest Pose
: Conversely, you may define the current pose as the new rest pose (i.e. "apply" current transformations to the *Edit Mode*). When you do so, the skinned objects/geometry is **also** reset to its default, undeformed state, which generally means you will have to skin it again.

Pose Selected as Rest Pose
: Same as *Pose as Rest Pose* but only applies to selected bones.

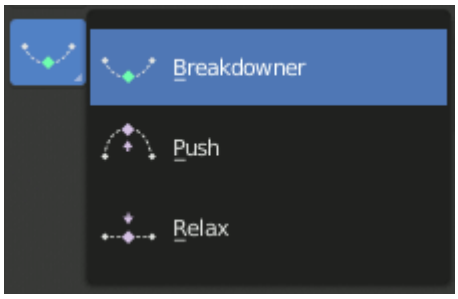
Visual Transform to Pose
: Applies the position of the bone after Constraints; allowing the constraints to be deleted and the bones will remain in their constrained positions.

Assign Custom Property Values as Default
: Assign the current values of custom properties as their defaults, for use as part of the rest pose state in NLA track mixing.

In-Betweens

There are several tools for editing poses in an animation.

There are also in *Pose Mode* a bunch of armature-specific editing options/tools, like auto-bones naming, properties switching/enabling/disabling, etc., that were already described in the armature editing pages. See the links above…



In-Betweens Tools.

Push Pose from Rest Pose

Reference

Mode:: Pose Mode
Menu:: *Pose ▸ In-Betweens ▸ Push Pose from Rest Pose*

Similar to *Push Pose from Breakdown* but interpolates the pose to the rest position instead. Only one keyframe is needed for this tool unlike two for the other.

Relax Pose to Rest Pose

Reference

Mode:: Pose Mode
Menu:: *Pose ▸ In-Betweens ▸ Relax Pose to Rest Pose*

Similar to *Relax Pose to Breakdown* but works to bring the pose back to the rest position instead. Only one keyframe is needed for this tool unlike two for the other.

Push Pose from Breakdown

Reference

Mode::	Pose Mode
Tool::	*Toolbar ▸ In-Betweens Tools ▸ Push*
Menu::	*Pose ▸ In-Betweens ▸ Push Pose from Breakdown*
Shortcut::	`Ctrl-E`

Push Pose interpolates the current pose by making it closer to the next keyframed position.

Relax Pose to Breakdown

Reference

Mode::	Pose Mode
Tool::	*Toolbar ▸ In-Betweens Tools ▸ Relax*
Menu::	*Pose ▸ In-Betweens ▸ Relax Pose to Breakdown*
Shortcut::	`Alt-E`

Relax pose is somewhat related to the above topic, but it is only useful with keyframed bones. When you edit such a bone (and hence take it "away" from its "keyed position"), using this tool will progressively "bring it back" to its "keyed position", with smaller and smaller steps as it comes near it.

Pose Breakdowner

Reference

Mode::	Pose Mode
Tool::	*Toolbar region ▸ In-Betweens Tools ▸ Breakdowner*
Menu::	*Pose ▸ In-Betweens ▸ Pose Breakdowner*
Shortcut::	`LMB`-drag

Creates a suitable breakdown pose on the current frame.

The Breakdowner tool can be constrained to work on specific transforms and axes, by pressing the following keys while the tool is active:

- `G`, `R`, `S`: move, rotate, scale
- `B`: Bendy bones
- `C`: custom properties
- `X`, `Y`, `Z`: to the corresponding axes

Blend to Neighbor

Reference

Mode::	Pose Mode
Menu::	*Pose ▸ In-Betweens ▸ Blend to Neighbor*
Shortcut::	`Shift-Alt-E`

Transitions the current pose with the neighboring keyframes in the timeline. In order for this operator to work, there must be a keyframe before and after the current frame.

Propagate

Reference

Mode::	Pose Mode
Menu::	*Pose ▸ Propagate*
Shortcut::	`Alt-P`

The Propagate tool copies the pose of the selected bones on the current frame over to the keyframes delimited by the *Termination Mode*. It automates the process of copying and pasting.

Termination Mode
: Modes which determine how it decides when to stop overwriting keyframes.

  To Next Keyframe
  : Simply copies the pose to the first keyframe after (but not including any keyframe on) the current frame.

  To Last Keyframe
  : Will simply replace the last keyframe (i.e. making action cyclic).

  Before Frame
  : To all keyframes between current frame and the *End frame* option. This option is best suited for use from scripts due to the difficulties in setting this frame value, though it is possible to set this manually via the Adjust Last Operation panel if necessary.

  Before Last Keyframe
  : To all keyframes from current frame until no more are found.

  On Selected Keyframes
  : Will apply the pose of the selected bones to all selected keyframes.

On Selected Markers

To all keyframes occurring on frames with Scene Markers after the current frame.

End Frame

Defines the upper-bound for the frame range within which keyframes will be affected (with the lower bound being the current frame).

Copy/Paste Pose

Reference

Mode::	Pose Mode
Menu::	*Pose* ▸ *Copy Pose*, *Pose* ▸ *Paste Pose*, *Pose* ▸ *Paste Pose Flipped*
Shortcut::	`Ctrl-C`, `Ctrl-V`, `Shift-Ctrl-V`

Blender allows you to copy and paste a pose, either through the *Pose* menu, or by using hotkeys.

Copy Pose
: Copy the current pose of selected bones into the pose buffer.

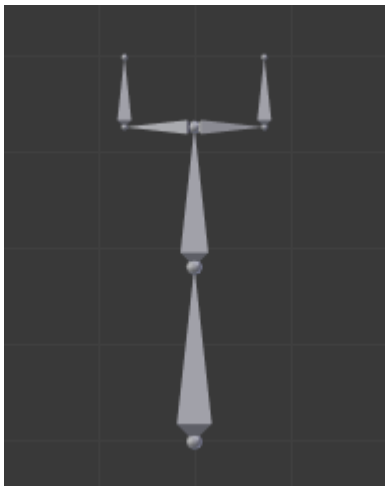
Paste Pose
: Paste the buffered pose to the currently posed armature.

Paste Pose Flipped
: Paste the *X axis mirrored* buffered pose to the currently posed armature.

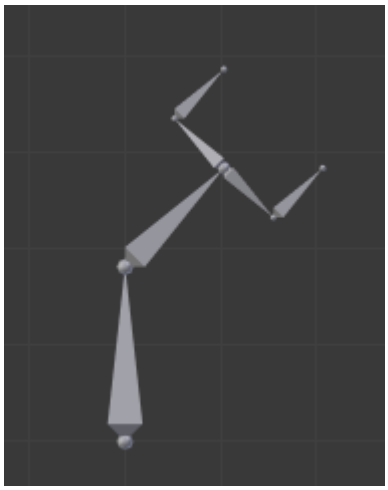
Here are important points:

- This tool works at the Blender session level, which means you can use it across armatures, scenes, and even files. However, the pose buffer is not saved, so you lose it when you close Blender.
- There is only one pose buffer.
- Only the selected bones are taken into account during copying (i.e. you copy only selected bones' pose).
- During pasting, on the other hand, bone selection has no importance. The copied pose is applied on a per-name basis (i.e. if you had a `forearm` bone selected when you copied the pose, the `forearm` bone of the current posed armature will get its pose when you paste it – and if there is no such named bone, nothing will happen…).

- What is copied and pasted is in fact the position, rotation or scale of each bone, in its own space. This means that the resulting pasted pose might be very different from the originally copied one, depending on:
    - The rest position of the bones.
    - And the current pose of their parents.

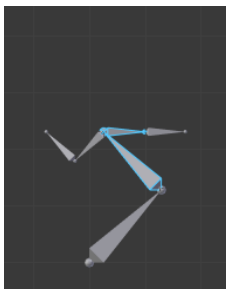


The rest position of the original armature.

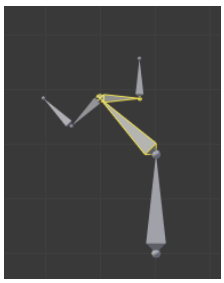


The rest position of the destination armature.

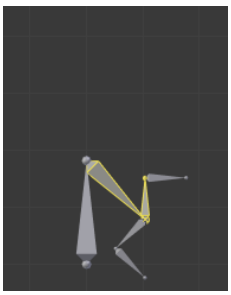
Examples of pose copy/paste.



The first copied pose (note that only two bones are selected and

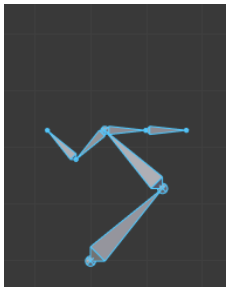


The pose pasted on the destination armature.



The pose mirror-pasted on the destination armature.

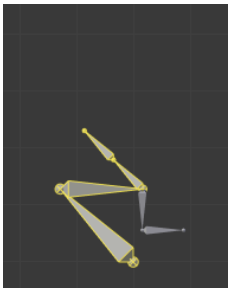
hence
copied).



The same pose as above is copied, but this time with all bones selected.



The pose pasted on the destination armature.



The pose mirror-pasted on the destination armature.

Pose Library

This section describes the pose library, which is based on the Asset Browser. For an overview of the asset system, see the Asset Libraries section. The pose library is meant to be used in Pose Mode. In other words, it only works when posing an armature, and not for general object animation.

Note

The pose library is implemented as an add-on. This add-on is enabled by default; disabling it will remove the pose library from Blender's user interface.

The "building blocks" of the pose library are actually implemented in Blender itself. The add-on only contains the user interface and the logic that determines what is stored in a pose asset. This was intentionally put into an add-on, so that artists or studios who want to change the behavior can do so with an add-on of their own.

What is a Pose Asset?

A *pose asset* is an action that has been marked as asset, and that contains exactly **one frame of animation** data. Usually these are created via the *Create Pose Asset* button (see below), but any action that is keyed on exactly one frame can be seen as pose asset.

Each pose in the library is stored in its own action data-block. This means that it can get its own name, its own preview image, and can be organized in Asset Catalogs.

Creating a Pose Library

A *pose library file* is typically a blend-file that is dedicated to poses. It can link in a character, props, etc., which can then not only be used to create the poses, but also for rendering previews.



Example pose library of the Sprite Fright character Ellie.

Pose Creation via Action Editor

To create a pose in the library from the Action Editor, **pose the character, select the relevant bones**, and click the *Create Pose Asset* button. This will create the new pose Action, which will contain keys for the current value of each bone's location, rotation, scale, and Bendy Bone properties.



To create a new pose asset, use the Create Pose Asset button in the Action editor.

The created Action is now assigned to the armature. This makes it possible to inspect which bones are included and to tweak anything. In that respect, it's an Action like any other, and you can add or remove keys as usual. Just make sure that the keys are all on the same frame, in order to keep this a "pose" instead of an "animation snippet"; the latter isn't supported at the moment.

True to its name, the *Create Pose Asset* button automatically marks the Action as Asset. Not only does this make it available in the pose library, it will also act as a *fake user* to ensure the Action isn't lost after you unassign it from the armature.

The pose asset can be renamed in the Asset Browser. There you can also right click on the thumbnail, then choose Assign Action to assign the Action to the active Object (see description above).

Note

The **Create Pose Asset** button creates a new Action. To make sure that this is actually visible in the user interface, so that you know that something happened, it tries to make sure that the Asset Browser shows the newly created pose asset. Because of this, it **requires that there is an Asset Browser visible, and that it's set to show the Current File asset library**.

This is especially relevant to pose assets, compared to other assets. You cannot mark an object as asset multiple times, but you can create ten pose assets from the same character pose.

Pose Creation from Existing Animation

Animators eat and breathe time, so there is a fair chance that you already have some poses lined up on the timeline. Creating a pose asset from

existing animation is pretty much the same as described above, with a few subtle differences:

- Go to the frame with the pose you want to turn into an asset.
- Select the relevant bones and click the **Create Pose Asset button** in the Action Editor.
- This creates an Action as before, but this time **it also includes any bone property that was keyed on the current frame**. In other words: any bone property (regular and custom) that's displayed in yellow in the user interface will be included in the pose asset. This makes it possible to also include properties that control IK/FK switching, for example. As with the pose, the current value is copied into the pose asset, and not the keyed value.
- Blender saves which action was previously assigned to the armature.
- The new pose Action is assigned to the armature so you can give it a name and inspect/adjust its contents.
- Click the *Restore Previous Action* button (back arrow icon) that appeared next to the Create Pose Asset button. This **reassigns the previous Action**, so that you're back at the animation you had before.

Pose Creation by Copying from Other File

As described in Design Limitations, Blender only writes data to the currently open blend-file. To copy a pose from some other file into a pose library file, see the following steps:

- Pose the character and select the relevant bones.
- Click the **Copy Pose as Asset button**, which is available in the Action Editor. This will create the pose asset (including its thumbnail) and store it in a temporary file somewhere.
- Choose an existing pose asset, and open its context menu. Click the **Open Blend File** option.
- A new Blender process will start, and automatically open the asset library file that contains the chosen pose. By the way, this works for all assets, not just poses!
- In the Asset Browser, click the **Paste as New Asset button**. This will load that temporary file, and load all the assets it can find in there. In

our case, it will only find a single pose, but future versions of Blender may extend this for other asset types. This is why the button is named so generically – it is not pose-specific.
- Give the pose a name, and click on the "refresh" button in the preview image panel to render a new preview if you want.
- **Save the file and quit Blender**.
- The original Blender is still running in the background and notices that the new Blender has quit. It **automatically refreshes the Asset Browser** to show the newly added pose.

Automatically Assigned Catalog

When you create a pose asset, Blender may automatically assign it to an asset catalog. This only works if there is an Asset Browser visible; Blender then assigns the pose asset to its active asset catalog. If there are multiple Asset Browsers open, it performs the following steps:

- If the current window has one Asset Browser, it uses that one.
- If the current window has multiple Asset Browsers, it uses the biggest one.
- Otherwise Blender goes over the other windows (if there are any), and do a similar search. The first window it sees that has an Asset Browser wins.

Controlling the Look of Preview Images

The pose library preview images are rendered with the active Scene camera. This approach was preferred over rendering a specific 3D Viewport for two main reasons:

- There is only one scene camera active at any time, making it predictable which camera is used.
- The camera, as well as the rest of the scene, can be set up specifically for rendering the thumbnails. Pose library files are intended for that purpose: to contain the poses and render their preview images.

The preview images are rendered using the Workbench Engine. Switch the scene to use that as render engine, and you'll see various options to influence the look. Select a pose asset and press the Generate Preview button to re-render the preview image with the current settings.

You can also animate settings such as MatCap rendering, light positions, and intensities, etc. Use this to your advantage!

Scene Animation for Preview Images

Sometimes it's handy to have a few different background colors or camera angles for your poses. Many facial poses are made with a specific camera angle in mind.

- Background color can be animated by placing a plane behind the character and animating its material. In this case just for fun, but for more serious applications this could be used to indicate a certain character, or a mood, or anything else.
- The active camera can be switched by using camera markers.

Both make it possible to choose a specific frame to pick the background color and camera angle. Pose the character, click the *Create Pose Asset* button, and the pose action will be keyed on the current frame. This means it's easy to edit the pose and refresh its preview image, because you know exactly which frame it was originally created on.

Using the Pose Library

The pose library can be used to pose a character in a few different ways. In short, you can fully apply a pose or blend it into the character's current pose interactively. How exactly these operations work depends on where you use them. This section will explain the use from both the Asset Browser and the 3D Viewport.

Use from the Asset Browser

The pose library can be used directly from the Asset Browser. The **Pose Library panels will appear** when the active object is an armature and in Pose Mode. The catalog system and the filter bar at the top can be used to search for specific poses.

The following operators can be accessed by `RMB` on a pose:

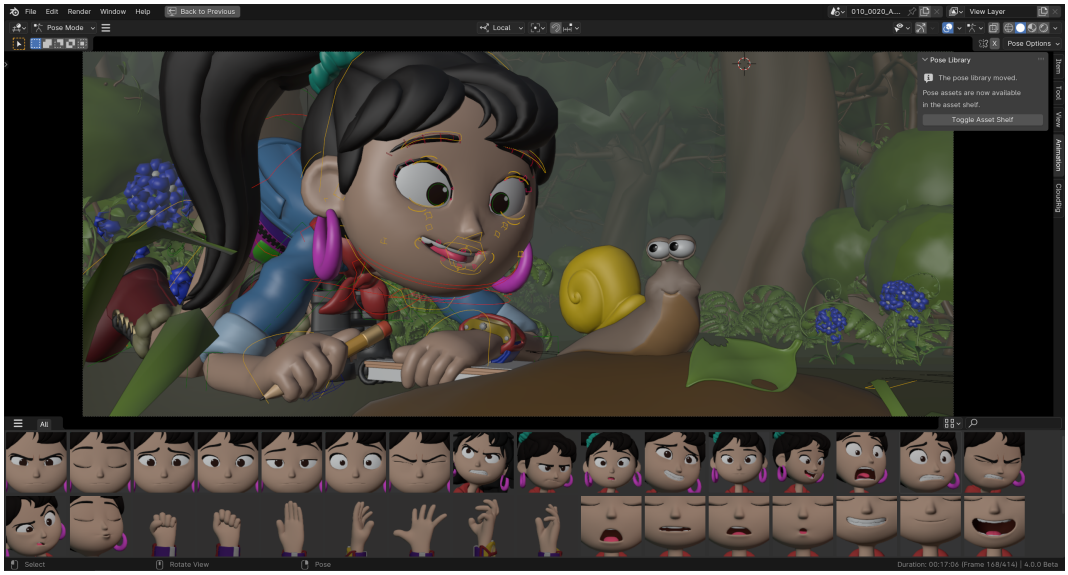
Apply Pose
: Applies the pose to the character. If there are any bones selected, the pose will be applied only to those bones. This makes it possible to create a "finger guns" pose by applying a fist pose to the hand, and then an "open hand" pose for only the index finger and thumb. Double-clicking a pose will also apply it.

Apply Pose Flipped
: Will mirror the pose from left to right and vice versa. This makes it possible, for example, to apply a left-hand pose to the right hand, reducing the number of poses you have to put into the library. This can of course also be applied for asymmetrical facial expressions that depend on the camera angle. While blending (see below), keep `Ctrl` pressed to blend the flipped pose.

Blend Pose
: Allows you to gradually blend a pose from the library into the character's pose. Click the button, then move the mouse left/right to determine the desired blend. A pose asset can be "subtracted" while blending. Drag to the right to blend as usual, drag to the left to subtract the pose. While blending, you can use `Tab` to toggle between the original and the blended pose. As usual in Blender, `LMB` or press `Return` to confirm; `RMB` or press `Esc` to cancel the operator. Blending can also exaggerate a pose, by pressing `E` (for Extrapolate) and applying a pose for more than 100%.

Select/Deselect Pose Bones
: Select or deselect the bones that are used in the pose. This can be used to create a selection set, or simply show what was part of the pose and what wasn't.

Use from 3D Viewport



The pose library in use from the Asset Shelf.

Note

The pose library previously lived in the Sidebar within the Pose Library panel. The panel still exists, but now contains a button to open the asset shelf.

In the 3D viewport, poses can be quickly applied from the Asset Shelf. Contrary to the Asset Browser, the shelf allows you to apply poses quicker.

Click on a pose to apply it. A single click is enough. You can also select and apply a pose via the cursor keys. This allows for fast exploration of the poses, to directly see the result on the active character.

Drag the pose thumbnail left to right to blend it into the character's current pose. Just release the mouse button to confirm.

Old Pose Library

In Blender 3.0, the Asset Browser based pose library, described above, replaced its predecessor pose library system. This section describes how to convert poses from the old pose library to the current system.

Converting Old Pose Libraries

Old-style pose libraries can be converted to pose assets in the following way:

- In the Action Editor, select the Action containing the pose library you want to convert.
- Make sure the scene camera is set up correctly for rendering preview images.
- In the Action Editor's Pose Library panel, click the "Convert Old-Style Pose Library" button.
- Open the Asset Browser, and see the poses have been converted.
- If you're happy with the result, remove the old pose library Action.
- Save the blend-file.

As usual, the blend-file should be saved to a directory marked as asset library in order to use the pose assets from other blend-files.

Note

This conversion does not assign the poses to any catalog, and so they will appear in the "Unassigned" section of the "Current File" asset library.

Flip Quats

Reference

Mode::	Pose Mode
Menu::	*Pose ▸ Flip Quats*
Shortcut::	`Alt-F`

Flip quaternion values to achieve desired rotations, while maintaining the same orientations.

Show/Hide

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Bone* ▸ *Viewport Display*
Menu::	… ▸ *Show/Hide*

You do not have to use bone layers to show/hide some bones. As with objects, vertices or control points, you can use `H`:

- `H` will hide the selected bone(s).
- `Shift-H` will hide all bones *but the selected one(s)*.
- `Alt-H` will show all hidden bones.

You can also use the *Hide* checkbox of the *Bone tab* ▸ *Viewport Display panel*.

Note that hidden bones are specific to a mode, i.e. you can hide some bones in *Edit Mode*, they will still be visible in *Pose Mode*, and vice versa. Hidden bones in *Pose Mode* are also invisible in *Object Mode*. And in *Edit Mode*, the bone to hide must be fully selected, not just its root or tip.

Tool Settings

Pose Options

Auto IK

Reference

Mode::	Pose Mode
Panel::	*Sidebar ▸ Tool ▸ Pose Options ▸ Auto IK*

Automatic IK is a tool for quick posing, when enabled, translating a bone will activate inverse kinematics and rotate the parent bone, and the parent's parent, and so on, to follow the selected bone. The IK chain can only extend from a child to a parent bone if the child is *connected* to it.

While moving bones, the length of the chain (the number of affected bones) can be increased or decreased using keyboard hotkeys. Pressing `PageUp` will increase in chain length by one and `PageDown` decreases the length by one. The chain length can also be controlled with `WheelUp` or `WheelDown`.

The initial chain length is 0, which effectively means follow the connections to parent bones as far as possible, with no length limit. So pressing increasing the chain length the first time sets the length to 1 (move only the selected bone), and at this point, decreasing the length point sets it back to 0 (unlimited) again. Thus, you have to increase the chain length *more than once* from the initial state to set a finite chain length greater than 1.

This is a more limited feature than using an IK constraint, which can be configured, but it can be useful for quick posing.

X-Axis Mirror

Reference

Mode::	Edit and Pose Mode
Panel::	*Sidebar* ▸ *Tool* ▸ *Options* ▸ *X-Axis Mirror*

This option enables automatic mirroring of editing actions along the X axis. You can enable this option in the *Tool tab* ▸ *Options panel*, while the armature is selected in *Edit Mode*. When you have pairs of bones of the same name with just a different "side suffix" (e.g. ".R"/".L", or "_right"/"_left" …), once this option is enabled, each time you transform (move, rotate, scale…) a bone, its "other side" counterpart will be transformed accordingly, through a symmetry along the armature local X axis. As most rigs have at least one axis of symmetry (animals, humans, …), it is an easy way to keep the model symmetrical.

Relative Mirror

Reference

Mode::	Edit and Pose Mode
Panel::	*Sidebar* ▸ *Tool* ▸ *Options* ▸ *Relative Mirror*

Accounts for any relative transformations when using *X-Axis Mirror*.

See also

Naming bones.

Known Limitations

Relative Mirror is not supported with Auto IK enabled.

Bone Constraints

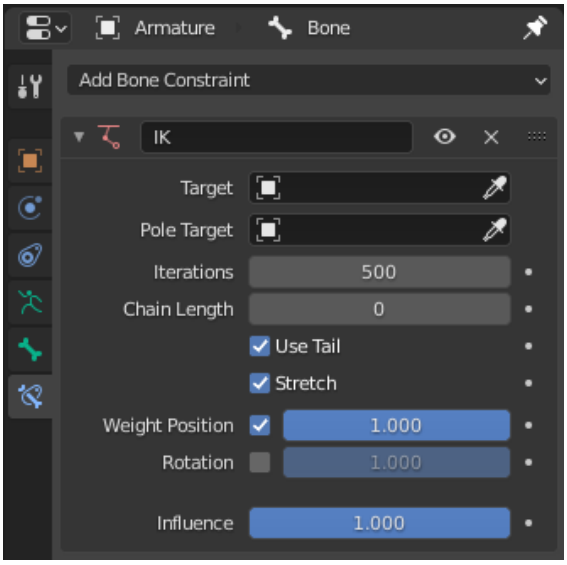
- Introduction

Inverse Kinematics

- Introduction
- Spline IK

Introduction

As bones behave like objects in *Pose Mode*, they can also be constrained. This is why the *Constraints* tab is shown in both *Object Mode* and *Edit Mode*. This panel contains the constraints *of the active bone* (its name is displayed at the top of the panel, in the *To Bone:…* static text field).



The Bone Constraints Properties in Pose Mode, with an Inverse Kinematics constraint added to the active bone.

Constraining bones can be used to control their degree of freedom in their pose transformations, using e.g. the *Limit* constraints. You can also use constraints to make a bone track another object/bone (inside the same object, or in another armature), etc. And the inverse kinematics feature is also mainly available through the *IK Solver* constraint, which is specific to bones.

For example, a human elbow cannot rotate backward (unless the character has broken their arm), nor to the sides, and its forward and roll rotations are limited in a given range. (E.g. depending on the rest position of your elbow, it may be from (0 to 160) or from (-45 to 135).)

So you should apply a *Limit Rotation* constraint to the forearm bone (as the elbow movement is the result of rotating the forearm bone around its root).

Using bones in constraints, either as owners or as targets, is discussed in detail in the constraints pages.

Introduction

Inverse Kinematics (IK) simplifies the animation process, and makes it possible to make more advanced animations with lesser effort.

Inverse Kinematics allow you to position the last bone in a bone chain and the other bones are positioned automatically. This is like how moving someone's finger would cause their arm to follow it. By normal posing techniques, you would have to start from the root bone, and set bones sequentially until you reach the tip bone: When each parent bone is moved, its child bone would inherit its location and rotation. Thus making tiny precise changes in poses becomes harder farther down the chain, as you may have to adjust all the parent bones first.

This effort is effectively avoided by use of IK.

IK is mostly done with bone constraints although there is also a simple Auto IK feature in Pose Mode. They work by the same method but the constraints offer more options and control. Please refer to the following pages for details about these constraints:

- IK Solver
- Spline IK

Armature IK Panel

Reference

Mode::	Pose Mode
Panel::	*Properties* ▸ *Armature* ▸ *Inverse Kinematics*

This panel is used to select the IK Solver type for the armature: *Standard* or *iTaSC*. Most the time people will use the *Standard* IK solver.

Standard

TODO.

iTaSC

iTaSC stands for instantaneous Task Specification using Constraints.

iTaSC uses a different method to compute the Jacobian, which makes it able to handle other constraints than just end effectors position and orientation: iTaSC is a generic multi-constraint IK solver. However, this capability is not yet fully exploited in the current implementation, only two other types of constraints can be handled: Distance in the Cartesian space, and Joint Rotation in the joint space. The first one allows maintaining an end effector inside, at, or outside a sphere centered on a target position, the second one is the capability to control directly the rotation of a bone relative to its parent. Those interested in the mathematics can find a short description of the method used to build the Jacobian here.

iTaSC accepts a mix of constraints, and multiple constraints per bone: the solver computes the optimal pose according to the respective weights of each constraint. This is a major improvement from the current constraint system where constraints are solved one by one in order of definition so that conflicting constraints overwrite each other.

Precision
: The maximum variation of the end effector between two successive iterations at which a pose is obtained that is stable enough and the solver should stop the iterations. Lower values means higher precision on the end effector position.

Iterations
: The upper bound for the number of iterations.

Solver
: Selects the inverse Jacobian solver that iTaSC will use.

SDLS

Computes the damping automatically by estimating the level of 'cancellation' in the armature kinematics. This method works well with the Copy Pose constraint but has the drawback of damping more than necessary around the singular pose, which means slower movements. Of course, this is only noticeable in Simulation mode.

DLS

Computes the damping manually which can provide more reactivity and more precision.

Damping Max

Maximum amount of damping. Smaller values means less damping, hence more velocity and better precision but also more risk of oscillation at singular pose. 0 means no damping at all.

Damping Epsilon

Range of the damping zone around singular pose. Smaller values means a smaller zone of control and greater risk of passing over the singular pose, which means oscillation.

Note

Damping and *Epsilon* must be tuned for each armature. You should use the smallest values that preserve stability.

Note

- The SDLS solver does not work together with a Distance constraint. You must use the DLS solver if you are going to have a singular pose in your animation with the Distance constraint.
- Both solvers perform well if you do not have a singular pose.

Animation

In Animation mode, iTaSC operates like an IK solver: it is stateless and uses the pose from F-Curves interpolation as the start pose before the IK convergence. The target velocity is ignored and the solver converges until the given precision is obtained. Still the new solver is usually faster than the old one and provides features that are inherent to iTaSC: multiple targets per bone and multiple types of constraints.

Simulation

The Simulation mode is the stateful mode of the solver: it estimates the target's velocity, operates in a 'true time' context, ignores rotation from keyframes (except via a joint rotation constraint) and builds up a state cache automatically.

Reiteration

Never
: The solver starts from the rest pose and does not reiterate (converges) even for the first frame. This means that it will take a few frames to get to the target at the start of the animation.

Initial
: The solver starts from the rest pose and re-iterates until the given precision is achieved, but only on the first frame (i.e. a frame which doesn't have any previous frame in the cache). This option basically allows you to choose a different start pose than the rest pose and it is the default value. For the subsequent frames, the solver will track the target by integrating the joint velocity computed by the Jacobian solver over the time interval that the frame represents. The precision of the tracking depends on the feedback coefficient, number of substeps and velocity of the target.

Always
: The solver re-iterates on each frame until the given precision is achieved. This option omits most of the iTaSC dynamic behavior: the maximum joint velocity and the continuity between frames is not guaranteed anymore in compensation of better precision on the

end effector positions. It is an intermediate mode between *Animation* and real-time *Simulation*.

Auto Step

Use this option if you want to let the solver set how many substeps should be executed for each frame. A substep is a subdivision on the time between two frames for which the solver evaluates the IK equation and updates the joint position. More substeps means more processing but better precision on tracking the targets. The auto step algorithm estimates the optimal number of steps to get the best trade-off between processing and precision. It works by estimation of the nonlinearity of the pose and by limiting the amplitude of joint variation during a substep. It can be configured with next two parameters:

Min

Proposed minimum substep duration (in second). The auto step algorithm may reduce the substep further based on joint velocity.

Max

Maximum substep duration (in second). The auto step algorithm will not allow substep longer than this value.

Steps

If Auto Step is disabled, you can choose a fixed number of substeps with this parameter. Substep should not be longer than 10 ms, which means the number of steps is 4 for a 25 fps animation. If the armature seems unstable (vibrates) between frames, you can improve the stability by increasing the number of steps.

Feedback

Coefficient on end effector position error to set corrective joint velocity. The time constant of the error correction is the inverse of this value. However, this parameter has little effect on the dynamic of the armature since the algorithm evaluates the target velocity in any case. Setting this parameter to 0 means ‘opening the loop’: the solver tracks the velocity but not the position; the error will accumulate rapidly. Setting this value too high means an excessive amount of correction and risk of instability. The value should be in the range 20-100. Default value is 20, which

means that tracking errors are corrected in a typical time of 100-200 ms. The feedback coefficient is the reason why the armature continues to move slightly in Simulation mode even if the target has stopped moving: the residual error is progressively suppressed frame after frame.

Max Velocity
: Indicative maximum joint velocity in radian per second. This parameter has an important effect on the armature dynamic. Smaller value will cause the armature to move slowly and lag behind if the targets are moving rapidly. You can simulate an inertia by setting this parameter to a low value.

Bone IK Panel

Reference

Mode::	Pose Mode
Panel::	*Properties* ▸ *Bone* ▸ *Inverse Kinematics*

This panel is used to control how the *Pose Bones* work in the IK chain.

IK Stretch
: Stretch influence to IK target.

Lock
: Disallow movement around the axis.

Stiffness
: Stiffness around the axis. Influence disabled if using *Lock*.

Limit
: Limit movement around the axis.

iTaSC Solver

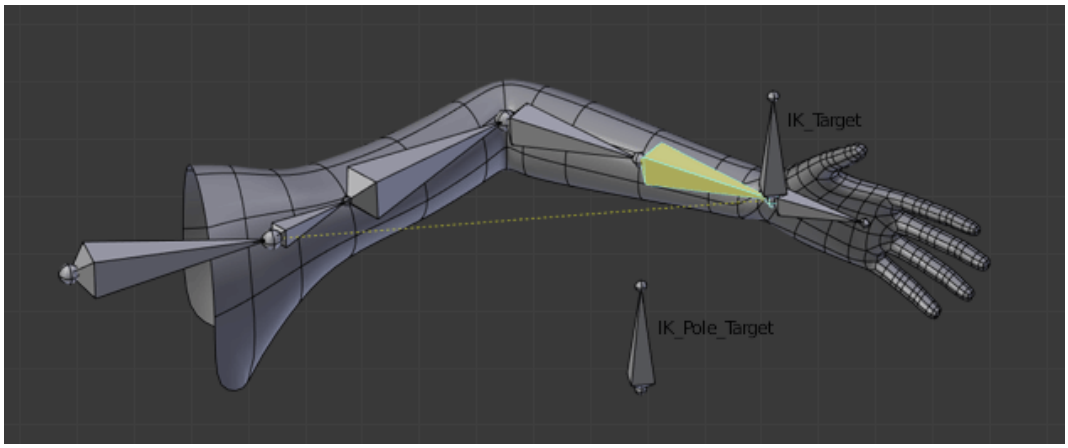
If the iTaSC IK Solver is used, the bone IK panel changes to add these additional parameters.

Control Rotation
: Activates a joint rotation constraint on that bone. The pose rotation computed from Action or UI interaction will be converted into a joint value and passed to the solver as target for the joint. This will give you control over the joint while the solver still tracks the other IK targets. You can use this feature to give a preferred pose for joints (e.g. rest pose) or to animate a joint angle by playing an action on it.

  Weight
  : The importance of the joint rotation constraint based on the constraints weight in case all constraints cannot be achieved at the same time. For example, if you want to enforce strongly the joint rotation, set a high weight on the joint rotation constraint and a low weight on the IK constraints.

Arm Rig Example

This arm uses two bones to overcome the twist problem for the forearm. IK locking is used to stop the forearm from bending, but the forearm can still be twisted manually by pressing `R Y Y` in *Pose Mode*, or by using other constraints.



IK Arm Example.

Note that, if a *Pole Target* is used, IK locking will not work on the root bone.

Spline IK

Spline IK is a constraint which aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well-integrated control offered by bones, Spline IK is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

Full description of the settings for the spline IK can be found on the Spline IK page.

Basic Setup

The Spline IK Constraint is not strictly an *Inverse Kinematics* method (i.e. IK Constraint), but rather a *Forward Kinematics* method (i.e. normal bone posing). However, it still shares some characteristics of the IK Constraint, such as operating on multiple bones, not being usable for Objects, and being evaluated after all other constraints have been evaluated. It should be noted that if a Standard IK chain and a Spline IK chain both affect a bone at the same time the Standard IK chain takes priority. Such setups are best avoided though, since the results may be difficult to control.

To setup Spline IK, it is necessary to have a chain of connected bones and a curve to constrain these bones to:

1. With the last bone in the chain selected, add a Spline IK Constraint from the Bone Constraints tab in the Properties.
2. Set the *Chain Length* setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
3. Finally, set the *Target* field to the curve that should control the curve.

Congratulations, the bone chain is now controlled by the curve.

Settings and Controls

For the precise list of options, see Spline IK constraint. This section is intended to introduce the workflow.

Roll Control

To control the Roll of the Spline IK chain, the standard methods of rotating the bones in the chain along their local Y axes still apply. For example, start at the farthest bone and simply rotate the bones in the chain around their local Y axes to adjust the roll of the chain from that point onward.

Applying Copy Rotation constraints on the bones also works.

Note

There are a couple of limitations to consider:

- Bones do not inherit a curve's tilt value to control their roll.
- There is no way of automatically creating a twisting effect where a dampened rotation is inherited up the chain. Consider using Bendy Bones instead.

Offset Controls

The entire bone chain can be made to follow the shape of the curve while still being able to be placed at an arbitrary point in 3D space when the *Chain Offset* option is enabled. By default, this option is not enabled, and the bones will be made to follow the curve in its untransformed position.

Length Control

The *Y Scale Mode* setting can be used to choose the way bones are scaled length-wise. The available options allow stretching the bone chain to fit the

curve, using the pre-IK scaling, or doing neither. In addition, the scale of the curve Object affects the result.

Thickness Controls

The thickness of the bones in the chain is controlled using the constraint's *XZ Scale Mode* setting. This setting determines the method used for determining the scaling on the X and Z axes of each bone in the chain.

The available modes are:

None
: This option keeps the X and Z scaling factors as 1.0.

Volume Preserve
: The X and Z scaling factors are taken as the inverse of the Y scaling factor (length of the bone), maintaining the 'volume' of the bone.

Bone Original
: This options just uses the X and Z scaling factors the bone would have after being evaluated in the standard way.

In addition to these modes, there is an option, *Use Curve Radius*. When this option is enabled, the average radius of the radii of the points on the curve where the joints of each bone are placed, are used to derive X and Z scaling factors. This allows the scaling effects, determined using the modes above, to be tweaked as necessary for artistic control.

Tips for Nice Setups

- For optimal deformations, it is recommended that the bones are roughly the same length, and that they are not too long, to facilitate a better fit to the curve. Also, bones should ideally be created in a way that follows the shape of the curve in its 'rest pose' shape, to minimize the problems in areas where the curve has sharp bends which may be especially noticeable when stretching is disabled.

- For control of the curve, it is recommended that hooks (in particular, Bone Hooks) are used to control the control points of the curve, with one hook per control point. In general, only a few control points should be needed for the curve (e.g. one for every 3-5 bones offers decent control).
- The type of curve used does not really matter, as long as a path can be extracted from it that could also be used by the Follow Path Constraint. This really depends on the level of control required from the hooks.

Lattice

Lattice – or commonly called deformation cage outside of Blender. A lattice consists of a three-dimensional non-renderable grid of vertices. Its main use is to apply a deformation to the object it controls with a Lattice Modifier. If the object is parented with *Lattice Deform* a Lattice Modifier is automatically applied.

Editing

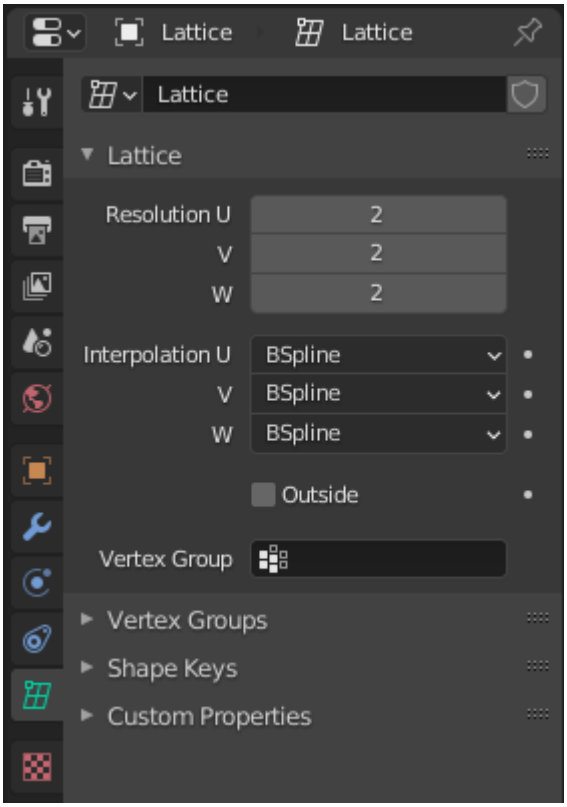
Flip (Distortion Free)
: Mirrors the vertices displacement from their base position.

  U, V, W

Make Regular
: Resets the whole lattice to a regular grid, where the cells are scaled to one cubic unit.

Properties

Lattice
: A Data-Block Menu.



Lattice properties.

Lattice

Points
: Rate of subdivision in the axes:

  U, V, W

Interpolation Type
: Selector for each axis. See Different types of interpolation..

  Linear, Cardinal, Catmull-Rom, B-Spline

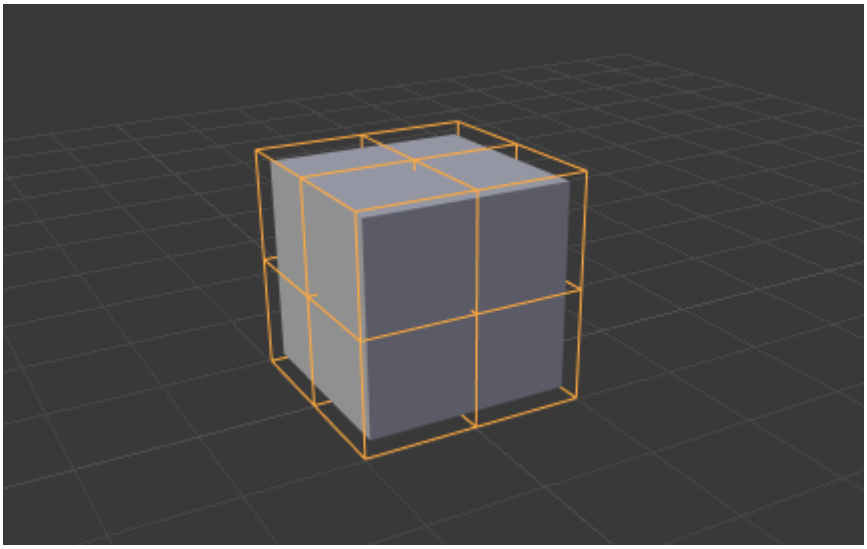
Outside
: Takes only the vertices on the surface of the lattice into account.

Vertex Group

The strength of the influence assigned as a weight to the individual vertices in the selected vertex group.

Usage

The lattice should be scaled and moved to fit around your object in Object Mode. Any scaling applied to the object in Edit Mode will result in the object deforming. This includes applying its scale with `Ctrl-A` as this will achieve the same result as scaling the lattice in Edit Mode, and therefore the object.



Lattice around the cube object in Object Mode.

Constraints

- Introduction
  - Adding & Removing Constraints
  - Tips
- Interface
  - Header
  - Common
  - Stack

Constraint Types

- Motion Tracking
  - Camera Solver Constraint
  - Object Solver Constraint
  - Follow Track Constraint
- Transform
  - Copy Location Constraint
  - Copy Rotation Constraint
  - Copy Scale Constraint
  - Copy Transforms Constraint
  - Limit Distance Constraint
  - Limit Location Constraint
  - Limit Rotation Constraint
  - Limit Scale Constraint
  - Maintain Volume Constraint
  - Transformation Constraint
  - Transform Cache Constraint
- Tracking
  - Clamp To Constraint
  - Damped Track Constraint
  - Inverse Kinematics Constraint
  - Locked Track Constraint
  - Spline IK Constraint

    - Stretch To Constraint
    - Track To Constraint
- Relationship
    - Action Constraint
    - Armature Constraint
    - Child Of Constraint
    - Floor Constraint
    - Follow Path Constraint
    - Pivot Constraint
    - Shrinkwrap Constraint

Introduction

Constraints are a way to control an object's properties (e.g. its location, rotation, scale), using either plain static values (like the "limit" ones), or another object, called "target" (like e.g. the "copy" ones).

Even though constraints are useful in static projects, their main usage is obviously in animation.

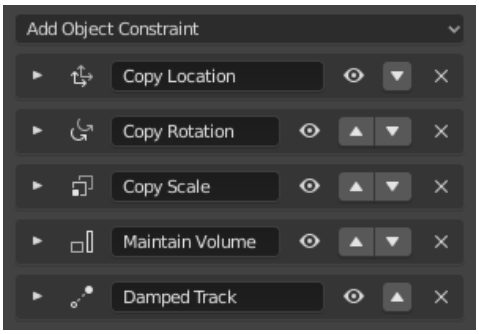
- You can control an object's animation through the targets used by its constraints (this is a form of indirect animation). Indeed, these targets can then control the constraint's owner's properties, and hence, animating the targets will indirectly animate the owner.
- You can animate constraints' settings. e.g. the *Influence* or when using an armature's bone as target, animate where along this bone (between root and tip) lays the real target point.

They can make the eyes of a tennis player track a tennis ball bouncing across the court, allow the wheels on a bus to all rotate together, help a dinosaur's legs bend at the knee automatically, and make it easy for a hand to grip the hilt of a sword and the sword to swing with the hand.

Constraints, in Blender, work with Objects and Bones. Read about using constraints in rigging in the Armature chapter.







Object	Bone	The Constraint Stack is evaluated from top to bottom.

Constraints work in combination with each other to form a Constraint Stack.

Adding & Removing Constraints

To add a constraint click on the *Add Object Constraint* menu in the Constraints tab. Alternatively, you can use the Add Constraint (with Targets) operator.

To copy constraints from one object to another use Copy Constraints to Selected Objects.

Any single constraint can be removed by clicking on the "X" button in the constraint's header. To remove all constants from an object use Clear Object Constraints.

Tip

Tracking constraints can be added/removed using the Track menu.

Tips

Constraints are a fantastic way to add sophistication and complexity to a rig.

But be careful not to rush in too quickly, piling up constraint upon constraint until you lose all sense of how they interact with each other.

Start simply. Get to know a single constraint inside and out. Copy Location Constraint is a good first constraint to explore it also has an animation

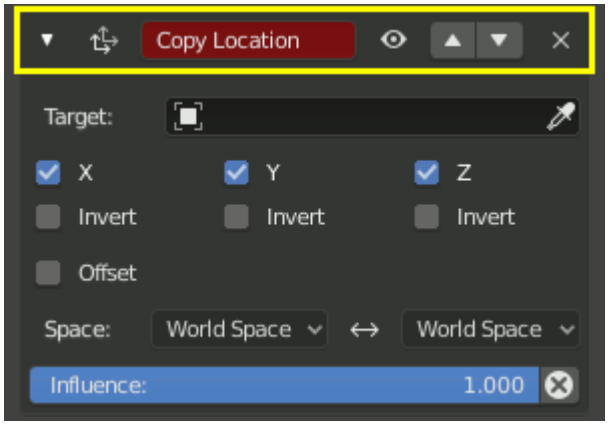
example. Take the time to understand every fundamental concept behind it, and the other constraints will make far more sense.

Interface

- Header
- Common
  - Target
  - Space
  - Influence
- Stack

Header

Every constraint has a header. The interface elements of the header are explained below using a Copy Location constraint as an example.



A Header sits at the top of every constraint.

Expand (down/right arrow icon)
: Show or Hide the settings of the constraint. Tidy up the constraint stack by hiding constraints that do not currently need attention. Constraints will continue to affect the scene even when hidden.

Icon
: The constraint type icon.

Name
: Give the constraint a meaningful name in this text field, which describes its purpose. Meaningful names help you and your team members understand what each constraint is supposed to do.

  The *red* background is a warning that the constraint is not yet functional. The background will turn *gray* when the constraint is functioning. When this Copy Location constraint has a valid target in the *target field* it will turn gray and begin to function.

Mute (eye icon)

Enable or Disable the constraint. Disabling a constraint will stop its affect on the scene.

Disabling a constraint is useful for turning off a constraint without losing all of its settings. Disabling means you can enable the constraint at a later time with the settings intact. Disabling is similar to setting the Influence to 0.0.

Extras

Apply `Ctrl-A`

Makes the constraint "real" by applying any transformations caused by the constraint to make the original object to match the results of the constraint and deletes the constraint.

Warning

Applying a constraint that is not first in the stack will ignore the stack order (it will be applied as if it was the first one), and may produce undesired results.

Duplicate `Shift-D`

Creates a duplicate of the constraint just below current one in the stack.

Copy to Selected

Copies the constraint from the Active object to all selected objects.

Move to First/Last

Moves the constraint to the first or last position in the constraint stack.

(Delete) `X`, `Delete`

Delete the constraint from the stack. The settings will be lost. The constraint will no longer affect the final outcome of the stack.

Move `::::`

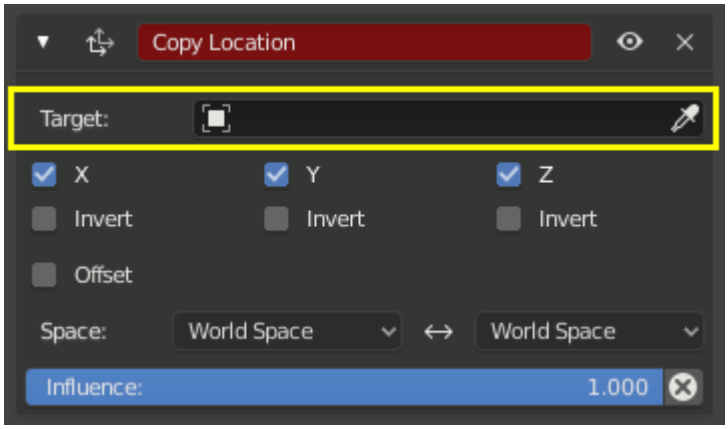
Move a constraint up or down in the constraint stack. Since the stack is evaluated from top to bottom, moving a constraint in the stack can significantly affect the final outcome of the stack.

- If there is only one constraint in the stack, the arrows will not be displayed.
- If the constraint is at the top of the stack, only the down arrow will be displayed.
- If the constraint is at the bottom of the stack, only the up arrow will be displayed.

Common

Target

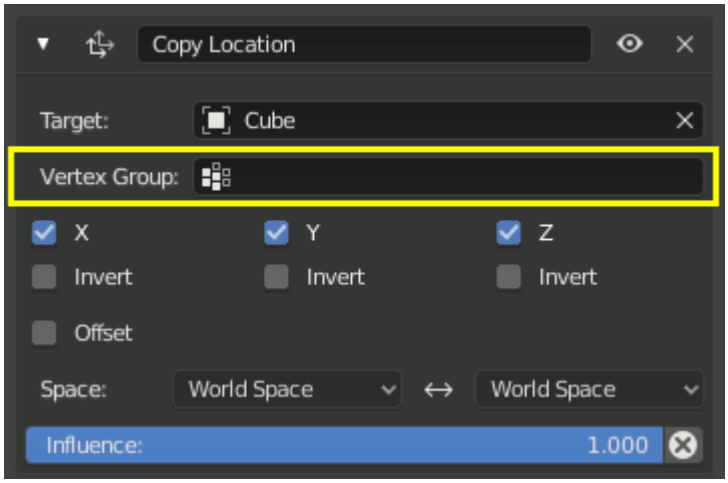
The Target Data ID field lets you link the constraint to a Target object of your choosing. This link provides data to the constraint so that it can begin to function. For example, the Copy Location Constraint needs location data to function. Fill in the Target field, and the Copy Location constraint will begin to use location data from the Target object.



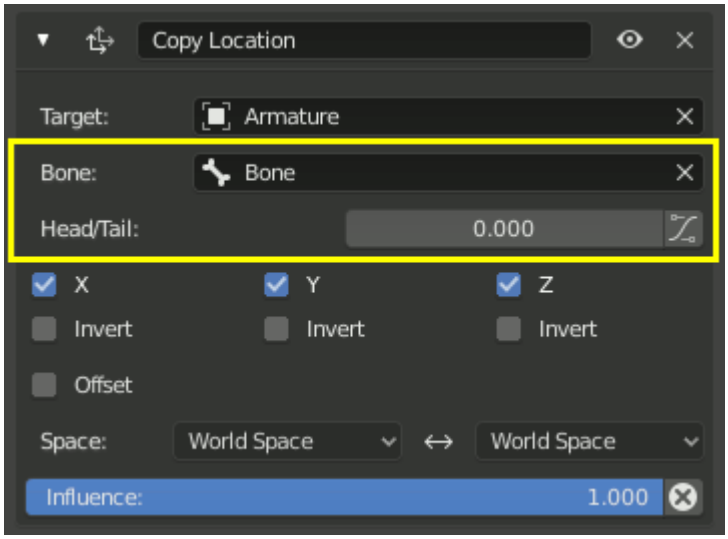
The Target field must be filled in for the constraint to function.

By default, the Target will use the Object Origin as the target point.

If the Target field links to a Mesh or Lattice object, a Vertex Group field will appear. Enter the name of a vertex group and the constraint will target the median point of this vertex group instead of the object's origin.



If the Target field links to an Armature, a Bone field will appear along with a *Head/Tail* slider. Enter the name of a bone and the constraint will target the bone instead of the entire armature object origin.



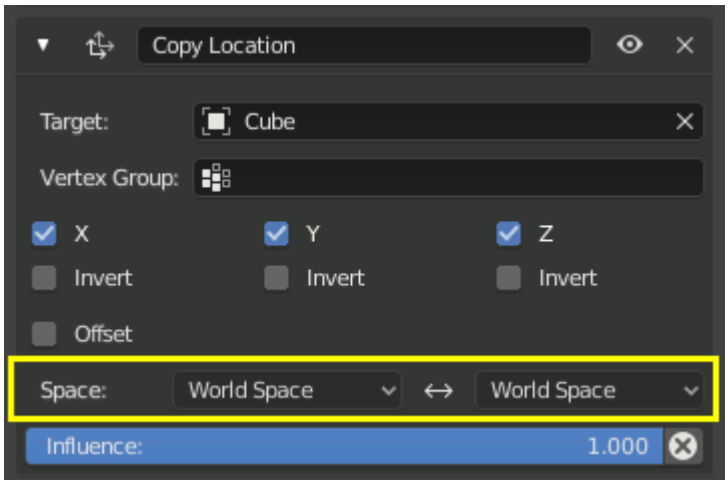
The slider moves the precise position of the target between the Head and Tail of the bone. Some constraints have a button next to the slider that enables using the curved shape of Bendy Bones.

Space

Constraints need a frame of reference in order to function. This frame of reference is called the “space” of the constraint. Choosing one space vs.

another will change this frame of reference and substantially alter the behavior of a constraint.

To understand how changing the space will change the behavior of the constraint, consider experimenting with two empties. Make sure they display as arrows so that you can see the local axes for each empty. Make sure to size one empty a little larger than the other so that they are both always visible even if directly on top of each other. Then add a constraint to one empty that targets the other and experiment thoroughly by moving, rotating and scaling the target in many different ways.



This constraint is set to use World Space as the frame of reference for both its *Target* space and its *Owner* space.

Target Space & Owner Space

The space used to evaluate the target of the constraint is called the *Target* space. The space used to evaluate the constrained object (the object that owns the constraint) is called the *Owner* space. Hover over the space select menu(s) to learn whether it affects the space of the target or the space of the owner.

When the constraints use a *Target* and/or/nor an *Owner* space there will be no, one or two selector(s). The Copy Location constraint in example use both Target **and** *Owner* space.

When a constraint uses both *Target* and *Owner* space, the Target and Owner can be any combination of space types.

Space Types

World Space

In this space type the world is the frame of reference for the object (or bone). Location is relative to the world origin. Rotation and Scale are oriented to the world axes. Transformations to the object, the object's parent and any other constraints higher up in the constraint stack are all taken into account.

Local Space

This space excludes all effects of the parent objects or bones, as well as the rest position and orientation of the bone itself. Only transformations applied to the object or bone itself are taken into account.

Warning

For objects without a parent Local Space has a special meaning, different from the normal behavior of local space for bones or objects that have a parent. This behavior is kept for backwards compatibility, but may be removed in the future and shouldn't be used.

Local with Parent *Bones Only*

The bone position and orientation is evaluated relative to its rest pose location and orientation, thus including both its own transformations and those caused by a possible parent relationship (i.e. the chain's transformations above the bone).

Pose Space *Bones Only*

The bone position and orientation is evaluated in the armature object local space (i.e. independently from the armature transformations in *Object Mode*). Hence, if the armature object has null transformations, *Pose Space* will have the same effect as *World Space*.

Custom Space

The position and orientation is evaluated relative to the current position and orientation of an arbitrary object or bone that is specified via additional input fields that appear when this option is selected. This can be used to evaluate the constraint using an arbitrary coordinate system.

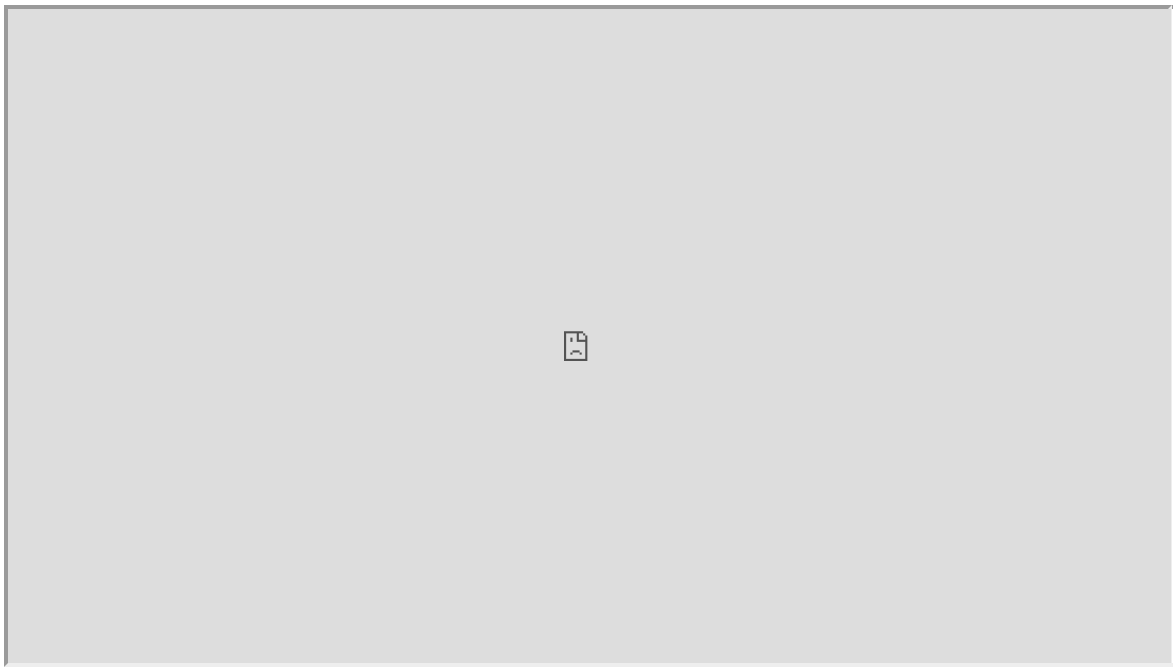
Local Space (Owner Orientation) *Bone Targets Only*

This space works like *Local Space*, with an additional coordinate space transformation that compensates for the difference in the rest pose orientations of the owner and target bones. If applied as the *Local Space* of the owner, this will produce the same global space movement as the target, provided parents are still at rest pose.

This option replaces the following setup with two additional bones:

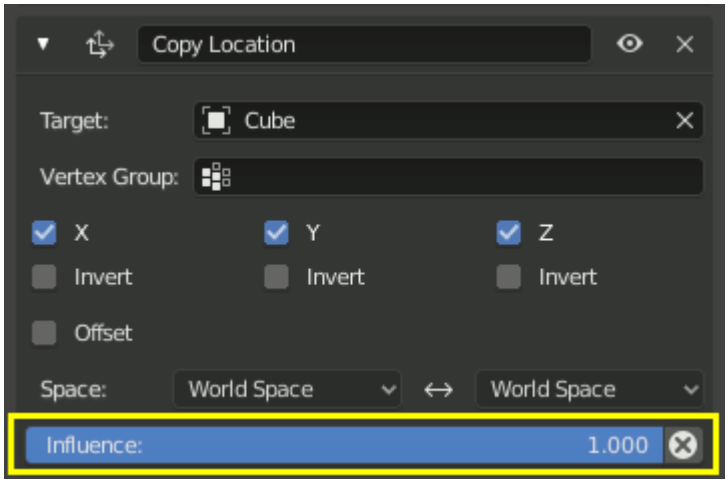
1. An extra child bone of the target, rotated the same as the owner in rest pose.
2. An extra sibling bone of the target, positioned same as the child in rest pose and using Copy Transforms in *World Space* from the child.
3. The constraint uses *Local Space* of the sibling instead of the original target.

This video demonstrates the difference from ordinary *Local Space*:



Influence

The influence slider determines how much the constraint will affect the constrained object (target).



An influence of 0.0 will have no effect. An influence of 1.0 will have the full effect.

Values between (0.0 and 1.0) will have a partial effect, but be careful. These partial effects can be difficult to control, especially as the constraint stack grows in complexity.

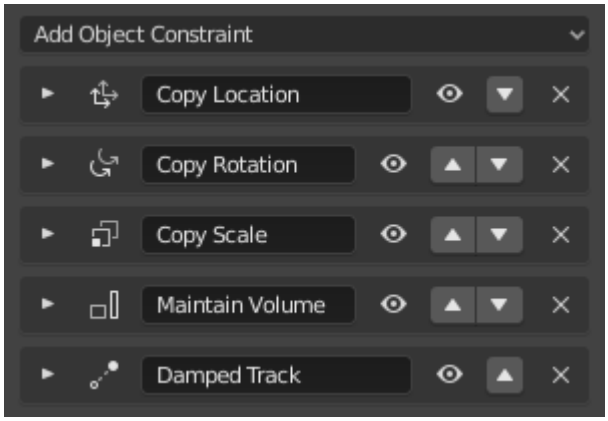
The influence value is animatable, allowing constraints to be turned off, or partially on as needed.

(Disable and Keep Transform)
: Disables the constraint while trying to preserve the current object position. This may not work perfectly if other constraints remain active.

Stack

The combination of all the constraints affecting an object is called the Constraints Stack. The Stack is in the Constraints panel, below the Add Constraint menu.

Constraints in the stack are evaluated from top to bottom. The order of each constraint has a substantial impact on the final outcome of the stack. Changing the order of the constraints can change the behavior of the entire stack.



The constraints in this example stack are evaluated from top to bottom starting with the “Copy Location” constraint and ending with the final “Damped Track” constraint.

To change the order of a constraint use the up/down arrows in the header.

Motion Tracking Constraints

- Camera Solver Constraint
  - Options
- Object Solver Constraint
  - Options
- Follow Track Constraint
  - Options
  - Example

Camera Solver Constraint

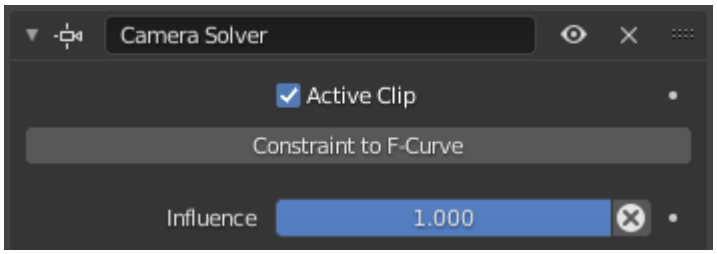
The *Camera Solver* constraint gives the owner of this constraint, the location and rotation of the “solved camera motion”.

The “solved camera motion” is where Blender reconstructs the position of the physical, real-world camera, when it filmed the video footage, relative to the thing being tracked.

Note

This constraint only works after you have set up a minimum of eight markers and pressed Solve Camera Motion (*Movie Clip Editor* ‣ *Toolbar* ‣ *Solve* ‣ *Solve Camera Motion*).

Options



Camera Solver Constraint panel.

Active Clip
: Receive tracking data from the scene’s Active Clip. If unchecked, an option appears to choose from the other clips.

Constraint to F-Curve
: Applies the constraint, creating Keyframes for the transforms.

Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Object Solver Constraint

The *Object Solver* constraint gives the owner of this constraint, the location and rotation of the “solved object motion”.

The “solved object motion” is where Blender thinks the physical, real-world (tracked) object was, relative to the camera that filmed it.

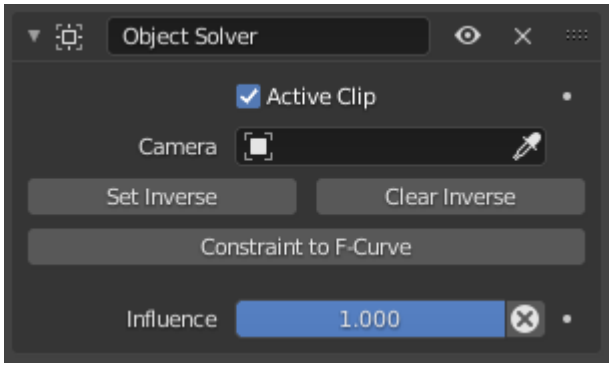
Can be used to add a mesh to video for example.

Note

This constraint only works after you have set up a minimum of eight markers and pressed Solve object Motion. Located at *Movie Clip Editor* ‣ *Toolbar* ‣ *Solve* ‣ *Solve Camera Motion*.

If it says *Solve Camera Motion* instead of *Solve Object Motion* then go into the *Movie Clip Editor* ‣ *Sidebar region* ‣ *Objects* and switch it from the camera, to an object.

Options



Object Solver Constraint panel.

Active Clip
: Receive tracking data from the scene's Active Clip. If unchecked, an option appears to choose from the other clips.

Object
: Select a tracked object to receive transform data from.

Camera
: Select the camera to which the motion is parented to (if left empty the active scene camera is used).

Set Inverse
: Moves the origin of the object to the origin of the camera.

Clear Inverse
: Moves the origin of the object back to the spot set in the Movie Clip Editor *Toolbar ▸ Solve ▸ Orientation ▸ Set Origin*.

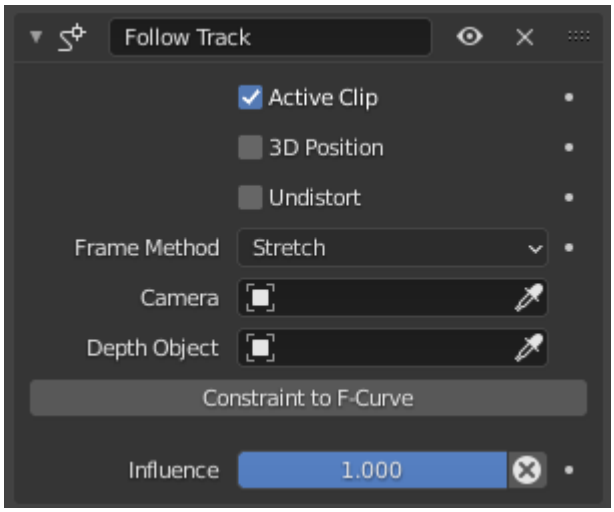
Constraint to F-Curve
: Applies the constraint, creating keyframes for the transforms.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Follow Track Constraint

By default the Follow Track constraint is making objects have the same position at a frame as the track has. The motion of this object happens on a single plane defined by the camera and the original position of the object.

Options



Follow Track Constraint panel.

Active Clip
: Receive tracking data from the scene's Active Clip. If unchecked, an option appears to choose from the other clips.

3D Position
: Use the 3D position of the track to parent to.

Undistorted
: Parent to the undistorted position of the 2D track.

Frame Method
: Defines how the footage is fitted in the camera frame.

Camera
: Select the camera to which the motion is parented to (if empty, the active scene camera is used).

Depth Object
: If this object is set, constrained objects will be projected onto the surface of this depth object which can be used to create facial makeup visual effects.

Constraint to F-Curve
: Creates F-Curves for the object that copies the movement caused by the constraint.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example

Follow Track Example Video

Transform Constraints

- Copy Location Constraint
- Copy Rotation Constraint
- Copy Scale Constraint
- Copy Transforms Constraint
- Limit Distance Constraint
- Limit Location Constraint
- Limit Rotation Constraint
- Limit Scale Constraint
- Maintain Volume Constraint
- Transformation Constraint
- Transform Cache Constraint

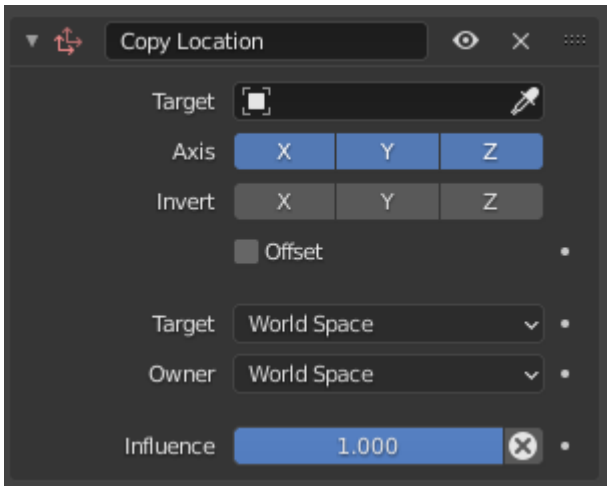
Copy Location Constraint

The *Copy Location* constraint forces its owner to have the same location as its target.

Important

Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options



Copy Location panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Axis
: These buttons control which axes are constrained.

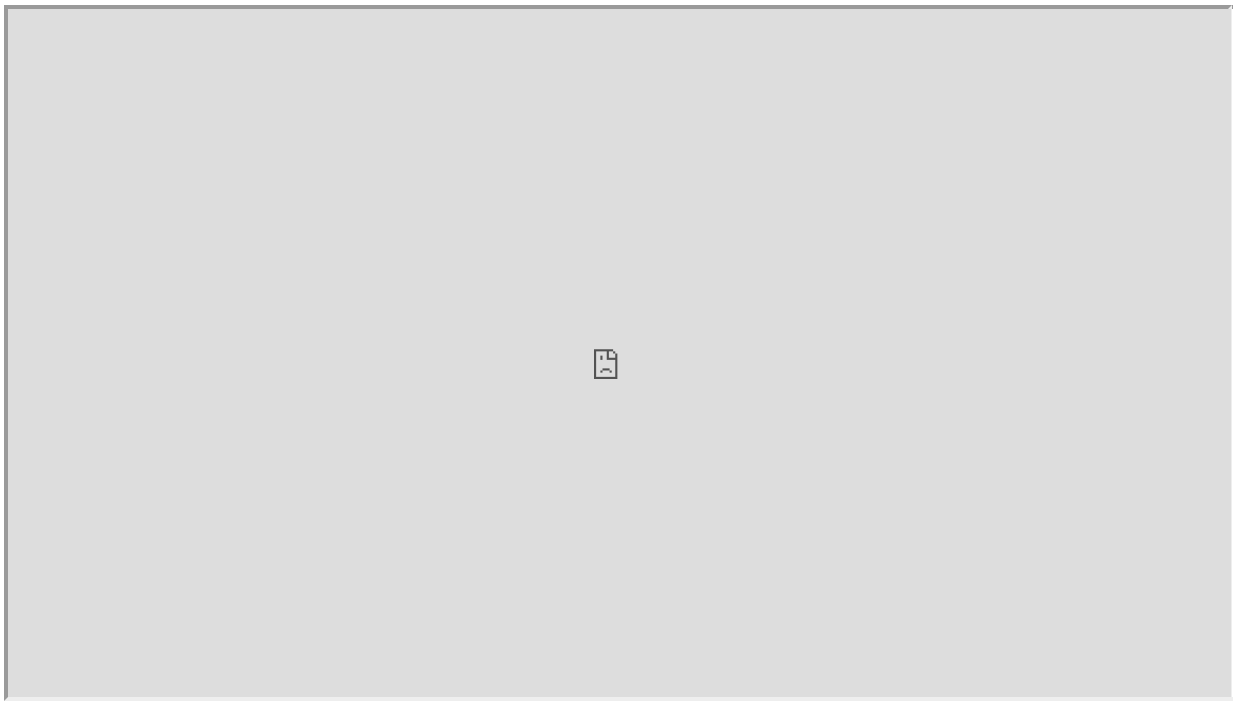
Invert
: Invert their respective corresponding axis coordinates.

Offset
: When enabled, this control allows the owner to be moved (using its current transform properties), relative to its target's position.

Target/Owner
: Standard conversion between spaces. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

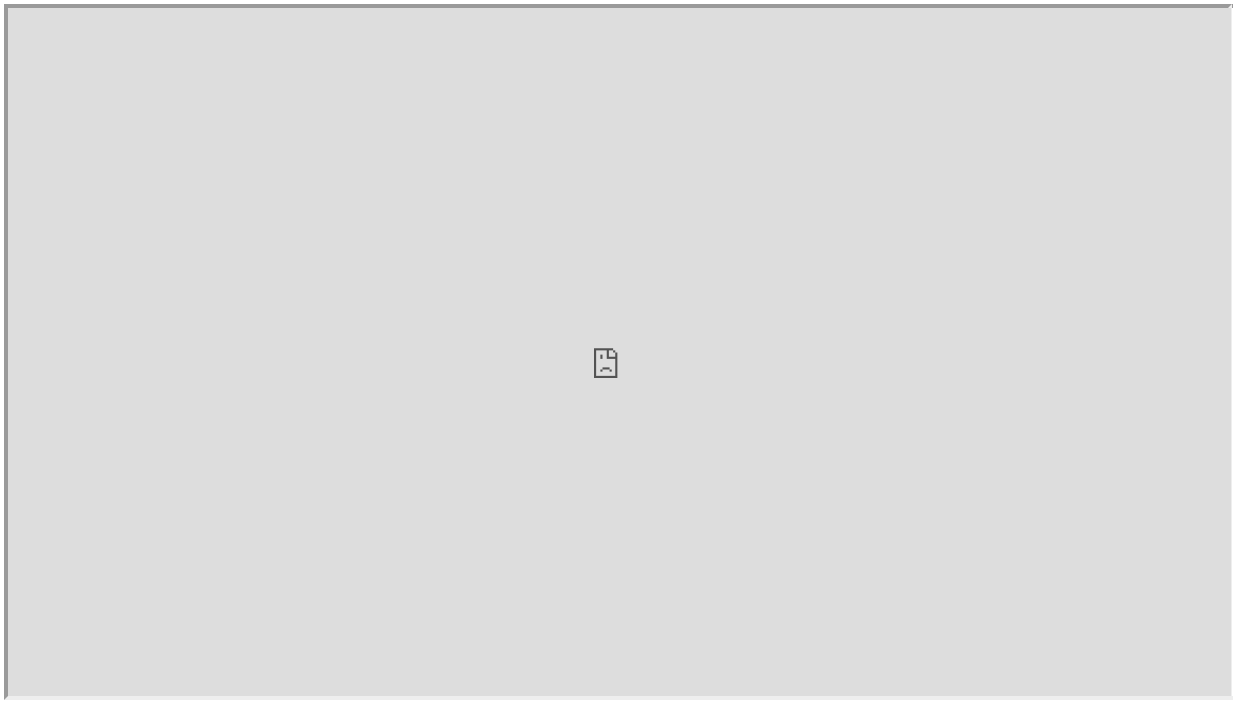
Examples



Animation

Let us animate a solar system with the *Copy Location* constraint and its *Offset* option. You can make the owner, called “moon”, describe perfect circles centered on the world origin (using e.g. *Location X*/*Y* sine and cosine F-Curves, see Built-in Function Modifier). Then copy the location of a target “earth” with the *Offset* checkbox enabled to model a satellite in a (simplified) orbit around its planet. Repeat these steps for more planets circling around its center star “sun”.

Following video is a small animation of a solar system created using (among a few others) the previously described technique:



Note that, this ‘solar’ system is not realistic at all (the wrong scale, the earth is rotating in the wrong direction around the sun, …).

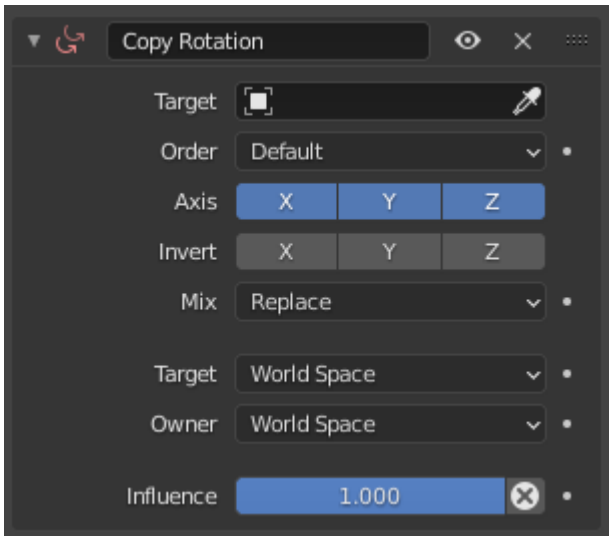
You can download the blend-file used to create this animation.

Furthermore you can also animate a few properties of each constraint using animation curves: e.g. you can animate the *Influence* of a constraint. It is used to first let the camera follow the moon, then the earth, and finally using two *Copy Location* constraints with *Offset* set.

Copy Rotation Constraint

The *Copy Rotation* constraint forces its owner to match the rotation of its target.

Options



Copy Rotation panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Order
: Allows specifying which Euler order to use during the copy operation. Defaults to the order of the owner.

Axis
: These buttons control which axes are constrained.

Invert

Invert their respective corresponding axis coordinates.

Mix
: Specifies how the new rotation is combined with the existing rotation.

  Replace
  : The new axis values replace existing values.

  Add
  : The new axis values are added to the existing values.

  Before Original
  : The new rotation is added before the existing rotation, as if it was applied to a parent of the constraint owner.

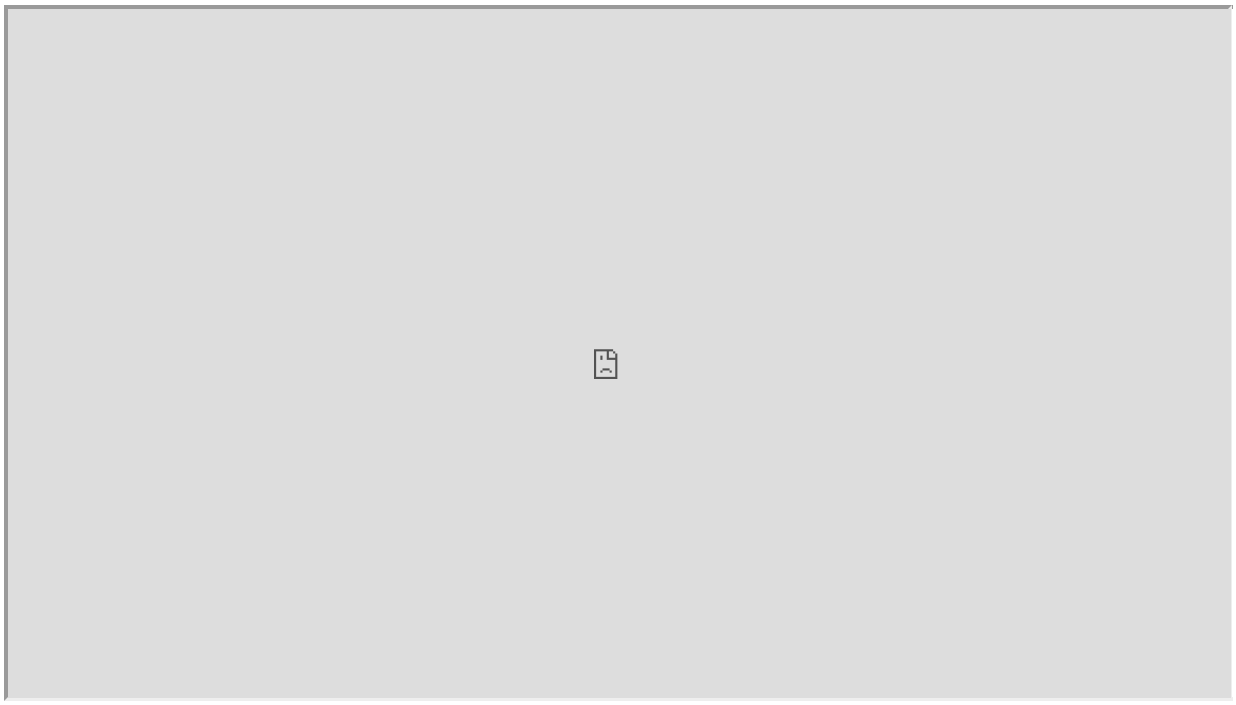
  After Original
  : The new rotation is added after the existing rotation, as if it was applied to a child of the constraint owner.

  Offset (Legacy)
  : This replicates the behavior of the original Offset checkbox. It was intended to be similar to the *Before Original* behavior, but does not work correctly with multiple axis rotations, and is thus deprecated.

Target/Owner
: Standard conversion between spaces. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



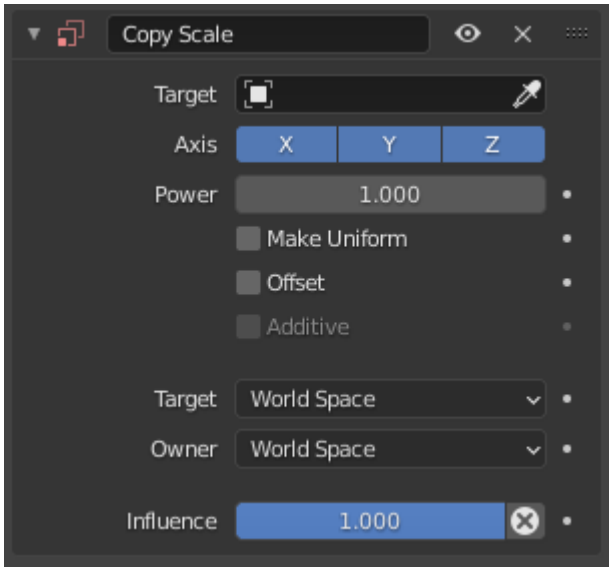
Copy Scale Constraint

The *Copy Scale* constraint forces its owner to have the same scale as its target.

Note

Here we talk of *scale*, not of *size*! Indeed, you can have two objects, one much bigger than the other, and yet both of them have the same scale. This is also true with bones: in *Pose Mode*, they all have a unitary scale when they are in rest position, represented by their visible length.

Options



Copy Scale panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Axis
: These buttons control which axes of the target scale are copied.

Power
: Allows raising the copied scale to the specified arbitrary power.

Make Uniform
: Instead of copying scale for individual axes, apply a uniform scaling factor to all axes of the owner that achieves the same overall change in volume.

Offset
: When enabled, the constraint combines the copied scale with the owner's scale, instead of overwriting it.

Additive
: Uses addition instead of multiplication in the implementation of the *Offset* option.

Target/Owner
: Standard conversion between spaces. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

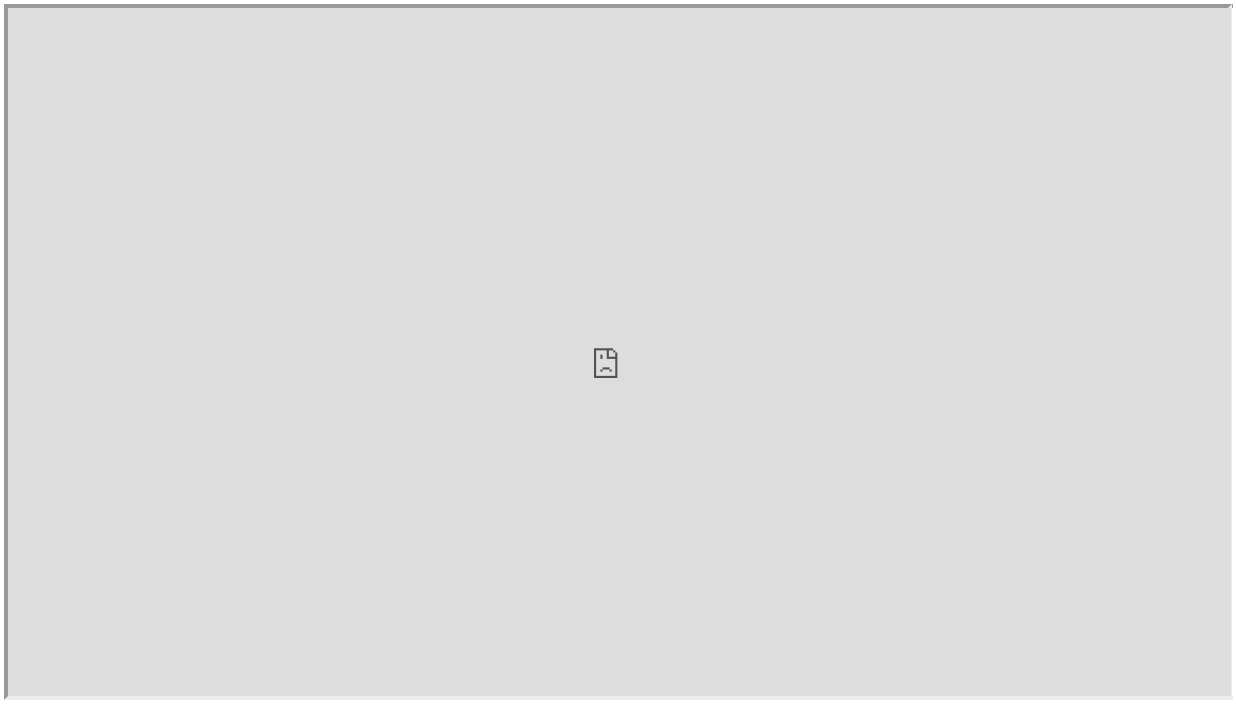
Note

Since scale is a multiplicative quantity, it should be combined using multiplication, and split into fractions or inverted via power. Thus the use of *Power* is more mathematically correct than *Influence*, which uses linear interpolation. The use of the *Additive* option is also not recommended.

Tip

To copy scale from one axis of the target to all axes of the owner, disable other axes, enable *Make Uniform*, and set *Power* to 3.

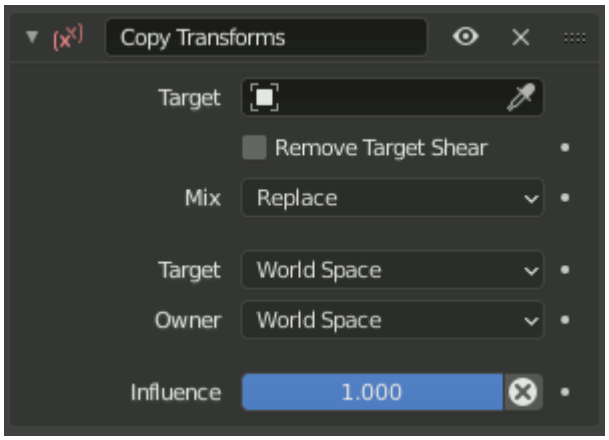
Example



Copy Transforms Constraint

The *Copy Transforms* constraint forces its owner to have the same transforms as its target.

Options



Copy Transforms panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Remove Target Shear
: Removes shearing from the target transformation after the target space conversion, ensuring it consists purely of translation, rotation and scale. Note that Copy Rotation always does this.

Mix
: Specifies how the copied transformation is combined with the existing transformation.

  Replace

The new transformation replaces the existing transformation.

Before/After Original (Full)

The new transformation is added before/after the existing transformation, as if it was applied to an imaginary parent/child of the constraint owner. Scale is handled like in the most basic Full Inherit Scale mode of bones, so combining non-uniform scale and rotation will create shear.

Before/After Original (Aligned)

The new transformation is added before/after the existing transformation, as if it was applied to an imaginary parent/child of the constraint owner. Scale is handled like in the Aligned Inherit Scale mode of bones to avoid creating shear.

This is equivalent to using the *Split Channels* option, but replacing the location component with the result of *Full*. If only uniform scale is used, the result is identical to *Full*.

Before/After Original (Split Channels)

Combines location, rotation and scale components of the transformation separately, similar to a sequence of three Copy Location, Copy Rotation and Copy Scale (with Offset) constraints bundled together in one operation; the result may be slightly different in case of sheared inputs.

Unlike *Aligned*, in this mode location channels are simply added together, so rotation and scale components of the input transformations cannot affect the resulting location.

Target/Owner

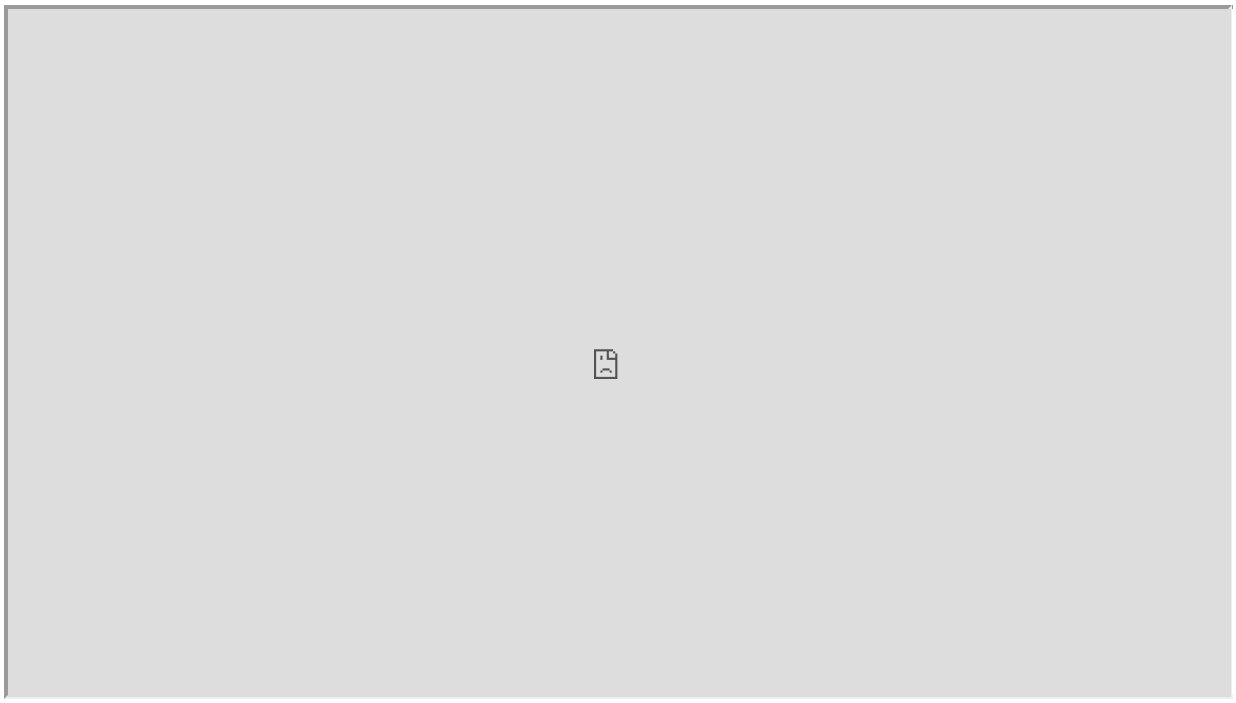
Standard conversion between spaces. See common constraint properties for more information.

Influence

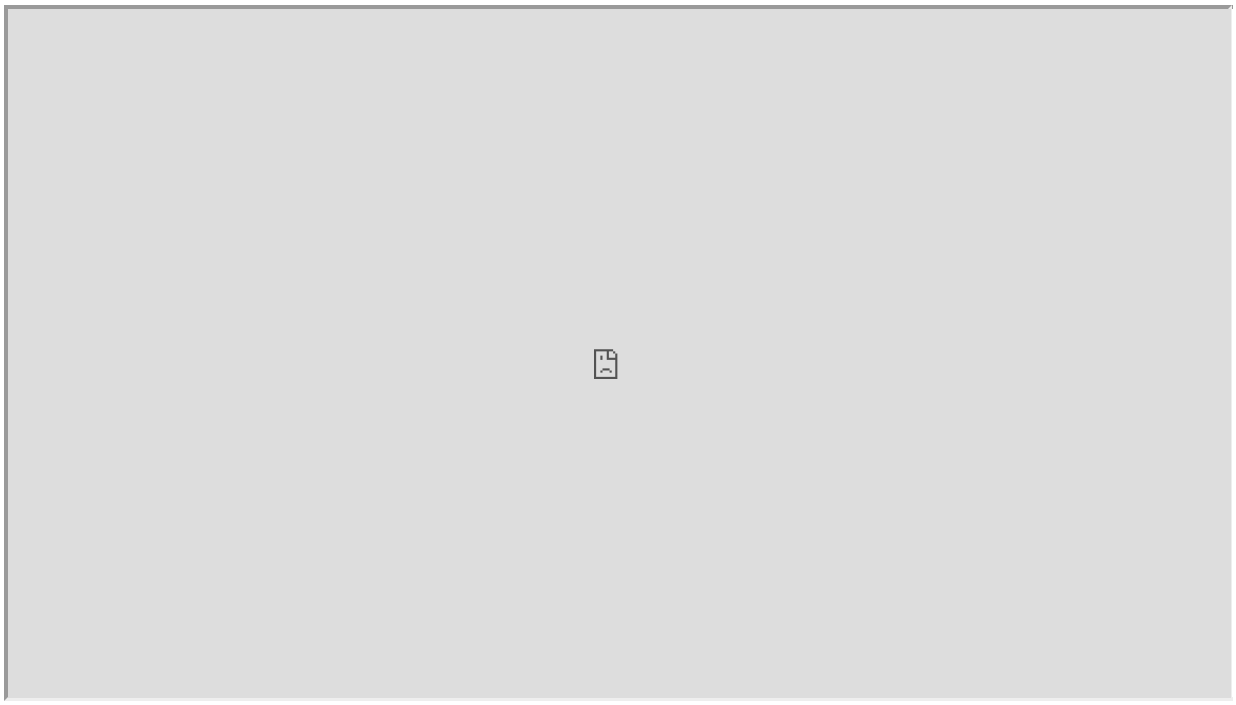
Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example

This video shows the difference between the mix modes. The right input is mixed after the left one:



A general demonstration of the constraint:



Limit Distance Constraint

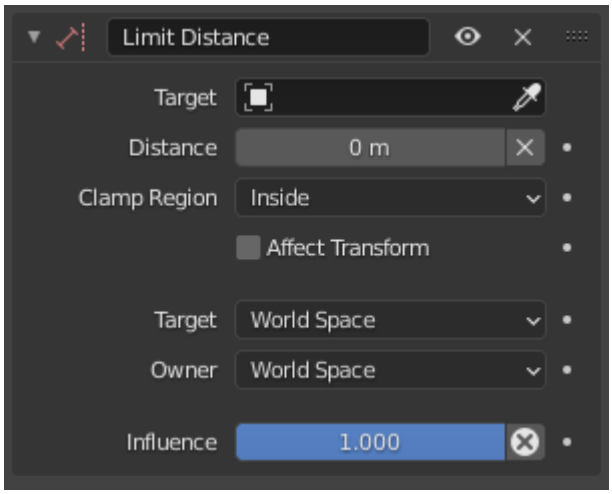
The *Limit Distance* constraint forces its owner to stay either further from, nearer to, or exactly at a given distance from its target. In other words, the owner's location is constrained either outside, inside, or at the surface of a sphere centered on its target.

When you specify a (new) target, the *Distance* value is automatically set to correspond to the distance between the owner and this target.

Important

Note that if you use such a constraint on a *connected* bone, it will have no effect, as it is the parent's tip which controls the position of your owner bone's root.

Options



Limit Distance panel.

Target

Data ID used to select the constraint's target, and is not functional (red state) when it has none. See common constraint properties for more information.

Distance

This number field sets the limit distance, i.e. the radius of the constraining sphere.

(Reset Distance)

Resets the *Distance* value,so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Clamp Region

The *Limit Mode* select menu allows you to choose how to use the sphere defined by the *Distance* setting and target's origin:

Inside

The owner is constrained *inside* the sphere.

Outside

The owner is constrained *outside* the sphere.

Surface

The owner is constrained *on the surface* of the sphere.

Affect Transform

Transform operators will take the constraint into account to immediately restrict the resulting transform property values.

Target/Owner

Standard conversion between spaces. See common constraint properties for more information.

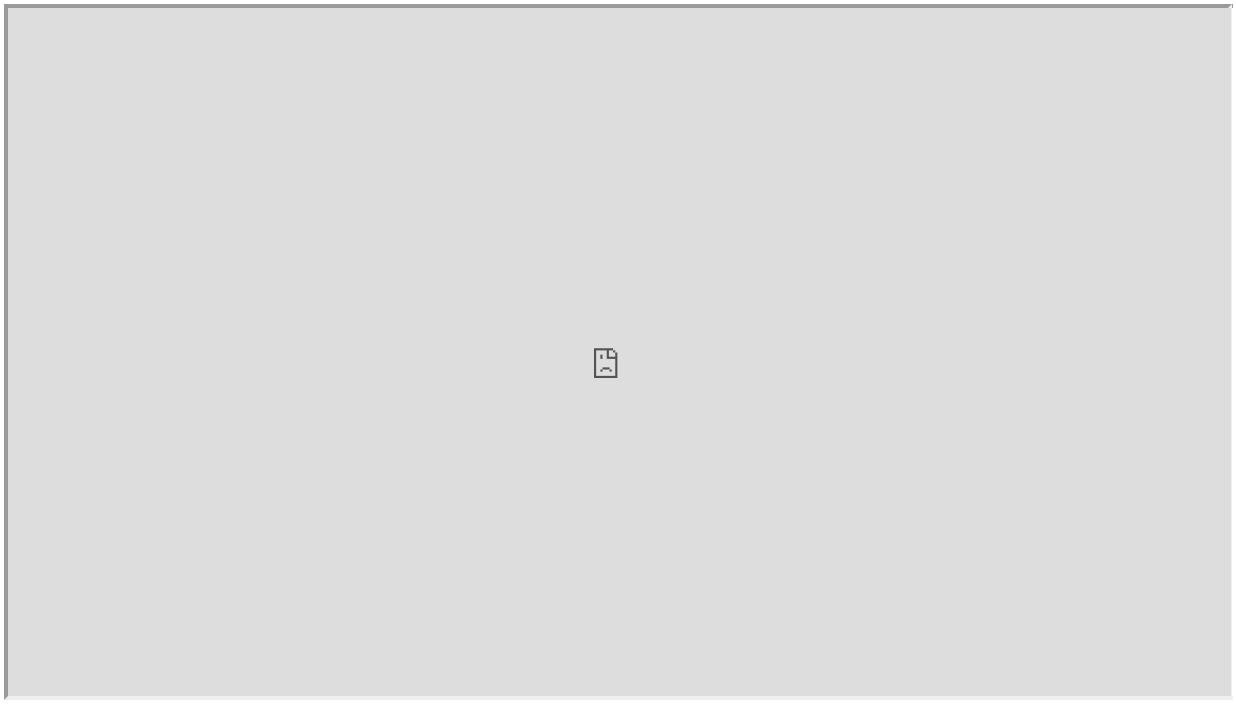
Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Tip

Evaluating both owner and target in a Custom Space using the root bone or any other suitable parent bone will automatically scale the effective distance with the relevant part of the rig.

Example



Limit Location Constraint

An object or *unconnected* bone can be moved around the scene along the X, Y and Z axes. This constraint restricts the amount of allowed translations along each axis, through lower and upper bounds.

The limits for an object are calculated from its origin, and the limits of a bone, from its root.

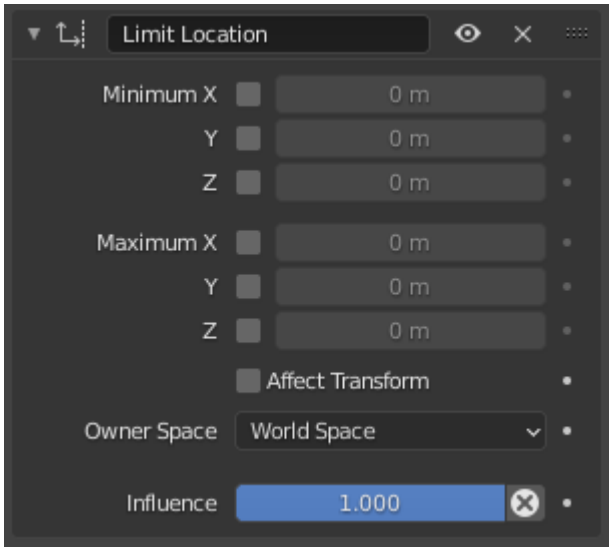
It is interesting to note that even though the constraint limits the visual and rendered location of its owner, its owner's data-block still allows (by default) the object or bone to have coordinates outside the minimum and maximum ranges. This can be seen in its *Transform* panel.

When an owner is selected and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally specified location.

Similarly, if its owner has an internal location that is beyond the limits, dragging it back into the limit area will appear to do nothing until the internal coordinates are back within the limit threshold (unless you enabled the *Affect Transform* option, see below).

Setting equal the min and max values of an axis, locks the owner's movement along that axis… Although this is possible, using the *Transformation Properties* axis locking feature is probably easier!

Options



Limit Location panel.

Minimum X, Y, Z
: These buttons enable the lower boundary for the location of the owner's origin along, respectively, the X, Y and Z axes of the chosen *Space*. The number field below them controls the value of their limit. Note that if a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Maximum X, Y, Z
: These buttons enable the upper boundary for the location of the owner's origin along, respectively, the X, Y and Z axes of the chosen *Space*. Same options as above.

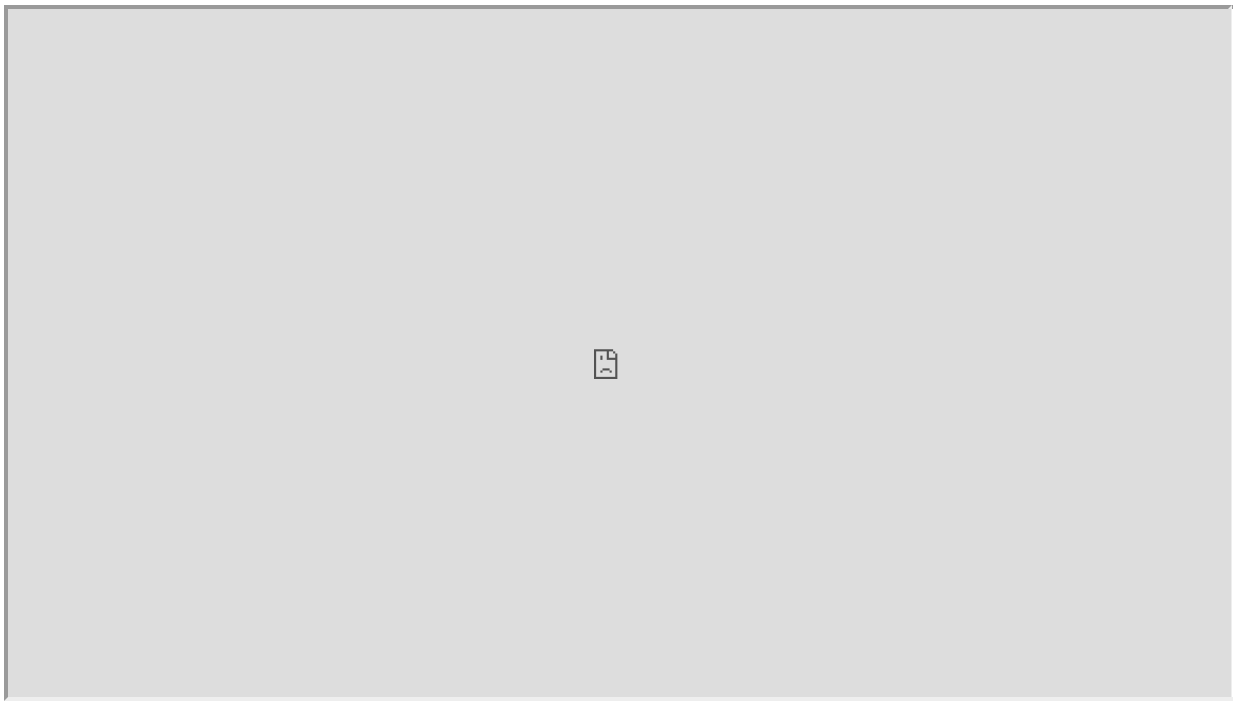
Affect Transform
: As pointed out before by default, even though visually constrained, the owner can still have coordinates out of bounds (as shown by the *Transform* panel). Well, when you enable this checkbox, this is no longer possible – the owner's transform properties are also limited by the constraint. However, note that, the constraint does not directly modify the coordinates: you have to select its owner one way or another for this to take effect…

Owner

This constraint allows you to choose in which space to evaluate its owner's transform properties. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



Limit Rotation Constraint

This constraint restricts the rotation of an object or bone to be inside specified angular limits. The limits are given as Euler rotation ranges (a min and max angle), and a separate range can be given for each of the three axes.

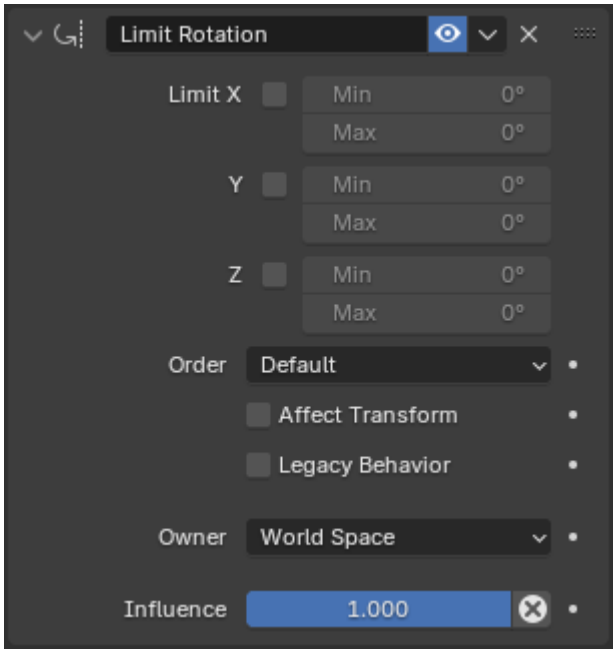
As with all constraints in Blender, this does not (by default) restrict the user-set rotation values of the object/bone as seen in the *Transform* panel. When the object/bone is rotated outside the limit range, it will be constrained to that range in its final displayed/rendered position, but the user-set rotation values will still be outside that range. If the constraint is removed, the object/bone will then jump back to match those user-set values.

Something unique about the Limit Rotation constraint (as compared to the Limit Location and Limit Scale constraints) is that rotations loop, and therefore the meaning of the limit range is subtly different. All constraints in Blender internally work on transform matrices, which can't distinguish between e.g. 180 and -180 degrees, or 0, 360, and 720 degrees. In other words, any angles that result in the same visual rotation are indistinguishable to the constraint system.

What this means for the Limit Rotation constraint is that when the user-set rotation is outside of the limit range, the final displayed rotation will snap to the closest *visual* rotation in that range, not the closest numerical angle. For example, if you have a limit range of 0 to 90 degrees then a user-set rotation of 340 degrees will actually snap to *0 degrees* because that is the closer *visual* rotation, even though 340 is numerically closer to 90.

Note that this constraint does not constrain the bone if it is manipulated by the IK solver. For constraining the rotation of a bone for IK purposes, see Inverse Kinematics.

Options



Limit Rotation panel.

Limit X, Y, Z

These buttons enable the rotation limit around respectively the X, Y and Z axes of the owner, in the chosen *Owner* space. The *Min* and *Max* number fields to their right control the value of their lower and upper boundaries, respectively.

Note

- If a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.
- Unlike the Limit Location constraint, you cannot separately enable lower or upper limits.
- The constraint can be used to simply remove shear from the owner transformation by leaving all limits disabled.

Order

Allows specifying which Euler order to use when applying the limits. Defaults to the order of the owner, or XYZ if the owner uses non-Euler rotations.

Affect Transform

The constraint is taken into account when the object is manually rotated using transformation tools in the editors. This prevents assigning transformation property values (as shown in the *Transform* panel) that exceed the specified limits.

Legacy Behavior

For backwards compatibility: make the constraint behave in the semi-broken way it did prior to Blender 4.2. This old behavior does not properly account for the looping nature of rotations, and therefore causes unpredictable/erratic rotation snapping. However, this behavior can still be useful in some specific circumstances when *Owner* is set to local space, and some older rig setups utilize that. However, that behavior is better and more robustly accomplished with drivers directly on the object/bone's rotation properties, so new rigs should favor that approach over using this option.

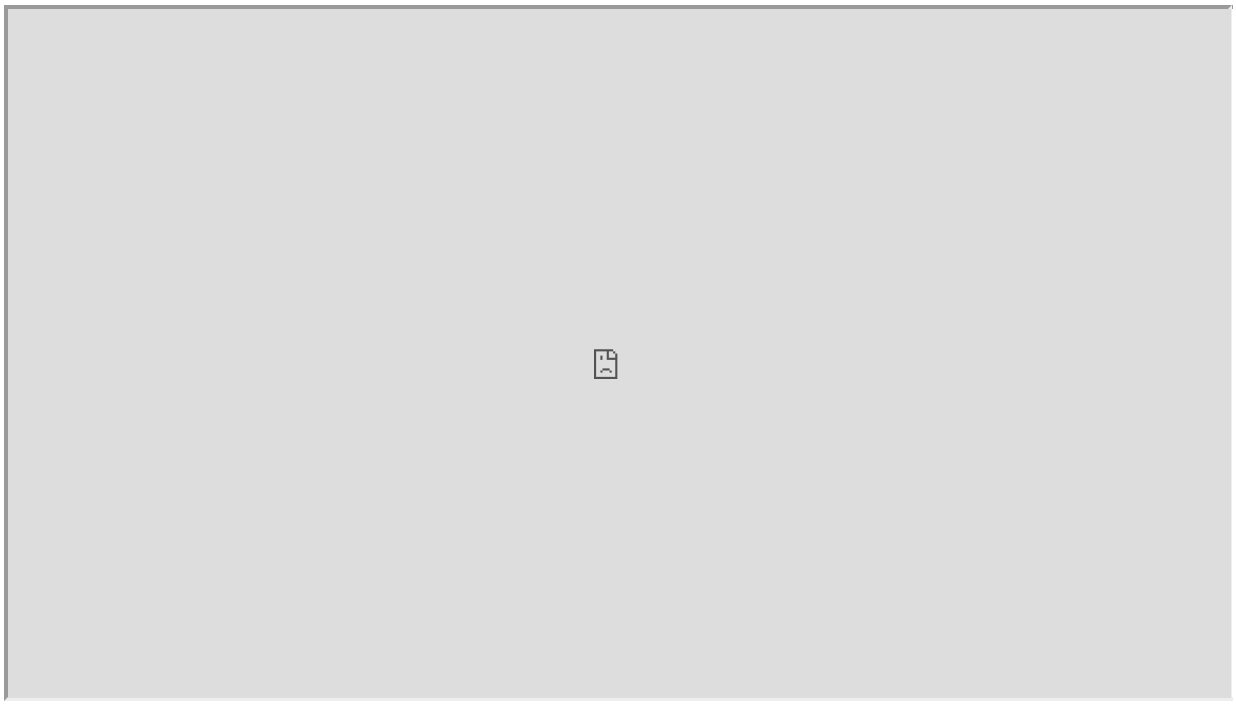
Owner

This constraint allows you to choose in which space evaluate its owner's transform properties. See common constraint properties for more information.

Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



Limit Scale Constraint

An object or bone can be scaled along the X, Y and Z axes. This constraint restricts the amount of allowed scaling along each axis, through lower and upper bounds.

Important

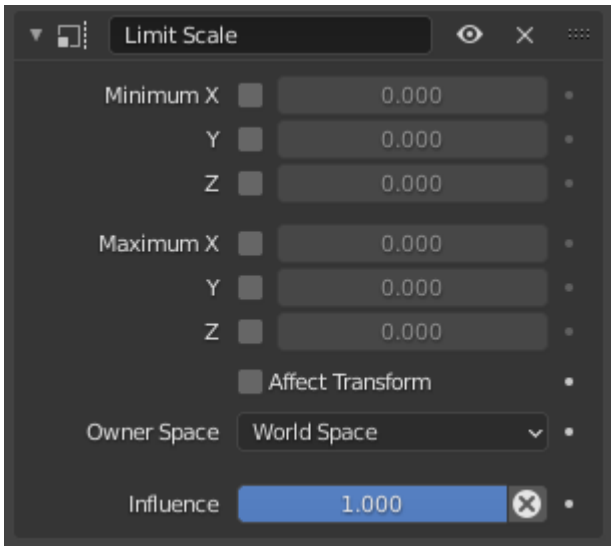
This constraint does not tolerate negative scale values (those you might use to mirror an object…): when you add it to an object or bone, even if no axis limit is enabled, nor the *Affect Transform* option, as soon as you scale your object, all negative scale values are instantaneously inverted to positive ones… And the boundary settings can only take strictly positive values.

It is interesting to note that even though the constraint limits the visual and rendered scale of its owner, its owner's data-block still allows (by default) the object or bone to have scale values outside the minimum and maximum ranges (as long as they remain positive!). This can be seen in its *Transform* panel. When an owner is scaled and attempted to be moved outside the limit boundaries, it will be constrained to those boundaries visually and when rendered, but internally, its coordinates will still be changed beyond the limits. If the constraint is removed, its ex-owner will seem to jump to its internally-specified scale.

Similarly, if its owner has an internal scale that is beyond the limits, scaling it back into the limit area will appear to do nothing until the internal scale values are back within the limit threshold (unless you enabled the *Affect Transform* option, see below, or your owner has some negative scale values).

Setting equal the min and max values of an axis locks the owner's scaling along that axis. Although this is possible, using the *Transformation Properties* axis locking feature is probably easier.

Options



Limit Scale panel.

Minimum/Maximum X, Y, Z

These buttons enable the lower boundary for the scale of the owner along respectively the X, Y and Z axes of the chosen *Space*. The *Min* and *Max* number fields to their right control the value of their lower and upper boundaries, respectively.

Note

If a min value is higher than its corresponding max value, the constraint behaves as if it had the same value as the max one.

Affect Transform

As pointed out before by default, even though visually constrained, and except for the negative values, the owner can still have scales out of bounds (as shown by the *Transform* panel). When you enable this checkbox, this is no longer possible, the owner transform properties are also limited by the constraint. However, note that, the constraint does not directly modify the scale values: you have to scale its owner one way or another for this to take effect.

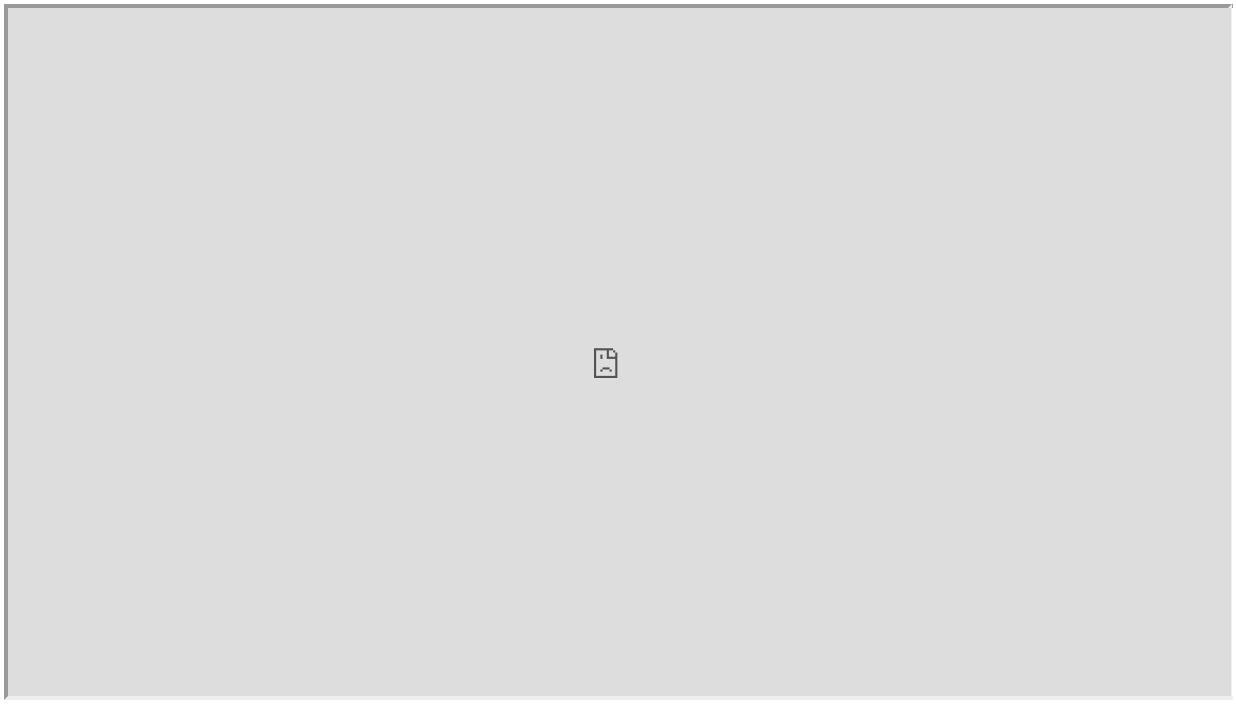
Owner

This constraint allows you to choose in which space to evaluate its owner's transform properties. See common constraint properties for more information.

Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



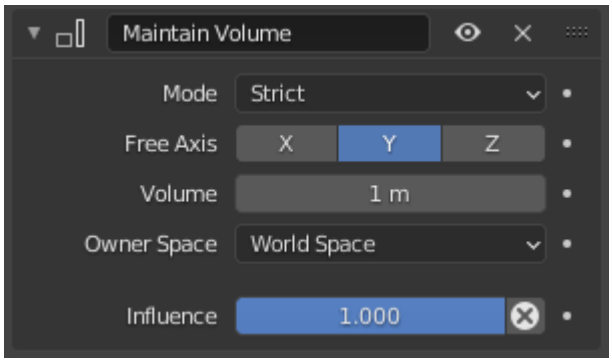
Maintain Volume Constraint

The *Maintain Volume* constraint limits the volume of a mesh or a bone to a given ratio of its original volume.

See also

Harkyman on the development of the Maintain Volume constraint.

Options



Maintain Volume Constraint.

Mode
: Specifies how the constraint handles scaling of the non-free axes.

  Strict
  : This mode overrides non-free axis scaling to strictly maintain the specified volume. Only the ratio between the scale of the non-free axes is passed through.

  Uniform
  : This mode maintains the volume as specified only when the pre-constraint scaling is uniform. Deviations from uniform scaling on

non-free axes are passed through.

Single Axis
: This mode maintains the volume only when the object is scaled just on its free axis. Any additional non-free axis scaling is passed through.

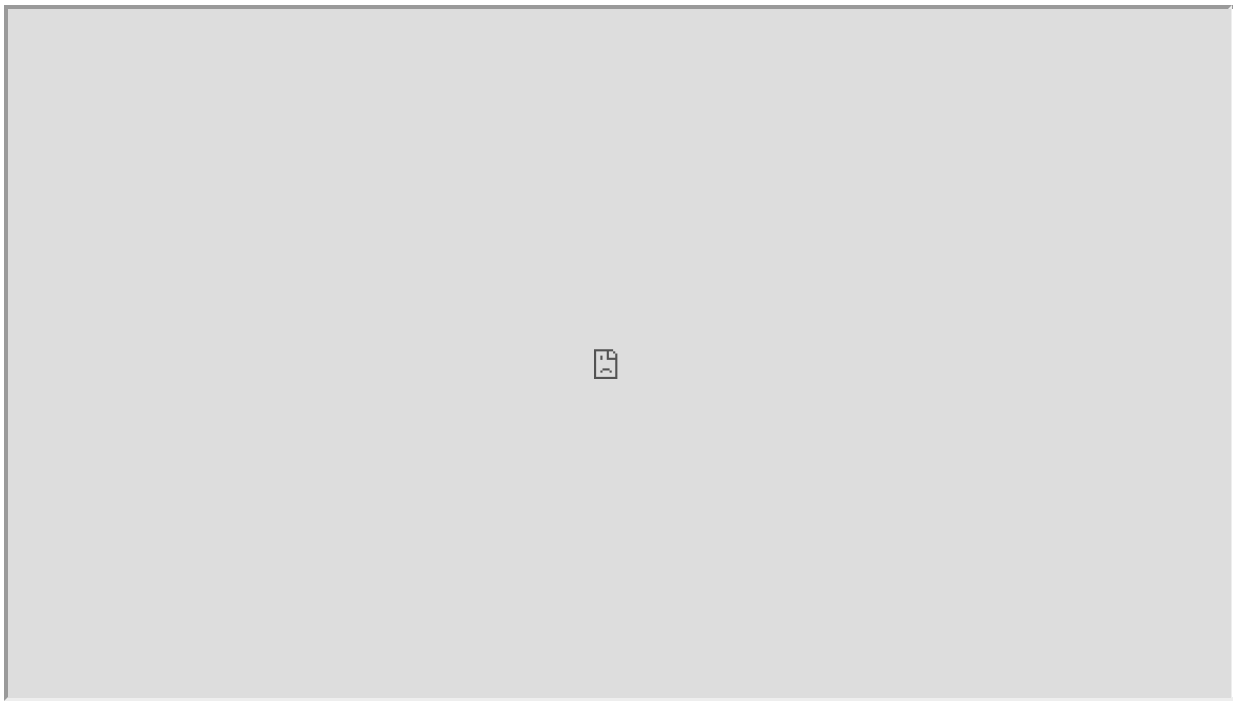
Free Axis
: The free-scaling axis of the object.

Volume
: The bone's rest volume.

Owner
: This constraint allows you to choose in which space to evaluate its owner's transform properties. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example

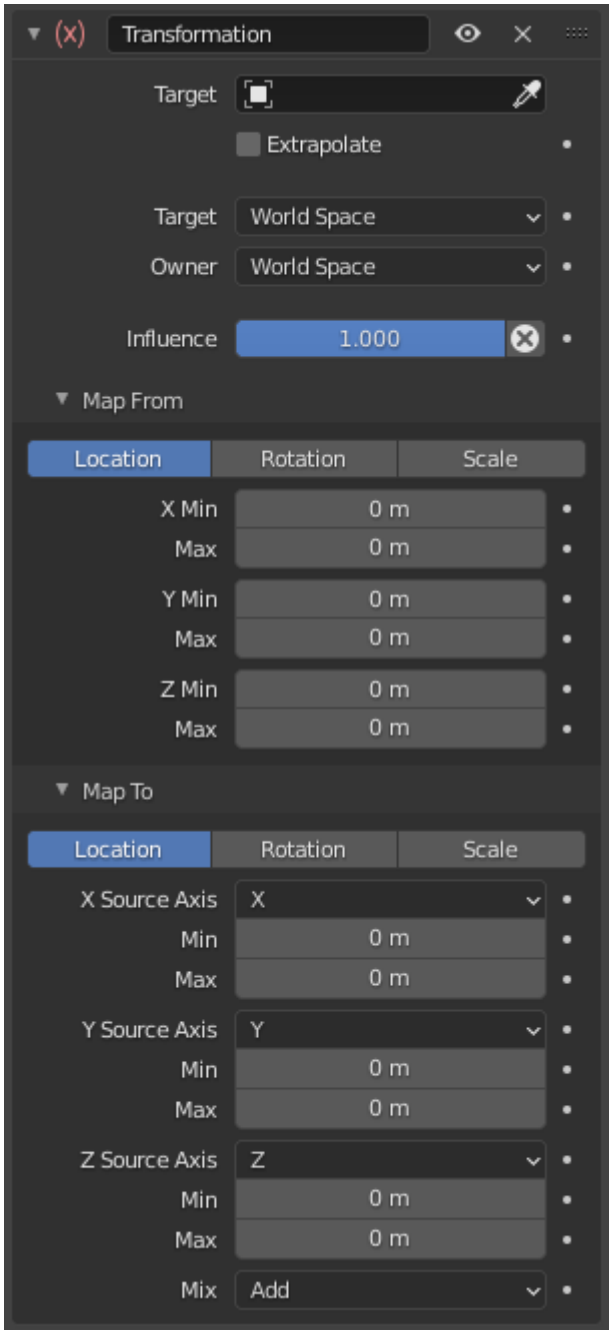


Transformation Constraint

This constraint is more complex and versatile than the other “transform” constraints. It lets you set the location, rotation or scale of an object/bone based on the location, rotation or scale of another, mixing and matching axes as you see fit. An example could be to set a gear’s X rotation based on the Y coordinate of a rail next to it.

The constraint works with input and output value ranges, one for each axis. It first clamps the input value to the *Map From* range, then offsets and scales it to the corresponding *Map To* range. This lets you, say, map a Y coordinate in the range (-3m, 3m) to an X rotation in the range (0, 180°).

Options



Transformation panel.

Target

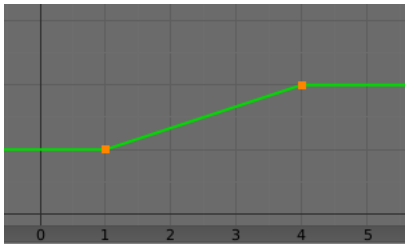
The reference object to read a transformation property from. If you don’t select one, the constraint’s icon will turn red and it will have no effect.

See common constraint properties for more information.

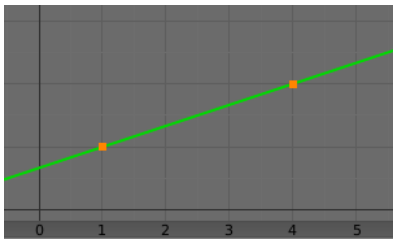
Bone
: If *Target* is an Armature, you can optionally choose a bone here to use the transformation of that bone instead of the armature object as a whole.

Extrapolate
: By default, the input and output values are clamped to the *Min*/*Max* values. When you enable *Extrapolate*, they're allowed to go beyond these limits. This is illustrated with the graphs below, where the X axis represents the input (*Map From* set to *Min* = 1 and *Max* = 4) and the Y axis represents the output (*Map To* set to *Min* = 1 and *Max* = 2).

The Extrapolate option.



Extrapolate disabled: the output values are limited to the Map To range.



Extrapolate enabled: the output values can extend beyond the limits.

Target/Owner
: Standard conversion between spaces. See common constraint properties for more information.

Influence
: Controls the percentage of effect the constraint has on the object. See common constraint properties for more information.

Map From

The transformation to read from the *Target* (or *Bone*).

Location, Rotation, Scale
: The type of transformation to read.

Mode *Rotation*
: The type of rotation to use, including different Euler orders, Quaternion, and other Rotation Channel Modes. Defaults to using the Euler order of the constraint owner.

  In the *Quaternion* mode, the channels are converted to weighted angles in the same way as the swing angles of the Swing and X/Y/Z Twist modes.

X/Y/Z Min, Max
: The input value range for each axis.

Map To

The transformation to apply to the owner.

Location, Rotation, Scale
: The type of transformation to apply.

Order *Rotation*
: Which Euler order to use. Defaults to the order of the constraint owner.

X/Y/Z Source Axis
: For each of the three output axes, lets you choose the input axis that it should take its value from. You can select the same input axis multiple times.

Min, Max
: The output value range for each axis.

Mix
: Specifies how the result of the constraint is combined with the existing transformation. The set of available choices varies based on the type of transformation.

Replace
: The result of the constraint replaces the existing transformation.

Multiply *Scale*
: The new values are multiplied with the existing axis values.

Add *Location Rotation*
: The new values are added to the existing axis values.

Before Original *Rotation*
: The new rotation is added before the existing rotation, as if it were applied to a parent of the constraint owner.

After Original *Rotation*
: The new rotation is added after the existing rotation, as if it were applied to a child of the constraint owner.

Note

- For historical reasons, the *Mix* mode defaults to *Add* for location and rotation, and *Replace* for scale.

- When using the rotation of the target as input, whatever the real values are, the constraint will always "take them back" into the (-180 to 180) range. E.g. if the target has a rotation of 420 degrees around its X axis, the values used as *X* input by the constraint will be:

  $$((420 + 180) \text{ modulo } 360) - 180 = 60 - 180 = -120$$

  As such, this constraint is not really suited for transforming an object based on a gear's rotation. Rotating a gear based on an object's transformation works fine, however.

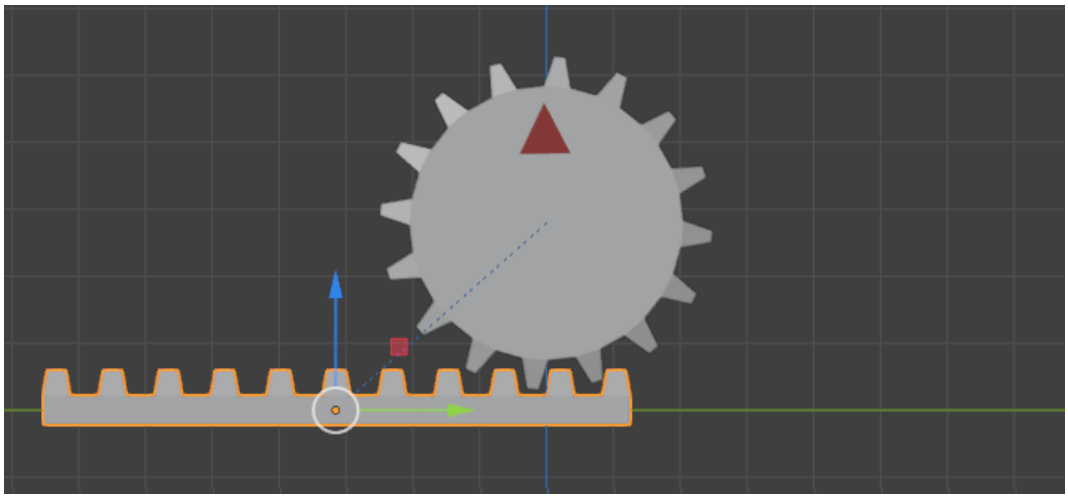
- Similarly, when using the scale transform properties of the target as input, whatever the real values are, the constraint will always take their absolute values (i.e. invert negative ones).

- When a *Min* value is higher than its corresponding *Max* one, both are considered equal to the *Max* one. This means you cannot create “reversed” mappings.

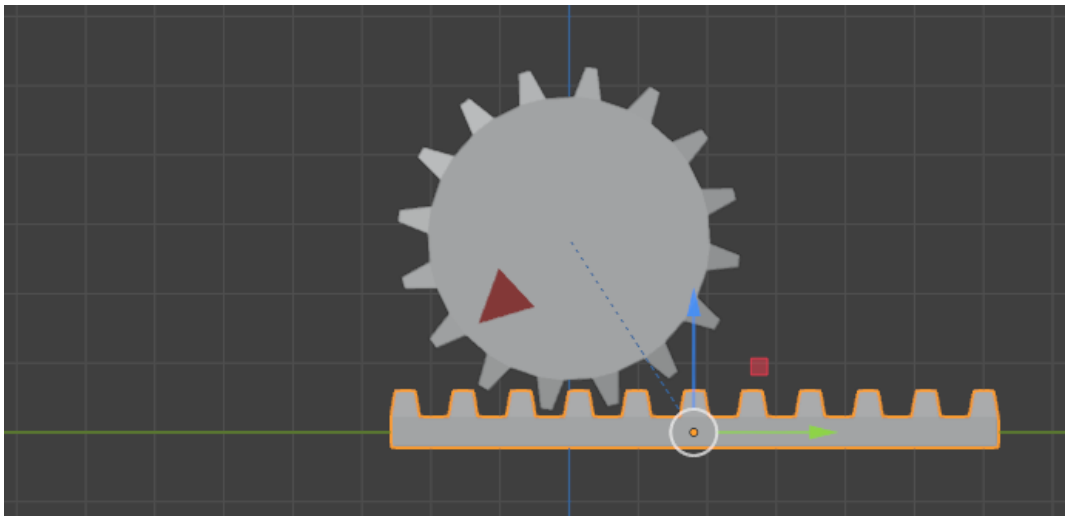
Example

In the following example, we add a constraint to a gear that sets its X rotation based on the Y position of a rail:

- Target: Rail object
- Map From: Location
  - Y Min: -3m
  - Y Max: 3m
- Map To: Rotation
  - X Source Axis: Y
  - X Min: 0°
  - X Max: 180°



Before moving the rail.



After moving the rail.

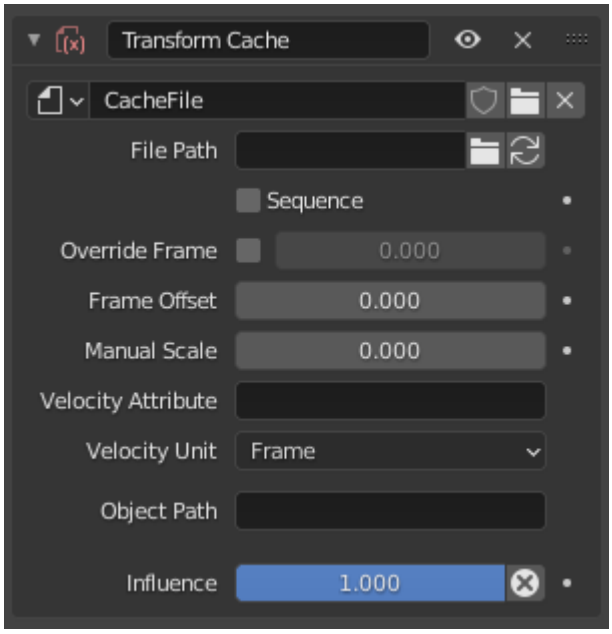
By default, the gear will stop rotating if the rail goes outside the (-3m, 3m) range. You can enable *Extrapolate* to change this.

Transform Cache Constraint

The *Transform Cache Constraint* is used to stream animations from Alembic or USD made at the transformation matrix level (for example rigid bodies, or camera movements).

When importing an Alembic or USD file, Transform Cache constraints are automatically added to objects with animated transforms. For time-varying meshes (so deforming animations), the Mesh Sequence Cache modifier is used.

Options



Transform Cache Constraint.

Cache File
: Data-block menu to select the Alembic or USD file.

File Path
: Path to the Alembic or USD file.

Sequence
: Whether or not the cache is separated in a series of files.

Override Frame
: Whether to use a custom frame for looking up data in the cache file, instead of using the current scene frame.

  The *Frame* value is the time to use for looking up the data in the cache file, or to determine which to use in a file sequence.

Frame Offset
: Subtracted from the current frame to use for looking up the data in the cache file, or to determine which file to use in a file sequence.

Manual Scale
: Value by which to enlarge or shrink the object with respect to the world's origin.

Velocity Attribute
: The name of the Alembic attribute used for generating motion blur data; by default, this is `.velocities` which is standard for most Alembic files.

  Note

  The *Velocity Attribute* option is currently for Alembic files only.

Velocity Unit
: Defines how the velocity vectors are interpreted with regard to time.

  Frame
  : The velocity unit was encoded in frames and does not need to be scaled by scene FPS.

  Second
  : The velocity unit was encoded in seconds and needs to be scaled by the scene FPS (1 / FPS).

Note

The *Velocity Unit* option is currently for Alembic files only.

Object Path

The path to the Alembic or USD object inside the archive or stage.

Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Tracking Constraints

- Clamp To Constraint
  - Options
  - Example
- Damped Track Constraint
  - Options
  - Example
- Inverse Kinematics Constraint
  - Options
  - Example
- Locked Track Constraint
  - Options
  - Example
- Spline IK Constraint
  - Options
  - Example
- Stretch To Constraint
  - Options
  - Example
- Track To Constraint
  - Options
  - Example

Clamp To Constraint

The *Clamp To* constraint clamps an object to a curve. The *Clamp To* constraint is very similar to the Follow Path constraint, but instead of using the evaluation time of the target curve, *Clamp To* will get the actual location properties of its owner (those shown in the *Transform* panel), and judge where to put it by "mapping" this location along the target curve.

One benefit is that when you are working with *Clamp To*, it is easier to see what your owner will be doing; since you are working in the 3D Viewport, it will just be a lot more precise than sliding keys around on an F-Curve and playing the animation over and over.

A downside is that unlike in the Follow Path constraint, *Clamp To* does not have any option to track your owner's rotation (pitch, roll, yaw) to the banking of the targeted curve, but you do not always need rotation on, so in cases like this it's usually a lot handier to fire up a *Clamp To*, and get the bits of rotation you do need some other way.

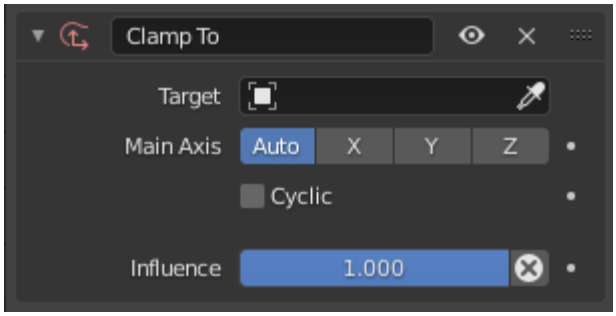
The mapping from the object's original position to its position on the curve is not perfect, but uses the following simplified algorithm:

- A "main axis" is chosen, either by the user, or as the longest axis of the curve's bounding box (the default).
- The position of the object is compared to the bounding box of the curve in the direction of the main axis. So for example if X is the main axis, and the object is aligned with the curve bounding box's left side, the result is 0; if it is aligned with the right side, the result is 1.
- If the cyclic option is unchecked, this value is clamped in the range 0-1.
- This number is used as the curve time, to find the final position along the curve that the object is clamped to.

This algorithm does not produce exactly the desired result because curve time does not map exactly to the main axis position. For example an object

directly in the center of a curve will be clamped to a curve time of 0.5 regardless of the shape of the curve, because it is halfway along the curve's bounding box. However, the 0.5 curve time position can actually be anywhere within the bounding box!

Options



Clamp To panel.

Target

The target Data ID indicates which curve object the Clamp To constraint will track along. The target must be a curve object type. See common constraint properties for more information.

Main Axis

This button group controls which global axis (X, Y or Z) is the main direction of the path. When clamping the object to the target curve, it will not be moved significantly on this axis. It may move a small amount on that axis because of the inexact way this constraint functions.

For example if you are animating a rocket launch, it will be the Z axis because the main direction of the launch path is up. The default *Auto* option chooses the axis which the curve is longest in (or X if they are equal). This is usually the best option.

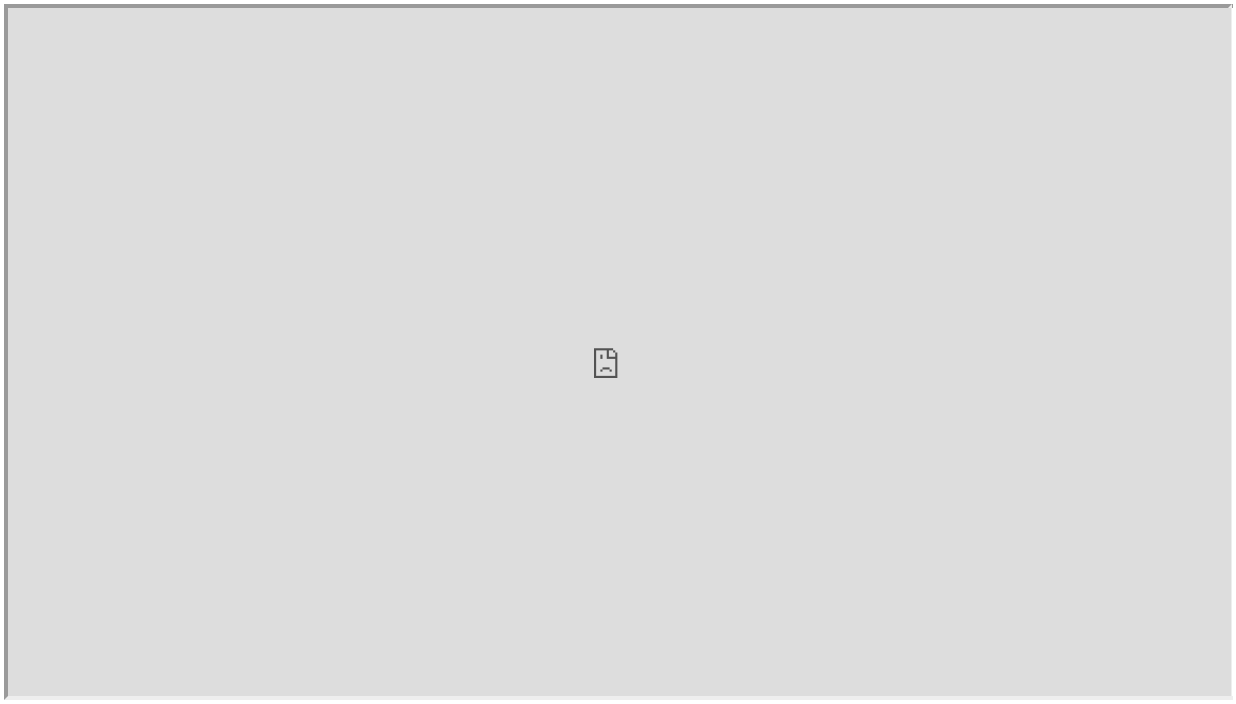
Cyclic

By default, once the object has reached one end of its target curve, it will be constrained there. When the *Cyclic* option is enabled, as soon as it reaches one end of the curve, it is instantaneously moved to its other

end. This is of course primarily designed for closed curves (e.g. circles), as this allows your owner to go around it over and over.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

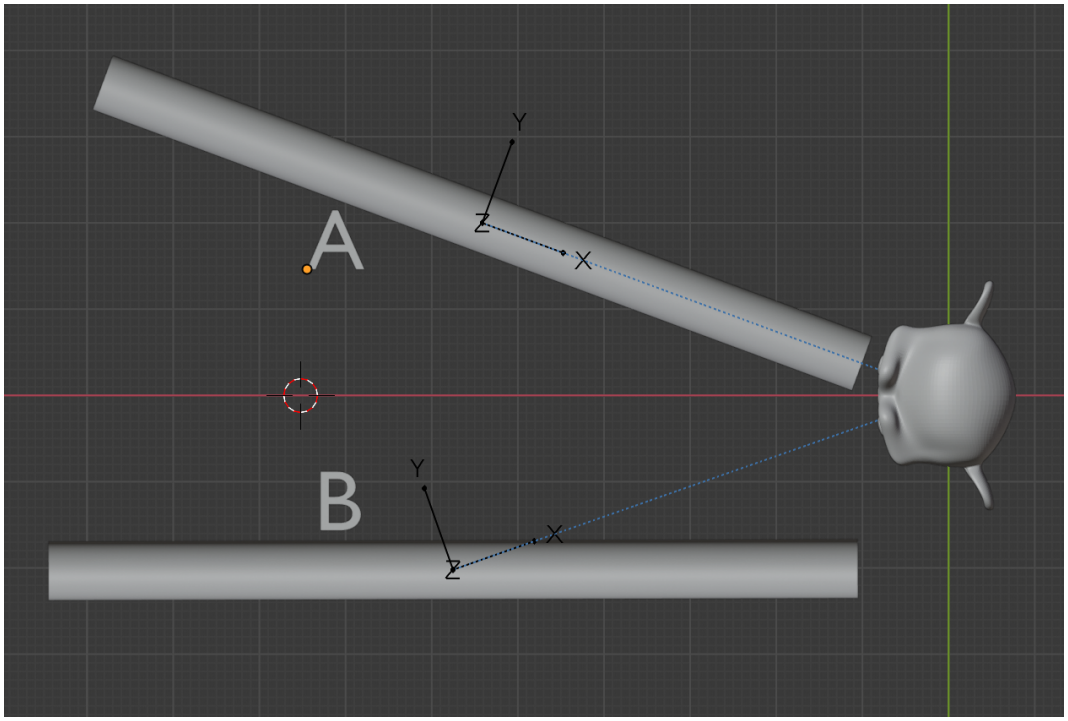
Example



Damped Track Constraint

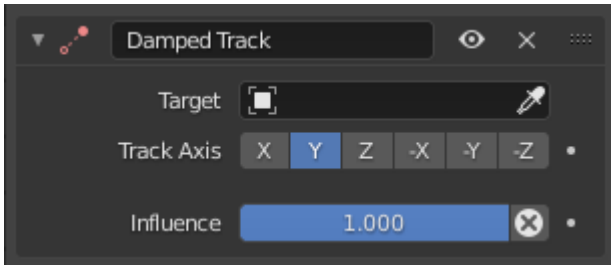
The *Damped Track* constraint constrains one local axis of the owner to always point towards *Target*. This constraint uses a pure Swing rotation, i.e. the shortest possible single axis rotation. In other 3D software you can find it with the name “Look at” constraint.

Although usually associated with bones, Damped Track can align objects to point to (and follow) other objects or bones. It is important to note that the constraint aligns the origin’s axes to point to the target’s origin point. This is illustrated in the following figure. In each case the objects are set as Damped Track to +X.



A: Object vertices aligned along axis origin. B: Object vertices aligned away from origin.

Options



Damped Track panel.

Target

[Data ID] used to select the constraint's target, and is not functional (red state) when it has none. See [common constraint properties] for more information.

Track Axis

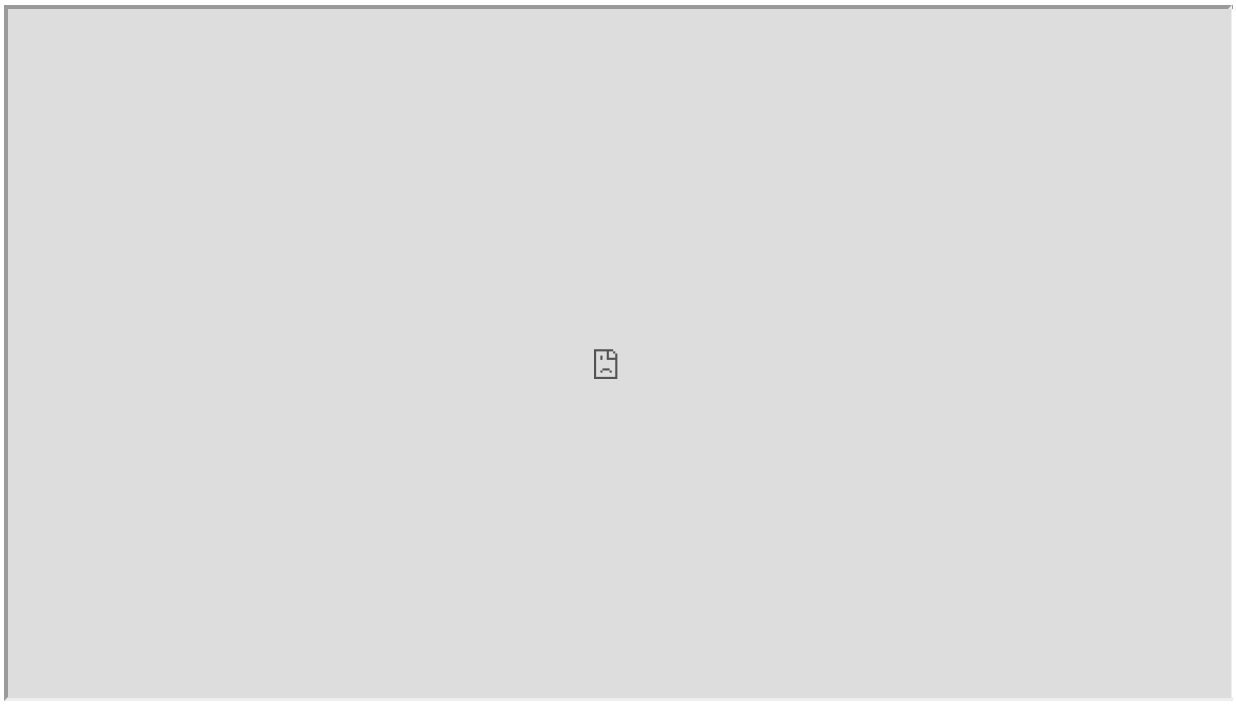
Once the owner object has had a Damped Track constraint applied to it, you must then choose which axis of the object you want to point at the Target object. The negative axis direction cause the object to point away from the Target object along the selected axis direction.

-X, -Y, -Z, X, Y, Z

Influence

Controls the percentage of affect the constraint has on the object. See [common constraint properties] for more information.

Example



Inverse Kinematics Constraint

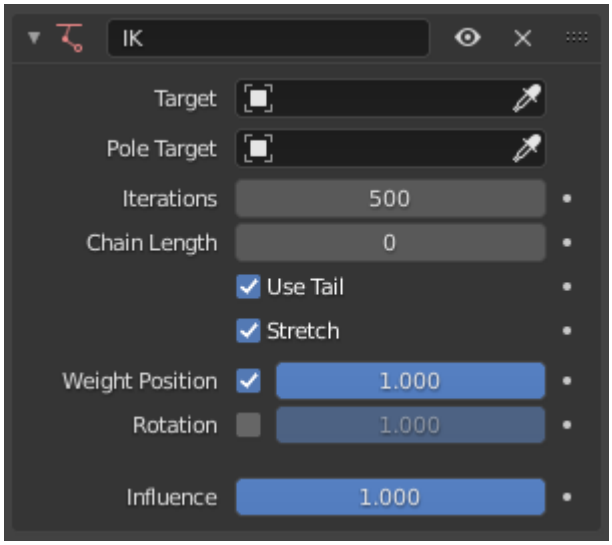
The *Inverse Kinematics* constraint implements the *inverse kinematics* armature posing technique. Hence, it is only available for bones. To quickly create an IK constraint with a target, select a bone in Pose Mode, and press `Shift-I`.

This constraint is fully documented in the Inverse Kinematics page, part of the rigging chapter.

Note

The IK constraints are special in that they modify multiple bones. For this reason, they ignore their position in the stack and always run after all other constraints on the affected bones. To apply constraints after IK, it is necessary to first copy the final transformation to a new bone chain, e.g. using Copy Transforms.

Options



Inverse Kinematics panel.

Target
: Data ID used to select the an armature. See common constraint properties for more information.

Pole Target
: Object for pole rotation.

Iterations
: Maximum number of solving iterations.

Chain Length
: How many bones are included in the IK effect. Set to 0 to include all bones.

Use Tail
: Include bone's tail as last element in chain.

Stretch
: Enable IK stretching.

Weight Position
: For Tree-IK: Weight of position control for this target.

Rotation
: Chain follows rotation of target.

Target
: Disable for targetless IK.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

iTaSC Solver

If the iTaSC IK Solver is used, the IK Solver Constraint changes to add these addition parameters.

IK Type

Copy Pose

Equivalent to the traditional end effector position and orientation constraint: the end effector is constrained to take the position, and optionally the orientation, of a given target, which is set in the target field.

Position/Rotation Locking

Allows to obtain various effect by not constraining the coordinates along certain axis.

Axis Reference

Specifies how to compute the axis coordinates.

Bone

The coordinates are the position and orientation of the target relative to the bone.

Target

The opposite of *Bone*, the coordinates are the position and orientation of the tip of the bone relative to the target.

Distance

Specify that the end effector will stay inside, at, or outside a sphere centered on the target object.

Limit Mode

Inside

The end effector will stay inside of the distance from the target object.

Outside

The end effector will stay outside of the distance from the target object.

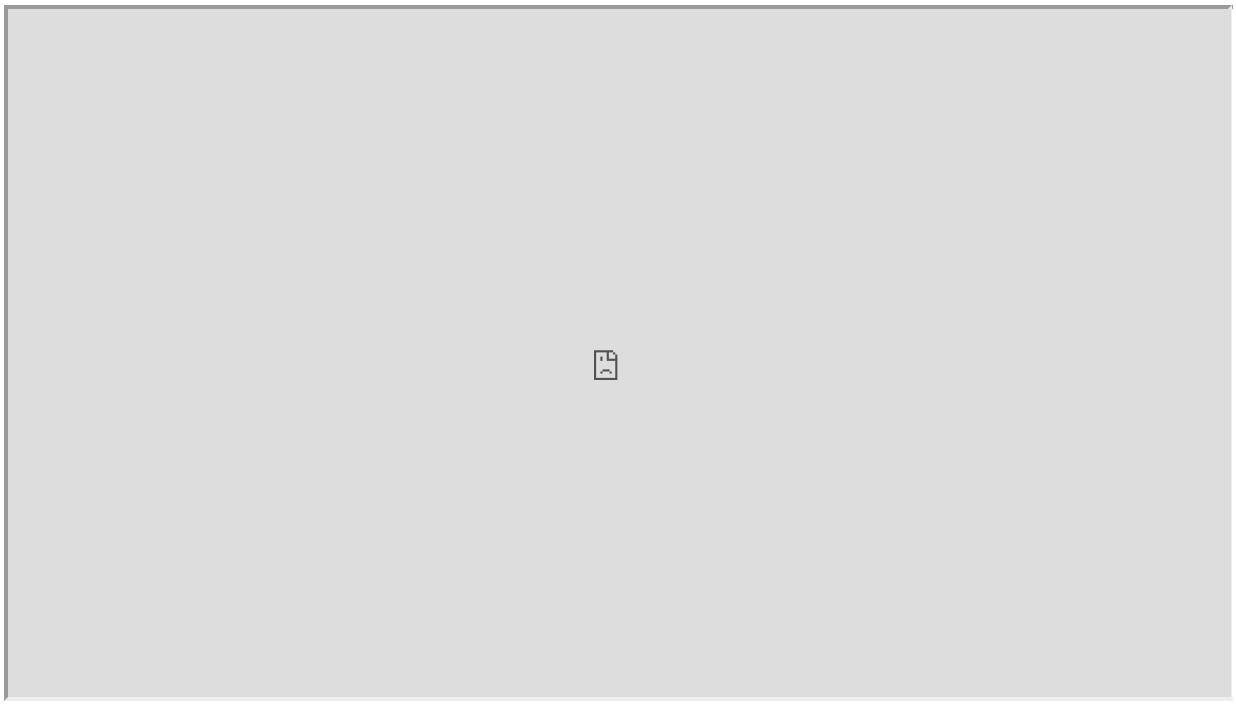
On Surface
The end effector will stay exactly at the distance from the target object.

Distance
The radius from the target object.

Note

The *Influence* parameter is not implemented if *Pole Target* is used.

Example



Locked Track Constraint

The *Locked Track* constraint is a bit tricky to explain, both graphically and textual. Basically, it is a Track To constraint, but with a locked axis, i.e. an axis that cannot rotate (change its orientation). Hence, the owner can only track its target by rotating around this axis, and unless the target is in the plane perpendicular to the locked axis, and crossing the owner, this owner cannot really point at its target.

Let us take the best real-world equivalent: a compass. It can rotate to point in the general direction of its target (the magnetic North, or a neighbor magnet), but it cannot point *directly at it*, because it spins like a wheel on an axle. If a compass is sitting on a table and there is a magnet directly above it, the compass cannot point to it. If we move the magnet more to one side of the compass, it still cannot point *at* the target, but it can point in the general direction of the target, and still obey its restrictions of the axle.

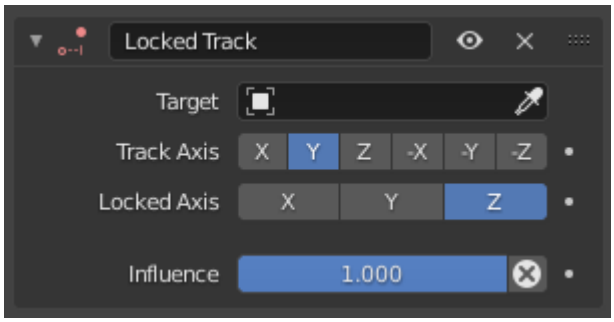
When using a *Locked Track* constraint, you can think of the target as a magnet, and the owner as a compass. The *Lock* axis will function as the axle around which the owner spins, and the *To* axis will function as the compass' needle. Which axis does what is up to you!

If you have trouble understanding the buttons of this constraint, read the tooltips, they are pretty good. If you do not know where your object's axes are, enable *Axis* in *Properties* ▸ *Armature* ▸ *Viewport Display*. Or, if you are working with bones, turn on the *Axes* in the armature's *Viewport Display* panel.

This constraint was designed to work cooperatively with the *Track To* constraint. If you set the axes buttons right for these two constraints, *Track To* can be used to point the axle at a primary target, and *Locked Track* can spin the owner around that axle to a secondary target.

This constraint also works very well for 2D billboarding.

Options



Locked track panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Track Axis
: The tracking local axis, i.e. the owner's axis to point at the target. The negative options force the relevant axis to point away from the target.

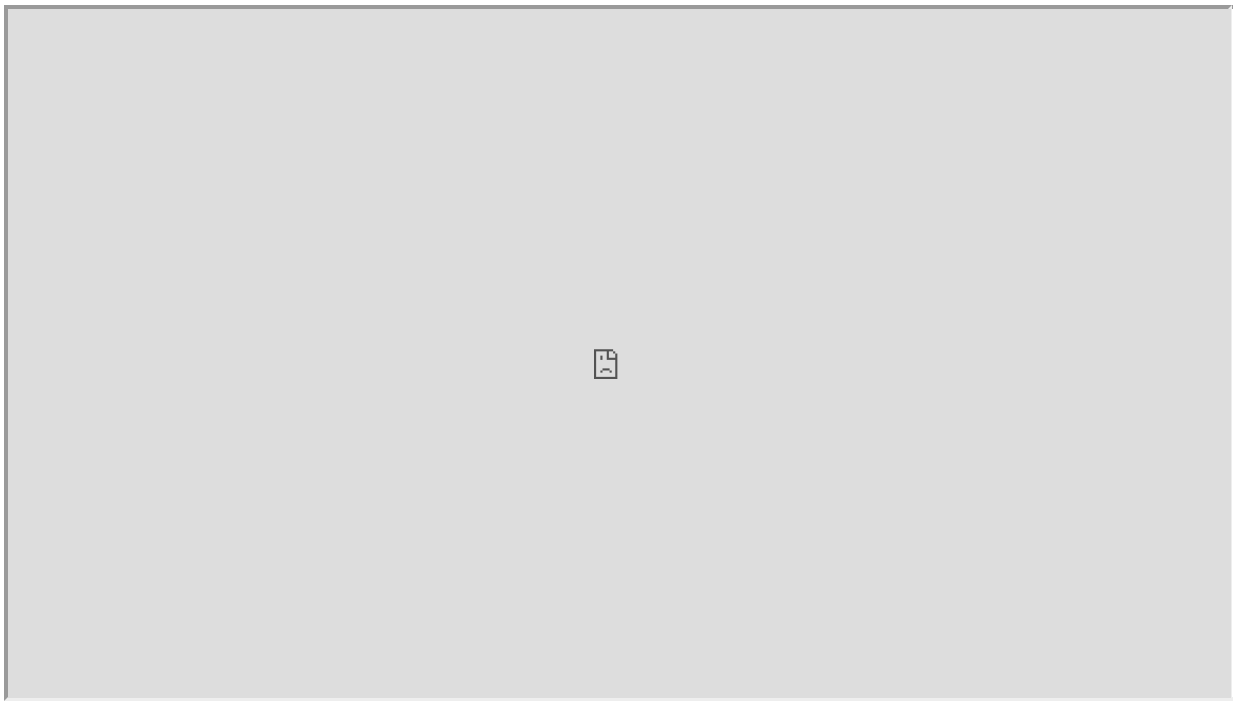
Locked Axis
: The locked local axis, i.e. the owner's axis which cannot be re-oriented to track the target.

Important

If you choose the same axis for *Tracking Axis* and *Locked Axis*, the constraint will no longer be functional (red state).

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



Spline IK Constraint

The *Spline IK* constraint aligns a chain of bones along a curve. By leveraging the ease and flexibility of achieving aesthetically pleasing shapes offered by curves and the predictability and well-integrated control offered by bones, *Spline IK* is an invaluable tool in the riggers' toolbox. It is particularly well suited for rigging flexible body parts such as tails, tentacles, and spines, as well as inorganic items such as ropes.

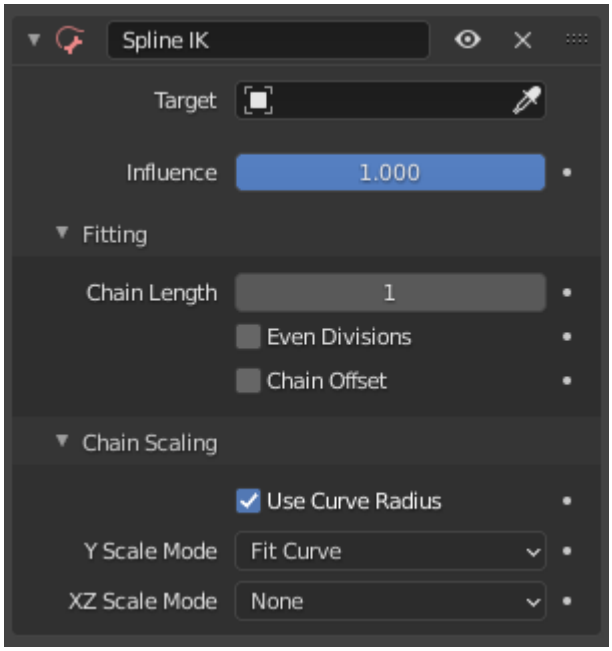
To set up *Spline IK*, it is necessary to have a chain of connected bones and a curve to constrain these bones to:

- With the last bone in the chain selected, add a *Spline IK* constraint from the *Bone Constraints* tab in the *Properties*.
- Set the 'Chain Length' setting to the number of bones in the chain (starting from and including the selected bone) that should be influenced by the curve.
- Finally, set *Target* to the curve that should control the curve.

Note

The IK constraints are special in that they modify multiple bones. For this reason, they ignore their position in the stack and always run after all other constraints on the affected bones. To apply constraints after IK, it is necessary to first copy the final transformation to a new bone chain, e.g. using Copy Transforms.

Options



Spline IK panel.

Target
: Data ID used to select the target curve. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Fitting

Chain Length
: How many bones are included in the chain.

Even Division
: Ignore the relative length of the bones when fitting to the curve.

Chain Offset
: Retain the offset of the root joint from the start point of the curve.

Chain Scaling

Use Curve Radius
: Average radius of the endpoints is used to tweak the X and Z scaling of the bones, on top of the X and Z scale mode.

Y Scale Mode
: Specifies how the length of a bone is scaled when it is fitted to the curve, in addition to the effects of target curve object scale and curvature.

  None
  : The bone is reset to its rest pose length.

  Fit Curve
  : The bones are stretched to cover the entire length of the curve.

  Bone Original
  : The original Y axis scale of the bone is used.

XZ Scale Mode
: Scaling that a bone undergoes in the other two directions.

  None
  : Do not scale the X and Z axes.

  Bone Original
  : Use the original scaling of the bones.

  Inverse Scale
  : Scale of the X and Z axes is the inverse of the Y scale.

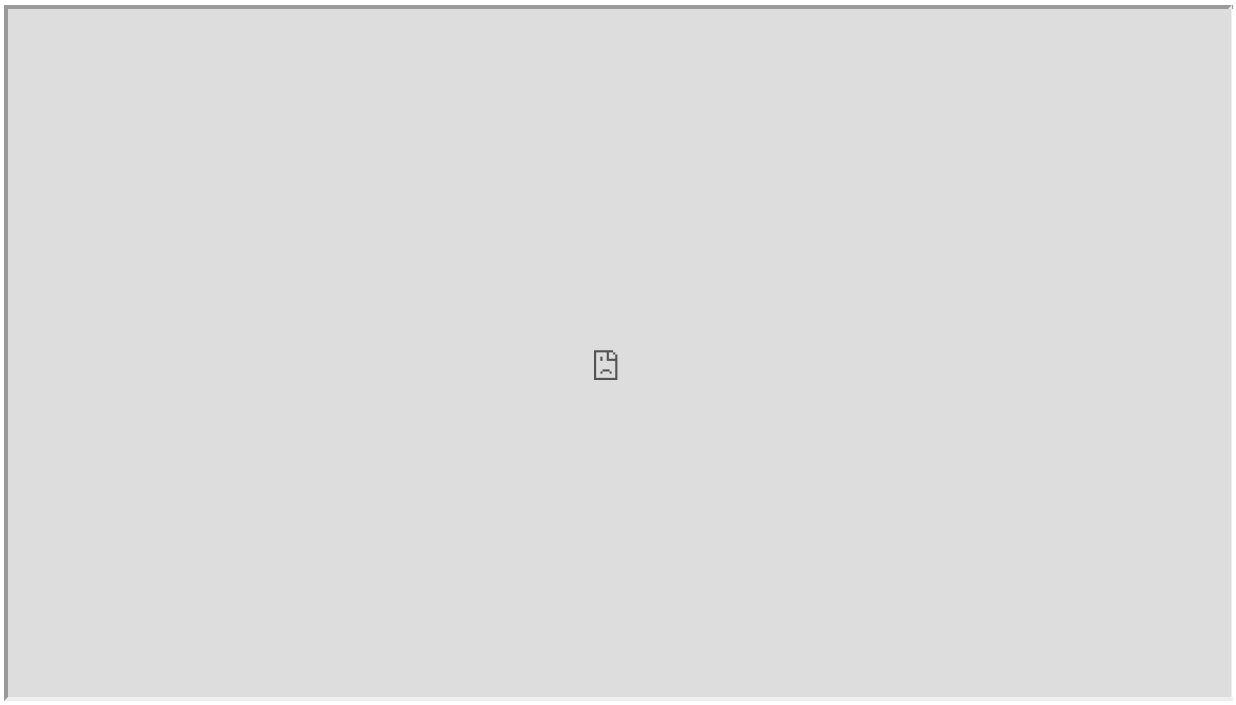
  Volume Preservation
  : Similar to the Stretch To constraint.

Use Original Scale
: Specifies that *Inverse Scale* or *Volume Preservation* should be applied on top of the original scaling of the bones, like in the Stretch To constraint.

See also

This subject is seen in-depth in the Armature Posing section.

Example



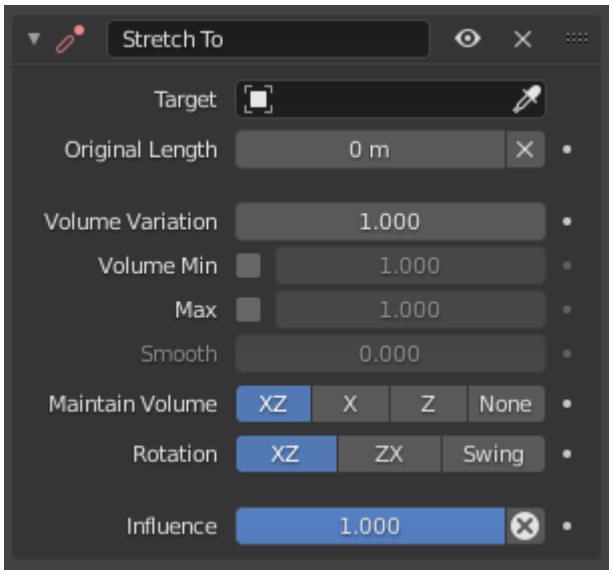
Stretch To Constraint

The *Stretch To* constraint causes its owner to rotate and scale its Y axis towards its target. So it has the same tracking behavior as the Track To constraint. However, it assumes that the Y axis will be the tracking and stretching axis, and does not give you the option of using a different one.

It also optionally has some raw volumetric features, so the owner can squash down as the target moves closer, or thin out as the target moves farther away. Note however, that it is not the real volume of the owner which is thus preserved, but rather the virtual one defined by its scale values. Hence, this feature works even with non-volumetric objects, like empties, 2D meshes or surfaces, and curves.

With bones, the "volumetric" variation scales them along their own local axes (remember that the local Y axis of a bone is aligned with it, from root to tip).

Options



Stretch To panel.

Target

Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Original Length

This number field sets the rest distance between the owner and its target, i.e. the distance at which there is no deformation (stretching) of the owner.

(Reset Original Length)

When clicked, this small button will recalculate the *Rest Length* value, so that it corresponds to the actual distance between the owner and its target (i.e. the distance before this constraint is applied).

Volume Variation

This number field controls the amount of "volume" variation exponentially to the stretching amount. Note that the 0.0 value is not allowed, if you want to disable the volume feature, use the *None* button (see below).

Volume Min, Max

Limits for the volume preservation to a minimum and maximum scaling each by a *Bulge* factor.

Smooth

Smoothness factor to make limits less visible.

Maintain Volume

These buttons control which of the X and/or Z axes should be affected (scaled up/down) to preserve the virtual volume while stretching along the Y axis. If you enable the *none* button, the volumetric features are disabled.

Rotation

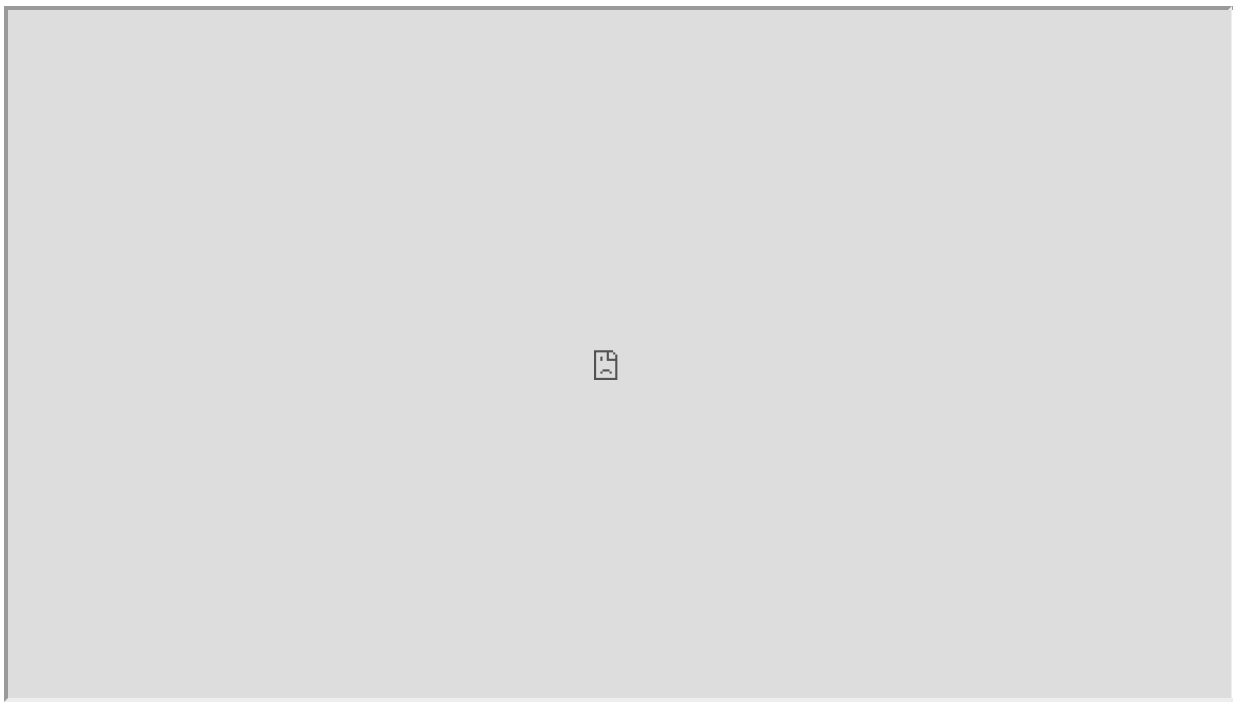
Specifies how the owner should be rotated to track the target with its Y axis.

XZ, ZX
: These buttons are equivalent to the *Up* ones of the Track To constraint: the owner is rotated around two local axes in the specified order.

Swing
: The constraint uses a single Swing rotation, equivalent to the Damped Track constraint.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



Track To Constraint

The *Track To* constraint applies rotations to its owner, so that it always points a given “To” axis towards its target, with another “Up” axis permanently maintained as much aligned with the global Z axis (by default) as possible. This tracking is similar to the “billboard tracking” in 3D (see note below).

This is the preferred tracking constraint, as it has a more easily controlled constraining mechanism.

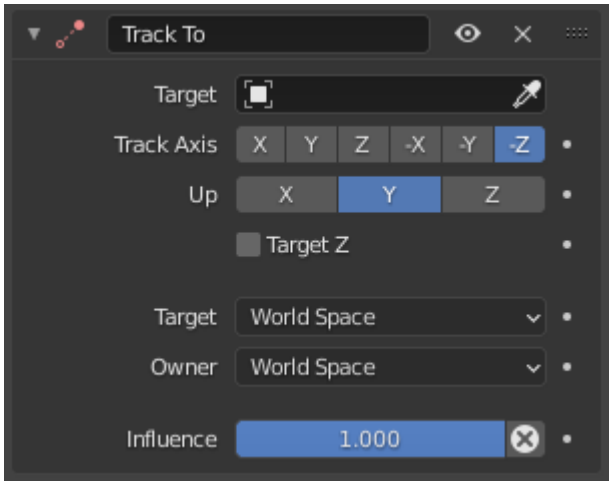
This constraint shares a close relationship to the Inverse Kinematics constraint in some ways.

Tip

Billboard Tracking

The term “billboard” has a specific meaning in real-time CG programming (i.e. video games!), where it is used for plane objects always facing the camera (they are indeed “trackers”, the camera being their “target”). Their main usage is as support for tree or mist textures: if they were not permanently facing the camera, you would often see your trees squeezing to nothing, or your mist turning into a mille-feuille paste, which would be funny but not so credible.

Options



Track To panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Track Axis
: The tracking local axis, i.e. the owner's axis to point at the target. The negative options force the relevant axis to point away from the target.

Up
: The "upward-most" local axis, i.e. the owner's axis to be aligned (as much as possible) with the global Z axis (or target Z axis, when the *Target* button is enabled).

Target Z
: By default, the owner's *Up* axis is (as much as possible) aligned with the global Z axis, during the tracking rotations. When this button is enabled, the *Up* axis will be (as much as possible) aligned with the target's local Z axis?

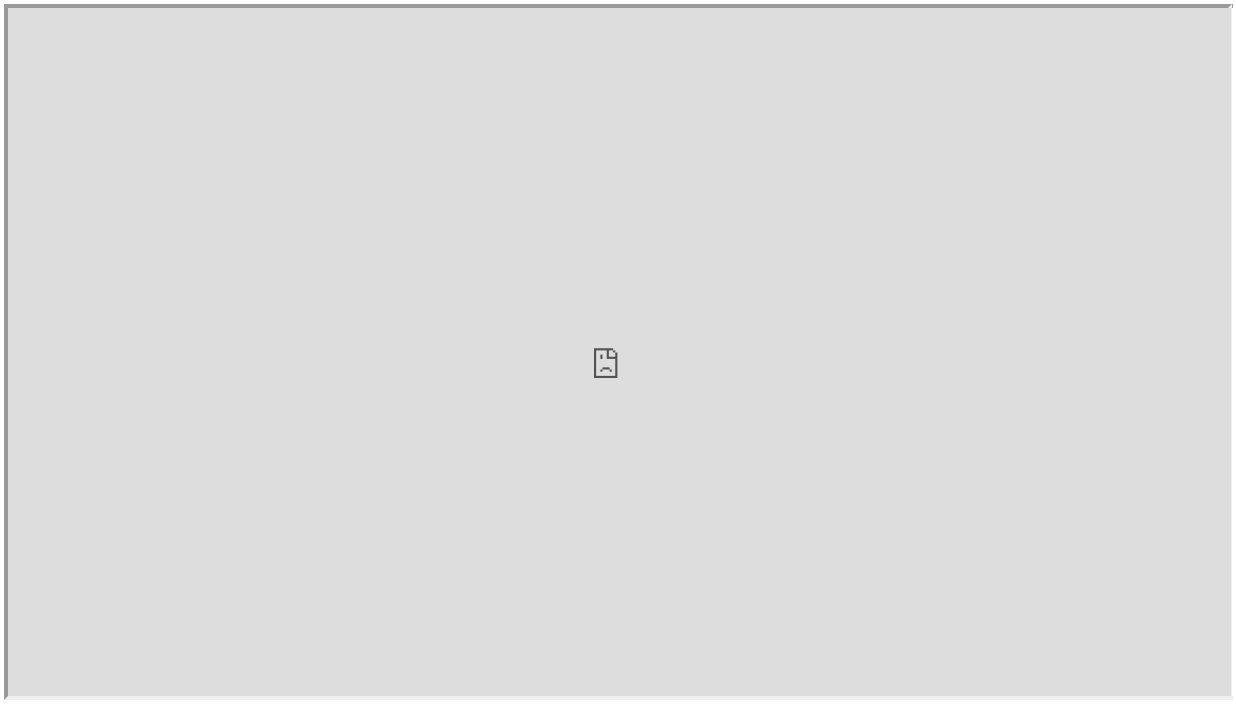
Target/Owner
: Standard conversion between spaces. See common constraint properties for more information.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Warning

If you choose the same axis for *Tracking Axis* and *Up*, the constraint will not be functional anymore (red state).

Example



Relationship Constraints

- Action Constraint
  - Options
  - Notes
  - Example
- Armature Constraint
  - Options
- Child Of Constraint
  - Options
  - Tips
  - Technical Note
  - Examples
- Floor Constraint
  - Options
  - Example
- Follow Path Constraint
  - Options
  - Example
- Pivot Constraint
  - Options
  - Example
- Shrinkwrap Constraint
  - Options
  - Example

Action Constraint

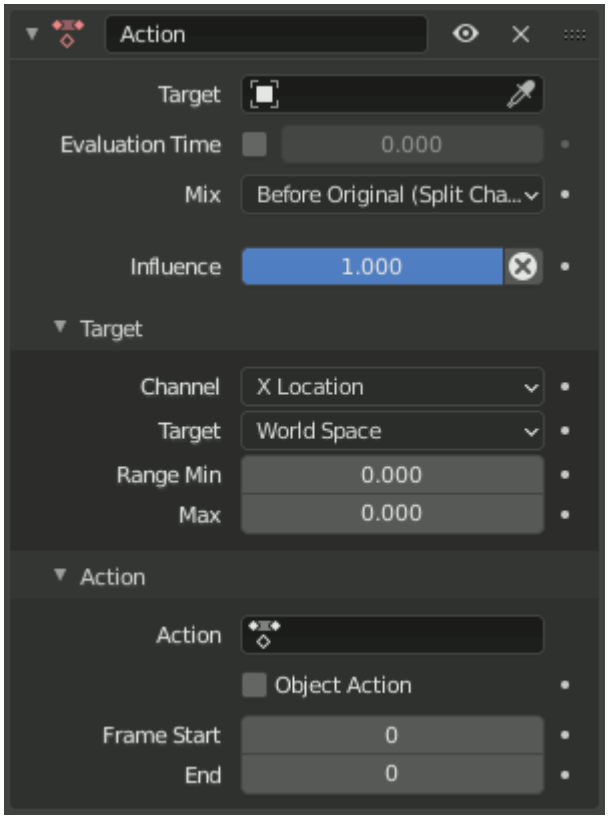
The *Action* constraint is powerful. It allows you control an Action using the transformations of another object.

The underlying idea of the *Action* constraint is very similar to the one behind the Drivers, except that the former uses a whole action (i.e. multiple F-Curves of the same type), while the latter controls a single F-Curve of their "owner"…

Note that even if the constraint accepts the *Mesh* action type, only the *Object*, *Pose* and *Constraint* types are really working, as constraints can only affect objects' or bones' transform properties, and not meshes' shapes. Also note that only the object transformation (location, rotation, scale) is affected by the action, if the action contains keyframes for other properties they are ignored, as constraints do not influence those.

As an example, let us assume you have defined an *Object* action (it can be assigned to any object, or even no object at all), and have mapped it on your owner through an *Action* constraint, so that moving the target in the (0.0 to 2.0) range along its X axis maps the action content on the owner in the (0 to 100) frame range. This will mean that when the target's *X* property is 0.0 the owner will be as if in frame 0 of the linked action; with the target's *X* property at 1.0 the owner will be as if in frame 50 of the linked action, etc.

Options



Action panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Evaluation Time
: This property allows objects to be driven without a constraint target by interpolating between the *Action Start* and *End* frames. The relative position between the start and end frame can be controlled using the value slider.

  This is very helpful for more complex rigging and mechanical rigs, as it means the Action constraint can be controlled directly with a Driver or Custom Property.

Mix

Specifies how the keyframed transformation from the action is combined with the existing transformation. These modes are the same as in the Copy Transforms constraint.

Before/After Original (Full)

The keyframed transformation is added before/after the existing transformation, as if it was applied to an imaginary parent/child of the constraint owner. Scale is handled like in the most basic Full Inherit Scale mode of bones, so combining non-uniform scale and rotation will create shear.

Before/After Original (Aligned)

The keyframed transformation is added before/after the existing transformation, as if it was applied to an imaginary parent/child of the constraint owner. Scale is handled like in the Aligned Inherit Scale mode of bones to avoid creating shear.

This is equivalent to using the *Split Channels* option, but replacing the location component with the result of *Full*. If only uniform scale is used, the result is identical to *Full*.

Before/After Original (Split Channels)

Combines location, rotation and scale components of the transformation separately, similar to a sequence of three Copy Location, Copy Rotation and Copy Scale (with Offset) constraints bundled together in one operation; the result may be slightly different in case of sheared inputs.

Unlike *Aligned*, in this mode location channels are simply added together, so rotation and scale components of the input transformations cannot affect the resulting location.

Warning

For technical reasons modes other than *After Original (Full)* and *After Original (Aligned)* may not work as expected for constraints on *objects* (not bones) without a parent.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Target

Channel
: This selector controls which transform property (location, rotation or scale along/around one of its axes) from the target to use as “action driver”.

Target
: This constraint allows you to choose in which space to evaluate its target’s transform properties.

Range Min, Max
: The lower and upper bounds of the driving transform property value.

Warning

Unfortunately, here again we find the constraint’s limitations:

- When using a rotation property as “driver”, these values are “mapped back” to the (-180.0 to 180.0) range.
- When using a scale property as “driver”, these values are limited to null or positive values.

Action

Action
: Select the name of the action you want to use.

Warning

Even though it might not be in red state (UI refresh problems…), this constraint is obviously not functional when this field does not contain a valid action.

Object Action

Bones **only**, when enabled, this option will make the constrained bone use the "object" part of the linked action, instead of the "same-named pose" part. This allows you to apply the action of an object to a bone.

Frame Start, End

The starting and ending frames of the action to be mapped.

Note

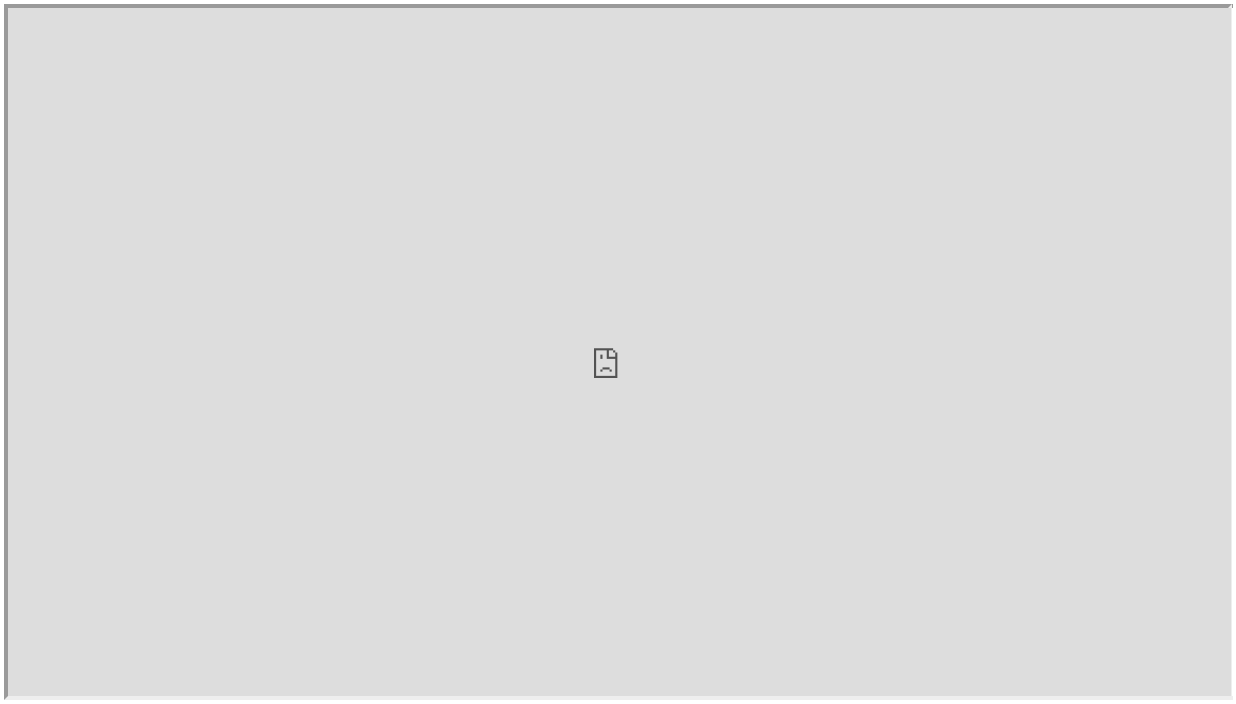
- These values must be strictly positive.
- By default, both values are set to 0, which disables the mapping (i.e. the owner just gets the properties defined at frame 0 of the linked action…).

Notes

- When the object or bone already has Action constraints, the next constraint using a newly keyframed action should be added before all others in order to get the same final combined transformation. This fact is not affected by the Mix mode.
- Unlike usual, you can have a *Start* value higher than the *End* one, or a *Min* one higher than a *Max* one: this will reverse the mapping of the action (i.e. it will be "played" reversed…), unless you have both sets reversed, obviously!
- When using a *Constraint* action, it is the constraint *channel's names* that are used to determine to which constraints of the owner apply the action. E.g. if you have a constraint channel named "trackto_empt1", its keyed *Influence* and/or *Head/Tail* values (the only ones you can key) will be mapped to the ones of the owner's constraint named "trackto_empt1".

- Similarly, when using a *Pose* action (which is obviously only meaningful and working when constraining a bone!), it is the bone's name that is used to determine which bone *channel's names* from the action to use (e.g. if the constrained bone is named "arm", it will use and only use the action's bone channel named "arm"…). Unfortunately, using a *Pose* action on a whole armature object (to affect all the keyed bones in the action at once) will not work…
- Actions can also be marked as Asset, but with certain limitations. For more info, see Pose Library.

Example



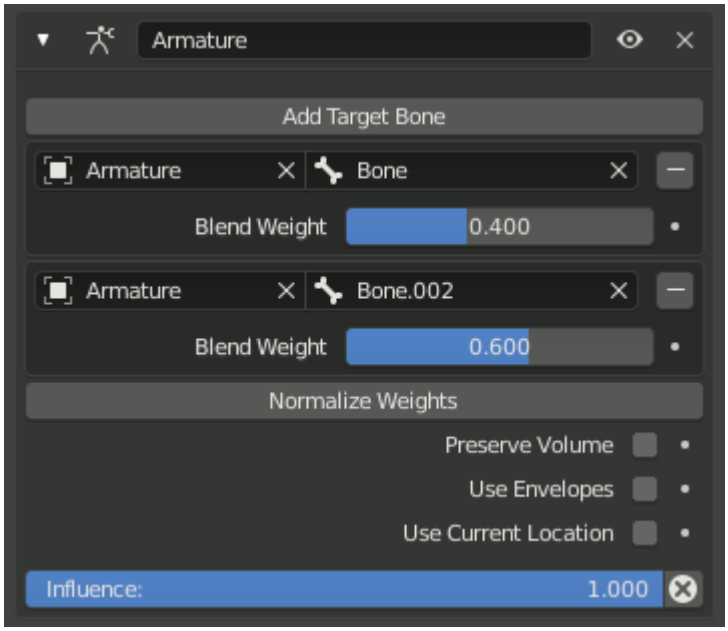
Armature Constraint

Armature is the constraint version of the Armature Modifier, exactly reproducing the weight-blended bone transformations and applying it to its owner orientation. It can be used like a variant of the Child Of constraint that can handle multiple parents at once, but requires all of them to be bones.

Note

Unlike the Armature modifier, the constraint does not take the Deform checkbox of bones into account, and can use any bone as target.

Options



Armature constraint.

Preserve Volume

Like the matching option of the modifier, it enables the use of quaternions for preserving the volume of the object during deformation.

Use Envelopes
: To approximate envelope-only behavior of the modifier, add all relevant bones with weight 1.0 and enable this option.

Note

Unlike the modifier, the constraint always requires explicitly listing all of its target bones with associated weights. This option merely enables envelopes for all bones, as if they had Envelope Multiply enabled.

Use Current Location
: Only for constraints on bones: Instead of using the rest location, use the current location of the owner bone to compute envelope weights or binding to B-Bone segments.

With envelope weights, this can be used to change the active "parent" bone of the owner bone dependent on its location. For non-bones this mode is always active, because they don't have a rest location.

Add Target Bone
: This button adds a new empty entry at the end of the target list.

Normalize Weights
: This button normalizes all weight values in the target list so that they add up to 1.0.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Bones

This specifies the list of bones used by the constraint to deform its owner. Each target bone has the following input fields and controls:

Target
: Unlike the modifier, the constraint can use bones coming from different armatures at the same time. See common constraint properties for more information.

Sub-target
: Name of the target bone.

(Remove Target)
: Removes the entry from the target list.

Weight
: Weight associated with the bone, equivalent to vertex groups in the modifier.

Child Of Constraint

Child Of is the constraint version of the standard parent/children relationship between objects (the one established through the `Ctrl-P` shortcut, in the 3D Viewport).

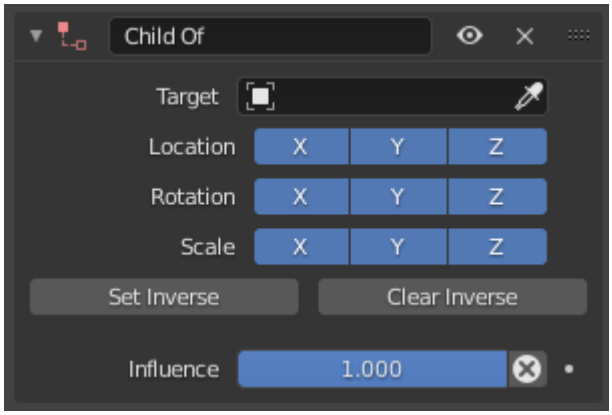
Parenting with a constraint has several advantages and enhancements, compared to the traditional method:

- You can have several different parents for the same object (weighting their respective influence with the *Influence* slider).
- As with any constraint, you can key (i.e. animate) its Influence setting. This allows the object which has a Child Of constraint upon it to change over time which target object will be considered the parent, and therefore have influence over it.

  Important

  Do not confuse this "basic" object parenting with the one that defines the chains of bones inside of an armature. This constraint is used to parent an object to a bone (the so-called object skinning), or even bones to bones. But do not try to use it to define chains of bones.

Options



Child Of panel.

Target
: The target object that this object will act as a child of. Data ID used to select the constraint's target, and is not functional (red state) when it has none. See common constraint properties for more information.

Location
: Each of these buttons will make the parent affect or not affect the location along the corresponding axis.

Rotation
: Each of these buttons will make the parent affect or not affect the rotation around the corresponding axis.

Scale
: Each of these buttons will make the parent affect or not affect the scale along the corresponding axis.

Set Inverse
: By default, when you parent your owner to your target, the target becomes the origin of the owner's space. This means that the location, rotation and scale of the owner are offset by the same properties of the target. In other words, the owner is transformed when you parent it to your target. This might not be desired! So, if you want to restore your owner to its before-parenting state, click on the *Set Inverse* button.

Clear Inverse

This button reverses (cancels) the effects of the above one, restoring the owner/child to its default state regarding its target/parent.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Tips

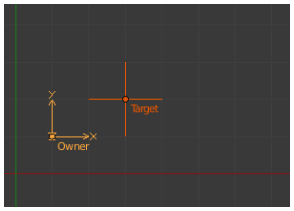
When creating a new parent relationship using this constraint, it is usually necessary to click on the *Set Inverse* button after assigning the parent. As noted above, this cancels out any unwanted transform from the parent, so that the owner returns to the location/rotation/scale it was in before the constraint was applied. Note that you should apply *Set Inverse* with all other constraints disabled (their *Influence* set to 0.0) for a particular *Child Of* constraint, and before transforming the target/parent (see example below).

About the toggle buttons that control which target's (i.e. parent's) individual transform properties affect the owner, it is usually best to leave them all enabled, or to disable all three of the given Location, Rotation and Scale transforms.

Technical Note

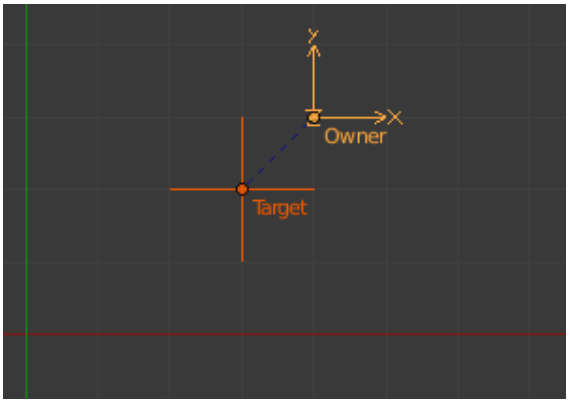
If you use this constraint with all channels on, it will use a straight matrix multiplication for the parent relationship, not decomposing the parent matrix into loc/rot/size. This ensures any transformation correctly gets applied, also for combinations of rotated and non-uniform scaled parents.

Examples



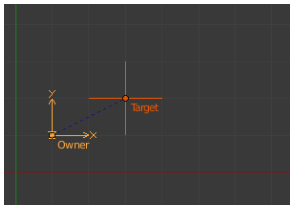
No constraint.

Note the position of Owner empty 1.0 unit along the X and Y axes.



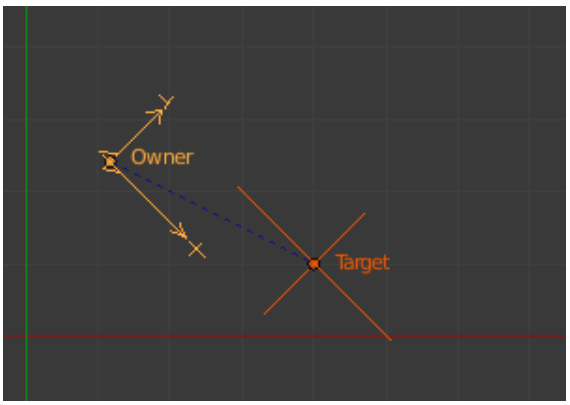
Child Of just added.

Here you can see that Owner empty is now 1.0 unit away from Target_1 empty along X and Y axes.



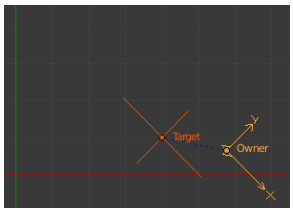
Offset set.

Set Inverse has been clicked, and Owner is back to its original position.



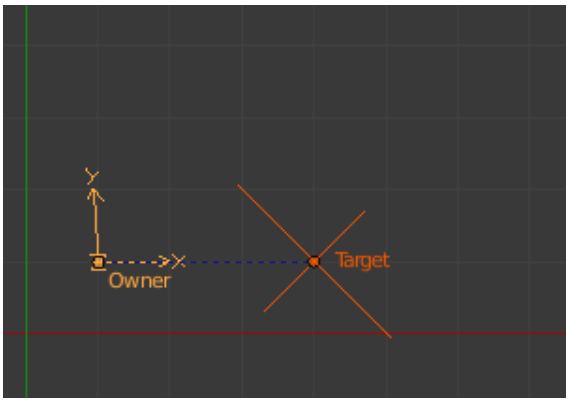
Target/parent transformed.

Target_1 has been moved along the XY plane, rotated around the Z axis, and scaled along its local X axis.



Offset cleared.

Clear Inverse has been clicked. Owner is fully again controlled by Target_1.



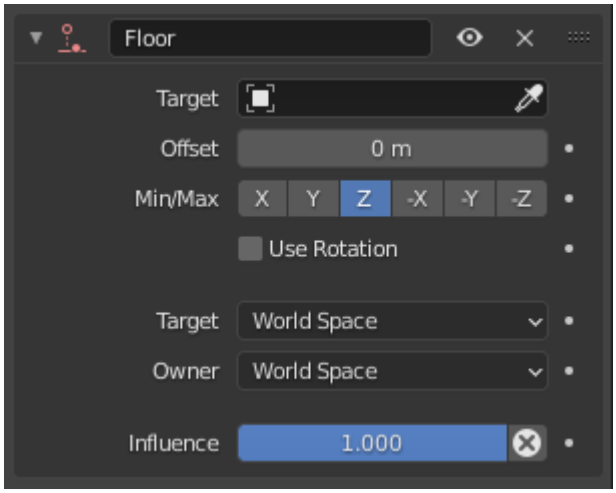
Offset set again.

Set Offset has been clicked again. As you can see, it does not gives the same result as in (Target/parent transformed). As noted above, use Set Inverse only once, before transforming your target/parent.

Floor Constraint

The *Floor* constraint allows you to use its target position (and optionally rotation) to specify a plane with a “forbidden side”, where the owner cannot go. This plane can have any orientation you like. In other words, it creates a floor (or a ceiling, or a wall)! Note that it is only capable of simulating entirely flat planes, even if you use the *Vertex Group* option. It cannot be used for uneven floors or walls.

Options



Floor panel.

Target
: Data ID used to select the constraints target, and is not functional (red state) when it has none. See common constraint properties for more information.

Offset
: Allows you to offset the “floor” plane from the target’s origin, by the given number of units. Use it e.g. to account for the distance from a foot bone to the surface of the foot’s mesh.

Max/Min

Controls which plane will be the “floor”. The names of the buttons correspond, indeed, to the *normal* to this plane (e.g. enabling Z means “XY plane”, etc.). By default, these normals are aligned with the *global* axes. However, if you enable *Use Rotation* (see above), they will be aligned with the *local target’s axes*. As the constraint does not only define an uncrossable plane, but also a side of it which is forbidden to the owner, you can choose which side by enabling either the positive or negative normal axis… e.g. by default Z, the owner is stuck in the positive Z coordinates.

Use Rotation

Forces the constraint to take the target’s rotation into account. This allows you to have a “floor” plane of any orientation you like, not just the global XY, XZ and YZ ones…

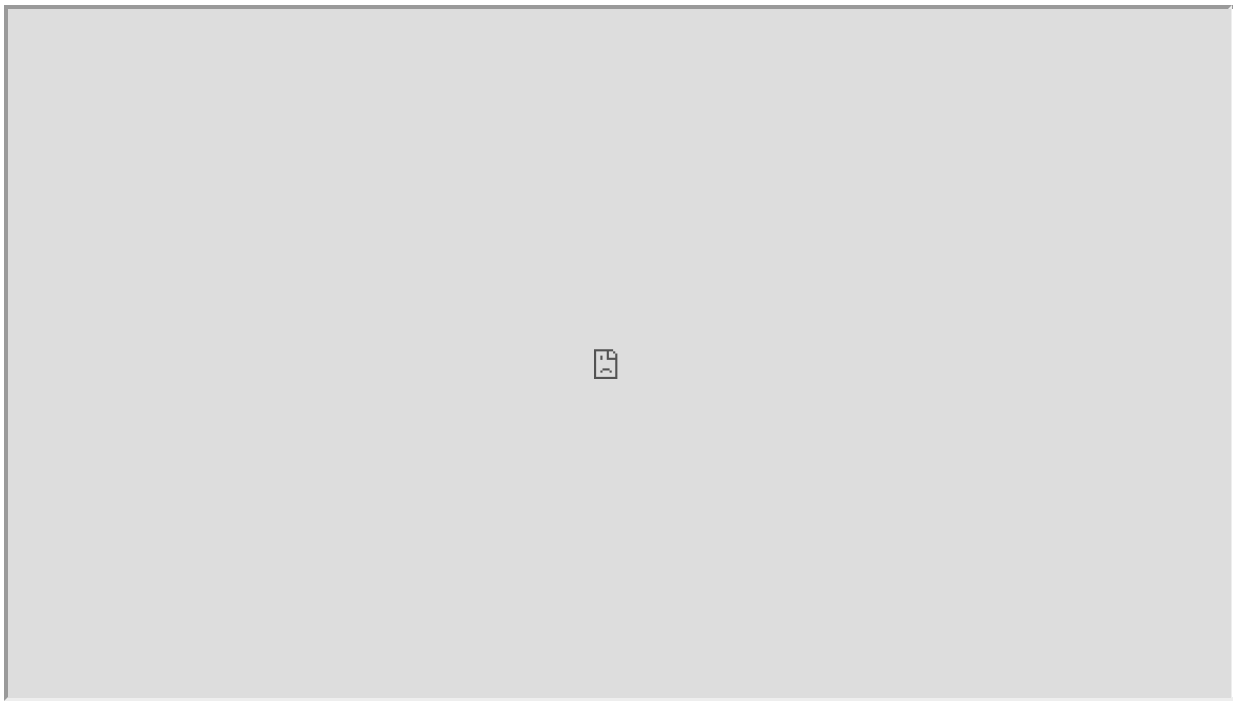
Target/Owner

Standard conversion between spaces. See common constraint properties for more information.

Influence

Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Example



Follow Path Constraint

The *Follow Path* constraint places its owner onto a *curve* target object, and makes it move along this curve (or path). It can also affect its owner's rotation to follow the curve's bends, when the *Follow Curve* option is enabled.

It could be used for complex camera traveling, a train on its rails and most other vehicles can also use "invisible" tracks, the links of a bicycle chain, etc.

The owner is always evaluated in the global (world) space:

- Its location (as shown in the *Transform* panel) is used as an offset from its normal position on the path. E.g. if you have an owner with the (1.0, 1.0, 0.0) location, it will be one unit away from its normal position on the curve, along the X and Y axis. Hence, if you want your owner *on* its target path, clear its location `Alt-G`!
- This location offset is also proportionally affected by the scale of the target curve. Taking the same (1.0, 1.0, 0.0) offset as above, if the curve has a scale of (2.0, 1.0, 1.0), the owner will be offset two units along the X axis (and one along the Y one)…
- When the *Follow Curve* option is enabled, its rotation is also offset to the one given by the curve. E.g. if you want the Y axis of your object to be aligned with the curve's direction, it must be in rest, non-constrained state, aligned with the global Y axis. Here again, clearing your owner's rotation `Alt-R` might be useful…

The movement of the owner along the target curve/path may be controlled in two different ways:

- The most simple is to define the number of frames of the movement, in the Path Animation panel of the Curve tab, via the *Frames* number field, and its start frame via the constraint's Offset option (by default, start frame: 1 [= offset of 0], duration: 100).

- The second way, much more precise and powerful, is to define an *Evaluation Time* interpolation curve for the *Target* path (in the *Graph Editor*). See the Graph Editor chapter to learn more about F-Curves.
- If you do not want your owner to move along the path, you can give to the target curve a flat *Speed* F-Curve (its value will control the position of the owner along the path).

Follow Path is another constraint that works well with the Locked Track one. One example is a flying camera on a path. To control the camera's roll angle, you can use a *Locked Track* and a target object to specify the up direction, as the camera flies along the path.

Note

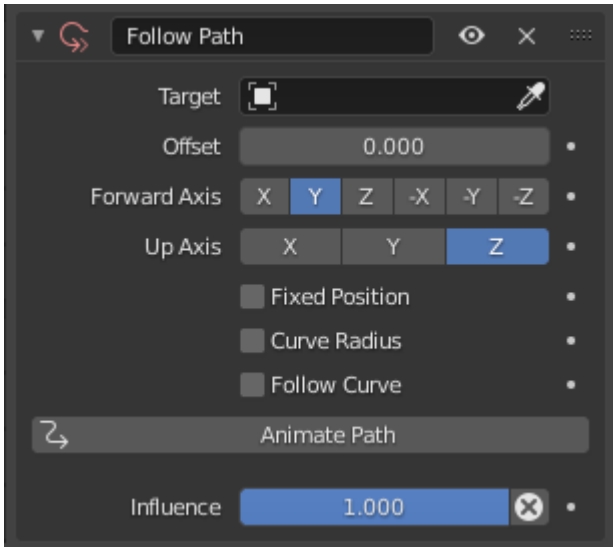
Follow Path & Clamp To

Do not confuse these two constraints. Both of them constraint the location of their owner along a curve, but *Follow Path* is an "animation-only" constraint, inasmuch as the position of the owner along the curve is determined by the time (i.e. current frame), whereas the Clamp To *constraint* determines the position of its owner along the curve using one of its location properties' values.

Note

Note that you also need to keyframe Evaluation Time for the Path. Select the path, go to the *Path Animation* panel in the curve properties, set the overall frame to the first frame of the path (e.g. frame 1), set the value of Evaluation time to the first frame of the path (e.g. 1), right-click on Evaluation time, select create keyframe, set the overall frame to the last frame of the path (e.g. frame 100), set the value of Evaluation time to the last frame of the path (e.g. 100), right-click on Evaluation time, select create keyframe.

Options



Follow Path panel.

Target
: Data ID used to select the constraint's target, which *must* be a curve object, and is not functional (red state) when it has none. See common constraint properties for more information.

Offset
: The number of frames to offset from the "animation" defined by the path (by default, from frame 1).

Forward Axis
: The axis of the object that has to be aligned with the forward direction of the path (i.e. tangent to the curve at the owner's position).

Up Axis
: The axis of the object that has to be aligned (as much as possible) with the world Z axis. In fact, with this option activated, the behavior of the owner shares some properties with the one caused by a Locked Track constraint, with the path as "axle", and the world Z axis as "magnet".

Fixed Position
: Object will stay locked to a single point somewhere along the length of the curve regardless of time.

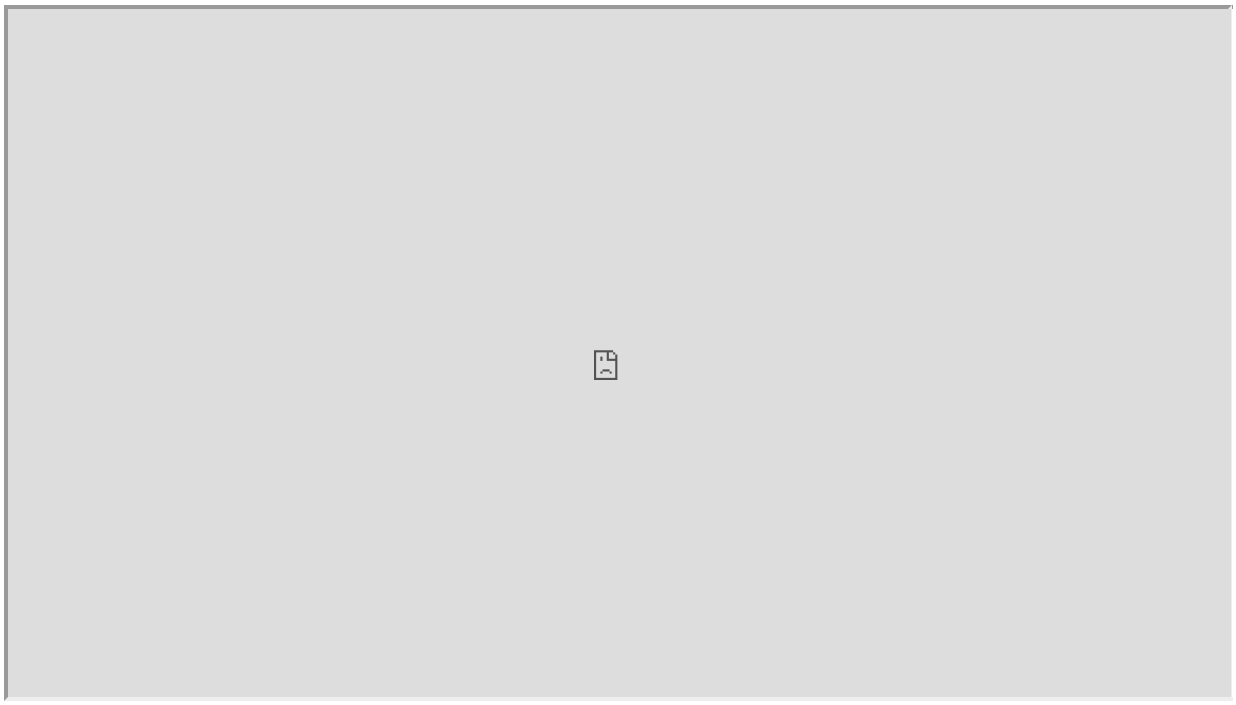
Curve Radius
: Objects scaled by the curve radius. See Curve Editing.

Follow Curve
: If this option is not activated, the owner's rotation is not modified by the curve; otherwise, it is affected depending on the Forward and Up Axes.

Animate Path
: Adds an F-Curve with options for the start and end frame. ToDo: from above.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

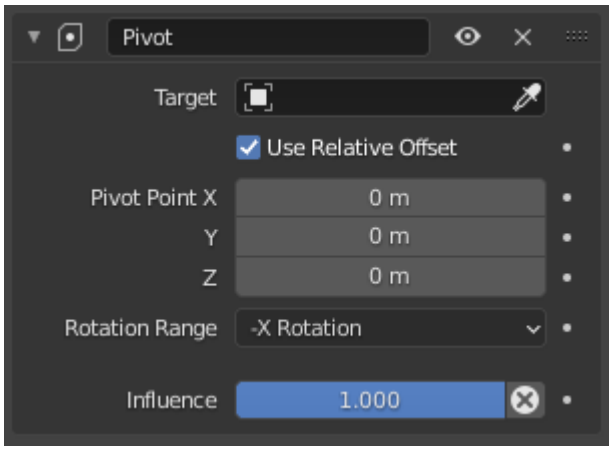
Example



Pivot Constraint

The *Pivot* constraint allows the owner to rotate around a target object. It was originally intended for pivot joints found in humans e.g. fingers, feet, elbows, etc.

Options



Pivot panel.

Target
: Data ID for the selection of the object to be used as a pivot point. See common constraint properties for more information.

Use Relative Offset
: Offset will be an absolute point in space instead of relative to the target.

Pivot Point X, Y, Z
: Offset of pivot from target.

Rotation Range
: Rotation range on which pivoting should occur.

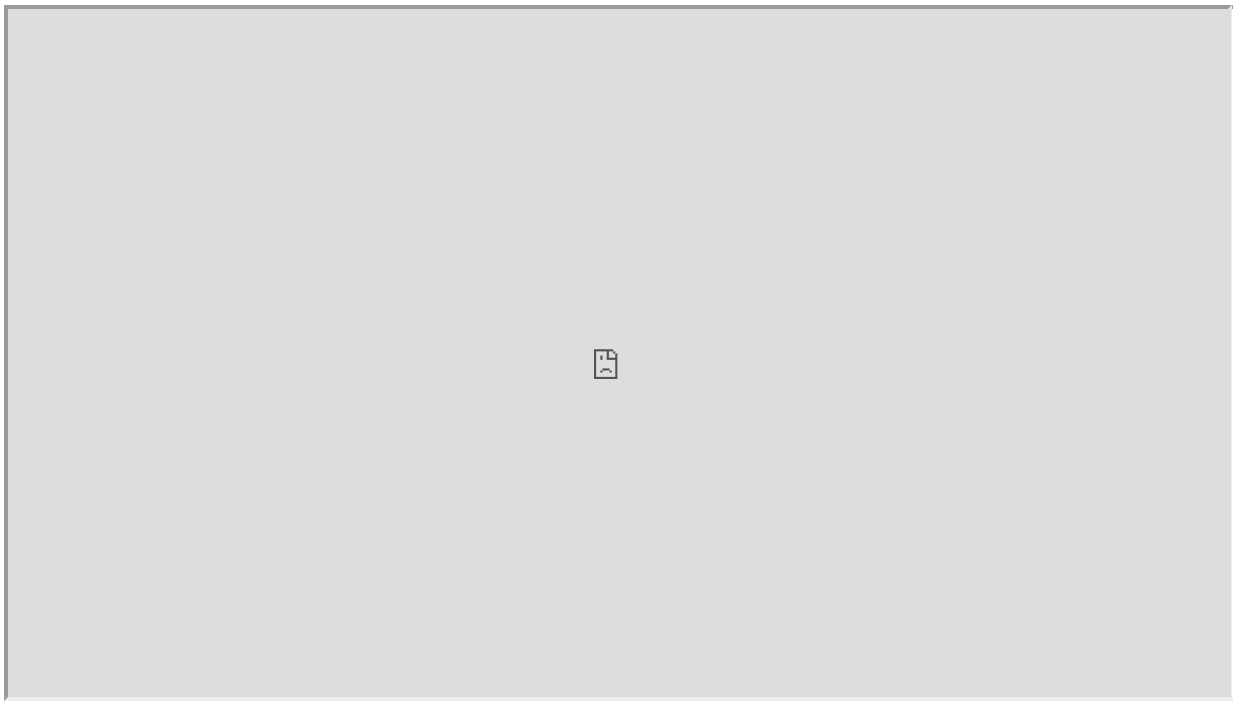
  Always

Use the pivot point in every rotation.

-X/-Y/-Z/X/Y/Z Rotation
: Use the pivot point in the corresponding direction around the corresponding axis.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

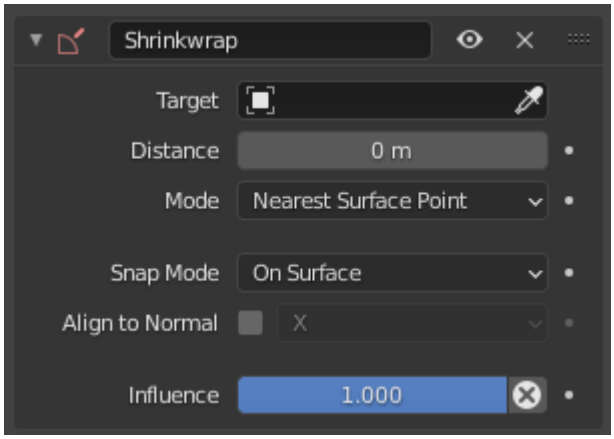
Example



Shrinkwrap Constraint

The *Shrinkwrap* constraint is the "object counterpart" of the Shrinkwrap Modifier. It moves the owner origin and therefore the owner object's location to the surface of its target. This implies that the target *must* have a surface; thus, you can only use meshes as targets.

Options



Shrinkwrap panel.

Target
: Data ID used to select the constraint's target, which *must* be a mesh object, and is not functional (red state) when it has none. See common constraint properties for more information.

Distance
: This number field controls the offset of the owner from the shrunk computed position on the target's surface.

Influence
: Controls the percentage of affect the constraint has on the object. See common constraint properties for more information.

Mode

This selector allows you to select which method to use to compute the point on the target's surface to which to move the owner's origin. You have these options:

Nearest Surface Point

The chosen target's surface's point will be the nearest one to the original owner's location. This is the default and most commonly useful option.

Projection

The target's surface point is determined by projecting the owner's origin along a given axis.

Project Axis
: This axis is controlled by the radio buttons that show up when you select this type. This mean the projection axis can only be aligned with one of the three axes, or their opposites. When the projection of the owner's origin along the selected direction does not hit the target's surface, the owner's location is left unchanged.

  +X, +Y, +Z, -X, -Y, -Z

Space
: Coordinate space in which the axis direction is specified.

Distance
: Distance cutoff after which projection is assumed to have failed, leaving the location unchanged.

Project Opposite
: In addition to the selected projection axis, project in the opposite direction and choose the closest hit.

Face Cull

This radio button allows you to prevent any projection over the “front side” (respectively the “back side”) of the target’s faces. The “side” of a face is determined by its normal (front being the side “from where” the normal “originates”).

Off, Front, Back

Invert Cull

When used with *Project Opposite* and *Face Culling*, it inverts the *Front* or *Back* cull choice for the opposite direction.

Nearest Vertex

This method is very similar to the *Nearest Surface Point* one, except that the owner’s possible shrink locations are limited to the target’s vertices.

This method doesn’t support the *Snap Mode* setting described below.

Target Normal Projection

This method is similar to *Nearest Surface Point*, but produces a much smoother projection in return for being significantly slower.

Instead of finding the closest point, it searches for the nearest point that has its interpolated smooth normal pointing towards or away from the original owner position. Non-manifold boundary edges are specially handled as infinitely thin cylinders that emit normals in all perpendicular directions; ignores flat shading.

Snap Mode

Most Shrinkwrap types support an additional setting to control how the owner is moved to the target point selected by the methods described above. Some of the choices only differ if *Distance* is not zero.

On Surface

The owner location is always changed. The offset is applied along the projection line connecting the original owner location and selected target point towards the original position.

Outside Surface
: Like *On Surface*, but the offset is always applied towards the outside of the target.

Above Surface
: Like *On Surface*, but the offset is applied along the smooth normal of the target.

Inside
: The owner is not moved if it is already inside the target. Offset shrinks the allowed volume towards the inside along the projection line.

Outside
: The owner is not moved if it is already outside the target. Offset expands the exclusion volume towards the outside along the projection line.

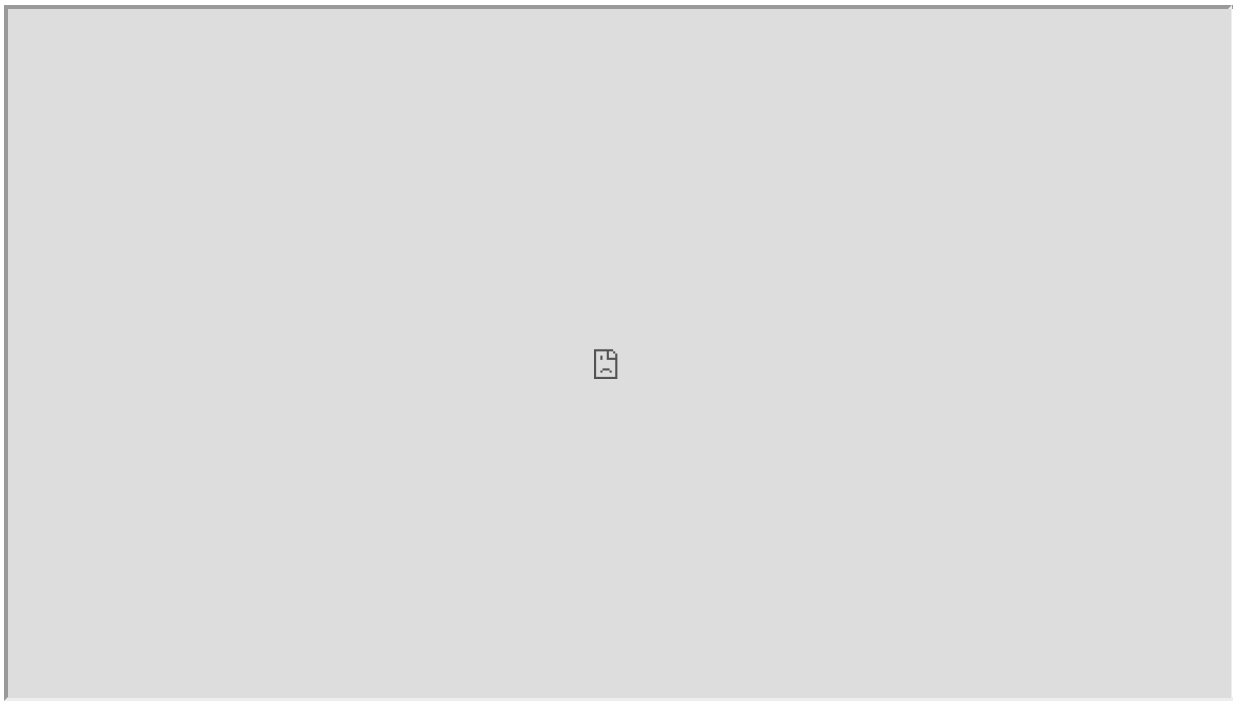
The *Inside* and *Outside* options can be used for very crude collision detection. The inside vs outside determination is done based on the target normal and is not always stable near 90 degree and sharper angles in the target mesh.

Align To Normal

Whenever *Snap Mode* is available, it is also possible to align the specified local axis of the object to the smooth normal of the target at the selected point. The axis is selected via radio buttons.

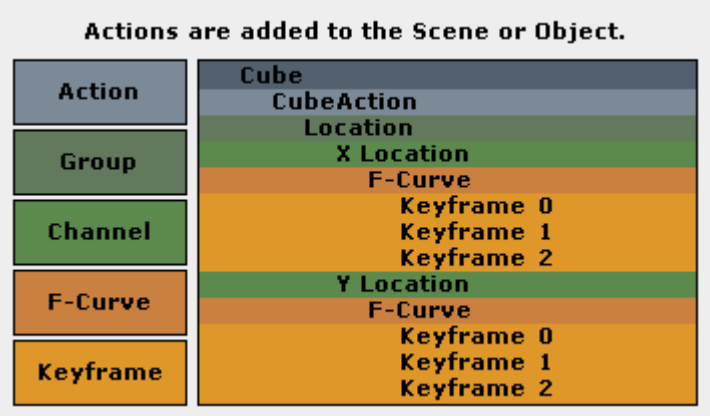
The alignment is performed via smallest rotation, like in Damped Track constraint.

Example



Actions

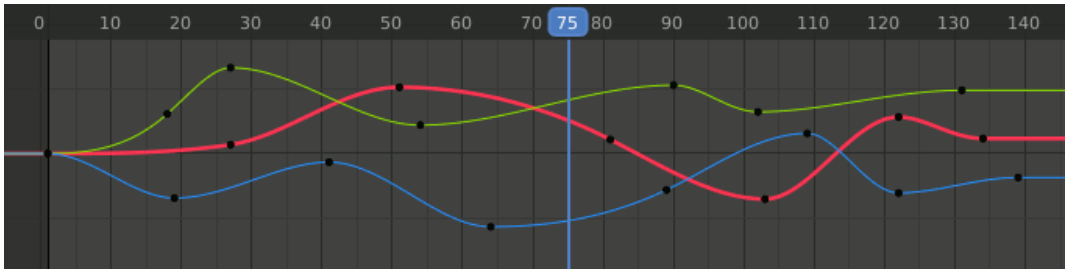
When animating objects and properties in Blender, Actions record and contain the data. As everything else in Blender, Actions are data-blocks.



Actions.

So when you animate an object by changing its location with keyframes, the animation is saved to the Action.

Each property has a channel which it is recorded to, for example, `Cube.location.x` is recorded to Channel X Location. The *X location* and *Y location* properties can be shared across multiple objects, if all objects have *X location* and *Y location* properties beneath them.



Graph Editor. Each channel has an F-Curve represented by the lines between the keyframes.

Actions
: Record and contain animation data.

Groups
: Are groups of channels.

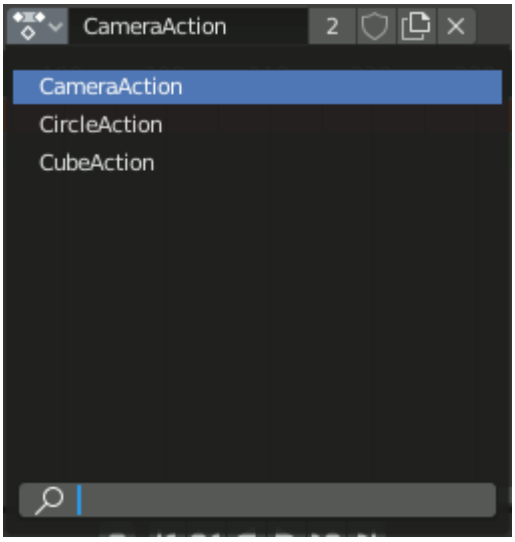
Channels
: Record properties.

F-Curves
: F-Curves are used to interpolate the difference between the keyframes.

Keyframes
: Keyframes are used to set the values of properties bound to a point in time.

Working with Actions

When you first animate an object by adding keyframes, Blender creates an *Action* to record the data.



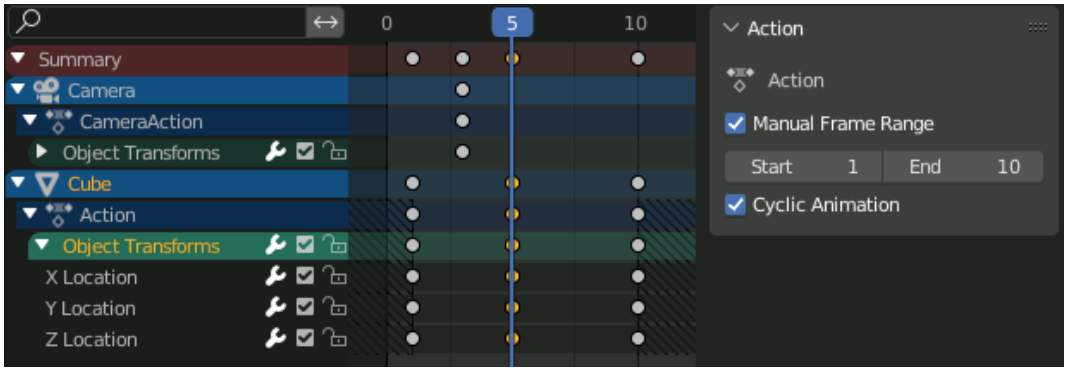
The Action data-block menu.

Actions can be managed with the *Action data-block menu* in the Dope Sheet Action Editor header, or the Sidebar region of the NLA Editor.

If you are making multiple actions for the same object, press the shield button for each action. This will give the actions a *Fake User* and will make Blender save the unlinked actions.

Objects can only use one *Action* at a time for editing. The NLA Editor is used to blend multiple actions together.

Properties



Actions with and without a Manual Frame Range in Dope Sheet.

It is possible to manually specify the intended useful frame range of an action via a panel available in the Dope Sheet or the NLA Editor when a channel or NLA track is selected.

Manual Frame Range

Manually specify the intended playback frame range for the action (this range is used by some tools, but does not affect animation evaluation). The manual frame range feature can be toggled with the checkbox.

When the range is set, it is used instead of the actual range occupied by key frames when adding a new track based on the action to NLA. It can also be used by exporters to determine the range of frames to export.

The range is displayed in the background of the editor as diagonal hash fill, to distinguish it from the solid fill of the current playback range.

The frame values are most commonly expected to be integers, but can be fractional.

Cyclic Animation

Specifies that the action is intended to be cyclic over the specified range. The first and last frames of the range should represent the same pose of the cycle one loop apart, i.e. the range should include the duplicated initial key of the loop.

Note

This option signifies intent and does **not** make the action cycle on its own. However, if Cycle-Aware Keying is enabled, it will automatically enable cyclic extrapolation and set up the loop period for curves newly added to the action.

Custom Properties

Create and manage your own properties to store data in the action's data block. See the Custom Properties page for more information.

Drivers

- Introduction
  - Graph View
  - Driver Configuration
  - Notes on Scripted Expressions
- Usage
  - Add Driver
  - Edit Driver
  - Open Drivers Editor
  - Copy & Paste
  - Copy Driver to Selected
  - Copy As New Driver
  - Expression
  - Removing Drivers
- Drivers Panel
  - Driver Settings
  - Driver Variables
  - Expressions
- Workflow & Examples
  - Transform Driver
  - Scripted Expression - Orbit a Point
  - Custom Function - Square Value
  - View Layer Attribute Lookup
  - Shape Key Drivers
- Troubleshooting
  - Scripted Expression
  - Rotational Properties are Radians

Introduction

Drivers are a way to control values of properties by means of a function, or a mathematical expression.

Effectively, drivers consist of:

- A **driver configuration** that specifies zero, one, or more input values using other properties or object transformation channels, and combines them using a predefined mathematical function or a custom Python expression.
- An **animation** F-Curve that maps the output of the driver configuration to the final value to apply to the driven property.

As an example, the rotation of Object 1 can be controlled by the scale of Object 2. It is then said that the scale of Object 2 *drives* the rotation of Object 1.

Not only can drivers directly set the value of a property to the value of a different one, they can also combine multiple values using a fixed function or a Python expression and further modulate it with a manually defined curve and/or a modifier stack.

Drivers are an extremely powerful tool for building rigs and are typically used to drive bone transforms and the influence of shape keys, action constraints and modifiers, often using custom properties as inputs.

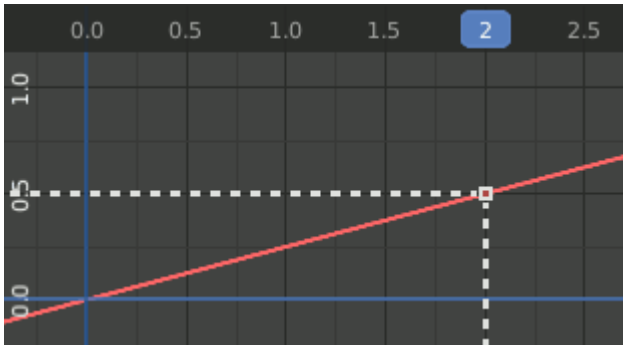
Graph View

The main area of the Drivers editor shows an F-Curve that represents the driver function.

The **X axis** maps to the output value of the driver configuration. The units depend on the setup.

The **Y axis** shows the value applied to the target property. The units depend on the property.

In the example image, if the driver value is 2.0 the property value will be 0.5.



Driver curve in the Drivers editor.

The default F-Curve is an identity map, i.e. the value produced by the driver configuration is applied to the driven property unchanged. If the driver output value is 2.0, the property will be 2.0.

The driver function can be defined artistically with Bézier curve handles or mathematically with trigonometric functions or polynomial expressions such as \(y = a + bx\). Furthermore, the function can also be procedurally modulated with noise or cyclic repetitions. See Modifiers for more details.

Driver Configuration

The Drivers panel shows the setup for a driver.

A driver can have zero, one, or more **variables**. Variables specify which properties, object transformation channels, or relative distances between objects, are used as inputs by the driver.

The driver **type** determines how the variables are used. The type can be:

- a built-in function: for example, the sum of the variables' values, or
- a scripted expression: an arbitrary Python expression that refers to the variables by their names.

This driver configuration outputs a single value which changes when the variables change. This value is then evaluated through the driver function curve to produce the result to be applied to the driven property.

Notes on Scripted Expressions

When a driver uses a *Scripted Expression*, Blender can evaluate it without using the fully featured Python interpreter if it is simple enough. This means that drivers are fast to evaluate with simple divisions, additions and other “simple” expressions. The built-in functions are always evaluated natively.

See Simple Expressions for a comprehensive list of expressions that can be evaluated natively.

When the expression is not simple, it will be evaluated using Python. As a consequence, the driver will be slower and there is a security risk if the author of the Python code is unknown. This is an important thing to take into consideration for heavy scenes and when sharing files with other people. See also: Auto run.

Usage

Drivers can be added to properties via their context menu, a shortcut, copy-pasted, or by typing an expression directly into the property's value.

After adding drivers, they are usually modified in the Drivers editor, or via a simplified *Edit Driver* popover invoked from the property context menu.

Add Driver

Reference

Menu:: *Context menu ▸ Add Driver*
Shortcut:: `Ctrl-D`

The usual way to add a driver to a property is to `RMB` click a property, then choose *Add Driver* in the context menu. Drivers can also be added by pressing `Ctrl-D` with the mouse over the property.

This operation adds a driver with a single variable (which needs to be filled in), and displays the *Edit Driver* popover.

Edit Driver

Reference

Menu:: *Context menu ▸ Edit Driver*

Displays a popover window that allows editing the custom expression and input variables of the driver without opening the full *Drivers Editor*.

Many drivers don't use their F-Curve component, so this reduced interface is sufficient.

Open Drivers Editor

Reference

Menu:: *Context menu ▸ Open Drivers Editor*

Opens a new window with the *Drivers Editor* and selects the driver associated with the property.

Copy & Paste

Reference

Menu:: *Context menu ▸ Copy Driver*
Menu:: *Context menu ▸ Paste Driver*

Drivers can be copied and pasted via the context menu. When adding drivers with the same settings, this can save time modifying settings.

Copy Driver to Selected

Reference

Menu:: *Context menu ▸ Copy Drivers to Selected*
Menu:: *Context menu ▸ Copy Driver to Selected*
Menu:: *Context menu ▸ Copy All Drivers to Selected*

Copy the property's driver from the active item to the same property of all selected items, if the same property exists.

Copy As New Driver

Reference

Menu:: *Context menu ▸ Copy As New Driver*

A driver that sets the property value to the value of a different property can be quickly created by using the *Copy As New Driver* context menu option of the input property, and then pasting the result onto the output property via *Paste Driver*.

It is also possible to add the new driver variable to an existing driver using the Paste Driver Variables button in the editor panel.

Expression

This is a quick way to add drivers with a scripted expression. First click the property you want to add a driver to, then type a hash # and a scripted expression.

Some examples:

- `#frame`
- `#frame / 20.0`
- `#sin(frame)`
- `#cos(frame)`

Removing Drivers

Reference

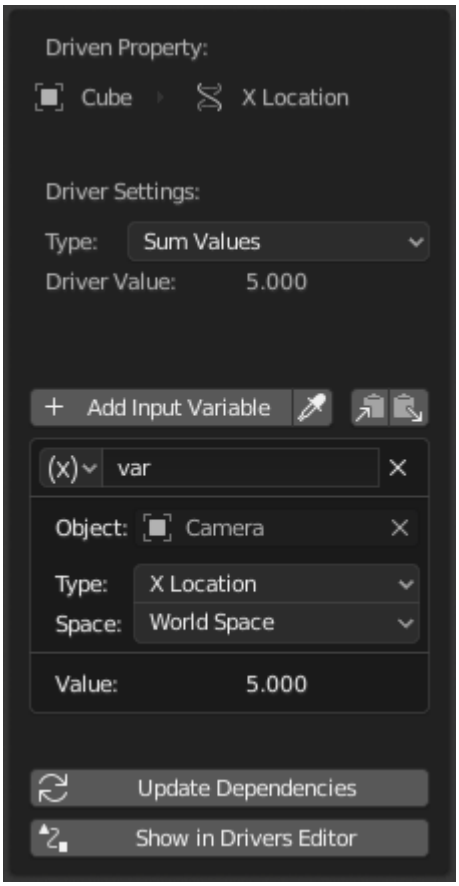
Menu:: *Context menu ▸ Delete Driver(s)*
Menu:: *Context menu ▸ Delete Single Driver*
Shortcut:: `Ctrl-Alt-D`

Removes driver(s) associated with the property, either for the single selected property or sub-channel, or all components of a vector.

Drivers Panel

Reference

Editor::



Edit Driver popover.

Graph editor
Mode:: Drivers
Panel:: *Sidebar region ▸ Drivers*
Shortcut:: `N`

Reference

Menu:: *Context menu ▸ Edit Driver*
Shortcut:: `Ctrl-D`

This panel is visible in Sidebar of the Drivers Editor or as a popover when adding a driver to a property.

It shows the property that is being driven, followed by a series of settings that determine how the driver works.

Driver Settings

Type

There are two categories of drivers:

- **Built-in functions** (*Average*, *Sum*, *Min* and *Max*)

  The driven property will have the value of the average, sum, lowest or highest (respectively) of the values of the referenced *Driver Variables*. If there is only one driver variable, these functions will yield the same result.

- **Custom** (*Scripted Expression*).

  An arbitrary Python expression that can refer to the *Driver Variables* by name. See Expressions.

Driver Value

The current result of the driver setup. Useful for debug purposes.

Variables

See Driver Variables.

Update Dependencies

Forces an update for the Driver Value dependencies.

Show in Drivers Editor

Opens the fully featured Drivers Editor. This button only appears in the popover version of the Drivers panel.

Driver Variables

Variables are references to properties, transformation channels, or the result of a comparison between transformations of two objects.

Drivers should access object data via *Driver Variables*, rather than direct references in the Python expression, in order for dependencies to be correctly tracked.



Add, Copy, Paste buttons.

Add Input Variable
: Adds a new Driver Variable.

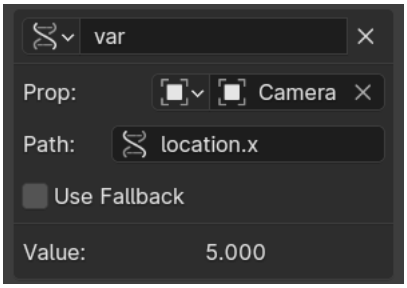
Copy/Paste Variables
: Copies the current variable list so it can be pasted into another driver's variable list.

Name
: Name for use in scripted expressions. The name must start with a letter, and only contain letters, digits, or underscores.

Variable Type
: The type of variable to use.

  Single Property
  : Retrieves the value of an RNA property, specified by a data-block reference and a path string.

    In case of transform properties, this will return the exact value of the UI property, while Transform Channel will take



parenting and/or constraints into account as needed.

See also Custom Properties.

ID Type
: The ID-block type. For example: Key, Image, Object, Material.

ID
: The ID of the ID-block type. For example: “Material.001”.

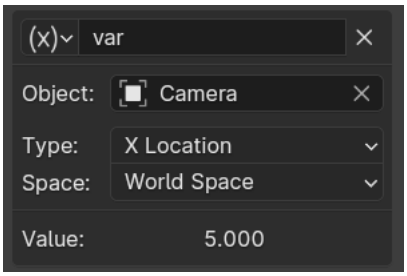
RNA Path
: The RNA name of the property, based on a subset of Python attribute access syntax. For example: `location.x` or `location[0]` for the X location animation channel value (before parenting or constraints), or `["prop_name"]` for a custom property.

Fallback
: If enabled, allows specifying a fallback value to use as the variable value if the RNA Path cannot be resolved, instead of causing a driver evaluation failure. For more info see Context Property below.

Tip

The easiest way to create a variable of this type is to use the Copy As New Driver context menu option of the input property, and paste the result into the driver via Paste Driver Variables.

Transform Channel
: Retrieves the value of a Transform channel from an object or bone.



ID
: ID of the object. For example: Cube, Armature, Camera.

Bone

For armatures, the name of the Armature bone. For example: “Bone”, “Bone.002”, “Arm.r”.

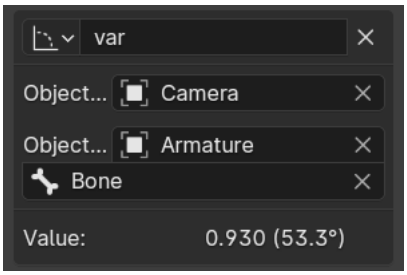
Type
: For example, X Location, X Rotation, X Scale.

  The *Average Scale* option retrieves the combined scale value, computed as the cubic root of the total change in volume. Unlike *X/Y/Z Scale*, this value can be negative if the object is flipped by negative scaling.

Mode (Rotation)
: For rotation channels, specifies the type of rotation data to use, including different explicit Euler orders. Defaults to using the Euler order of the target. See Rotation Channel Modes.

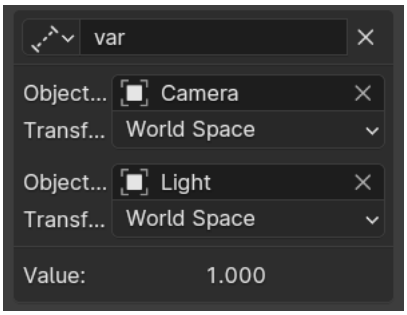
Space
: World Space, Transform Space, Local Space.

Rotational Difference
: Provides the value of the rotational difference between two objects or bones, in radians.



Bone
: For armatures, the name of the Armature bone. For example: “Bone”, “Bone.002”, “Arm.r”.

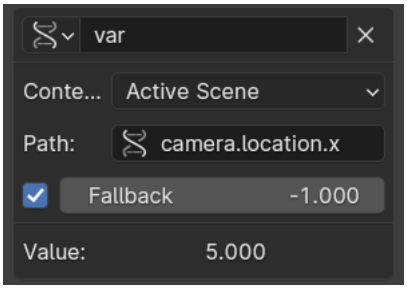
Distance
: Provides the value of the distance between two objects or bones.



Bone
: For armatures, the name of the Armature bone. For example: “Bone”, “Bone.002”, “Arm.r”.

Space
: World Space, Transform Space, Local Space.

Context Property
: Provides the value of a property that is implicitly referring to either a scene or a view layer of the currently evaluating animation system. This is a weak reference which does not lead to the scene or view layer referenced from the driver to be linked when linking animation data.



An example when such properties comes in play is referring to a transformation of the active camera. It is possible to set up a driver in a character file, and make the driver use the set camera when the character is linked into a set.

Context
: Active Scene, Active View Layer.

RNA Path
: The RNA name of the property, based on a subset of Python attribute access syntax. For example: `camera.location.x` or `camera.location[0]` for the camera X location animation channel value (before parenting or constraints), or `["prop_name"]` for a custom property.

Fallback
: If enabled, allows specifying a fallback value to use as the variable value if the RNA Path cannot be resolved, instead of causing a driver evaluation failure.

  This feature can be very useful for making drivers more robust when implementing scene-global options using custom properties. When the object is linked into a different scene, these custom properties may not exist there, and the fallbacks can be used to provide sensible default values.

Fallbacks can also be used to emulate the lookup behavior of the View Layer mode of the material Attribute Node.

Tip

Although the values of the x/y/z animation channels for the camera location can be accessed via `camera.location[0/1/2]`, retrieving its world space location and orientation after parenting and constraints currently requires using `camera.matrix_world`. This property can be understood easily by viewing the matrix as an array of four vectors in *World* space:

- `matrix_world[0][0/1/2]` is the *Screen Right* direction vector (camera local X).
- `matrix_world[1][0/1/2]` is the *Screen Up* direction vector (camera local Y).
- `matrix_world[2][0/1/2]` is the **opposite** of the direction the camera is pointing.
- `matrix_world[3][0/1/2]` is the *location* of the camera.

Value
: Shows the value of the variable.

Rotation Channel Modes

Rotation Transform Channels support a number of operation modes, including:

Auto Euler
: Uses the Euler order of the target to decompose rotation into channels.

XYZ Euler, …
: Explicitly specifies the Euler rotation order to use.

Quaternion
: Provides the Quaternion representation of the rotation.

Swing and X/Y/Z Twist

Decomposes the rotation into two parts: a Swing rotation that aims the specified axis in its final direction, followed by a Twist rotation around that axis. This is often necessary for driving corrective Shape Keys and bones for organic joint rotation.

This decomposition is often produced in rigs by using a helper bone with a Damped Track Constraint to extract the swing part, and its child with Copy Transforms to extract the twist component.

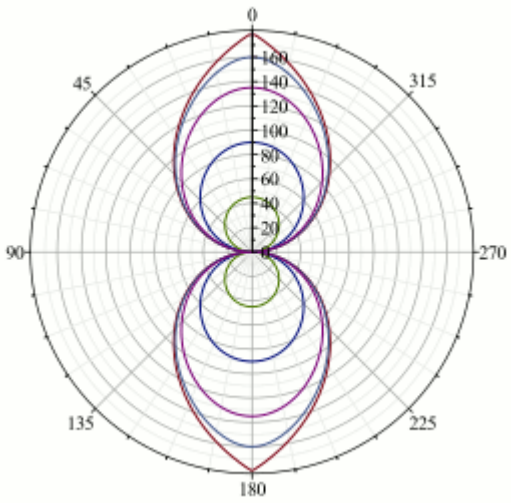
The channels values for *Swing and Y Twist* are:

Y Rotation
: True angle of the twist rotation.

W Rotation
: True angle of the swing rotation, independent of its direction.

X Rotation, Z Rotation
: Weighted angles that represent the amount of swing around the X/Z axis.

  The magnitude of the angle equals *W Rotation* when the rotation is purely around that axis, and fades out to zero as the direction changes toward the other axis, following the falloff curves from the graph on the right.



Falloff curves for weighted angles.

Mathematically, the swing angles are computed from quaternion components, using $2 \arccos(w)$ for *W* and $2 \arcsin(x)$ etc. for the others. The component of the swing rotation that corresponds to the twist axis is always 0, and is replaced by the twist angle.

Expressions

Expression
: A text field where you can enter an arbitrary Python expression that refers to *Driver Variables* by their names.

  The expression has access to a set of standard constants and math functions from `math`, `bl_math` and other modules, provided in the *Driver Namespace*. For an example of adding a custom function to the namespace, see the driver namespace example.

  For performance reasons it is best to use the Simple Expressions subset as much as possible.

Use Self
: If this option is enabled, the variable `self` can be used for drivers to reference their own data. Useful for objects and bones to avoid having creating a *Driver Variable* pointing to itself.

  Example: `self.location.x` applied to the Y rotation property of the same object will make the object tumble when moving.

  Note that dependencies for properties accessed via `self` may not be fully tracked.

Simple Expressions

Blender can evaluate a useful subset of Python driver expressions directly, which significantly improves performance, especially on multi-core systems. To take advantage of this, the driver expression must only use the following features:

Variable Names
: Use only ASCII characters.

Literals
: Floating-point and decimal integer.

Globals
: `frame`

Constants
: `pi`, `True`, `False`

Operators
: `+`, `-`, `*`, `/`, `==`, `!=`, `<`, `<=`, `>`, `>=`, `and`, `or`, `not`, conditional operator/ ternary if

Standard Functions
: `min`, `max`, `radians`, `degrees`, `abs`, `fabs`, `floor`, `ceil`, `trunc`, `round`, `int`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `exp`, `log`, `sqrt`, `pow`, `fmod`

Blender Provided Functions
: `lerp`, `clamp`, `smoothstep`

Simple expressions are evaluated even when Python script execution is disabled.

When an expression outside of this subset is used, Blender displays a “Slow Python expression” warning. However, as long as the majority of drivers use simple expressions, using a complex expression in select few is OK.

See also

- Extending Blender with Python.
- Python and its documentation.
- functions.wolfram.com.

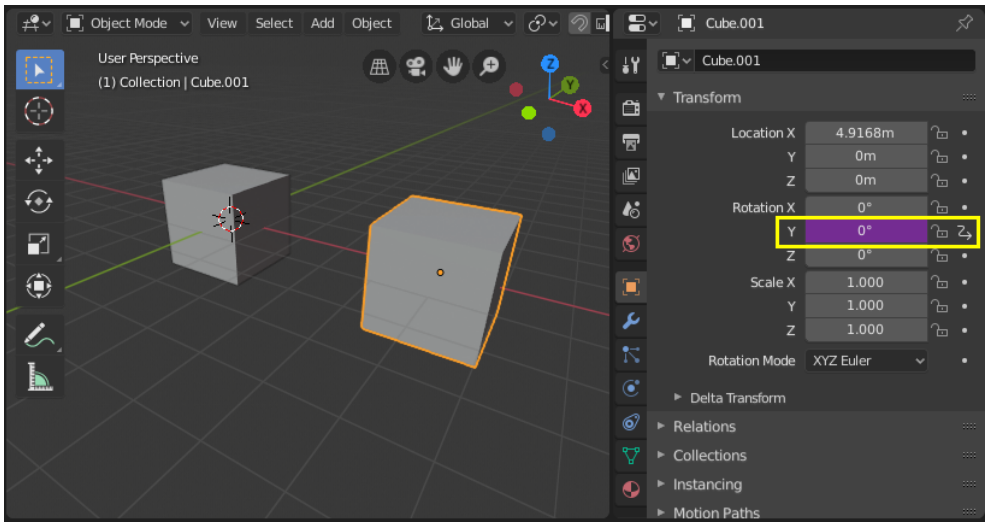
Workflow & Examples

Simple Drivers can be configured from the pop-over that appears when adding a new Driver. When adding multiple Drivers or for more advanced configurations, it is useful to have open the Drivers Editor.

Transform Driver

Control a property with an object's transform. In this example, the Y rotation of Object 2 will be driven by the X position of Object 1. Starting from a simple setup with two objects:

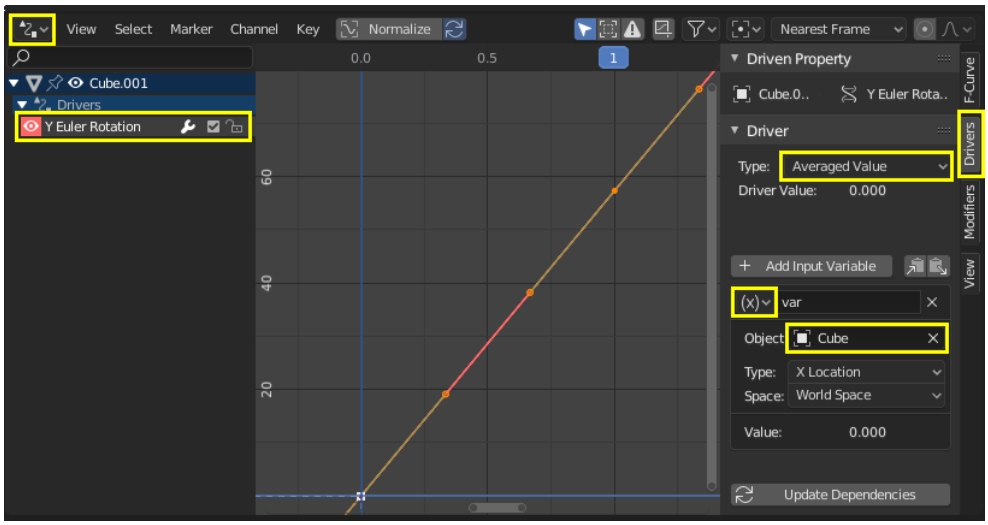
1. Add a Driver to the *Rotation Y* property of the second object via the context menu or with `Ctrl-D`.



2. Open the *Drivers Editor* and select the *Y Euler Rotation* property in the channels region.

3. Open the Sidebar region and select the *Drivers* tab.

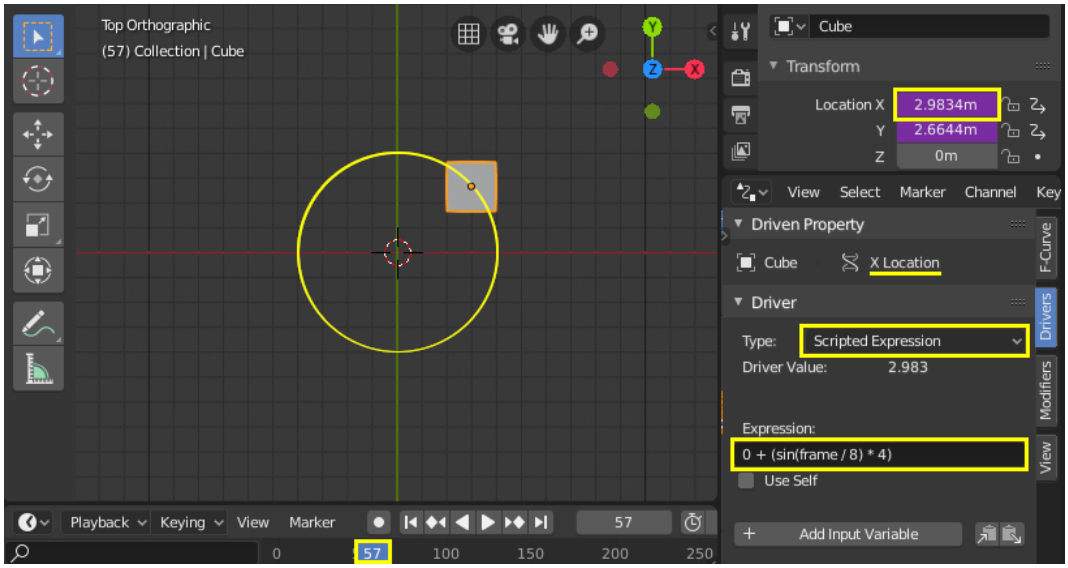
4. Configure the driver to be the *Averaged Value* of a *Transform Channel* of the first object.



5. Experiment with moving the first object and notice how it affects the Y rotation of the second object.

Scripted Expression - Orbit a Point

Orbit an object's position around a point with a custom *Scripted Expression*. The object's position will change when scrubbing the timeline. Using trigonometry, circular motion can be defined in 2D using the sine and cosine functions. (See Unit Circle.) In this example, the current frame is used as the variable that induces the motion. `frame` is a Simple Expression that corresponds to `bpy.context.scene.frame_current`.



1. Add a driver to the X Location property.
    1. Set the *Driver Type* to *Scripted Expression*.
    2. Add the expression `0 + (sin(frame / 8) * 4)`, where:
        - `frame/8` : is the current frame of the animation, divided by 8 to slow the orbit down.
        - `(sin( )*4)` : multiplies the result of `sin(frame/8)` by 4 for a bigger circle.
        - `0 +` : is used to control the offset to the orbit center point.
2. Add a driver to the Y Location property with the expression `0 + (cos(frame / 8) * 4)`.
3. Scrub the timeline to see the effect. Experiment with the variables to control the size and center of the orbit.

Custom Function - Square Value

Create a custom function to get the square of a value (i.e. $value^2$). Adding the function to the *Driver Namespace* allows it to be used from driver expressions. The *Driver Namespace* has a list of built-in functions for use in driver expressions, as well as constants such as π and e. These can be inspected via the Python Console:

```
>>> bpy.app.driver_namespace[' <tab>
                             acos']
                             acosh']
                             asin']
                             asinh']
                             atan']
                             ...
```

To add a new function to the *Driver Namespace*, the function itself needs to be implemented and then added to the `bpy.app.driver_namespace`.

1. Add the following to the Text Editor inside Blender and press *Run Script*.

    ```
    import bpy

    def square(val):
    ```

```
    """Returns the square of the given value"""
    return val * val

# Add function to driver_namespace.
bpy.app.driver_namespace['square'] = square
```

2. Add a driver with a *Scripted Expression* such as `square(frame)`.

3. Observe the effect when scrubbing the timeline.

There are more custom function examples available in Blender's Text Editor *Templates* ▸ *Python* ▸ *Driver Functions*. Since Simple Expressions cannot access custom functions, using them only makes sense for complex computations.

Warning

Trying to replace built-in entries of the driver namespace may result in undefined behavior.

View Layer Attribute Lookup

The material Attribute Node in the View Layer mode automatically searches for the attribute property in multiple locations. This, for example, can allow setting a certain value of the custom attribute at the Scene or World level, and then overriding it differently for one View Layer.

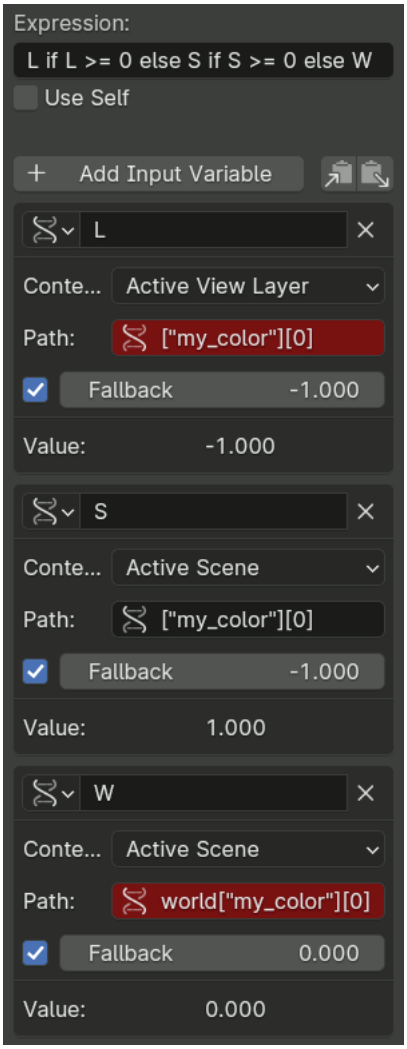
Context Properties of drivers don't implement this behavior, so if necessary it has to be manually emulated via fallback values and a conditional expression (conditions are Simple Expressions).

For an attribute named `attr`, the node tries the following six RNA path lookups in order:

- `["attr"]` in the active View Layer (custom property).
- `attr` in the active View Layer (built-in property).
- `["attr"]` in the active Scene.

- `attr` in the active Scene.
- `world["attr"]` in the active Scene.
- `world.attr` in the active Scene.

Depending on the specific property it may be sufficient to check only a subset of these locations. For example, the image on the right shows how to access an attribute that is known to definitely be a custom property with a color value.

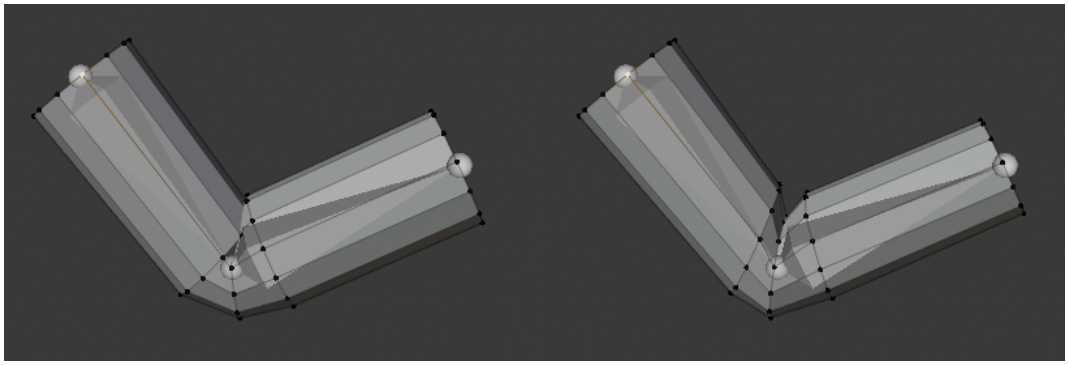


Driver variables accessing locations that are not final in the lookup chain should use fallback values that are invalid for the attribute (e.g. negative color values), which can then be checked by the conditional expression. The final variable should fallback to a valid default value to be used when the property is not set at all.

Shape Key Drivers

Improved Mesh Deformation

Fix intersection problems that happen when using armatures and weight painting, especially at joints. Shape keys can also be used to tweak and refine a rig, for example to suggest muscle formations. In this example, a shape key is used to improve the deformation at the elbow of a rudimentary arm.



Left: Skeletal mesh deformation without correction. Right: Corrective shape key applied

Setup

1. Add a mesh (in this example, a cylinder with loop cuts).
2. Add an armature with a chain of bones.
3. Skin the mesh to the armature using weight painting.

(Note: to parent the mesh to the armature: select the mesh first, then the armature and use `Ctrl-P` to parent with auto weights.)

Experiment with posing the armature and observe the deformation at the joint. To fix intersection problems or angles that look unsatisfactory, you can associate a Shape Key with a pose.

Shape Key

1. Pose the armature such that the problems are visible. Be sure to cover the extreme poses that you want to support for the rig.
2. With the mesh selected, add a new *Shape Key* in addition to the *Basis* key. *Properties* ▸ *Mesh tab* ▸ *Shape Keys*
3. In order to author the shape key on top of the armature deformation, enable both *Edit Mode Display* and *Cage Editing* in the Armature modifier. *Properties* ▸ *Modifiers tab* ▸ *Armature Modifier* ▸ *Header*
4. Enter Edit Mode and select the new shape key in the properties panel. Adjust the vertices as desired. Select the *Basis* key to toggle between the original mesh and your edits. (Note: be careful to apply edits only to your shape and not to the original mesh or other existing keys.)

Once you are satisfied with how the deformation looks for the problematic pose, you'll need to configure a driver to activate the shape smoothly when entering that position.

Driver

1. Add a driver to the *Value* of the shape key you've created.
2. Open the Drivers Editor and select the driver.

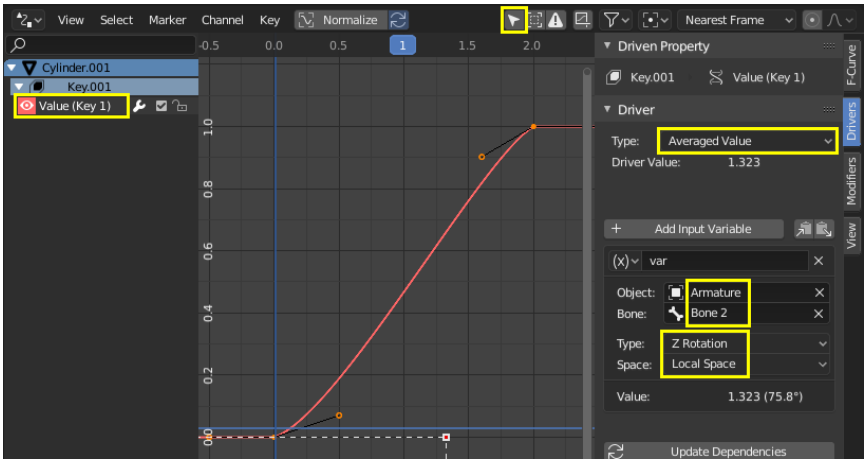
Method 1 – Direct mapping to a bone rotation value

A simple way to configure the driver is with a direct correspondence of the value of a bone's rotation channel to the shape key activation *Value*. This method has the disadvantage of relying on a single channel of a bone's rotation which might be insufficient to precisely express the condition under which the shape key should be activated.

1. In the Drivers tab, select the *Averaged Value* of the rotation of the bone you are posing.

   Understand the rotation axis that you are interested in by enabling axes display in the armature or by observing the bone's transform values in the Properties.

   Select the rotation channel and set it to local, meaning, the bone's rotation value relative to its parent bone.



2. Manually set points in the driver curve by selecting a handle and dragging it or inserting values in the *F-Curve* tab. The Y

axis represents the shape key *Value*, which should go from 0.0 to 1.0. The X axis is usually the frame, but for this driver it represents the rotation value in radians. You can have more than two points in the curve and tweak the transitions with the handles in the curve view (`G` to move).

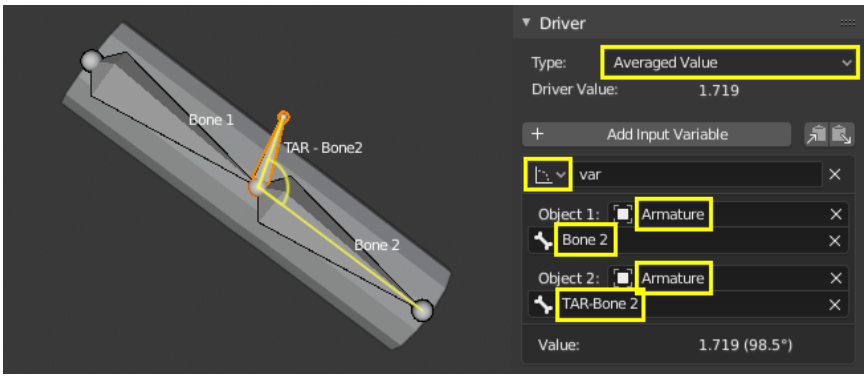
3. To verify that the driver behaves correctly, deselect the option to only show drivers for selected objects. This way, you can pose the armature and keep an eye on the driver.

Method 2 – Rotational difference to a target bone

This method requires an additional *target* or *corrective* bone, but it better expresses the spatial condition in 3D space of the bone that is causing the problem.

1. In armature Edit Mode, add a new bone extruded from Bone 1, in the position at which Bone 2 should have the shape key active. This type of bones usually follow a naming convention such as “TAR-” (target) or “COR-” (corrective).

2. In the Drivers tab, select the *Averaged Value* of the rotational difference between the bone you are rotating and the target bone. A rotational difference is the minimum angle between two objects in World Space. It is therefore important that the bones have the same root, so that the only thing affecting the angle between the bones is the rotation of one of them. When the deformation bone (Bone 2) reaches the target rotation (TAR-Bone 2) the rotational difference will be 0°.



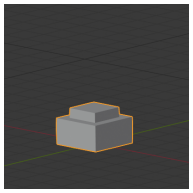
3. Manually adjust the driver curve handles so that the shape key *Value* (Y axis) is 1.0 when the rotational difference (X axis) is 0°. The *Value* should be 0.0 when the arm is extended, at which point the rotational difference should be around 90° or more (in radians).

4. See the steps in Method 1 on how to adjust the curve handles and confirm that the functionality is working. Pose the armature to verify that the ranges are correct.

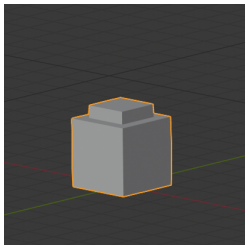
Chained Relative Shape Keys

Activate different shape keys in succession. In this example, moving a single bone will activate first *Key 1* and then *Key 2*. See also relative shape keys mix additively.

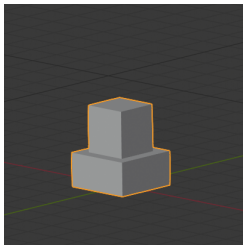
Shape Keys
: Add two shape keys to a mesh, besides the *Basis*.



Basis.

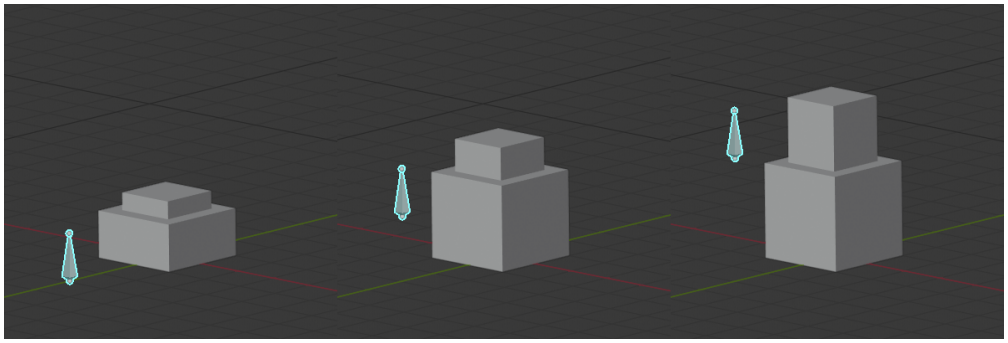


Key 1: top faces moved up by 1 m.



Key 2: inner top moved up by 1 m.

Drivers
: Add an armature with a single bone to control the shape keys. The goal is to activate the keys in succession as this bone moves up.



As shown in the picture above, when the bone is halfway up, both *Key 1* and *Key 2* have an influence. It is a matter of preference if *Key 1* should be at its maximum *Value* before *Key 2* starts to become active, or how much they should overlap. This example shows a seamless blend.

For a seamless blend where there is overlap, *Key 1* should have a *Value* of 0.0 when the bone is at the bottom and increase linearly to 1.0 until the bone is past the midpoint height. *Key 2* should have a value of 0.0 before the midpoint height and then increase at the same rate than *Key 1* until reaching *Value* 1.0 when the bone is at maximum height.

1. Add a driver to the *Value* of *Key 1* and *Key 2*. In the *Drivers* tab, configure both drivers to be the *Averaged Value* of a variable with the bone's Z location.

2. Determine the range of the bone's motion in the World Z axis by moving it up so that it is aligned with the top of the mesh when both keys are active. Here we will use [0.0, 2.5].

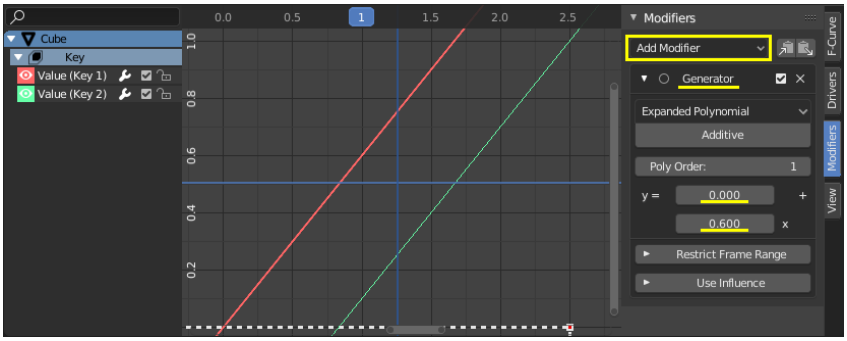
3. Configure the driver functions so that the *Value* of the shape keys (Y axis) is as desired for the bone's height (X axis).

   The driver functions should be linear, therefore, they can be defined analytically with a function of type \(y = a + bx\), where \(a\) is an offset in \(y\) and \(b\) is the slope.

   1. In the *Modifiers* tab, add a *Generator* of type *Extended Polynomial* for both drivers.

2. Play with the values of \(a\) and \(b\) so that the curves go from [0.0, 1.0] in the Y axis and from [0.0, 2.5] in the X axis. The curves should overlap in the mid area of the X axis and they should have the same slope (\(b\)).

   Possible values are *Key 1*: \(y = 0.0 + 0.6x\) and *Key 2*: \(y = -0.5 + 0.6x\).

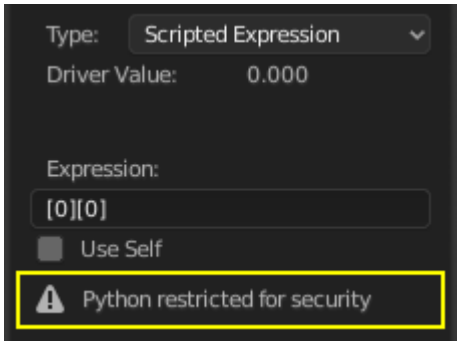


   Note that the functions go outside the range [0.0, 1.0] for the shape keys' *Value*, but that has no effect because *Value* is clamped in a *Range* in the *Shape Keys* panel.

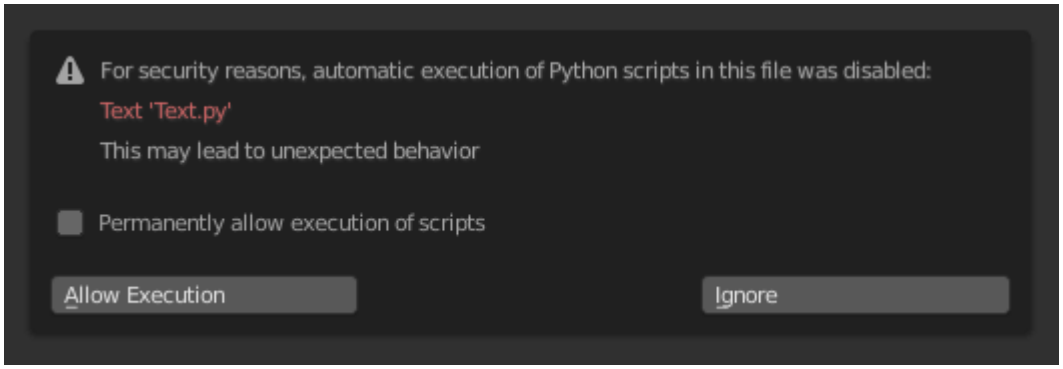
Troubleshooting

Some common problems people may run into when using drivers.

Scripted Expression



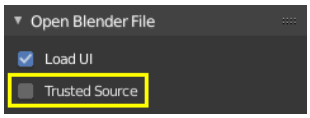
A security warning in the Drivers panel.



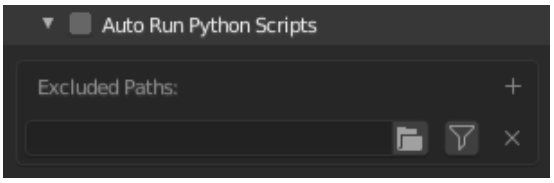
An Auto-run warning in the Info editor's header.

By default Blender will restrict execution of Python scripts.

If using a *Scripted Expression* Driver Type that doesn't follow the Simple Expressions subset, you will have to open the file as *Trusted Source*, or set *Auto Run Python Scripts* in *Preferences* ‣ *Save & Load* ‣ *Blender Files*.



The Trusted Source checkbox in the File Browser.



The Auto Run Python Scripts checkbox in the Preferences.

Rotational Properties are Radians

Parts of the User Interface may use different units of measurements for angles, rotation. In the Graph Editor, while working with Drivers, all angles are Radians.

Markers

Markers are used to denote frames with key points or significant events within an animation. E.g. it could be that a character's animation starts, the camera changes position, or a door opens. Markers can be given names to make them more meaningful at a quick glance. They are available in many of Blender's editors.

Note

Unlike keyframes, markers are always placed at a whole frame number, you cannot set a marker at frame 2.5.

Markers can be created and edited in the following editors:

- Graph Editor
- Dope Sheet
- NLA Editor
- Video Sequence Editor
- Timeline

Note

A marker created in one of these editors will also appear in all others that support them.

Types

Besides standard markers, *pose markers* are another type of markers, which are specific to armatures and shape keys. They are used to denote poses in the Action Editor mode and Shape Keys Editor of Dope Sheet.

Visualization

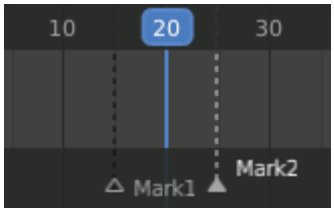
In the supported editors, if at least one is created, markers are visualized in a separate row at their bottom. This area can be disabled per editor via the *View ▸ Show Markers* menu option.

Note

While the markers area is disabled, markers operators are not available in that editor, and in the header the *Marker* menu is hidden.

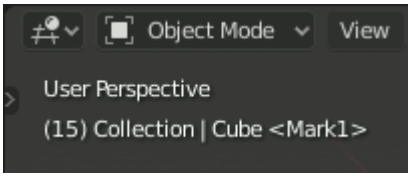
Standard

Regular markers are shown as small white triangles, empty if unselected or filled if selected, and with a dashed line that covers the editor height at the corresponding frame. If they have a name, this is shown to their right in white.



3D Viewport

The 3D Viewport does not allow you to create, edit or remove markers, but it shows their name in the Object Info in the upper left corner, when on their frame.



Pose Markers

Pose markers show a diamond-shaped icon in the Dope Sheet. In the NLA editor pose markers are shown as a red dashed line inside the relative action strip.



Add Marker

Reference

Mode::	All modes
Menu::	*Marker* ‣ *Add Marker*
Shortcut::	`M`

The simplest way to add a marker is to move to the frame where you would like it to appear, and press `M`.

Hint

Markers can also be added during playback.

Pose Markers

If *Show Pose Markers* is checked, a pose marker and a new pose in the Old Pose Library are added.

Selecting

Reference

Mode::	All modes
Shortcut::	`LMB`

Click `LMB` on the marker's triangle to select it. Use `Shift-LMB` to select multiple markers.

In the Graph Editor, Dope Sheet, NLA Editor, Timeline, and Video Sequence Editor, you can also select all markers with `A` while hovering the mouse over the marker row, and apply selection tools on them like Box

Select, etc. (as usual, LMB to select, RMB to deselect). The corresponding options are found in the Select menu of these editors.

Editing

Duplicate Marker

Reference

Mode::	All modes
Menu::	*Marker ▸ Duplicate Marker*
Shortcut::	Shift-D

You can duplicate the selected markers by pressing Shift-D. Once duplicated, the new ones are automatically placed in select mode, so you can move them to the desired location.

Note

Note that unlike most other duplications in Blender, the names of the duplicated markers are not altered at all (no .001 numeric counter append).

Duplicate Marker to Scene

Reference

Mode::	All modes
Menu::	*Marker ▸ Duplicate Marker to Scene…*

Duplicates the selected markers into another scene.

Delete Marker

Reference

Mode::	All modes
Menu::	*Marker ▸ Delete Marker*
Shortcut::	`X`

To delete the selected markers simply press `X`, and confirm the pop-up message with `LMB`.

Rename Marker

Reference

Mode::	All modes
Menu::	*Marker ▸ Rename Marker*
Shortcut::	`F2`

Having dozens of markers scattered throughout your scene's time will not help you much unless you know what they stand for. You can name a marker by selecting it, pressing `F2`, typing the name, and press `Return`

Move Marker

Reference

Mode::	All modes
Menu::	*Marker ▸ Move Marker*
Shortcut::	`G`

Once you have one or more markers selected, press `G`, while hovering with the mouse over the marker bar, to move them, and confirm the move with `LMB` or `Return` (as usual, cancel the move with `RMB`, or `Esc`). Or drag them with the `LMB`.

By default, you move the markers in one-frame steps, but if you hold `Ctrl`, the markers will move in steps corresponding to 1 second (according to the

scene's *FPS*).

Select

Reference

Mode:: All modes
Menu:: *Marker ▸ Select*

Convenient operators for selecting Marks; see Selecting for more information on selecting Markers.

All `A`
: Selects all Markers.

None `Ctrl-A`
: Deselects any already selected Markers.

Invert `Ctrl-I`
: Select all unselected Markers and deselects all selected Markers.

Before Current Frame
: Selects all Markers to the left of the current frame and the Marker on the current frame if it exists.

After Current Frame
: Selects all Markers to the Right of the current frame and the Marker on the current frame if it exists.

Show Pose Markers

Reference

Editor:: Dope Sheet
Mode:: Action Editor or Shape Keys Editor mode
Menu:: *Marker ▸ Show Pose Markers*

Shows markers belonging to the active action instead of scene markers.

Make Markers Local

Reference

Mode:: All modes
Menu:: *Marker ▸ Make Markers Local*

It is possible to convert standard markers into pose markers with *Marker ▸ Make Markers Local*. Note that the original marker will be gone. If you want to keep it, make a duplicate before you convert.

Jump to Next/Previous Marker

Reference

Mode:: All modes
Menu:: *Marker ▸ Jump to Next/Previous Marker*

Moves the Playhead to the next/previous marker relative to the current frame.

Bind Camera to Markers

Reference

Editor:: Timeline
Menu:: *Marker ▸ Bind Camera to Markers*
Shortcut:: `Ctrl-B`

Bind Camera to Markers is a special operator only available in the *Timeline*. The operator allows markers to be used to set the active object as the active camera.

To use this operator, select the object to become the active camera and select a marker to bind the active camera to. If no marker is selected when the operator is applied, a marker will be added. When an object is bound to a marker, the marker will be renamed to the name of the active object. These markers also have a camera icon next to the left of the name to easily distinguish them from other informative markers.

These markers can be moved to change the frame at which the active camera is changed to the object the marker is bound to.

Shape Keys

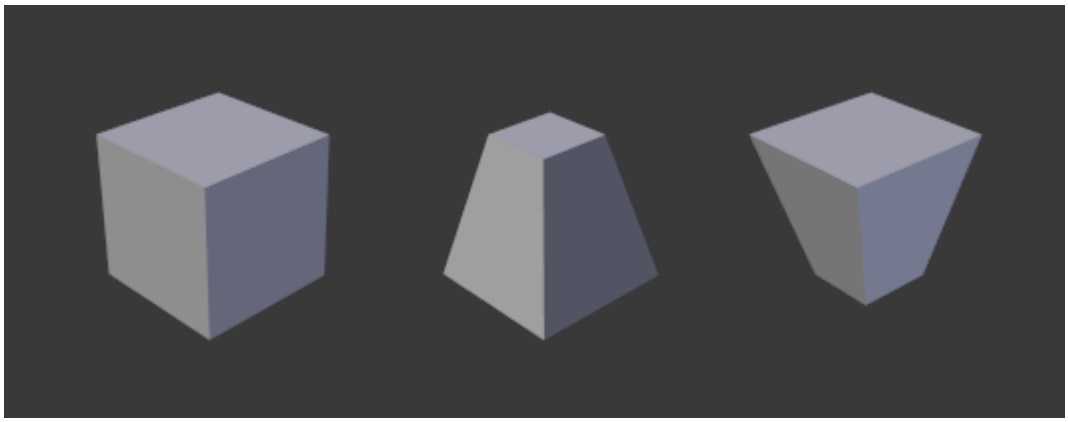
- Introduction
  - Workflow
  - Relative or Absolute Shape Keys
- Shape Keys Panel
  - Relative Shape Keys
  - Absolute Shape Keys
- Workflow
  - Relative Shape Keys
  - Absolute Shape Keys

Introduction

Shape keys are used to deform objects into new shapes for animation. In other terminology, shape keys may be called “morph targets” or “blend shapes”.

The most popular use cases for shape keys are in character facial animation and in tweaking and refining a skeletal rig. They are particularly useful for modeling organic soft parts and muscles where there is a need for more control over the resulting shape than what can be achieved with combination of rotation and scale.

Shape keys can be applied on object types with vertices like mesh, curve, surface and lattice.



Example of a mesh with different shape keys applied.

Workflow

Shape keys are authored in the Shape Keys panel which is accessed in the Object Data tab of the Properties (e.g. the Mesh tab for mesh objects).

A shape key is modified by first selecting a shape key in the panel, and then moving the object’s vertices to a new position in the 3D Viewport.

The panel has controls for affecting the current *Value* (influence, weight) of a shape. It is possible to see a shape in isolation or how it combines with others.

Adding and Removing Vertices

It is not possible to add or remove vertices in a shape key. The number of vertices and how they connect is specified by the mesh, curve, surface or lattice. A shape key merely records a position for each vertex and therefore shapes always contain all the object's vertices.

When adding a vertex, all shape keys will record it with the position in which it is created. Workflow-wise, adding and deleting vertices after creating shape keys is possible, but it is best to leave the creation of shape keys for when the mesh is finished or its topology is stable.

Adding Shape Keys

When adding a new shape key with the + button next to the list, the new shape will be a copy of the Basis shape, independently of the current result visible in the 3D Viewport.

When adding a new shape key from *Specials* ▸ *New Shape from Mix*, the shape will start of with the vertex configuration that is visible at that moment.

When doing facial animation with relative shape keys, it can be useful to first create a shape key with a complex extreme pose (e.g. anger or surprise), and then break this complex shape into components by applying a temporary vertex group to the complex shape and creating a copy with *New Shape from Mix*. This technique helps reducing conflicts between different shape keys that would otherwise produce a double effect.

Relative or Absolute Shape Keys

A mesh (curve, surface or lattice) has a stack of shape keys. The stack may be of *Relative* or *Absolute* type.

Relative

Mainly used for muscles, limb joints, and facial animation.

Each shape is defined relative to the Basis or to another specified shape key.

The resulting effect visible in the 3D Viewport, also called *Mix*, is the cumulative effect of each shape with its current value. Starting with the Basis shape, the result is obtained by **adding** each shape's weighted **relative** offset to its reference key.

Value

Represents the weight of the blend between a shape key and its reference key.

A value of 0.0 denotes 100% influence of the reference key and 1.0 of the shape key. Blender can extrapolate the blend between the two shapes above 1.0 and below 0.0.

Basis

Basis is the name given to the first (top-most) key in the stack.

The Basis shape represents the state of the object's vertices in their original position. It has no weight value and it is not keyable. This is the default *Reference Key* when creating other shapes.

Absolute

Mainly used to deform the objects into different shapes over time.

Each shape defines how the object's shape will be at *Evaluation Time* specified in its *Value*.

The resulting shape, or *Mix*, is the interpolation of the previous and next shape given the current *Evaluation Time*.

Value

Represents the *Evaluation Time* at which that shape key will be active.

Basis

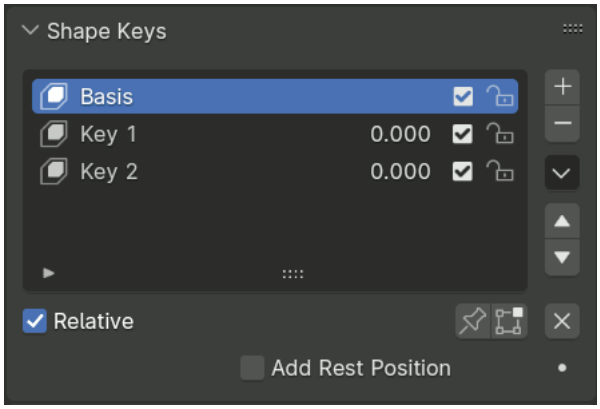
Basis is the name given to the first (topmost) key in the stack.

The Basis shape represents the state of the object's vertices in their original position.

Shape Keys Panel

Reference

Editor:: Properties
Mode:: All modes
Panel:: *Object Data* ▸ *Shape Keys*



Shape Keys panel.

The Shape Keys panel is used for authoring shape keys.

Active Shape Key Index
A List View.

Value/Frame (number)
In Relative mode: Value is the current influence of the shape key used for blending between the shape (value=1.0) and its reference key (value=0.0). The reference key is usually the Basis shape. The weight of the blend can be extrapolated above 1.0 and below 0.0.

In Absolute mode: Value is the *Evaluation Time* at which the shape will have maximum influence.

Mute (check mark)
If unchecked, the shape key will not be taken into consideration when mixing the shape key stack into the result visible in the 3D Viewport.

Lock Shape (lock icon)
Shape keys can be locked to protect them from accidental modification due to inadvertently selecting the wrong key for editing

in the list. Most common sculpt and edit mode operators and tools that move vertices abort with an error if the active shape key is locked.

Note

Operators that always modify all shape keys in exactly the same way, like Apply Object Transforms, don't check shape key locks. Neither currently do most edit mode operators that modify topology, because the topology is expected to usually be finalized before shape keys are created.

Shape Key Specials

New Shape from Mix

Add a new shape key with the current deformed shape of the object. This differs from the + button of the list, as that one always copies the Basis shape independently of the current mix.

Mirror Shape Key

If your mesh is symmetrical, in *Object Mode*, you can mirror the shape keys on the X axis. This will not work unless the mesh vertices are perfectly symmetrical. Use the *Mesh ▸ Symmetrize* tool in *Edit Mode*.

Mirror Shape Key (Topology)

Same as *Mirror Shape Key* though it detects the mirrored vertices based on the topology of the mesh. The mesh vertices do not have to be perfectly symmetrical for this action to work.

Join as Shapes (Transfer Mix)

Transfer the current resulting shape from a different object.

Select the object to copy, then the object to copy into. Use this action and a new shape key will be added to the active object with the current mix of the first object.

Transfer Shape Key

Transfer the active shape key from a different object regardless of its current influence.

Select the object to copy, then the object to copy into. Use this action and a new shape key will be added to the active object with the active shape of the first object.

Delete All Shape Keys
: Removes all Shape Keys and any effect that they had on the mesh.

Apply All Shape Keys
: Saves the current visible shape to the mesh data and deletes all Shape Keys.

Relative
: Set the shape keys to *Relative* or *Absolute*. See Relative or Absolute Shape Keys.

Shape Key Lock (pin icon)
: Show the active shape in the 3D Viewport without blending. *Shape Key Lock* gets automatically enabled while the object is in *Edit Mode*.

Shape Key Edit Mode (edit mode icon)
: If enabled, when entering *Edit Mode* the active shape key will **not** take maximum influence as is default. Instead, the current blend of shape keys will be visible and can be edited from that state.

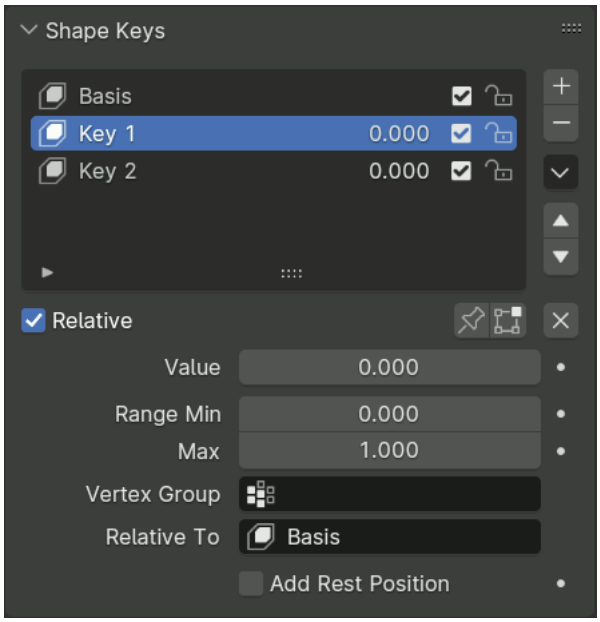
Add Rest Position
: Creates an Attribute in the vertex domain called `rest_position` which is a copy of the `position` attribute before shape keys and modifiers are evaluated. Only mesh objects support this option.

Relative Shape Keys

See Relative or Absolute Shape Keys.

With relative shape keys, the value shown for each shape in the list represents the current weight or influence of that shape in the current *Mix*.

(Clear Shape Keys)
: Set all influence values, or weights, to zero. Useful to quickly guarantee that the result shown in the 3D Viewport is not affected by shapes.



Relative Shape Keys options.

Value
: The weight of the blend between the shape key and its reference key (usually the Basis shape).

  A value of 0.0 denotes 100% influence of the reference key and 1.0 of the shape key.

Range
: Minimum and maximum range for the influence value of the active shape key. Blender can extrapolate results when the *Value* goes lower than 0.0 or above 1.0.

Vertex Group
: Limit the active shape key deformation to a vertex group. Useful to break down a complex shape into components by assigning temporary vertex groups to the complex shape and copying the result into new simpler shapes.

Relative To
: Select the shape key to deform from. This is called the *Reference Key* for that shape.

Note

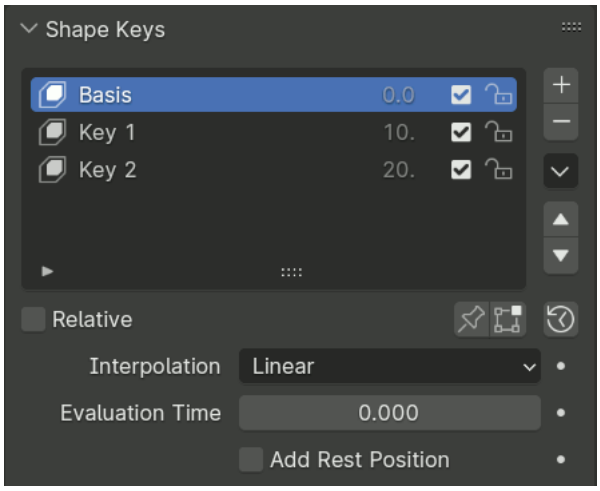
Rather than storing offsets directly, internally relative keys are stored as snapshots of the mesh shape. The relative deformation offsets are computed by subtracting *Reference Key* from that snapshot.

Therefore, replacing the *Reference Key* has the effect of subtracting the difference between the new and old reference from the relative deformation of the current key.

Absolute Shape Keys

See Relative or Absolute Shape Keys.

With absolute shape keys, the value shown for each shape in the list represents the *Evaluation Time* at which that shape key will be active.

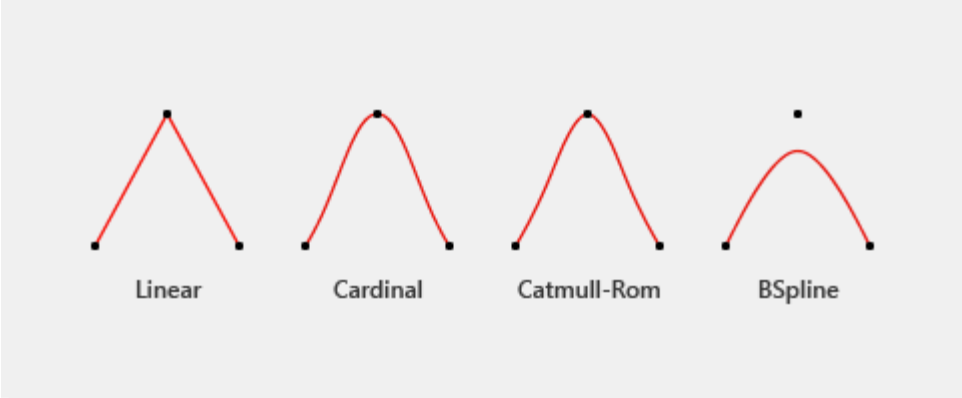


Absolute Shape Keys options.

Re-Time Shape Keys (clock icon)
: Absolute shape keys are timed, by order in the list, at a constant interval. This button resets the timing for the keys. Useful if keys were removed or re-ordered.

Interpolation
: Controls the interpolation between shape keys.

  Linear, Cardinal, Catmull-Rom, B-Spline



Different types of interpolation.

The red line represents interpolated values between keys (black dots).

Evaluation Time

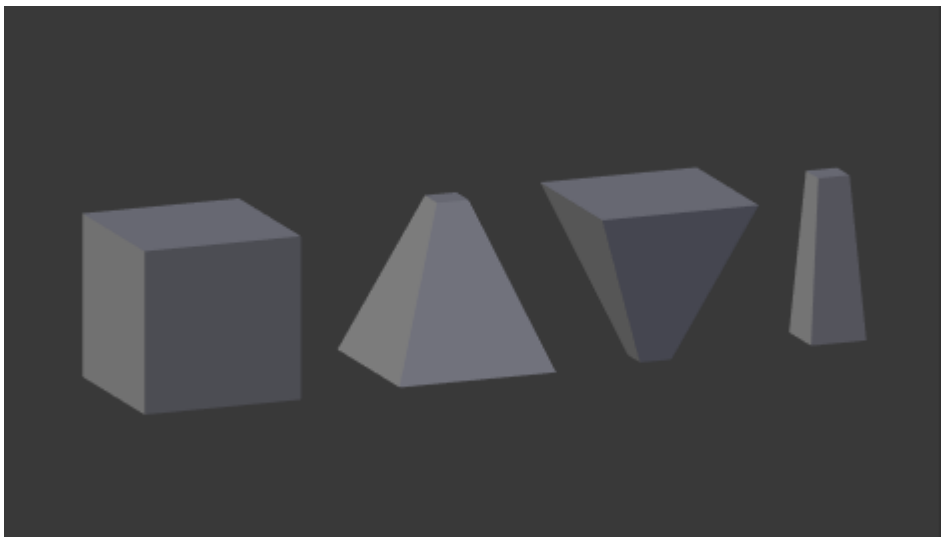
Controls the shape key influence. Scrub to see the effect of the current configuration. Typically, this property is keyed for animation or rigged with a driver.

Workflow

Relative Shape Keys

1. In *Object Mode*, add a new shape key via the *Shape Key* panel with the + button.
2. “Basis” is the rest shape. “Key 1”, “Key 2”, etc. will be the new shapes.
3. Switch to *Edit Mode*, select “Key 1” in the *Shape Key* panel.
4. Deform mesh as you want (do not remove or add vertices).
5. Select “Key 2”, the mesh will be changed to the rest shape.
6. Transform “Key 2” and keep going for other shape keys.
7. Switch back to *Object Mode*.
8. Set the *Value* for “Key 1”, “Key 2”, etc. to see the transformation between the shape keys.

In the figure below, from left to right shows: “Basis”, “Key 1”, “Key 2” and mix (“Key 1” `1.0` and “Key 2” `0.8`) shape keys in Object Mode.

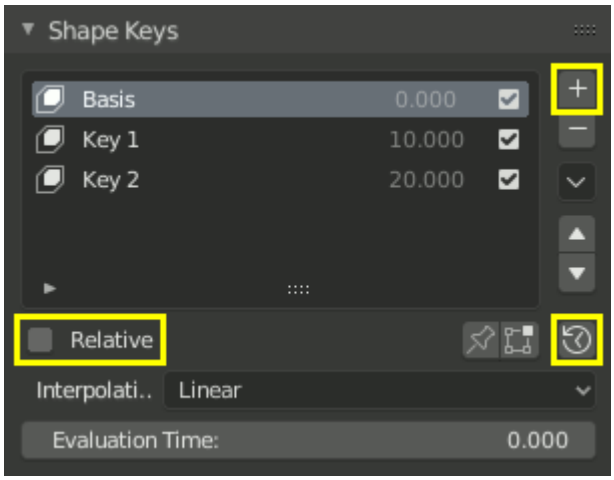


Relative shape keys example.

For more practical examples, see how to combine shape keys and drivers.

Absolute Shape Keys

1. Add sequence of shape keys as described above for relative shape keys.
2. Uncheck the *Relative* checkbox.
3. Click the *Reset Timing* button.
4. Switch to *Object Mode*.
5. Drag *Evaluation Time* to see how the shapes succeed one to the next.



Absolute shape keys workflow.

By adding a driver or setting keyframes to *Evaluation Time* you can create an animation.

See also

Shape Key Operators

There are two modeling tools used to control shape keys and are found in Edit Mode.

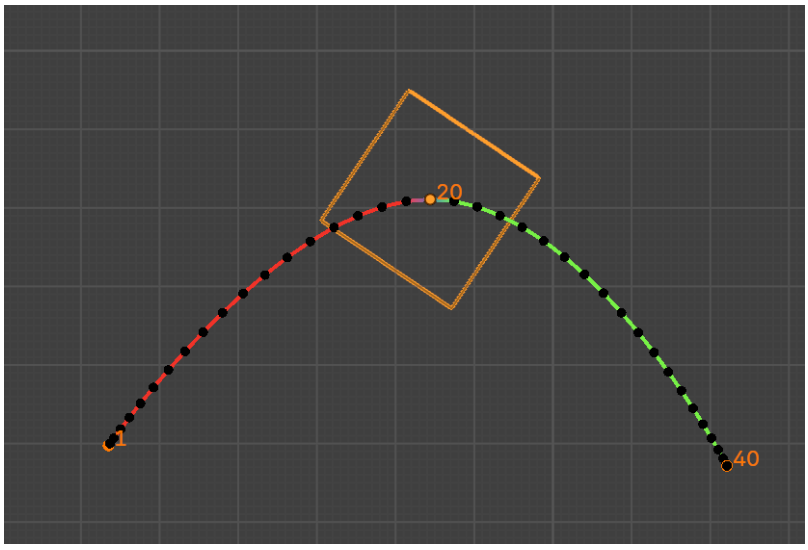
Motion Paths

Reference

Editor:: 3D Viewport, Properties
Mode:: Object Mode
Panel:: *Properties* ▸ *Object Properties* ▸ *Motion Paths*

Reference

Editor:: 3D Viewport, Properties
Mode:: Pose Mode
Panel:: *Properties* ▸ *Armature* ▸ *Motion Paths*
Menu:: *Pose* ▸ *Motion Paths*



An animated cube with its motion path displayed.

The Motion Paths tool allows you to visualize the motion of points as paths over a series of frames. These points can be object origins and bone joints.

To create or remove motion paths, it is necessary to first select the bones. Then:

1. To show the paths (or update them, if needed), click on the *Calculate Path* button.
2. To hide the paths, click on the *Clear Paths* button.

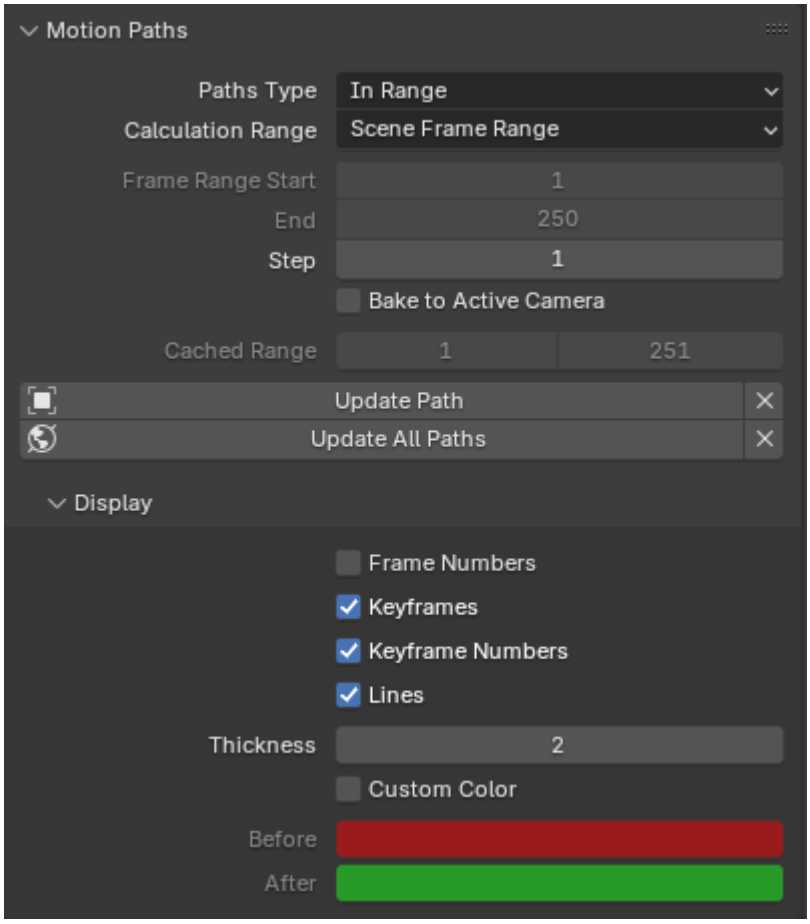
Note

Remember that only selected bones and their paths are affected by these actions!

The paths are shown in red for the section in the past and green for the section in the future. These colors follow the user preference options “Before Current Frame” and “After Current Frame”, which can be found in the 3D Viewport section. Each frame is displayed by a small dot on the paths.

The paths are automatically updated when you edit your poses/keyframes, and they are also active during animation playback. Playing the animation affects the paths only when using the *Around Frame* type.

Options



The Motion Paths panel in the Armature tab.

Paths Type

Type of range to show for Motion Paths.

Around Frame:: Display paths of points within a fixed number of frames around the current frame. When you enable this button, you get paths for a given number of frames before and after the current one

In Range:: Display paths of points within specified range.

Calculation Range

The range of the motion path. Only active when *Paths Type* is set to *In Range*. Changing this option only takes effect when updating the path, via the *Update Path* or *Update All Paths* buttons.

All Keys:: Generate a motion path ranging from the first keyframe to the last. Only the keys of the active object / bone are considered.

Selected Keys:: Same as *All Keys* except that it ranges from the first to the last *selected* keyframe.

Scene Frame Range:: Use the start & end frames of the scene, or the preview range if active.

Manual Range:: Manually set the start and end frame.

Frame Range Start, End

Starting and Ending frame of range of paths to display/calculate (not for the *Around Frame* type).

Although the start and end frame are always editable, updating the motion path will change these according to the *Calculation Range* setting. To ensure your chosen frame range is maintained, choose *Manual Range* there.

Frame Range Before, After

Number of frames to show before and after the current frame (only for the *Around Frame* type).

Step

Allows displaying one point for every *n* frames on the path. Mostly useful when you enable the frame number display (see below), to avoid cluttering the 3D Viewport.

Bake to Active Camera

When enabled the motion path is calculated in screen space for the active scene camera. Note that the resulting motion path will only be useful for that single camera. Switching cameras using markers is not supported. It will only bake to the camera that is active when the bake was started.

Cache/Bone Cache From, To

These are the start/end frames of the range in which motion paths are shown. You cannot modify this range without deleting the motion path

first.

Calculate
: If no paths have been calculated, Calculate Paths will create a new motion path in cache based on the options specified in the pop-up menu or the Adjust Last Operation panel. Note, if the current context is an Armature calculating the objects motion paths, and not the bones, this operator will calculate the motion paths for all the bones within the armature as well.

  Start, End
  : These are the start/end frames of the range in which motion paths are shown. The start frame is *inclusive*, so if you set *Start* to 1, you will really see the frame 1 as starting point of the paths…

  Bake Location
  : Which point on the bones is used when calculating paths. Only available for bones while in Pose Mode.

    Heads:: Calculates the path position of the bone's heads.

    Tails:: Calculates the path position of the bone's heads.

Update Paths
: In the case a path has already been calculated, this operator will update the path shape to the current animation. To change the frame range of the calculated path, you need to delete the path and calculate it again.

  (Clear Paths)
  : Clears paths on all objects/bones or just the selected ones when holding `Shift`.

Update All Paths
: Recalculates the motion paths for all visible objects and poses.

Display

Frame Numbers
: When enabled, a small number appears next to each frame dot on the path, which is of course the number of the corresponding frame.

Keyframes
: When enabled, big yellow square dots are displayed on motion paths, showing the keyframes of their bones (i.e. only the paths of keyed bones at a given frame get a yellow dot at this frame).

+ Non-Grouped Keyframes
: For bone motion paths, it searches the whole Action for keyframes instead of in groups with matching name only (this is slower).

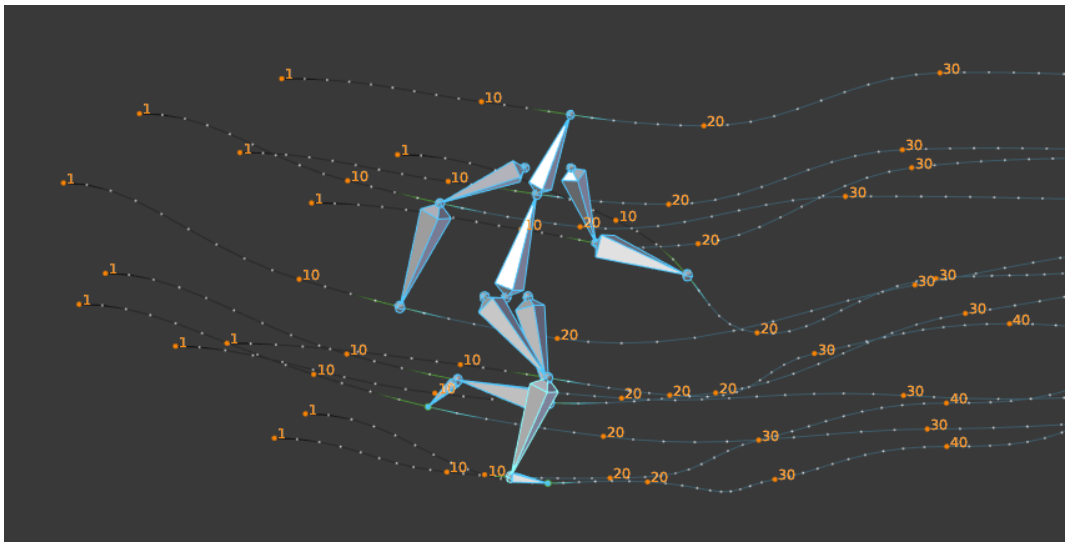
Keyframe Numbers
: When enabled, you will see the numbers of the displayed keyframes, so this option is obviously only valid when *Show Keys* is enabled.

Lines
: Toggles whether the lines between the points are shown.

Thickness
: Line thickness for motion path.

Custom Color
: Use custom color for this motion path. The custom color can be modified for time before and after the current frame.

Example



An example of a motion path of an armature.

Physics

- Introduction
  - Quick Effects
- Rigid Body
  - Introduction
  - Properties
  - Rigid Body World
  - Rigid Body Constraints
  - Tips
- Cloth
  - Introduction
  - Settings
  - Examples
- Soft Body
  - Introduction
  - Settings
  - Forces
  - Collision
  - Examples
- Fluid
  - Introduction
  - Type
  - Materials
- Particle System
  - Introduction
  - Particle System Panel
  - Emitter
  - Hair
  - Texture Influence
  - Particle Edit Mode
- Dynamic Paint
  - Introduction
  - Brush
  - Canvas

- Forces
  - Gravity
  - Force Fields
- Collision
  - Options
  - Examples
  - Hints
- Baking Physics Simulations
  - Options
  - Baking
- Simulation Nodes
  - Baking
  - Examples

Introduction

Blender's physics system allows you to simulate a number of different real-world physical phenomena. You can use these systems to create a variety of static and dynamic effects such as:

- Hair, grass, and flocks
- Rain
- Smoke and dust
- Water
- Cloth
- Jello
- etc.

Quick Effects

Reference

Editor::	3D Viewport
Mode::	Object Mode
Menu::	*Object* ▸ *Quick Effects*

Sets up a basic simulation scene or effect including the selected objects. The tool will add essential objects like domains or particle systems both with predefined settings, so that there will be instant viewable result.

Quick Fur

Adds a fur setup to the selected objects. The fur setup is based on Geometry Nodes and built with Hair Node Groups that come with Blender as bundled assets.

Density
: Surface density of generated hair curves.

Length
: Length of the generated hair curves.

Hair Radius
: The width of the hair, used for rendering engines.

View Percentage
: Factor applied on the density for the viewport.

Apply Hair Curves
: Applies the modifier that uses the Generate Hair Curves node group.

Noise
: Deforms hair curves using a noise texture. See the Hair Curves Noise node group for more information.

Frizz
: Deforms hair curves using a random vector per point to frizz them. See the Frizz Hair Curves node group for more information.

Rigid Body

- Introduction
  - Creating a Rigid Body
  - Working with Rigid Bodies
- Properties
  - Settings
  - Collisions
  - Dynamics
- Rigid Body World
  - Settings
  - Rigid Body Cache
  - Rigid Body Field Weights
- Rigid Body Constraints
  - Introduction
  - Types
- Tips
  - Animation
  - Simulation Stability
  - Combining Rigid Bodies with Other Simulations
  - Scaling Rigid Bodies

Introduction

The rigid body simulation can be used to simulate the motion of solid objects. It affects the position and orientation of objects and does not deform them.

Unlike the other simulations in Blender, the rigid body simulation works closer with the animation system. This means that rigid bodies can be used like regular objects and be part of parent-child relationships, animation constraints and drivers.

Creating a Rigid Body

Reference

Mode::	All Modes
Panel::	*Properties* ▸ *Physics* ▸ *Rigid Body*
Menu::	*Object* ▸ *Rigid Body*

Only mesh objects can be part of a rigid body simulation. To create rigid bodies, either click on the *Rigid Body* button in the *Physics* tab of the Properties or use *Add Active*/*Add Passive* in the *Object* ▸ *Rigid Body* menu.

There are two types of rigid bodies: active and passive. *Active* bodies are dynamically simulated, while *passive* bodies remain static. Both types can be driven by the animation system when using the *Animated* option.

During the simulation, the rigid body system will override the position and orientation of dynamic rigid body objects. Note however, that the location and rotation of the objects are not changed, so the rigid body simulation acts similar to a constraint. To apply the rigid body transformations you can use the *Apply Object Transform* operator.

The scale of the rigid body object also influences the simulation, but is always controlled by the animation system.

Rigid body physics on the object can be *removed* with the *Rigid Body* button in the *Physics* tab in the Properties or in the *Object* ▸ *Rigid Body* menu.

Working with Rigid Bodies

Several object operators exist for working with rigid bodies, these operators can be found in the Rigid Body object menu. These operators include functions to add/remove rigid bodies, modify their properties, and add Rigid Body Constraints.

Properties

Reference

Panel:: *Physics ▸ Rigid Body*

Type
: Role of the rigid body in the simulation.

Active:: The object is dynamic and is directly controlled by simulation results.

Passive:: The object remains static and is directly controlled by animation system, thus does not have Dynamics properties.

- Settings
- Collisions
  - Surface Response
  - Sensitivity
  - Collections
- Dynamics
  - Deactivation

Settings

Reference

Panel:: *Physics* ▸ *Rigid Body* ▸ *Settings*

Mass
: Specifies how heavy the object is and “weights” irrespective of gravity.

> **Tip**
>
> There are predefined mass presets available with the Calculate Mass operator.

Dynamic
: Enables/disables rigid body simulation for the object.

Animated
: Allows the rigid body to additionally be controlled by the animation system.

Collisions

Reference

Panel:: *Physics* ▸ *Rigid Body* ▸ *Collisions*

Shape

Determines the collision shape of the object; these can be broken into two categories: primitive shapes and mesh based shapes.

Primitive shapes (*Box*, *Sphere*, *Capsule*, *Cylinder*, and *Cone*) are best in terms of memory and performance but do not necessarily reflect the actual shape of the object. They are calculated based on the object's bounding box. The center of gravity is always in the geometric center of the shape. Primitive shapes can be shown in the 3D Viewport by enabling Bounds.

Mesh based shapes (*Convex Hull* and *Mesh*) are calculated based on the geometry of the object so they are a better representation of the object. The center of gravity for these shapes is the object origin.

Box:: Box-like shapes (e.g. cubes), including planes (e.g. ground planes). The size per axis is calculated from the bounding box.

Sphere:: Sphere-like shapes. The radius is the largest axis of the bounding box.

Capsule:: This points up the Z axis.

Cylinder:: This points up the Z axis. The height is taken from the Z axis, while the radius is the larger of the X or Y axes.

Cone:: This points up the Z axis. The height is taken from the Z axis, while the radius is the larger of the X or Y axes.

Convex Hull::

A mesh-like surface encompassing (e.g. shrink-wrapped over) all vertices (best results with fewer vertices). A convex approximation of the object, which has good performance and stability.

Mesh:: Mesh consisting of triangles only, allowing for more detailed interactions than convex hulls. Allows simulating concave objects, but is rather slow and unstable.

Compound Parent:: Takes the collision shapes from the object's children and combines them. This makes it possible to create concave shapes from primitive shapes. This usually results in a faster simulation than the *Mesh* collision shape while also being generally more stable.

Source
: Source of the mesh used to create the collision shape.

Base:: The base mesh of the object.

Deform:: Includes any deformations added to the mesh (shape keys, deform modifiers).

Final:: Includes all deformations and modifiers.

Deforming
: Mesh shapes can deform during simulation.

Surface Response

Friction
: Resistance of object to movement. Specifies how much velocity is lost when objects collide with each other.

Bounciness
: Tendency of object to bounce after colliding with another (0 to 1) (rigid to perfectly elastic). Specifies how much objects can bounce after collisions.

Sensitivity

The collision margin is used to improve the performance and stability of rigid bodies. Depending on the shape, it behaves differently: some shapes embed it, while others have a visible gap around them.

The margin is *embedded* for these shapes:

- Sphere
- Box
- Capsule
- Cylinder
- Convex Hull: Only allows for uniform scale when embedded.

The margin is *not embedded* for these shapes:

- Cone
- Active Triangle Mesh
- Passive Triangle Mesh: Can be set to 0 most of the time.

Margin
: Threshold of distance near the surface where collisions are still considered (best results when nonzero).

Collections

Allows rigid body collisions allocate on different groups (maximum 20).

Dynamics

Reference

Panel:: *Physics ▸ Rigid Body ▸ Dynamics*

Used to control the physics of the rigid body simulation. This panel is available only for *Active* type of rigid bodies.

Damping Translation
: Amount of linear velocity that is lost over time.

Rotation
: Amount of angular velocity that is lost over time.

Deactivation

Enable deactivation of resting rigid bodies. Allows the object to be deactivated during the simulation (improves the performance and stability, but can cause glitches).

Start Deactivated
: The rigid body starts deactivated. It will be activated when in proximity of moving active rigid body objects. The proximity check uses the object's bounding box to determine if a moving object is close enough to activate it.

Linear Velocity
: Specifies the linear deactivation velocity below which the rigid body is deactivated and the simulation stops simulating the object.

Angular Velocity
: Specifies the angular deactivation velocity below which the rigid body is deactivated and the simulation stops simulating the object.

Rigid Body World

Reference

Panel:: *Scene* ▸ *Rigid Body World*

The *Rigid Body World* is a group of rigid body objects, which holds settings that apply to all rigid bodies in this simulation.

When you add rigid body physics to an object, primary there is created a group of objects with default “RigidBodyWorld” name. Rigid body objects automatically are added to this group when you add rigid body physics for them. You can create several Rigid Body World Collections and allocate the rigid body objects with the Collections panel.

Rigid body objects and constraints are only taken into account by the simulation if they are in the collection specified in the *Collection* field of the *Rigid Body World* panel in the *Scene* tab.

Settings

Rigid Body World
: Enable/disable evaluation of the rigid body simulation based on the rigid body objects participating in the specified group of Rigid Body World.

Remove Rigid Body World
: Remove rigid body simulation from the current scene.

Collection
: Containing rigid body objects participating in this simulation.

Constraints
: Containing rigid body object constraints participating in the simulation.

Simulation quality and timing settings:

Speed
: Can be used to speed up/slow down the simulation.

Split Impulse
: Enable/disable reducing extra velocity that can build up when objects collide (lowers the simulation stability a little so use only when necessary). Limits the force with which objects are separated on collision, generally produces nicer results, but makes the simulation less stable (especially when stacking many objects).

Substeps Per Frame
: Number of simulation steps taken per frame (higher values are more accurate but slower). This only influences the accuracy and not the speed of the simulation.

Solver Iterations
: Amount of constraint solver iterations made per simulation step (higher values are more accurate but slower). Increasing this makes constraints and object stacking more stable.

Rigid Body Cache

Reference

Panel:: *Scene ▸ Rigid Body World ▸ Cache*

The *Cache* subpanel specifies the frame range in which the simulation is active. Can be used to bake the simulation.

Start/End
: First and last frame of the simulation.

Bake
: Calculates the simulation and protects the cache. You need to be in *Object Mode* to bake.

Delete Bake
: Active after the baking of simulation. Clears the baked cache.

Calculate to Frame
: Bake physics to current frame.

Current Cache to Bake
: Bake from Cache.

Bake All Dynamics
: Bake all physics.

Delete All Bakes
: Deletes all baked caches of all objects in the current scene.

Update All to Frame
: Update cache to current frame.

If you have not saved the blend-file, the cache is created in memory, so save your file first or the cache may be lost.

Rigid Body Field Weights

Reference

Panel:: *Scene ▸ Rigid Body World ▸ Field Weights*

As other physics dynamics systems, rigid body simulation are also influenced by external force effectors.

Rigid Body Constraints

- Introduction
  - Connect
  - Physics Menu
  - Common Options
- Types
  - Fixed Constraint
  - Point Constraint
  - Hinge Constraint
  - Slider Constraint
  - Piston Constraint
  - Generic Constraint
  - Generic Spring Constraint
  - Motor Constraint

Introduction

Constraints (also known as joints) for rigid bodies connect two rigid bodies. The physics constraints are meant to be attached to an Empty object. The constraint then has fields which can be pointed at the two physics-enabled object which will be bound by the constraint. The empty object provides a location and axis for the constraint distinct from the two constrained objects. The location of the entity hosting the physics constraint marks a location and set of axes on each of the two constrained objects. These two anchor points are calculated at the beginning of the animation and their position and orientation remain fixed in the *local* coordinate system of the object for the duration of the animation. The objects can move far from the constraint object, but the constraint anchor moves with the object. If this feature seems limiting, consider using multiple objects with a non-physics *Child of* constraint and animate the relative location of the child.

Connect

The quickest way to constrain two objects is to select both and click the *Connect* button in *Object ▸ Rigid Body*. This creates a new empty object (named "Constraint") with a physics constraint already attached and pointing at the two selected objects.

Physics Menu

Also you can create *Rigid Body Constraint* on one of the two constrained objects with *Rigid Body Constraint* button of the *Physics* tab in the Properties. This constraint is dependent on the object location and rotation on which it was created. This way, there are no empty object created for the constraint. The role of the empty object is put on this object. The constrained object can be then be set as a *Passive* type for better driving of the constraint.

Additional parameters appear in the *Rigid Body Constraint* panel of the *Physics* tab in the Properties for the selected empty object or the one of the two constrained objects with the created constraint.

Common Options

Reference

Panel:: *Physics* ▸ *Rigid Body Constraint*

Settings

Enabled
: Specifies whether the constraint is active during the simulation.

Disable Collisions
: Allows constrained objects to pass through one another.

Breakable
: Allows constraint to break during simulation. Disabled for the *Motor* constraint. This can be used to simulate destruction.

Threshold
: Impulse strength that needs to be reached before the constraint breaks.

Limits

By using limits you can constrain objects even more by specifying a translation/rotation range on/around respectively one axis (see below for each one individually). To lock one axis, set both limits to 0.

Objects

First
: First object to be constrained.

Second
: Second object to be constrained.

Override Iterations

Allows making constraints stronger (more iterations) or weaker (less iterations) than specified in the rigid body world.

Iterations
: Number of constraint solver iterations made per simulation step for this constraint.

Types

- Fixed Constraint
- Point Constraint
- Hinge Constraint
  - Options
- Slider Constraint
  - Options
- Piston Constraint
  - Options
- Generic Constraint
  - Options
- Generic Spring Constraint
  - Options
- Motor Constraint
  - Options

Fixed Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Fixed

This constraint cause the two objects to move as one. Since the physics system does have a tiny bit of slop in it, the objects do not move as rigidly as they would if they were part of the same mesh.

Point Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Point

The objects are linked by a point bearing allowing any kind of rotation around the location of the constraint object, but no relative translation is permitted. The physics engine will do its best to make sure that the two points designated by the constraint object on the two constrained objects are coincident.

Hinge Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Hinge

The hinge permits one degree of freedom between two objects. Translation is completely constrained. Rotation is permitted about the Z axis of the object hosting the Physics constraint (usually an Empty, distinct from the two objects that are being linked). Adjusting the position and rotation of the object hosting the constraint allows you to control the anchor and axis of the hinge.

The Hinge is the only single-axis rotational constraint that uses the Z axis instead of the X axis. If something is wrong with your hinge, check your other constraints to see if this might be the problem.

Options

Limits
: Z Angle
: Enables/disables limit rotation around Z axis.

: Lower
: Lower limit of Z axis rotation.

: Upper
: Upper limit of Z axis rotation.

Slider Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Slider

The Slider constraint allows relative translation along the X axis of the constraint object, but permits no relative rotation, or relative translation along other axes.

Options

Limits
X Axis
Enables/disables limit translation around X axis.

Lower
Lower limit of X axis translation.

Upper
Upper limit of X axis translation.

Piston Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Piston

A piston permits translation along the X axis of the constraint object. It also allows rotation around the X axis of the constraint object. It is like a combination of the freedoms of a slider with the freedoms of a hinge (neither of which is very free alone).

Options

Limits
: X Axis
: Enables/disables limit translation around X axis.

: Lower
: Lower limit of X axis translation.

: Upper
: Upper limit of X axis translation.

: X Angle
: Enables/disables limit rotation around X axis.

: Lower
: Lower limit of X axis rotation.

: Upper
: Upper limit of X axis rotation.

Generic Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Generic

The generic constraint has a lot of available parameters.

The X, Y, and Z axis constraints can be used to limit the amount of translation between the objects. Clamping the min/max to zero has the same effect as the Point constraint.

Clamping the relative rotation to zero keeps the objects in alignment. Combining an absolute rotation and translation clamp would behave much like the Fixed constraint.

Using a nonzero spread on any parameter allows it to oscillate in that range throughout the course of the simulation.

Options

Limits
: Angular
: X Angle, Y Angle, Z Angle
: Enables/disables limit rotation around X, Y or Z axis respectively.

: Lower
: Lower limit of rotation for X, Y or Z axis respectively.

: Upper
: Upper limit of rotation for X, Y or Z axis respectively.

Linear

X Axis, Y Axis, Z Axis

Enables/disables limit translation on X, Y or Z axis respectively.

Lower

Lower limit of translation for X, Y or Z axis respectively.

Upper

Upper limit of translation for X, Y or Z axis respectively.

Generic Spring Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Generic Spring

The generic spring constraint adds some spring parameters for the X/Y/Z axes to all the options available on the Generic constraint. Using the spring alone allows the objects to bounce around as if attached with a spring anchored at the constraint object. This is usually a little too much freedom, so most applications will benefit from enabling translation or rotation constraints.

If the damping on the springs is set to 1, then the spring forces are prevented from realigning the anchor points, leading to strange behavior. If your springs are acting weird, check the damping.

Options

Limits
: X/Y/Z Axis
  : Enables/disables limit translation on X, Y or Z axis respectively.

    Lower
    : Lower limit of translation for X, Y or Z axis respectively.

    Upper
    : Upper limit of translation for X, Y or Z axis respectively.

: X/Y/Z Angle
  : Enables/disables limit rotation around the X, Y or Z axis respectively.

Lower

Lower limit of rotation for X, Y or Z axis respectively.

Upper

Upper limit of rotation for X, Y or Z axis respectively.

Springs

X/Y/Z Axis

Enables/disables springs translation on X, Y or Z axis respectively.

Stiffness

Spring Stiffness of the translation on X, Y or Z axis respectively. Specifies how “bendy” the spring is.

Damping

Spring Damping of the translation on X, Y or Z axis respectively. Amount of damping the spring has.

X/Y/Z Angle

Enables/disables springs rotation around the X, Y or Z axis respectively.

Stiffness

Spring Stiffness of the rotation around the X, Y or Z axis respectively. Specifies how “bendy” the spring is.

Damping

Spring Damping of the rotation around the X, Y or Z axis respectively. Amount of damping the spring has.

Motor Constraint

Reference

Panel:: *Physics ▸ Rigid Body Constraint*
Type:: Motor

The motor constraint causes translation and/or rotation between two entities. It can drive two objects apart or together. It can drive simple rotation, or rotation and translation (although it will not be constrained like a screw since the translation can be blocked by other physics without preventing rotation).

The rotation axis is the X axis of the object hosting the constraint. This is in contrast with the Hinge which uses the Z axis. Since the Motor is vulnerable to confusing perturbations without a matching Hinge constraint, special care must be taken to align the axes. Without proper alignment, the motor will appear to have no effect (because the hinge is preventing the motion of the motor).

Options

Linear Motor/Angular Motor

Enable

Enable linear or angular motor respectively.

Target Velocity

Target linear or angular motor velocity respectively.

Max Impulse

Maximum linear or angular motor impulse respectively.

Tips

As with all physics-enabled objects, pay close attention to the *Animated* checkbox in the *Rigid Body* panel of the *Physics* tab in the Properties. A common mistake is to use keyframe animation on a *Passive* physics object without checking the *Animated* box. The object will move, but the physics engine will behave as if the *Passive* is still in its starting place, leading to disappointment.

Animation

The most common trick is to Keyframe animate the location or rotation of an *Active* physics object as well as the *Animated* checkbox. When the curve on the *Animated* property switches to disabled, the physics engine takes over using the object's last known location, rotation and velocities.

Animating the strengths of various other parameters (a Motor's Target Velocity, a Hinge's limits, etc.) can be used to accomplish a wide variety of interesting results.

Enabling a constraint during the physics simulation often has dramatic results as the physics engine tries to bring into alignment two objects which are often dramatically out of alignment. It is very common for the affected objects to build up enough kinetic energy to bounce themselves out of camera.

Rigid body dynamics can be baked to normal keyframes with *Bake To Keyframes* in the *Object ▸ Rigid Body* menu.

Simulation Stability

The simplest way of improving simulation stability is to increase the steps per second. However, care has to be taken since making too many steps can

cause problems and make the simulation even less stable (if you need more than 1000 steps, you should look at other ways to improve stability).

Increasing the number of solver iterations helps making constraints stronger and also improves object stacking stability.

It is best to avoid small objects, as they are currently unstable. Ideally, objects should be at least 20 cm in diameter. If it is still necessary, setting the collision margin to 0, while generally not recommended, can help making small object behave more naturally.

When objects are small and/or move very fast, they can pass through each other. Besides what is mentioned above it's also good to avoid using mesh shapes in this case. Mesh shapes consist of individual triangles and therefore do not really have any thickness, so objects can pass through more easily. You can give them some thickness by increasing the collision margin.

Combining Rigid Bodies with Other Simulations

Since the rigid body simulation is part of the animation system, it can influence other simulations just like the animation system can.

In order for this to work, the rigid body object needs to have a Collision Modifier. Simply click on *Collision* in the *Physics* tab.

Scaling Rigid Bodies

Rigid body objects can be scaled, also during the simulation. This work well in most cases, but can sometimes cause problems.

If dynamic scaling is not needed, rigid body objects should have the scale applied by using the *Apply Scale* tool `Ctrl-A`.

Cloth

- Introduction
  - Workflow
  - Springs
- Settings
  - Physical Properties
  - Cache
  - Shape
  - Collisions
  - Property Weights
  - Field Weights
- Examples
  - Using Simulation to Shape/Sculpt a Mesh
  - Smoothing of Cloth
  - Cloth on Armature
  - Cloth with Animated Vertex Groups
  - Cloth with Dynamic Paint
  - Using Cloth for Soft Bodies
  - Cloth with Wind

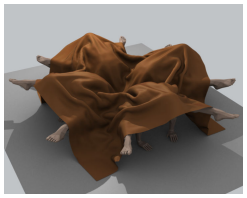
Introduction

Cloth simulation is one of the hardest aspects of computer graphics, it is a deceptively simple real-world item that is taken for granted, but it actually has very complex internal and environmental interactions. Cloth is commonly modeled as 2D mesh to simulate real world objects such as fabrics, flags, banners. And yet cloth can also be used to model 3D objects such as teddy bears, pillows, balloons, or balls.

Cloth interacts with and is affected by other moving objects, the wind and other forces, as well as a general aerodynamic model, all of which is under your control.



Cloth example.



Cloth on carved wooden men (made by motorsep).



Cloth example.

Once Cloth physics have been added to a mesh, a Cloth Modifier will be added to the object's modifier stack. As a modifier then, it can interact with other modifiers, such as *Armature* and *Smooth*. In these cases, the ultimate shape of the mesh is computed in accordance with the order of the modifier stack. For example, you should smooth the cloth *after* the modifier computes the shape of the cloth.

You can *Apply* the Cloth Modifier to freeze, or lock in, the shape of the mesh at that frame, which removes the modifier. For example, you can

drape a flat cloth over a table, let the simulation run, and then apply the modifier. In this sense, you are using the simulator to save yourself a lot of modeling time.

Results of the simulation are saved in a cache, so that the shape of the mesh, once calculated for a frame in an animation, does not have to be recomputed again. If changes to the simulation are made, you have full control over clearing the cache and re-running the simulation. Running the simulation for the first time is fully automatic and no baking or separate step interrupts the workflow.

Computation of the shape of the cloth at every frame is automatic and done in the background; thus you can continue working while the simulation is computed. However, it is CPU-intensive and depending on the power of your PC and the complexity of the simulation, the amount of CPU needed to compute the mesh varies, as does the lag you might notice.

Note

Do Not Jump Ahead

If you set up a cloth simulation but Blender has not computed the shapes for the duration of the simulation, and if you jump ahead a lot of frames forward in your animation, the cloth simulator may not be able to compute or show you an accurate mesh shape for that frame, if it has not previously computed the shape for the previous frame(s).

Workflow

A general process for working with cloth is to:

1. Model the cloth object as a general starting shape.
2. Designate the object as a “cloth” in the *Physics* tab of the Properties.
3. Model other deflection objects that will interact with the cloth. Ensure the Deflection modifier is last on the modifier stack, after any other mesh deforming modifiers.

4. Light the cloth and assign materials and textures, UV unwrapping if desired.
5. If desired, give the object particles, such as steam coming off the surface.
6. Run the simulation and adjust settings to obtain satisfactory results. The Timeline editors playback controls are great for this step.
7. Optionally age the mesh to some point in the simulation to obtain a new default starting shape.
8. Make minor edits to the mesh on a frame-by-frame basis to correct minor tears.

Tip

To avoid unstable simulation, make sure that the cloth object does not penetrate any of the deflection objects.

Springs

Internally, cloth physics is simulated with virtual springs that connect the vertices of a mesh. There are four types of springs that control how the cloth bends. These four types are defined below and illustrated in the following image:

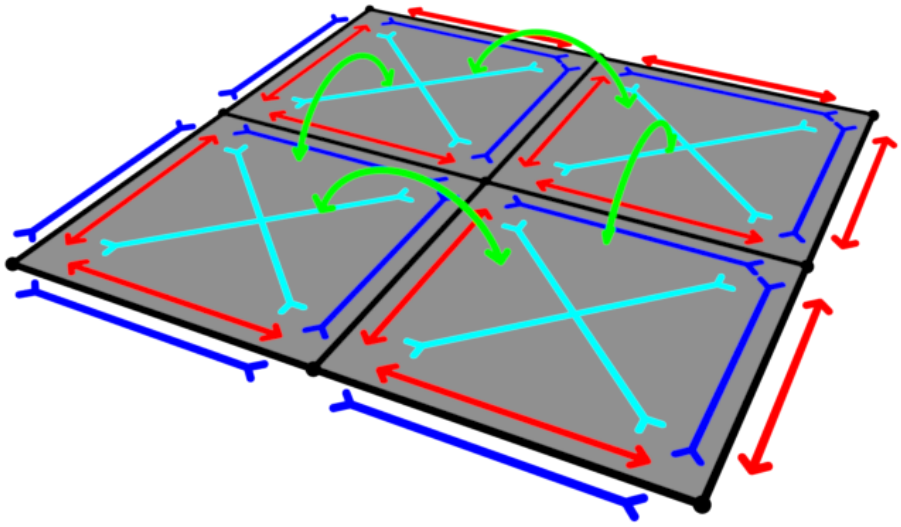


Illustration of cloth springs; tension springs (blue), compression springs (red), shear springs (cyan), and angular bending springs (green).

Tension Springs
: Control the stiffness of the cloth.

Compression Springs
: Control the amount of force required to collapse or compress the cloth.

Shear Springs
: Like compression springs but it controls the angular deformation.

Angular Bending Springs
: Control how resilient the cloth is to folding or crumpling.

All four of these spring types can be controlled independently in the Physical Properties panel. While these settings control surface springs, optionally, internal springs can be used for 3D meshes and behave similarly to Soft Bodies.

Settings

Reference

Panel:: *Physics ▸ Cloth*

Presets
: Contains a number of preset cloth examples.

Quality Steps
: Set the number of simulation steps per frame. Higher values result in better quality, but will be slower.

Speed Multiplier
: Adjust how fast time progresses in the cloth simulation.

- Physical Properties
  - Stiffness
  - Damping
  - Internal Springs
  - Pressure
- Cache
  - Editing the Cached Simulation
- Shape
- Collisions
  - Object Collisions
  - Self-Collisions
- Property Weights
- Field Weights

Physical Properties

Reference

Panel:: *Physics ▸ Cloth ▸ Physical Properties*

Vertex Mass
: The mass of the cloth material.

Air Viscosity
: Air has some thickness which slows falling things down.

Bending Model
: **Linear::** Cloth model with linear bending springs (old).
: **Angular::** Cloth model with angular bending springs.

Stiffness

Tension
: How much the material resists stretching.

Compression
: How much the material resists compression.

Structural
: Overall stiffness of the cloth (only in linear bending model).

Shear
: How much the material resists shearing.

Bending
: Wrinkle coefficient. Higher creates more large folds.

Damping

Tension
: Amount of damping in stretching behavior.

Compression
: Amount of damping in compression behavior.

Structural
: Amount of damping in stretching behavior (only in linear bending model).

Shear
: Amount of damping in shearing behavior.

Bending
: Amount of damping in bending behavior.

Internal Springs

As stated in the introduction, cloth physics are simulated through Springs connecting vertices on the surface of a mesh. But these springs only interact on the surface and only apply to 2D surfaces. 3D or *Internal Springs* can be used to make a mesh behave similarly to a Soft Body. Internal springs can be enabled by toggling the checkbox in the *Internal Springs* panel header.

Max Spring Creation Length
: The maximum length an internal spring can have during creation. If the distance between internal points is greater than this, no internal spring will be created between these points. A length of zero means that there is no length limit.

Max Creation Diversion
: The maximum angle that is allowed to use to connect the internal points can diverge from the vertex normal.

Check Surface Normals
: Requires the points the internal springs connect to have opposite normal directions.

Tension
: How much the material resists stretching.

Compression
: How much the material resists compression.

Vertex Group
: The *Tension* and *Compression* of internal springs can be controlled via a Vertex Group to specify which the portions of the mesh have internal springs or the spring strength.

Max Tension
: Maximum tension stiffness value.

Max Compression
: Maximum Compression stiffness value.

Pressure

Cloth pressure allows the simulation of soft-shelled objects such as balloons or balls that are filled with a type of fluid. This fluid is modeled as a gas; to emulate an incompressible liquid set *Pressure Scale* as high as possible without breaking the simulation. Cloth pressure can be enabled by toggling the checkbox in the *Pressure* panel header.

Note

Non-manifold meshes will work with cloth pressure however, pressure will escape out of the mesh holes and cause drifting or propulsion forces. One way to get around this is by using the *Vertex Group* to exclude the non-manifold portions of the mesh.

Pressure
: The uniform pressure that is constantly applied to the mesh. This value is specified in units of *Pressure Scale*, and can be negative to simulate

implosions or any other case where an object has outside pressure pushing inwards.

Custom Volume
: Use the *Target Volume* parameter as the initial volume for the cloth, instead of computing it from the mesh itself.

Target Volume
: The mesh volume where the inner/outer pressure will be the same. If set to zero, changes in the volume of the object will not affect pressure.

Pressure Scale
: Ambient pressure (in kPa) that exists both inside and outside the object, balancing out when the volume matches the target. Increase the value to make the object resist changes in volume more strongly.

Fluid Density
: Specifies the density of the fluid contained inside the object (in kg/liter = 1000 kg/m^3, use 1 for water), used to generate a hydrostatic pressure gradient that simulates the weight of the fluid. If the value is negative, it instead models buoyancy from a surrounding fluid.

  The fluid is not actually simulated, so while the setting helps to achieve a more plausible object shapes at rest, it cannot create realistic fluid dynamics effects. It can also be used to give more weight to a soft body like object with heavy and sufficiently flexible filling, even if it is not a fluid by itself.

  The volume of the object is not preserved. If that is desired it should be used together with *Pressure Scale*. *Fluid density* times *object size* times 50 is a good start value for *Scale* to make sure that no more than 10% volume change if the object does not experience higher acceleration than standard gravity.

Vertex Group
: Cloth pressure can be controlled via a Vertex Group to specify which the portions of the mesh to apply pressure. Zero weight means no pressure while a weight of one means full pressure.

Note, faces with a vertex that has zero weight will be excluded from the *Target Volume* calculation.

Cache

Reference

Panel:: *Physics* ▸ *Cloth Cache*

After you have set up the deflection mesh for the frame range you intend to run the simulation (including animating that mesh *via* armatures), you can now tell the cloth simulation to compute (and avoid) collisions. Select the cloth object and in the *Object* tab, *Physics* tab, set the *Start* and *End* settings for the simulation frames you wish to compute, and click the *Bake* button.

Cache settings for cloth are the same as with other dynamic systems. See Particle Cache for details.

Note

If you move or edit the cloth object **after** you have already run the simulations, you must clear the cache; otherwise, Blender will use the position of the current/cached mesh's vertices when trying to represent where they are.

Note

Subdivision Surface Modifier

A bake/cache is done for every subdivision level so please use the **equal** subdivision level for render and preview.

Note

You cannot change *Start* or *End* without clearing the bake simulation. When the simulation has finished, you will notice you have the option to

free the bake, edit the bake and re-bake.

Editing the Cached Simulation

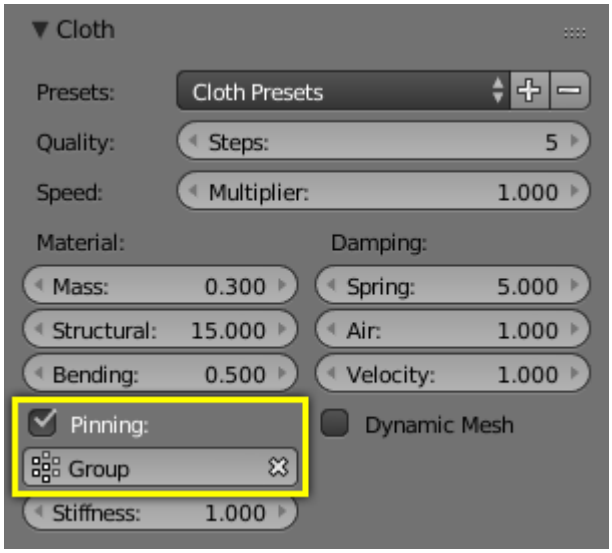
Important

Editing the cached simulation is not currently working, see: blender/blender#77114 for details.

Shape

Reference

Panel:: *Physics ▸ Cloth ▸ Shape*



Cloth Shape.

Pin Group
: Vertex group to use for pinning.

The shape of the cloth can be controlled by pinning cloth to a Vertex Group. There are several ways of doing this including Weight Painting areas you want to pin. The weight of each vertex in the group controls how strongly it is pinned.

Stiffness
: Target position stiffness.

Sewing
: Another method of restraining cloth similar to pinning is sewing springs. Sewing springs are virtual springs that pull vertices in one part

of a cloth mesh toward vertices in another part of the cloth mesh. This is different from pinning which binds vertices of the cloth mesh in place or to another object. A clasp on a cloak could be created with a sewing spring. The spring could pull two corners of a cloak about a character's neck. This could result in a more realistic simulation than pinning the cloak to the character's neck since the cloak would be free to slide about the character's neck and shoulders.

Sewing springs are created by adding extra edges to a cloth mesh that are not included in any faces. They should connect vertices in the mesh that should be pulled together. For example the corners of a cloak.

Max Sewing Force

Maximum force that can be applied by sewing springs. Zero means unbounded, but it is not recommended to leave the field at zero in most cases, as it can cause instability due to extreme forces in the initial frames where the ends of the sewing springs are far apart.

Shrinking Factor

Factor by which to shrink the cloth, specifying a negative value controls the amount for the cloth to grow.

Dynamic Mesh

Allows animating the rest shape of cloth using shape keys or modifiers (e.g. an Armature modifier or any deformation modifier) placed above the Cloth modifier. When it is enabled, the rest shape is recalculated every frame, allowing unpinned cloth to squash and stretch following the character with the help of shape keys or modifiers, but otherwise move freely under control of the physics simulation.

Normally cloth uses the state of the object in the first frame to compute the natural rest shape of the cloth, and keeps that constant throughout the simulation. This is reasonable for fully realistic scenes, but does not quite work for clothing on cartoon style characters that use a lot of squash and stretch.

Rest Shape Key

Allows starting the cloth simulation using a specific Shape Key as the rest state, instead of the shape that results from evaluating shape keys and preceding modifiers in the regular way. This option is mutually exclusive with *Dynamic Mesh*.

This can be used to start the simulation with the cloth in a pre-draped state without applying that shape as a plastic deformation that relaxes all springs as a side effect.

This property is only visible if the mesh has shape keys.

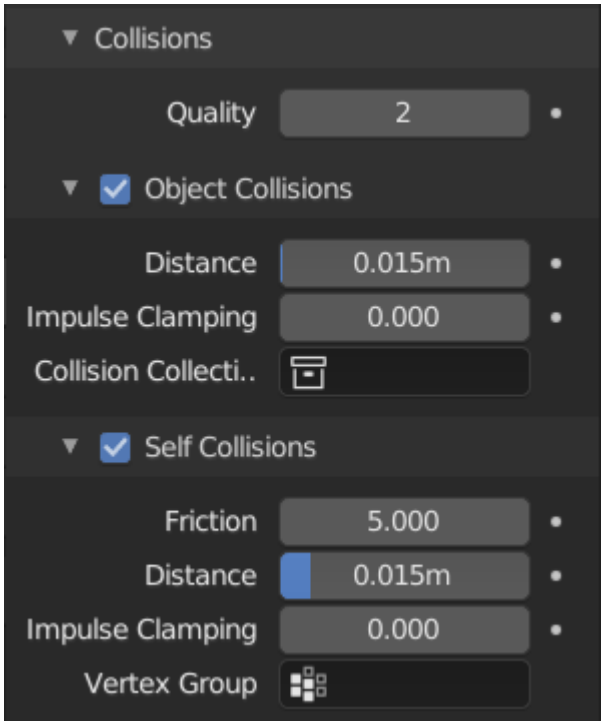
Collisions

Reference

Panel:: *Physics* ▸ *Cloth* ▸ *Collision*

In most cases, a piece of cloth does not just hang there in 3D space, it collides with other objects in the environment. To ensure proper simulation, there are several items that have to be set up and working together:

- The *Cloth* object must be told to participate in collisions.
- Optionally (but recommended) tell the cloth to collide with itself.
- Other objects must be visible to the *Cloth* object *via* shared layers.
- The other objects must be mesh objects.
- The other objects may move or be themselves deformed by other objects (like an armature or shape key).
- The other mesh objects must be told to deflect the cloth object.
- The blend-file must be saved in a directory so that simulation results can be saved.
- You then *Bake* the simulation. The simulator computes the shape of the cloth for a frame range.
- You can then edit the simulation results, or make adjustments to the cloth mesh, at specific frames.
- You can make adjustments to the environment or deforming objects, and then re-run the cloth simulation from the current frame forward.



Cloth Collisions panel.

Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the cloth.

Object Collisions

If the cloth object needs to be deflected by some other object. To deflect a cloth, the object must be enabled as an object that collides with the cloth object. To enable objects to collide with cloth objects enable collision physics for the collider object (not on the cloth object).

Note

If your colliding object is not a mesh object, such as a NURBS surface, or a text object, you must convert it to a mesh object using Convert.

Distance
: The distance another object must get to the cloth for the simulation to repel the cloth out of the way. Smaller values might give errors but gives some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping
: Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Vertex Group
: Faces that have all vertices assigned to this Vertex Group are excluded from collision with objects.

Collision Collection
: Only objects that are a part of this Collection can collide with the cloth. Note that these objects must also have Collision physics enabled.

Self-Collisions

Real cloth cannot penetrate itself, so you normally want the cloth to self-collide. Enable this to tell the cloth object that it should not penetrate itself. This adds to the simulation's compute time, but provides more realistic results.

Tip

A flag, viewed from a distance does not need this enabled, but a close-up of a cape or blouse on a character should have this enabled.

Friction
: A coefficient for how slippery the cloth is when it collides with itself. For example, silk has a lower coefficient of friction than cotton.

Distance

As cloth at this distance begins to repel away from itself. Smaller values might give errors but gives some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping
: Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Vertex Group
: Faces that have all vertices assigned to this Vertex Group are excluded from self-collision.

See also

Example blend-file: Cloth self-collisions.

Troubleshooting

If you encounter some problems with collision detection, there are a few ways to fix them:

- The fastest solution is to increase the *Distance* for Object/Self Collisions. This will be the fastest way to fix the clipping; however, it will be less accurate and will not look as good. Using this method tends to make it look like the cloth is resting on air, and gives it a very rounded look.
- A second method is to increase the *Quality* (in the *Cloth* panel). This results in smaller steps for the simulator and therefore to a higher probability that fast-moving collisions get caught. You can also increase the Collisions *Quality* to perform more iterations to get collisions solved.
- If none of the methods help, you can easily edit the cached/baked result in *Edit Mode* afterwards.
- If the Cloth is torn by the deforming mesh; increase the stiffness settings.

Property Weights

Reference

Panel:: *Physics ▸ Cloth ▸ Property Weights*

This panel is used to constrain certain cloth properties to a certain vertex group. The properties that they control can be found in a combination of the *Physical Properties* and *Shape* panels.

Structural Group
: Defines a vertex group to control over structural stiffness.

Max Tension
: Maximum tension stiffness value.

Max Compression
: Maximum Compression stiffness value.

Shear Group
: Vertex group for fine control over shear stiffness.

Max Shearing
: Maximum shear scaling value.

Bending Group
: Vertex group for fine control over bending stiffness.

Max Bending
: Maximum bending stiffness value.

Shrinking Group
: Vertex group for shrinking cloth.

Max Shrinking

Max amount to shrink cloth by, specifying a negative value controls the max amount for the cloth to grow.

Field Weights

Reference

Panel:: *Physics ▸ Cloth ▸ Field Weights*

As other physics dynamics systems, Cloth simulation is also influenced by external force effectors.

Examples

To start with cloth, the first thing you need, of course, is some fabric. So, let us delete the default cube and add a plane. In order to get some good floppy and flexible fabric, you will need to subdivide it several times, about eight is a good number. So `Tab` into *Edit Mode* and subdivide the mesh a couple of times.

Now, we will make this cloth by going to the Physics tab. Scroll down until you see the *Cloth* panel, and press the *Cloth* button. Now, a lot of settings will appear, most of which we will ignore for now.

That is all you need to do to set your cloth up for animating, but if you playback the animation, the drop of your newly created fabric will be quite unspectacular. That is what we will cover in the next two sections about pinning and colliding.

Using Simulation to Shape/Sculpt a Mesh

You can *Apply* the Cloth Modifier at any point to freeze the mesh in position at that frame. You can then re-enable the cloth, setting the start and end frames from which to run the simulation forward.

Another example of aging is a flag. Define the flag as a simple grid shape and pin the edge against the flagpole. Simulate for 50 frames or so, and the flag will drop to its “rest” position. Apply the *Cloth* Modifier. If you want the flag to flap or otherwise move in the scene, re-enable it for the frame range when it is in camera view.

Smoothing of Cloth

Now, if you followed this from the previous section, your cloth is probably looking a little blocky. In order to make it look nice and smooth like the picture you need to apply a Smooth and/or Subdivision Surface Modifier in

the *Modifiers* tab. Then, in the Toolbar, find the *Edit* panel and Press *Smooth*.

Cloth on Armature

Clothing can be simulated and pinned to an armature. For example, a character could have a baggy tunic pinned to the character's waist with a belt.

The typical workflow for pinning:

1. Set the armature to its bind pose.
2. Model clothing that encloses but does not penetrate the character's mesh.
3. Parent the clothing objects to the armature. The armature will now have several child meshes bound to it.
4. Create a new vertex group on each cloth object for its pinned vertices.
5. Add vertices to be pinned to this vertex group and give these vertices nonzero weights (you probably want weight = 1). For example the belt area of the tunic would be in the vertex group and have weight one.
6. Designate the clothing objects as "cloth" in the Physics tab of the Properties. Make sure the Cloth Modifier is below the Armature Modifier in the modifier stack.
7. In the cloth Shape panel select the vertex group.
8. Add collision physics to the character's mesh.
9. The clothing is now ready; non-pinned vertices will be under control of the Cloth modifier. Pinned vertices will be under control of the Armature modifier.

Note

When animating or posing the character you must begin from the bind pose. Move the character to its initial pose over several frames so the physics engine can simulate the clothing moving. Very fast movements and teleport jumps can break the physics simulation.

Regression blend-file.

Cloth with Animated Vertex Groups

Cloth with animated pinned vertices: Regression blend-file. Unsupported: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will **not** work (e.g. from goal = 0 to goal = 0.5).

Cloth with Dynamic Paint

Cloth with Dynamic Paint using animated vertex groups: Regression blend-file. Unsupported: Starting with a goal of 0 and increasing it, but still having the vertex not pinned will **not** work (e.g. from goal = 0 to goal = 0.5) because the necessary "goal springs" cannot be generated on-the-fly.

Using Cloth for Soft Bodies

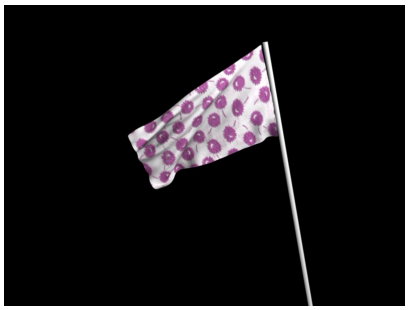


Using cloth for soft bodies.

Cloth can also be used to simulate soft bodies. It is for sure not its main purpose but it works nonetheless. The example image uses standard *Rubber* material, no fancy settings, just `Alt-A`.

Blend-file for the example image: Using Cloth for soft bodies.

Cloth with Wind



Flag with wind applied.

Regression blend-file for Cloth with wind and self-collisions (also the blend for the image above): Cloth flag with wind and self-collisions.

Soft Body

- Introduction
  - Typical Scenarios for using Soft Bodies
  - Creating a Soft Body
  - Interaction in Real-Time
  - Tips
- Settings
  - Object
  - Simulation
  - Cache
  - Goal
  - Edges
  - Self Collision
  - Solver
- Forces
  - Exterior
  - Interior
- Collision
  - Collisions with Other Objects
  - Self-Collisions
- Examples
  - A Bouncing Cube

Introduction

Soft body simulation is used for simulating soft deformable objects. It was designed primarily for adding secondary motion to animation, like jiggle for body parts of a moving character.

It also works for simulating more general soft objects that bend, deform and react to forces like gravity and wind, or collide with other objects.

While it can simulate cloth and other stiff types of deformable objects to an extent, the Cloth Simulation can do it better with a solver specifically designed for this purpose.

The simulation works by combining existing animation on the object with forces acting on it. There are exterior forces like gravity or force fields and interior forces that hold the vertices together. This way you can simulate the shapes that an object would take on in reality if it had volume, was filled with something, and was acted on by real forces.

Soft bodies can interact with other objects through *Collision*. They can interact with themselves through *Self-Collision*.

The result of the soft body simulation can be converted to a static object. You can also *bake edit* the simulation, i.e. edit intermediate results and run the simulation from there.

Typical Scenarios for using Soft Bodies



The wind cone is a soft body, as the suspension.

Animation

Soft bodies are well suited for:

- Jiggle on moving characters.
- Elastic and deformable objects made of materials like rubber or gelatin.
- Tree branches moving in the wind, swinging ropes, and the like.
- Flags, wide sleeves, cushions or other simple fabric reacting to forces.

Creating a Soft Body

Soft body simulation works for all objects that have vertices or control points (meshes, curves, surfaces, and lattices).

To add a soft body simulation to an object, go to the *Physics* tab in the Properties and activate the *Soft Body* button. For a reference of all the settings see this page.

You start a soft body simulation by playback animation with `Alt-A`, and stop the simulation with `Esc` or `Alt-A`.

Interaction in Real-Time

To work with a soft body simulation, you will find it handy to use the Timeline editor. You can change between frames and the simulation will always be shown in the actual state. You can interact in real-time with the simulation, e.g. by moving collision objects or shaking a soft body object.

You can then select the soft body object while running the simulation and *Apply* the modifier in the *Modifiers* tab of the Properties. This makes the deformation permanent.

Tips

- Soft bodies work especially well if the objects have an even vertex distribution. You need enough vertices for good collisions. You change the deformation (the stiffness) if you add more vertices in a certain region.
- The calculation of collisions may take a long time. If something is not visible, why calculate it?
- To speed up the collision calculation it is often useful to collide with an additional, simpler, invisible, somewhat larger object.
- Use soft bodies only where it makes sense. If you try to cover a body mesh with a tight piece of cloth and animate solely with soft body, you will have no success. Self-collision of soft body hair may be activated, but that is a path that you have to wander alone. We will deal with Collisions in detail later.
- Try and use a *Lattice* or a *Curve Guide* soft body instead of the object itself. This may be magnitudes faster.

Settings

Reference

Panel:: *Physics ▸ Soft Body*

Collision Collection
: If set, soft body collides with objects from the collection, instead of using objects that are on the same layer.

- Object
- Simulation
- Cache
- Goal
  - Settings
  - Strengths
- Edges
  - Aerodynamics
  - Stiffness
- Self Collision
- Solver
  - Diagnostics
  - Helpers

Object

Friction
: The friction of the surrounding medium. Generally friction dampens a movement. The larger the friction, the more viscous is the medium. Friction always appears when a vertex moves relative to its surround medium.

Mass
: Mass value for vertices. Larger mass slows down acceleration, except for gravity where the motion is constant regardless of mass. Larger mass means larger inertia, so also braking a soft body is more difficult.

Control Point
: You can paint weights and use a specified vertex group for mass values.

Simulation

Speed

You can control the internal timing of the soft body system with this value. It sets the correlation between frame rate and tempo of the simulation. A free falling body should cover a distance of about five meters after one second and travel at a speed of ten meters per seconds.

You can adjust the scale of your scene and simulation with this correlation. If you render with 25 frames per second, you will have to set *Speed* to 1.3.

Cache

Reference

Panel:: *Physics ▸ Soft Body ▸ Cache*

Soft Body physics simulations use a unified system for caching and baking. See Particle Cache and General Baking documentation for reference.

Goal

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Goal*

Enabling this tells Blender to use the motion from animations (F-Curves, armatures, parents, lattices, etc.) in the simulation. The "goal" is the desired end position for vertices based on this animation.

See exterior forces for details.

Vertex Group
: Use a vertex group to allow per-vertex goal weights (multiplied by the *Default* goal).

Settings

Stiffness
: The spring stiffness for *Goal*. A low value creates very weak springs (more flexible "attachment" to the goal), a high value creates a strong spring (a stiffer "attachment" to the goal).

Damping
: The friction coefficient for *Goal*. Higher values give damping of the spring effect (little jiggle), and the movement will soon come to an end.

Strengths

Default
: Goal weight/strength for all vertices when no *Vertex Group* is assigned. If you use a vertex group the weight of a vertex defines its goal.

Min/Max

When you use a vertex group, you can use the *Minimum* and *Maximum* to fine-tune (clamp) the weight values. The lowest vertex weight will become *Minimum*, the highest value becomes *Maximum*.

Edges

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Edges*

Allow the edges in a mesh object to act like springs. See interior forces.

Springs
: Use a specified vertex group for spring strength values.

Pull
: The spring stiffness for edges (how much the edges are allowed to stretch). A low value means very weak springs (a very elastic material), a high value is a strong spring (a stiffer material) that resists being pulled apart.

  A value of 0.5 is latex, 0.9 is like a sweater, 0.999 is a highly-starched napkin or leather. The soft body simulation tends to get unstable if you use a value of 0.999, so you should lower this value a bit if that happens.

Push
: How much the soft body resists being scrunched together, like a compression spring. Low values for fabric, high values for inflated objects and stiff material.

Damp
: The friction for edge springs. High values (max of 50) dampen the *Push*/*Pull* effect and calm down the cloth.

Plasticity
: Permanent deformation of the object after a collision. The vertices take a new position without applying the modifier.

Bending
: This option creates virtual connections between a vertex and the vertices connected to its neighbors. This includes diagonal edges. Damping also applies to these connections.

Length
: The edges can shrink or be blown up. This value is given in percent, 0 disables this function. 100% means no change, the body keeps 100% of its size.

Collision
: Edge
: Checks for edges of the soft body mesh colliding.

: Face
: Checks for any portion of the face of the soft body mesh colliding (which is computationally intensive). While *Face* enabled can solve collision errors, there does not seem to be any dampening settings for it. So parts of the soft body object near a collision mesh tend to "jitter" as they bounce off and fall back, even when there is no motion of any meshes. Edge collision has dampening, so that can be controlled, but Deflection dampening value on a collision object does not seem to affect the face collision.

Aerodynamics

Force from surrounding media. See exterior forces for details.

Type
: **Simple::** Edges receive a drag force from the surrounding media.
: **Lift Force::** Edges receive a lift force when passing through the surrounding media.

Factor
: How much aerodynamic force to use. Try a value of 30 at first.

Stiffness

For quad faces, the diagonal edges are used as springs. This stops quad faces to collapse completely on collisions (what they would do otherwise).

Shear
: Stiffness of the virtual springs created for quad faces.

Self Collision

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Self Collision*

Note

Self-Collision is working only if you have activated *Use Edges*.

When enabled, allows you to control how Blender will prevent the soft body from intersecting with itself. Every vertex is surrounded with an elastic virtual ball. Vertices may not penetrate the balls of other vertices. If you want a good result you may have to adjust the size of these balls. Normally it works pretty well with the default options.

Calculation Type

Manual:: The *Ball Size* directly sets the ball size.

Average:: The average length of all edges attached to the vertex is calculated and then multiplied with the *Ball Size* setting. Works well with evenly distributed vertices.

Minimal/Maximal:: The ball size is as large as the smallest/largest spring length of the vertex multiplied with the *Ball Size*.

Average Min Max:: Size = ((Min + Max)/2) × *Ball Size*.

Ball Size

Fraction of the length of attached edges. The edge length is computed based on the chosen algorithm. This setting is the factor that is multiplied by the spring length. It is a spherical distance (radius) within which, if another vertex of the same mesh enters, the vertex starts to deflect in order to avoid a self-collision.

Set this value to the fractional distance between vertices that you want them to have their own “space”. Too high of a value will include too many vertices at all times and slow down the calculation. Too low of a level will let other vertices get too close and thus possibly intersect because there will not be enough time to slow them down.

Stiffness

How elastic that ball of personal space is. A high stiffness means that the vertex reacts immediately to another vertex enters their space.

Dampening

How the vertex reacts. A low value just slows down the vertex as it gets too close. A high value repulses it.

Note

Collisions with other objects are set in the (other) Collision panel. To collide with another object they have to share at least one common layer.

Solver

Reference

Panel:: *Physics ▸ Soft Body ▸ Solver*

The settings in the *Soft Body Solver* panel determine the accuracy of the simulation.

Step Size Min
: Minimum simulation steps per frame. Increase this value, if the soft body misses fast-moving collision objects.

Max
: Maximum simulation steps per frame. Normally the number of simulation steps is set dynamically (with the *Error Limit*) but you have probably a good reason to change it.

Auto-Step
: Use velocities for automatic step sizes. Helps the Solver figure out how much work it needs to do based on how fast things are moving.

Error Limit
: Rules the overall quality of the solution delivered. The most critical setting that defines how precise the solver should check for collisions. Start with a value that is half the average edge length. If there are visible errors, jitter, or over-exaggerated responses, decrease the value. The solver keeps track of how “bad” it is doing and the *Error Limit* causes the solver to do some “adaptive step sizing”.

Diagnostics

Print Performance to Console
: Prints on the console how the solver is doing.

Estimate Transforms
: Estimate matrix, split to `COM`, `ROT`, `SCALE`.

Helpers

These settings control how the soft body will react (deform) once it either gets close to or actually intersects (cuts into) another collision object on the same layer.

Choke
: Calms down (reduces the exit velocity of) a vertex or edge once it penetrates a collision mesh.

Fuzzy
: Fuzziness while on collision, high values make collision handling faster but less stable. Simulation is faster, but less accurate.

Forces

- Exterior
  - Example
  - Force Fields
  - Aerodynamics
  - Goal
  - Technical Details
- Interior
  - Stiffness
  - Bending Stiffness

Exterior

Exterior forces are applied to the vertices (and nearly exclusively to the vertices) of soft body objects. This is done using Newton's Laws of Physics:

- If there is no force on a vertex, it stays either unmoved or moves with constant speed in a straight line.
- The acceleration of a vertex depends on its mass and the force. The heavier the mass of a vertex the slower the acceleration. The larger the force the greater the acceleration.
- For every action there is an equal and opposite reaction.

Well, this is done only in the range of computing accurateness, there is always a little damping to avoid overshoot of the calculation.

Example

We will begin with a very simple example: the default cube.

- To judge the effect of the external forces you should at first turn off the *Goal*, so that the vertices are not retracted to their original position.
- Start playback to run the simulation.

What happens? The cube moves in negative Z direction. Each of its eight vertices is affected by a global, constant force – the gravitation. Gravitation without friction is independent from the weight of an object, so each object you would use as a soft body here would fall with the same acceleration. The object does not deform, because every vertex moves with the same speed in the same direction.

Force Fields

Soft body vertices interact with all the Force Fields applied (usually to particles) in the layer, such as wind, force fields, and what ever physics field effect is on a common layer.

Soft Body Field Weights

Reference

Panel:: *Physics* ▸ *Soft Body* ▸ *Field Weights*

The *Soft Body Field Weights* panel allows you to control how much influence each type of external force field has on the soft body system.

Effector Collection
: Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity
: Control how much the Global Gravity has an effect on the system.

All
: Scale all of the effector weights.

Aerodynamics

Edges can be affected by wind as they move, and sail or flutter in a breeze. A simple aerodynamic model of a flag sailing in the wind.

This special exterior force is not applied to the vertices but to the connecting edges. Technically, a force perpendicular to the edge is applied. The force scales with the projection of the relative speed on the edge (dot product). Note that the force is the same if wind is blowing or if you drag the edge through the air with the same speed. That means that an edge moving in its own direction subject to no force, and an edge moving perpendicular to its own direction is subjected to maximum force.

The angle and the relative speed between medium and edge is used to calculate the force on the edge. This force results that vertices with few connecting edges (front of a plane) fall faster than vertices with more connecting edges (middle of a plane). If all vertices have the same amount of edges in a direction they fall with equal speed.

The *Aerodynamics* settings are set in the Soft Body Edges panel.

Goal

A "goal" is a shape that a soft body object tries to conform to. It acts like a pin on a chosen set of vertices, controlling how much of an effect soft body has on them.

Enabling *Soft Body Goal* tells Blender to use the position (or animated position) of a vertex in the simulation. Animating the vertices can be done in all the usual ways (F-Curves, armatures, parents, lattices, etc.) before the soft body simulation is applied. The "goal" is the desired end position for vertices. How a soft body tries to achieve this goal can be defined using stiffness forces and damping.

See the Goal Settings for details.

Goal Strength

The *Goal Strength* defines how much motion from an animation system gets applied.

A *Goal* value of 1.0 means no soft body simulation, the object act like any regular animated object (i.e. the vertex is kept at its original position). When setting *Goal* to 0.0 (or no goal), the vertex is only influenced by physical laws according to soft body simulation.

By setting goal values between 0.0 and 1.0, you can blend between having the object affected only by the animation system, and having the object affected only by the soft body effect.

Goal also serves as a memory, to make sure soft objects don't deform too much, ending up in the non-soft animated shape. Using the *Vertex Group* weight system, you can define a *Goal* weight per vertex. To make this look more natural, spring forces can be defined to control how far vertices can move from their original position.

Often Weight Paint is used to adjust the weight comfortably. For non-mesh objects the *Weight* parameter of their vertices/control points is used instead; Use the Context menu in Edit Mode or the *Transform* panel in the Sidebar region. The weight of *Hair* particles can also be painted in Particle Edit Mode.

Technical Details

In the Soft Body world, vertices of meshes are treated as particles having a mass. Their movement is determined by the forces affecting them. Beside other forces the individual particles can interact with another along edges using a physical model which is very close to shock absorbers used in cars. The working parts are:

- A spring trying to keep the particles at a certain distance. How hard the spring tries to do that is controlled by the soft body parameter *Stiffness*.
- A damping element to calm the movement down. The resistance the element builds up against motion is controlled by the soft body parameter *Damping*.

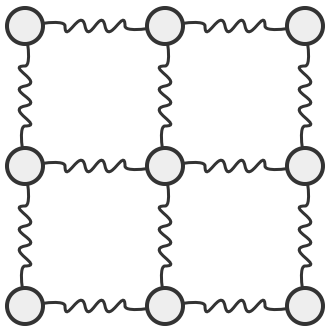
Interior

By default, the edges of a soft-body mesh act like springs. This means that, like a mechanical spring, they can stretch under tension and squeeze under pressure. Their initial length is also their “ideal” or “rest” length, which they try to return to.

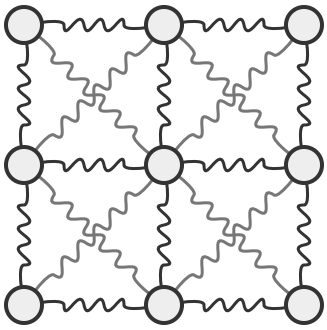
Having edges act like springs is what holds the mesh together. If you were to disable this behavior (as well as the Goal), each vertex would be free to go anywhere independently of the others, which would stretch the mesh until it’s no longer recognizable.

Having springs along edges alone typically isn’t enough, however: vertices in quads are still free to move towards their diagonal opposite, potentially collapsing the quad into a line.

You could solve this by creating diagonal edges everywhere, but fortunately, you don’t have to: simply enable the *Stiffness* option to have Blender create diagonal springs internally. This way, you don’t have to change your mesh.



Base springs along edges.



Additional springs when Stiffness is enabled.

Another method of preventing mesh collapse is applying *Bending Stiffness*, which adds rotational resistance: making edges try to keep their relative angles.

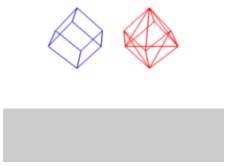
Both of these methods are described in more detail below. You can configure them, as well as other settings, in the Soft Body Edges panel.

Stiffness

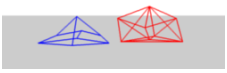
To show the effect of the Stiffness setting, we will drop two cubes onto a plane (see Collisions). The blue cube uses quads, while the red one uses tris. Both cubes have their Goal setting disabled.

If *Stiffness* is disabled, the quad-only cube will collapse completely, while the tri cube only temporarily deforms from the impact:

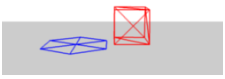
Without Stiffness.



Frame 1.



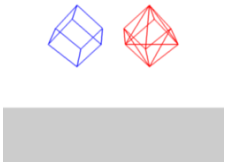
Frame 36.



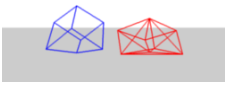
Frame 401.

If *Stiffness* is enabled, the quad cube maintains its shape as well thanks to the extra springs:

With Stiffness.



Frame 1.



Frame 36.



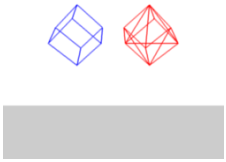
Frame 401.

Bending Stiffness

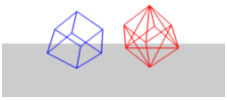
The second method to stop an object from collapsing is to give it *Bending Stiffness*. Just like the other settings, this can be combined with *Stiffness* to add bending resistance to the diagonal springs as well.

We first do the same cube experiment as before, using only *Bending Stiffness*:

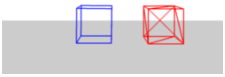
Bending Stiffness.



Frame 1.

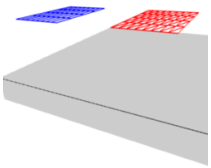


Frame 36.

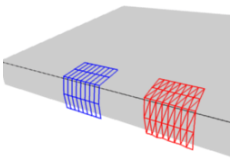


Frame 401.

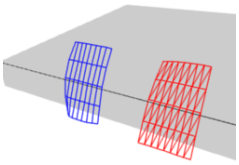
Both cubes keep their shape. Now, we try the same thing with subdivided planes, again a quad-based one and a triangulated one:



Two planes falling.



No bending stiffness.



High bending stiffness (10).

Without any *Bending Stiffness*, the faces can rotate freely as though their edges were hinges. Enabling *Stiffness* to add diagional springs would not change this (just as triangulating doesn't).

With a high *Bending Stiffness*, however, the edges resist this rotation, and the planes act more like planks than towels.

Collision

There are two different collision types that you may use: collision between different objects and internal collision. We should set one thing straight from the start – the primary targets of the collision calculation are the vertices of a soft body. So if you have too few vertices too few collision takes place. Secondarily, you can use edges and faces to improve the collision calculation.

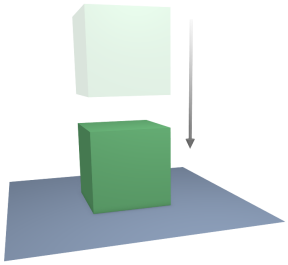
Collisions with Other Objects

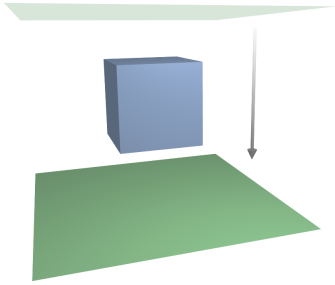
For a soft body to collide with another object there are a few prerequisites:

- If *Collision Collection* is set, the object must belong to the collection.
- The collision object has to be a mesh object.
- You have to activate the *Collision* in the *Physics* tab for the collision object. The collision object may also be a soft body.

Examples

A soft body cube colliding with a plane (Fig. A soft body cube colliding with a plane.) works pretty well, but a soft body plane falls right through a cube that it is supposed to collide with (Fig. A soft body plane colliding with a cube, so no interaction at all.).





A soft body cube colliding with a plane.	A soft body plane colliding with a cube, so no interaction at all.

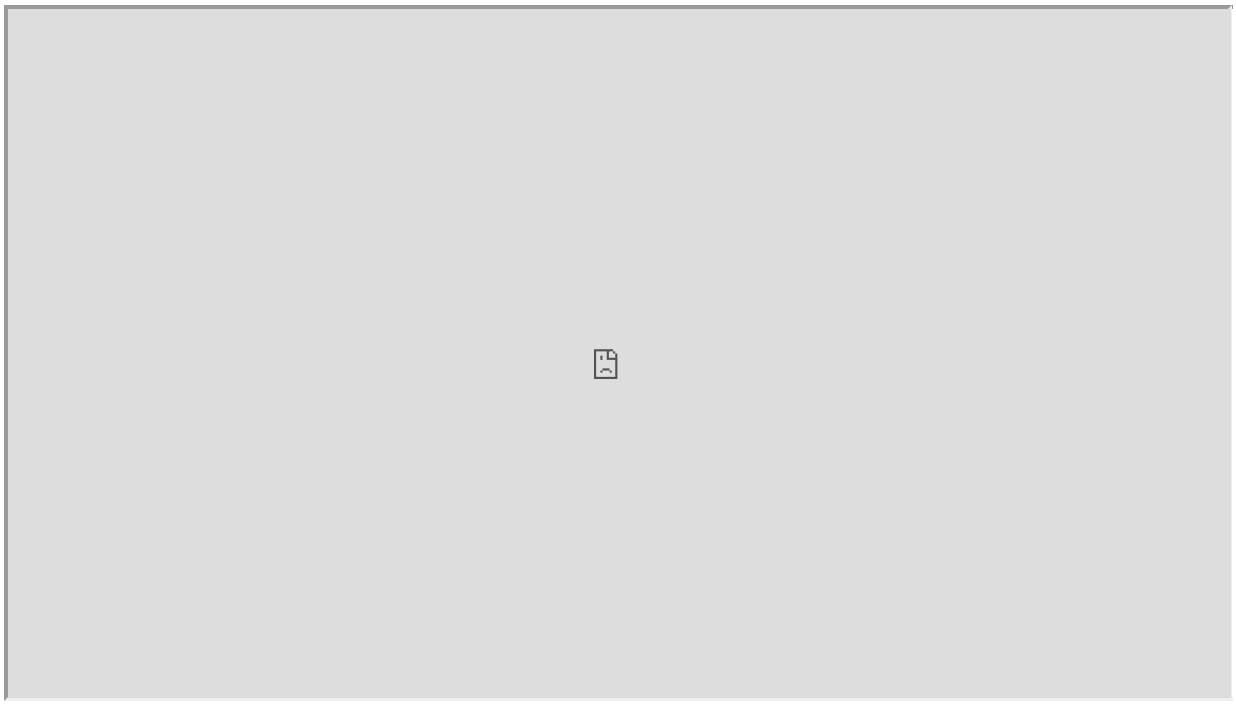
Why is that? Because the default method of calculation only checks to see if the four vertices of the soft body plane collides with the cube as the plane is pulled down by gravity. You can activate *Collision: Face* (in the *Soft Body Edges* panel) to enable collision between the face of the plane and the object instead, but this type of calculation takes much longer.

Let us have a closer look at the collision calculation, so you can get an idea of how we might optimize it.

Calculating Collisions

Soft body simulations are by default done on a per-vertex basis. If the vertices of the soft body do not collide with the collision object, there will be no interaction between the two objects.

In the video below, you can see the vertices colliding with a plane. If a vertex penetrates the zone between *Outer* and *Inner*, it is repulsed by a force in the direction of the face normal. The position that a vertex finally ends up in is dependent on the forces that act upon it. In the example (the first vertex on the left in the video below) gravity and the repulsion force of the face balance out. The speed at which the vertex is pulled out of the collision zone is influenced by the *Choke* parameter in the Soft Body Solver settings.



See also

Download the blend-file.

Now let's see what happens if we make vertices heavier and let them travel at a faster speed. In the video above, you can see vertices traveling at different speeds. The two on the far right (fifth and sixth) are traveling so fast that they pass right through the collision zone (this is because of the default solver precision, which we can fix later). You will notice that the fourth vertex also travels quite fast and because it is heavier it breaches the inner zone. The first three vertices collide correctly.

You can set up your collision so that edges and even faces are included in the collision calculation in the *Soft Body Edges* panel with the Collision *Face* and *Edge* options. The collision is then calculated differently. It is checked whether the edge or face intersects with the collision object, the collision zones are not used.

Good Collisions

If the collision you have set up is not behaving properly, you can try the following:

- The soft body object must have more subdivisions than the collision object. Add loop cuts to the soft body object in strategic areas that you know are most likely to be involved in a collision.
- Check the direction of the face normals.
- If the collision object has sharp spikes, they might penetrate the soft body.
- The resolution of the solver must match the speed at which soft body vertices are traveling. Lower the parameter *Error Limit* and carefully increase *Min Step*.
- *Outer* and *Inner* should be large enough, but zones of opposite faces should not overlap, or you have forces in opposite directions.
- If you use strong forces you should use large zones.
- Set *Choke* to a high enough value (all the way up if necessary) if you have difficulties with repelled vertices.
- Colliding faces are difficult to control and need long calculation times. Try not to use them.

Often it is better to create a simplified mesh to use as your collision object, however, this may be difficult if you are using an animated mesh.

Self-Collisions

For information on self-collision please refer to the Self Collision settings.

Examples

Here are some simple examples showing the power of soft body physics.

A Bouncing Cube

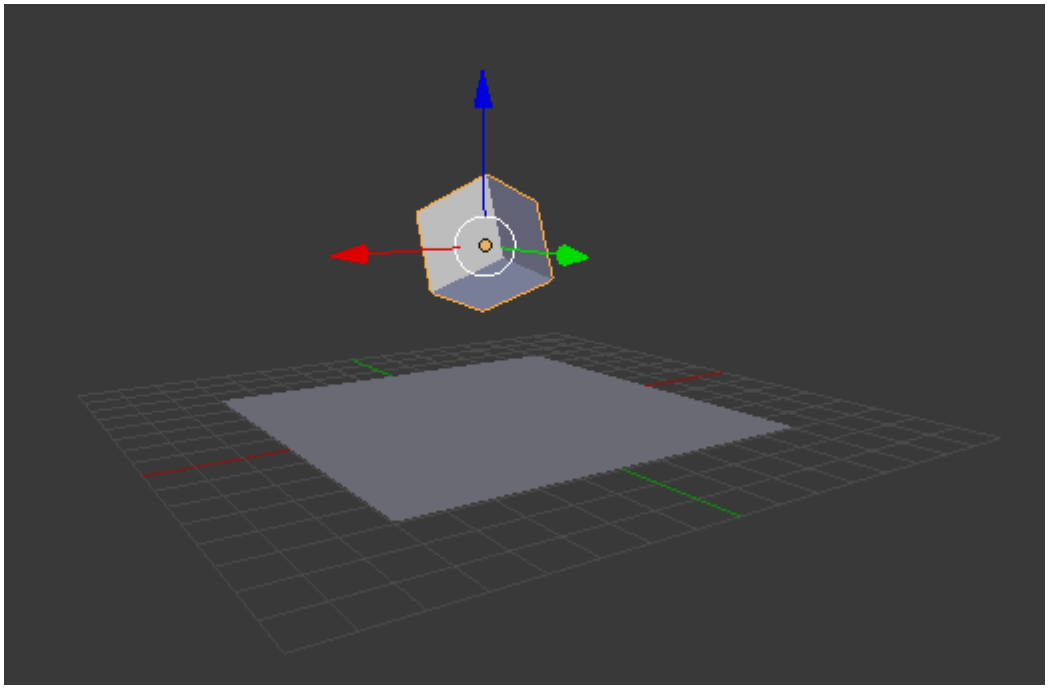
The Process

First, change your start and end frames to 1 and 150.

Then, add a plane, and scale it five times. Next, go to the physics tab, and add a collision. The default settings are fine for this example.

Now add a cube, or use the default cube, then enter *Edit Mode* to subdivide it three times. Add a Bevel Modifier to it to smoothen the edges and then to add a little more, press `R` twice, and move your cursor a bit.

When finished, your scene should look like this:



The scene, ready for soft body physics.

Everything is ready to add the soft body physics. Go to *Properties* ▸ *Physics* and choose *Soft Body*. Uncheck the *Soft Body Goal*, and check *Soft Body Self Collision*. Also, under *Soft Body Edges*, increase the Bending to 10.

Playing the animation will now give a slow animation of a bouncing cube. To speed things up, we need to bake the soft body physics.

Under *Soft Body Cache* change the values of your start and end frames. In this case 1 and 150. Now, to test if everything is working, you can take a cache step of 5 or 10, but for the final animation it is better to reduce it to 1, to cache everything.

You can now bake the simulation, give the cube materials and textures and render the animation.

The Result

The rendered bouncing cube

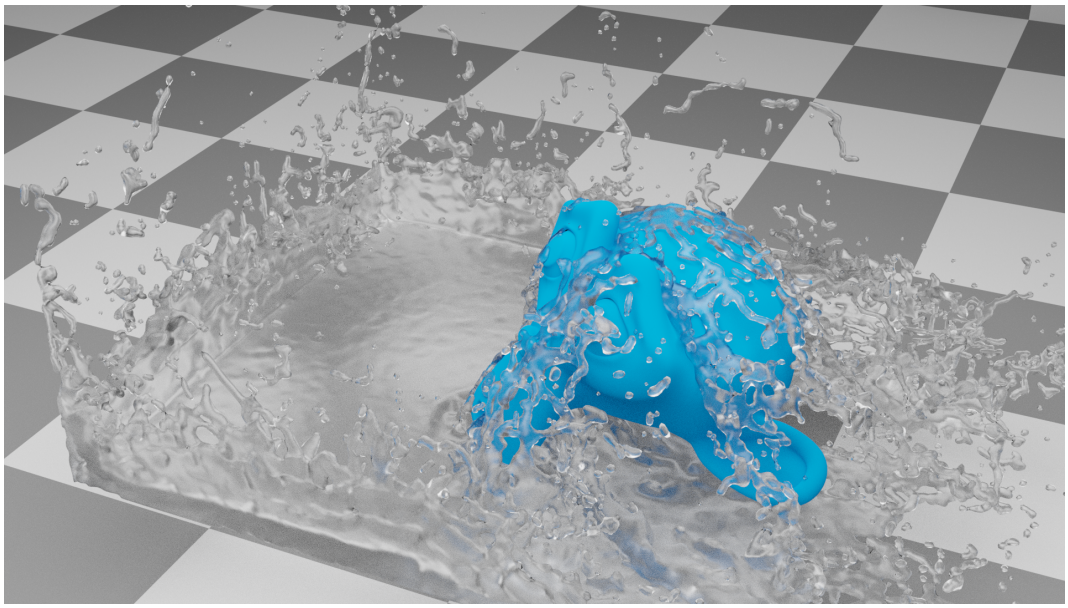
Fluid

- Introduction
  - Liquid Simulations
  - Gas Simulations
  - Workflow
- Type
  - Domain
  - Flow
  - Effector
- Materials
  - Smoke Material

Introduction

Liquid Simulations

Fluid physics are used to simulate physical properties of liquids especially water. While creating a scene in Blender, certain objects can be marked to become a part of the fluid simulation. For a fluid simulation you have to have a domain to define the space that the simulation takes up. In the domain settings you will be able to define the global simulation parameters (such as viscosity and gravity).



Example of a liquid simulation.

Gas Simulations

Gas or smoke simulations are a subset of the fluids system, and can be used for simulating collections of airborne solids, liquid particulates and gases, such as those that make up smoke. It simulates the fluid movement of air and generates animated Voxel textures representing the density, heat, and

velocity of other fluids or suspended particles (e.g. smoke) which can be used for rendering.



Example of a fire simulation.

Gases or smoke are emitted inside of a Domain from a mesh object or particle system. The smoke movement is controlled by airflow inside the domain, which can be influenced by Effector objects. Smoke will also be affected by the scene's gravity and force fields. Airflow inside the domain can affect other physics simulations via the Fluid Flow force field.

Workflow

At least a Domain object and one Flow object are required to create a fluid simulation.

1. Create a Domain object that defines the bounds of the simulation volume.
2. Set up Flow objects which will emit fluid.
3. Set up Effector objects to make the fluid interact with objects in the scene.
4. Assign a material to the domain object.
5. Save the blend-file.

6. Bake the Cache for the simulation.

Note

There are Quick Liquid and Quick Smoke tools which will automatically create a domain object with a basic liquid or smoke and fire material.

Type

- Domain
  - Settings
  - Gas Settings
  - Liquid Settings
  - Guides
  - Collections
  - Cache
  - Field Weights
- Flow
  - Settings
- Effector
  - Settings

Domain

- Settings
  - Border Collisions
  - Gas
  - Fire
  - Liquid
- Gas Settings
  - Adaptive Domain
  - Noise
  - Viewport Display
- Liquid Settings
  - Diffusion
  - Particles
  - Mesh
- Guides
- Collections
- Cache
  - Advanced
- Field Weights

Settings

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Settings*
Type:: Domain

The domain object contains the entire simulation. Fluid simulations cannot leave the domain, it will either collide with the edge or disappear, depending on the domain's settings.

Keep in mind that large domains need higher resolutions and longer bake times. You will want to make it just large enough that the simulation will fit inside it, but not so large that it takes too long to compute the simulation.

To create a domain, add a cube and transform it until it encloses the area where you want the simulation to take place. Translation, rotation, and scaling are all allowed. To turn it into a fluid domain, click *Fluid* in the *Properties* ▸ *Physics* tab, then select *Domain* as the fluid *Type*.

Note

You *can* use other shapes of mesh objects as domain objects, but the fluid simulator will use the shape's Bounding Box as the domain bounds. In other words, the actual shape of the domain will still be rectangular.

Domain Type
: A fluid domain can control either liquid or gas flows. Liquid domains take all liquid flow objects that intersect with the domain into consideration. Gas domains consider all intersecting *Smoke*, *Fire*, and *Smoke & Fire* flow objects. It is not possible to change the domain type dynamically.

Resolution Divisions

The fluid domain is subdivided into many "cells" called Voxels which make up "pixels" of fluid. This setting controls the number of subdivisions in the domain. Higher numbers of subdivisions are one way of creating higher resolution fluids.

Since the resolution is defined in terms of "subdivisions", larger domains will need more divisions to get an equivalent resolution to a small domain. For example, a one meter cube with 64 *Resolution Divisions* will need 128 divisions to match a 2 meter cube. The dimension used as the base division is the longest dimension of the objects bounding box. To help visualize the voxel size, the *Resolution Divisions* can be previewed with a small cube shown in the 3D Viewport, to show the size of these divisions.

Time Scale

Controls the speed of the simulation. Low values result in a "slow motion" simulation, while higher values can be used to advance the simulation faster (good for generating fluids to be used in still renders).

Use Adaptive Time Steps

Lets the solver automatically decide when to perform multiple simulation steps per frame. It takes into account the maximum and minimum number of time steps, the current *Frame Rate*, and the *Time Scale*.

CFL Number

Determines the maximum velocity per grid cell and is measured in grid cells per time step. Fluid is only allowed to move up to this velocity in one time step. If this threshold is exceeded the solver will subdivide the simulation step.

In general, greater CFL (Courant–Friedrichs–Lewy) numbers will minimize the number of simulation steps and the computation time. Yet it will yield less physically accurate behavior for fast fluid flows. Smaller CFL numbers result in more simulation steps per frame, longer simulation times but more accurate behavior at high velocities (e.g. fast fluid flow colliding with obstacle).

Note

When lowering the *CFL* number it is recommended to increase the maximum number of time steps. Similarly, when increasing the *CFL* number the minimum number of time steps should be adjusted.

Timesteps Maximum
: Maximum number of allowed time steps per frame. If needed, the solver will divide a simulation step up to this number of sub-steps.

Timesteps Minimum
: Minimum number of allowed time steps per frame. The solver will always perform at least this number of simulation steps per frame.

Gravity
: By default the fluid solver will use the global scene gravity. This behavior can be disabled in the scene settings. Disabling the global gravity will enable the fluid gravity options.

Empty Space *Gas Only*
: Voxels with values under this value are considered empty space. More empty space optimizes rendering. With OpenVDB caching it also reduces cache sizes.

Delete in Obstacle
: Remover any volume of fluid that intersects with an obstacle inside the domain.

Border Collisions

Reference

Panel::	*Physics* ▸ *Fluid* ▸ *Settings* ▸ *Border Collisions*
Type::	Domain (Gas)

Controls which sides of the domain will allow fluid “pass through” the domain, making it disappear without influencing the rest of the simulation, and which sides will deflect fluids.

Gas

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Gas*
Type:: Domain (Gas)

Buoyancy Density
: Buoyant force based on gas density.

  - Values above 0 will cause the gas to rise (simulating gas which is lighter than ambient air).
  - Values below 0 will cause gas to sink (simulating gas which is heavier than ambient air).

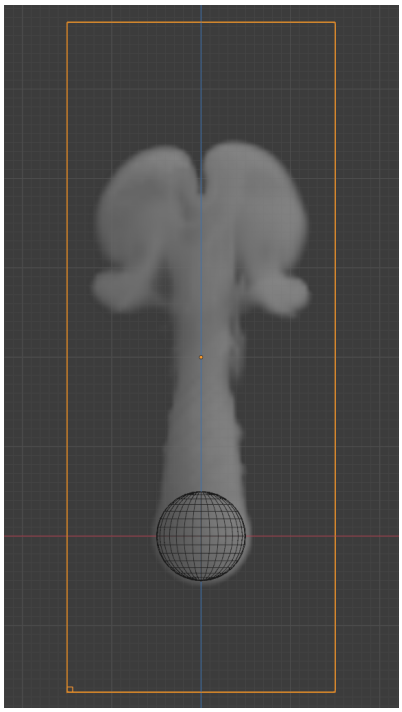
Buoyancy Heat
: Controls how much gas is affected by temperature. The effect this setting has on gas depends on the per flow object Initial Temperature:

  - Values above 0 will result in the gas rising when the flow object *Initial Temperature* is set to a positive value, and gas sinking when the flow object *Initial Temperature* is set to a negative value.
  - Values below 0 will result in the opposite of positive values, i.e. gas emitted from flow objects with a positive *Initial Temperature* will sink, and gas from flow objects with a negative *Initial Temperature* will rise.

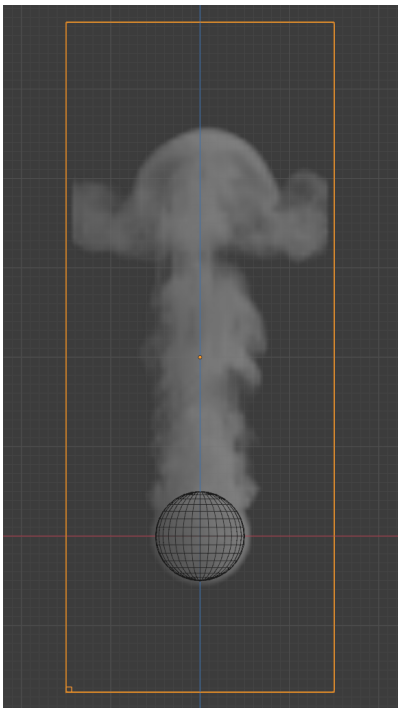
  Note that gas from multiple flow objects with different temperatures will mix and warm up or cool down until an equilibrium is reached.

Vorticity
: Controls the amount of turbulence in the gas. Higher values will make lots of small swirls, while lower values make smoother shapes.

Comparison of different amounts of vorticity.



Domain with a vorticity of 0.0.



Domain with a vorticity of 0.2.

Dissolve

Allow gas to dissipate over time.

Time
: Speed of gas dissipation in frames.

Slow
: Dissolve gas in a logarithmic fashion. Dissolves quickly at first, but lingers longer.

Fire

Reference

Type:: Domain
Panel:: *Physics ▸ Fluid ▸ Gas ▸ Fire*

Reaction Speed
: How fast fuel burns. Larger values result in smaller flames (fuel burns before it can go very far), smaller values result in larger flames (fuel has time to flow farther before being fully consumed).

Flame Smoke
: Amount of extra smoke created automatically to simulate burnt fuel. This smoke is best visible when using a "Fire + Smoke" Flow Object.

Vorticity
: Vorticity for flames in addition to the global fluid Vorticity.

Temperature Maximum
: Maximum temperature of flames. Larger values result in faster rising flames.

Minimum
: Minimum temperature of flames. Larger values result in faster rising flames.

Smoke Color
: Color of smoke emitted from burning fuel.

Liquid

Reference

Type:: Domain
Panel:: *Physics ▸ Fluid ▸ Liquid*

Liquid settings control the behavior of the particles which the simulation consists of. Enabling the liquid checkbox will automatically create a particle system for the simulation. This particle system visualizes the flow of the simulation. Visualizing the liquid particles is optional. The fluid

simulation will make use of all the fields without an attached particle system too.

Note

Disabling the liquid checkbox will delete the attached particle system and its settings.

Simulation Method
: Determines the liquid particle simulation method.

  FLIP
  : Produces a very splashy simulation with lots of particles dispersed in the air.

  APIC
  : Produces a very energetic but also more stable simulation. Vortices within the liquid will be preserved better than with *FLIP*.

FLIP Ratio *Simulation FLIP Only*:
: How much FLIP velocity to use when updating liquid particle velocities. A value of 1.0 will result in a completely FLIP based simulation. Completely FLIP based simulations produce more chaotic splashes and are preferable when simulating greater quantities of liquid. When using smaller values the behavior will be less turbulent and splashes are more subtle. This is optimal when simulating scenes where the liquid is supposed to be on a small scale.

System Maximum
: Maximum number of fluid particles that are allowed in the simulation. If this field is set to a nonzero value the simulation will never contain more than this number of fluid particles. Otherwise, with a value of zero the solver will always sample new particles when needed.

Particle Radius

The radius of one liquid particle in grid cells units. This value describes how much area is covered by a particle and thus determines how much area around it can be considered as liquid. A greater radius will let particles cover more area. This will result in more grids cell being tagged as liquid instead of just being empty.

Whenever the simulation appears to leak or gain volume in an undesired, non physically accurate way it is a good idea to adjust this value. That is, when liquid seems to disappear this value needs to be increased. The inverse applies when too much liquid is being produced.

Sampling

Factor that is used when sampling particles. A higher value will sample more particles. Note that particle resampling occurs at every single simulation step.

Randomness

New particles are sampled with some randomness attached to their position which can be controlled by this field. Higher values will sample the liquid particles more randomly in inflow regions. With a value of 0.0 all new particles will be sampled uniformly inside their corresponding grid cells.

When trying to create a laminar inflow (with little randomness) or more turbulent flows (with greater randomness) this value can be useful.

Particles Maximum

The maximum number of liquid particles per grid cell. During a simulation the number of liquid particles in a cell can fluctuate: Particles can flow into other cells or can get deleted if they move outside the narrow band. Resampling will add new particles considering this maximum.

This value sets the upper threshold of particles per cell. It is also a good way to estimate how many particles there can be in your simulation (one needs to take grid resolution into account too). This can be useful before baking and when planning a simulation.

Minimum

The minimum number of liquid particles per grid cell. Similarly to the maximum particle threshold, this value ensures that there are at least a certain amount of particles per cell.

Narrow Band Width

Controls the width in grid cell units of the narrow band that liquid particles are allowed to flow in. A high value will result in a thicker band and can result in an inflow region completely filled with particles. Unless the goal of the simulation is to visualize the liquid particles it is recommended to not increase the band width significantly as more particles slow down the simulation.

In some situations increasing this value can help create volume when the simulation appears to leak. In all other cases it is best to keep the narrow band as thin as possible since the liquid surface contains most details and simulating particles inside the liquid is not an optimal use of computing resources.

See also

The narrow band is an implementation of Narrow Band FLIP for Liquid Simulations.

Fractional Obstacles

Enables finer resolution in fluid / obstacle regions (second order obstacles). This option reduces the “stepping effect” that results when an obstacle lies inclined inside the domain. It also makes liquid flow more smoothly over an obstacle.

Obstacle Distance

Determines how far apart fluid and obstacles are. This value can be used to achieve a more fluid motion over inclined obstacles: Depending on the slope of the obstacle increasing this value can help liquid particles flow better over an obstacle. Setting this field to

a negative value will let fluid move towards the inside of an obstacle.

Obstacle Threshold
: Value to control the smoothness of the fractional obstacle option. Smaller value reduce the “stepping effect” but may result particles sticking to the obstacle.

Bake Data, Free Data
: This option is only available when using the Modular cache type. *Bake Data* simulates and stores the base of the fluid simulation on drive. Both gas and liquid simulations can add refinements on top of this (e.g. gas simulations can add noise, liquid simulations can add a mesh or secondary particles or both).

  The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

  Once the simulation has been baked, the cache can be deleted by pressing *Free Data*. It is possible to pause or resume a *Bake All* process.

Gas Settings

- Adaptive Domain
- Noise
- Viewport Display
  - Slice
  - Grid Display
  - Vector Display
  - Advanced *Gridlines Only*

Adaptive Domain

Reference

Type:: Domain
Panel:: *Physics ▸ Fluid ▸ Adaptive Domain*

When enabled, the domain will adaptively shrink to best fit the gas, saving computation time by leaving voxels without gas out of the simulation. Unless the *Add Resolution* is used, the adaptive domain will not exceed the bounds of the original domain.

Add Resolution
: Number of voxels to add around the outside of the domain.

Margin
: Amount of extra space to leave around gas, measured in voxels. With very fast-moving gas larger margins may be required to prevent the gas from being cut off by the adaptive boundary, but note this will increase the number of voxels which need to be computed.

Threshold
: Smallest amount of gas a voxel can contain before it is considered empty and the adaptive domain is allowed to cut it out of the simulation.

Noise

Reference

Type:: Domain
Panel:: *Physics* ▸ *Fluid* ▸ *Noise*

Adding noise to the gas simulation creates a finer detailed looking simulation on top of the base. This makes it possible to add more details to gases (i.e. fire or smoke or both) without changing the overall fluid motion.

See also

Fluid noise is an implementation of Wavelet Turbulence for Fluid Simulation.

Besides enabling parts of the interface, checking *Noise* lets the cache know which simulation data to read. If, for example, *Noise* is enabled but there is no noise simulation data to read it will show an empty domain. The checkbox does not reset the cache and can be used to switch the view between base resolution and noise view.

Upres Factor
: Factor by which to enhance the resolution of the noise. The scaling factor is coupled to the Resolution Divisions.

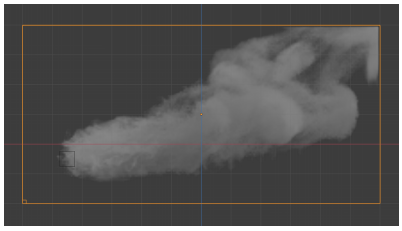
Strength
: Strength of the noise. Higher values result in more turbulent vortices.

Scale
: Scale of the noise. Greater values result in larger vortices.

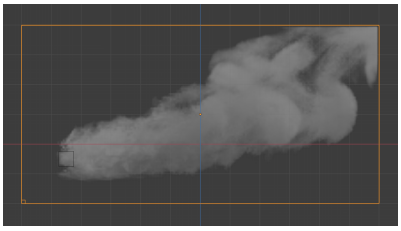
Time

Animation time of the noise. This value has an influence on where the noise field is evaluated. It can be used as a seed to give wavelet noise a slightly different look in two domains that are otherwise the same.

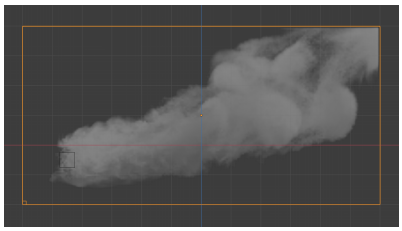
Smoke plume with varying animation time. While the fluid motion of all four smoke plumes are alike each example has a unique look.



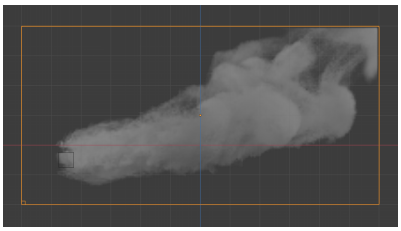
Animation Time: 0.1



Animation Time: 1.0



Animation Time: 2.5

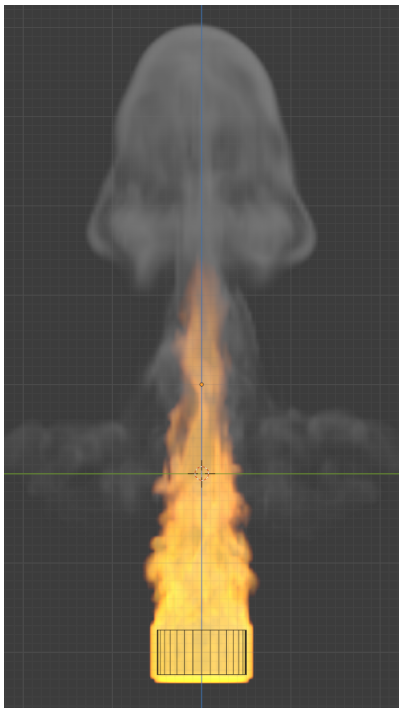


Animation Time: 10.0

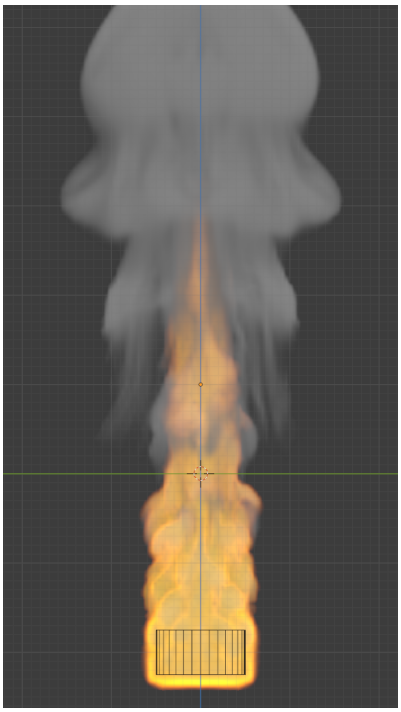
Note

Resolution Divisions and *Upres Factor* are not equivalent. By using different combinations of these resolution settings, you can obtain a variety of different styles of smoke.

Comparison of fire simulations with and without noise at the same grid resolution.



Resolution Divisions:
200, without noise



Resolution Divisions:
100, Noise scale: 2.

Low division simulations with lots of *Upres Factor* divisions generally appear smaller in real-world scale and can be used to achieve pyroclastic plumes such as in the following image:



Viewport Display

Thickness
: Factor that scales the thickness of the grid that is currently being displayed.

Interpolation
: Interpolation method to use for the visualization of the fluid grid.

  Linear
  : Linear interpolation between voxels. Gives good smoothness and speed.

  Cubic
  : Cubic interpolation between voxels. Gives smoothed high quality interpolation, but is slower.

  Closest
  : No interpolation between voxels. Gives raw voxels.

Slice per Voxel
: Determines how many slices per voxel should be generated.

Slice

Renders only a single 2D section of the domain object.

Axis
: Auto
  : Adjust slice direction according to the view direction.

  X/Y/Z
  : Slice along the X/Y/Z axis.

Position

Position of the slice relative to the length of the respective domain side.

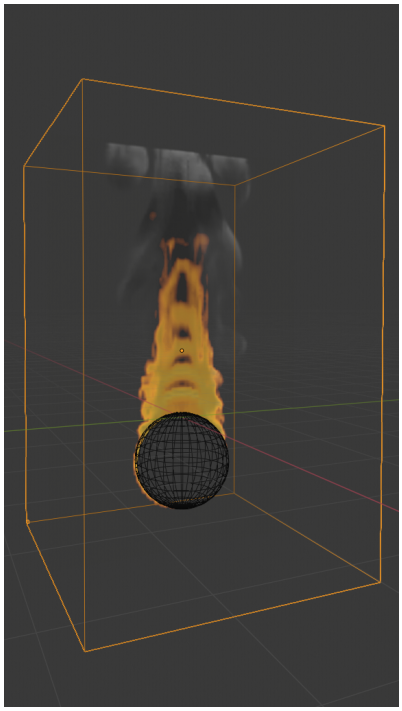
Gridlines *Closest Interpolation Only*
: Display gridlines to differentiate the underlying cells in the current slice of the fluid domain.

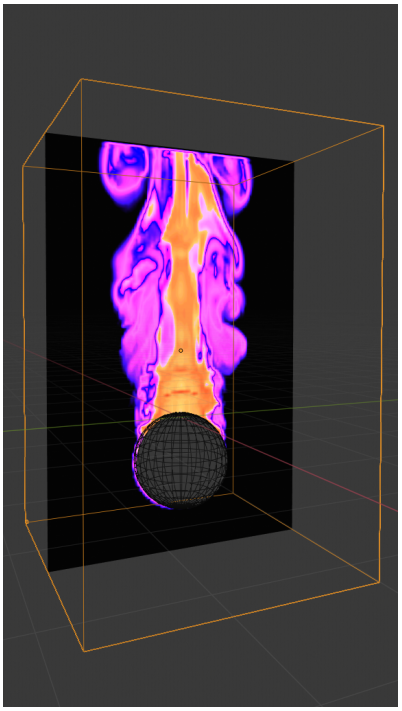
Grid Display

Use a specific color map for the visualization of the simulation field. This comes in handy during debugging or when making more advanced adjustments to the simulation. For instance, if the actual color of a fire simulation is barely visible in the viewport then changing the color profile can help to see the real size of the flame.

Field
: The simulation field used in the display options (e.g. density, fuel, heat).

Comparison of a fire simulation with and without color mapping.





Slice view of “fire” grid without color mapping.	Slice view of “fire” grid with color mapping.

Scale
: Scale the selected simulation field by this value.

Vector Display

Visualization options for the vector fields.

Display As
: Streamlines
  : Choose to display the vectors as “Streamlines”.

  Needle
  : Choose to display the vectors as “Needles”.

  MAC Grid
  : Choose to display the vector field as “Marker-And-Cell Grid”.

    X/Y/Z
    : Show an individual X/Y/Z component of the MAC grid.

Magnitude *Streamlines or Needle Only*
: Scale the display vectors by the magnitude of the vectors they represent.

Field
: The vector field represented by the display vectors (e.g. fluid velocity, external forces).

Scale
: Scale the vectors by this size in the viewport.

Advanced *Gridlines Only*

Advanced coloring options for gridlines.

Color Gridlines

Flags

Color gridlines with flags.

Highlight Range *Grid Display Only*

Highlight the cells with values of the displayed grid within the range. Values between the *Lower Bound* and *Upper Bound* (inclusive) are considered to be within the range.

Lower Bound

Lower bound of the highlighting range.

Upper Bound

Upper bound of the highlighting range.

Color

Color used to highlight the cells.

Cell Type

Choose to highlight only a particular type of cells.

Liquid Settings

- Diffusion
  - High Viscosity Solver
- Particles
- Mesh

Diffusion

Reference

Type:: Domain
Panel:: *Physics ▸ Fluid ▸ Diffusion*

Liquid diffusion defines the physical properties of a liquid and in turn define how a liquid interacts with its environment. The main factors of diffusion are the *Viscosity* and *Surface Tension*. These properties can be adjusted to create virtual liquids that behave like water, oil, honey, or any other liquid. A couple presets exist to change the diffusion for different substances are predefined and can be changed in the preset menu. Fluid Diffusion settings can be enabled/disabled in the panel header.

Viscosity

The viscosity refers to the "thickness" of the fluid and actually the force needed to move an object of a certain surface area through it at a certain speed.

For manual entry, please note that real-world viscosity (the so-called dynamic viscosity) is normally measured in Pascal-seconds (\(Pa\cdot s\)), or in Poise units (P, equal to 0.1 \(Pa\cdot s\)), and commonly centiPoise units (cP, equal to 0.001 \(Pa\cdot s\)).

Blender, on the other hand, uses the kinematic viscosity which is the dynamic viscosity divided by the density, \(\frac{Pa\cdot s}{kg/m^{3}}\), which is \(m^{2}/s\). So for example, the viscosity of water at room temperature is 1.002 cP, or 0.001002 \(Pa\cdot s\); the density of water is about 1000 \(kg/m^{3}\), which gives a kinematic viscosity of 0.000001002 \(m^{2}/s\) – so the entry would be 1.002 times 10 to the minus six (1.002×10^{-6} in scientific notation).

The table below gives some examples of fluids together with their dynamic and kinematic viscosities.

Blender viscosity unit conversion.

Fluid	Dynamic viscosity (in cP)	Kinematic viscosity (Blender, in \(m^{2}/s\))
Water (20 °C)	1.002×10^{0} (1.002)	1.002×10^{-6} (0.000001002)
Oil SAE 50	5.0×10^{2} (500)	5.0×10^{-5} (0.00005)
Honey (20 °C)	1.0×10^{4} (10,000)	2.0×10^{-3} (0.002)
Chocolate Syrup	3.0×10^{4} (30,000)	3.0×10^{-3} (0.003)
Ketchup	1.0×10^{5} (100,000)	1.0×10^{-1} (0.1)
Melting Glass	1.0×10^{15}	1.0×10^{0} (1.0)

Tip

You can find the kinematic viscosity of more materials in the proper units by asking Wolfram Alpha, e.g. "kinematic viscosity of alcohol in m^2/s".

To simplify the input of these numbers, the viscosity is changed by entering values in scientific notation by entering a base value and the exponent of that number.

Base
: The base of the viscosity value (e.g. 1.002 in the case of water (20 °C)).

Exponent
: The exponent of the viscosity value that will be multiplied by 10^{-1} (e.g. 6 in the case of water (20 °C)).

Note

Viscosity Varies

The default values in Blender are considered typical for those types of fluids and “look right” when animated. However, actual viscosity of some fluids, especially sugar-laden fluids like chocolate syrup and honey, depend highly on temperature and concentration. Oil viscosity varies by SAE rating. Glass at room temperature is basically a solid, but glass at 1500 °C flows (nearly) like water.

Warning

The simulator is not suitable for non-fluids, such as materials that do not “flow”. Simply setting the viscosity to very large values will not result in rigid body behavior, but might cause instabilities.

Surface Tension

Surface tension in grid units. Higher value will produce liquids with greater surface tension.

High Viscosity Solver

The high viscosity liquid solver can be used to simulate fluids with increased viscosity, replicating the behavior of substances like honey or molasses. This specialized solver enhances the accuracy of slow-moving and thick liquid simulations.

Strength

The viscosity of the liquid. Higher values result in more viscous fluids.

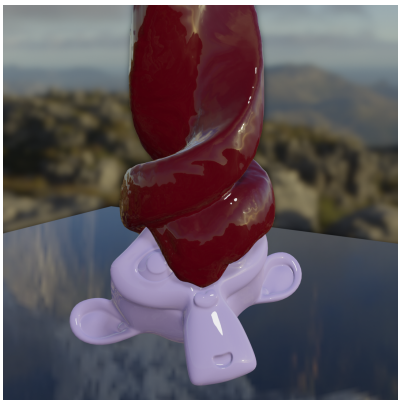
Note

A strength value of 0 will still apply some viscosity. Uncheck the *High Viscosity Solver* to disable the high viscosity liquid solver simulation step completely.

Rotating liquid inflow with varying viscosities.



Strength of 0.2 (at frame 65).



Strength of 0.4 (at frame 200).

Particles

Spray
: Create spray particles during the secondary particle simulation. Spray particles are those that appear to fly through the air above the liquid surface when there is a bigger splash.

Foam
: Create foam particles during the secondary particle simulation. Foam particles are those that solely move on the liquid surface.

Bubbles
: Create bubble particles during the secondary particle simulation. Bubble particles are those that move below the liquid surface.

Note

Enabling a secondary particle type will also create a particle system for that type of particles. Disabling a particle type will delete this particle system including its settings.

Combined Export
: Select particle types that should go into the same particle system. This option has no effect on the outcome of the simulation. It only changes the way particle systems are allocated in the particle settings.

Upres Factor
: Factor by which to enhance the resolution of the particle simulation. The scaling factor is coupled to the Resolution Divisions (i.e. the particle simulation is this times bigger than the base simulation).

Wave Crest Potential Maximum
: Upper clamping threshold for marking fluid cells as wave crests. A higher value results in less marked cells.

Wave Crest Potential Minimum
: Lower clamping threshold for marking fluid cells as wave crests. A lower value results in more marked cells.

Trapped Air Potential Maximum
: Upper clamping threshold for marking fluid cells where air is trapped. A higher value results in less marked cells.

Trapped Air Potential Minimum
: Lower clamping threshold for marking fluid cells where air is trapped. A lower value results in more marked cells.

Kinetic Energy Potential Maximum
: Upper clamping threshold that indicates the fluid speed where cells start to emit particles. A higher value results in generally less particles.

Kinetic Energy Potential Minimum
: Lower clamping threshold that indicates the fluid speed where cells start to emit particles. A lower value results in generally more particles.

Potential Radius
: Radius to compute potential for each cell. Higher values are slower but create smoother potential grids.

Particle Update Radius
: Radius to compute position update for each particle. Higher values are slower but particles move less chaotic.

Wave Crest Particle Sampling
: Maximum number of particles generated per wave crest cell per frame.

Trapped Air Particle Sampling
: Maximum number of particles generated per trapped air cell per frame.

Particle Life Maximum
: Highest possible particle lifetime.

Particle Life Minimum

Lowest possible particle lifetime.

Bubble Buoyancy
: Amount of buoyancy force that rises bubbles. A high value results in bubble movement mainly upwards.

Bubble Drag
: Amount of drag force that moves bubbles along with the fluid. A high value results in bubble movement mainly along with the fluid.

Particles in Boundary
: Delete
  : Delete secondary particles that are inside obstacles or left the domain.

  Push Out
  : Push secondary particles that left the domain back into the domain.

Bake Particles, Free Particles
: This option is only available when using the Modular cache type.

  The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

  Once the simulation has been baked, the cache can be deleted by pressing *Free Particles*. It is possible to pause or resume a *Bake Particles* process.

Mesh

The liquid mesh is, besides the liquid particles, another way to visualize the liquid simulation. It is generated directly from the liquid particles and often uses a higher resolution than the base Resolution Divisions.

Besides enabling parts of the interface, checking *Mesh* lets the cache know which simulation data to read. If, for example, *Mesh* is enabled but there is no mesh simulation data to read it will show the original domain. The checkbox does not reset the cache and can be used to switch the view between the original domain and the baked liquid mesh.

It is important to keep in mind that the shape of the mesh depends on a combination of *all* these parameters. E.g. changing the Particle Radius will lead to a different interpretation of the concavity values.

Upres Factor
: Factor by which to enhance the resolution of the mesh. The scaling factor is coupled to the Resolution Divisions (i.e. the mesh is this times bigger than the base simulation).

Particle Radius
: The radius of one liquid particle in grid cells units. This value describes how much area is covered by a particle and thus determines how much area around it can be considered as liquid. A greater radius will let particles cover more area. This will result in meshes covering more volume around liquid particles.

  This property refers to the same Particle Radius described in the liquid domain settings. Yet for the mesh, it is useful to interpret the particle radius on its own. For one, the mesh can have a resolution different from the base resolution through the Upres Factor. For another, it is often desirable to be able to control the mesh size around a single liquid particle.

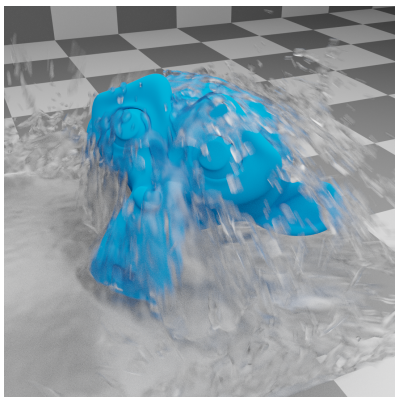
Use Speed Vectors

Creates a `velocity` Attribute which records the velocity of each vertex per frame. These will be used (automatically) when rendering with motion blur enabled.

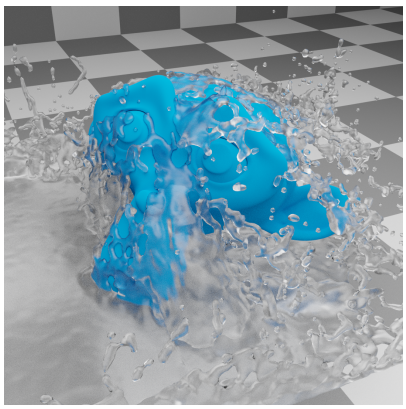
Note

In order to render motion blur with Cycles, Deformation Motion Blur must be enabled.

Comparison of a liquid splash with and without motion blur (rendered with Cycles).



Motion blur enabled.



Motion blur disabled.

Mesh Generator

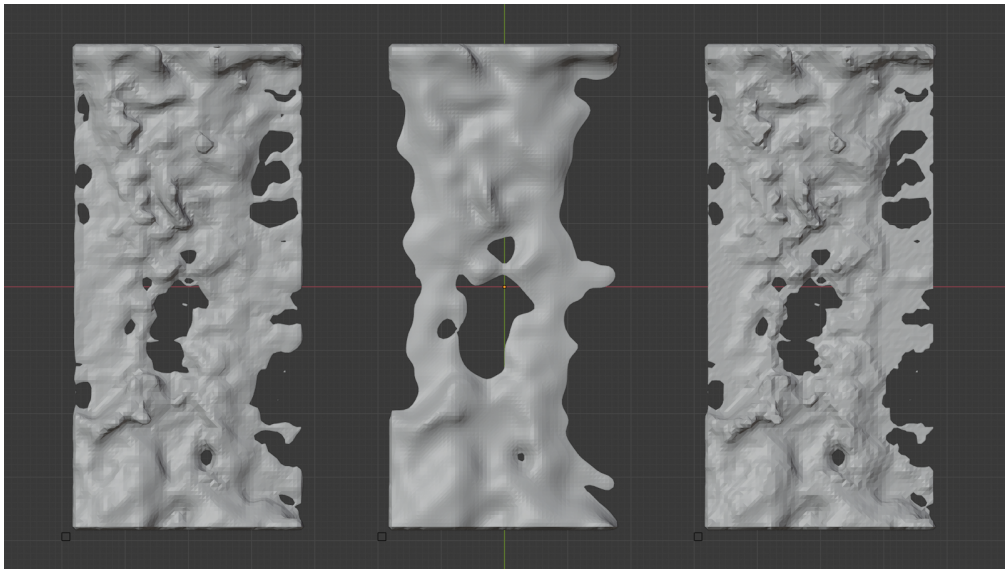
The mesh generator method determines the accuracy of the mesh. The *Final* option produces a higher quality mesh and provides more configuration option than the *Preview* option which in turn is faster but not as smooth.

Smoothing Positive

Positive mesh smoothing iterations. Higher values will make the mesh outline increasingly smooth. Yet higher values can prevent small details (e.g. smaller liquid drops) from getting meshed.

Smoothing Negative

Negative mesh smoothing iterations. Higher values will make the mesh outline sharper. High values will preserve details, however, the mesh outline will become more ragged (e.g. a single mesh particle will become less rounded and have more flat sides).



Comparison of a liquid drop hitting a surface (viewed from top) with varying smoothing values. Left: 1, 1 (Smoothing Positive, Smoothing Negative). Middle: 10, 1. Right: 1, 10. Note the slightly sharper corners in the right splash (compared to the left one).

Concavity Upper

Upper mesh concavity bound. High values tend to smoothen and fill out concave regions.

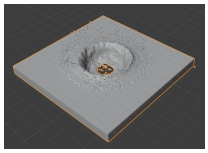
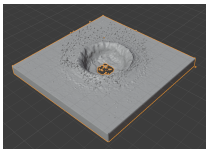
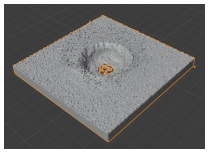
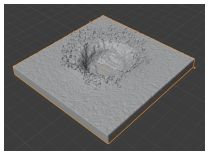
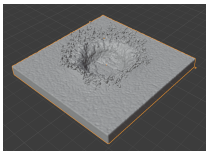
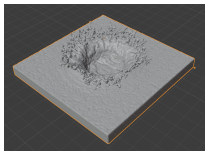
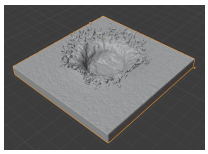
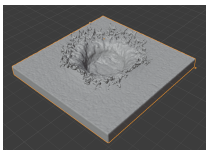
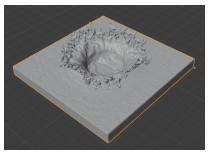
Concavity Lower

Lower mesh concavity bound. High values tend to smoothen and fill out concave regions.

Using a lower concavity which is greater the upper concavity can result in distorted, non-manifold meshes. Unless the artist sees value in this kind of mesh, such concavity value combinations should be avoided.

Crown splash with varying upper and lower concavity settings. Note that setting the concavity values to the same value produces a very

granular mesh.

 Upper: 1.0, Lower: 0.0.	 Upper: 1.0, Lower: 0.5.	 Upper: 1.0, Lower: 1.0.
 Upper: 1.5, Lower: 0.0.	 Upper: 1.5, Lower: 0.5.	 Upper: 1.5, Lower: 1.0.
 Upper: 2.0, Lower: 0.0.	 Upper: 2.0, Lower: 0.5.	 Upper: 2.0, Lower: 1.0.

Bake Mesh, Free Mesh

This option is only available when using the Modular cache type.

The progress will be displayed in the status bar. Pressing `Esc` will abort the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Mesh*. It is possible to pause or resume a *Bake Mesh* process.

Guides

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Guides*
Type:: Domain

Fluid guides are used to apply forces onto the simulation. They are like simple external forces but also seek to preserve the physically accurate flow of the fluid. The *Guides* panel allows you to adjust guiding forces globally, i.e. for the entire domain. Enabling the guides hints the fluid solver to use the more accurate, but also computationally more expensive pressure solving step.

Even when there are no guiding objects baked or there is no guiding domain attached, the fluid solver will still perform the more expensive pressure guiding algorithm if guiding is enabled. It is therefore recommended to only enable *Guides* when there is a clear intention to use guiding in the simulation.

See also

Fluid guiding is an implementation of Primal-Dual Optimization for Fluids.

Weight
: Controls the lag of the guiding. A larger value (also known as the 'alpha' guiding value) results in a greater lag.

Size
: This setting determines the size of the vortices that the guiding produces. A greater guiding size (also known as the blur radius or 'beta' guiding value) results in larger vortices.

Velocity Factor

All guiding velocities are multiplied by this factor. That is, every cell of the guiding grid, which has the same size as the domain object, is multiplied by this factor.

Velocity Source

Guiding velocities can either come from objects that move inside the domain or from other fluid domains.

Effector

All effector objects inside the domain will be considered for the global guiding velocity grid. Once effector objects have been baked it is not possible to change the fluid domain resolution anymore.

Domain

When using another fluid domain as the guiding velocity source this domain may have a different resolution and may also be of a different type (e.g. the guiding domain is of type *Gas* while the actual domain with the guiding effect in it is of type *Liquid*).

In order to use a domain as the velocity source, this domain needs to be baked already.

Guide Parent

When using *Domain* as the velocity source, this field serves to select the guiding domain object.

Bake Guides, Free Guides

This option is only available when using the Modular cache type and when using *Effector* as the Velocity Source. *Bake Guides* writes vertex velocities of effector objects to drive. It is meant to be used before baking the fluid simulation.

The progress will be displayed in the status bar. Pressing `Esc` will pause the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free Guides*. It is possible to pause or resume a *Bake Guides*

process.

Collections

Reference

Type:: Domain
Panel:: *Properties ▸ Physics ▸ Fluid ▸ Collections*

Flow
: If set, only objects in the specified Collection will be allowed to act as Flow objects in this domain.

Effector
: If set, only objects in the specified Collection will be allowed to act as Effector objects in this domain.

Cache

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Cache*
Type:: Domain

The *Cache* panel is used to Bake the fluid simulation and stores the outcome of a simulation so it does not need to be recalculated.

Baking takes a **lot** of compute power (hence time). Depending on the scene, it is recommended to allocate enough time for the baking process.

If the mesh has modifiers, the rendering settings are used for exporting the mesh to the fluid solver. Depending on the setting, calculation times and memory use might exponentially increase. For example, when using a moving mesh with *Subdivision Surface* as an obstacle, it might help to decrease simulation time by switching it off, or to a low subdivision level. When the setup/rig is correct, you can always increase settings to yield a more realistic result.

Note

Fluid simulations use their own cache. All other physics simulations make use of the General Baking operators.

Cache Directory
: Directory to store baked simulation files in. Inside this directory each simulation type (i.e. mesh, particles, noise) will have its own directory containing the simulation data.

Type
: The type of the cache determines how the cache can be baked.

Replay
: The cache will be baked as the simulation is being played in the viewport.

Modular
: The cache will be baked step by step: The bake operators for this type are spread across various panels within the domain settings (e.g. the bake tool for the mesh can be found in the Mesh panel).

All
: The cache will be baked with a single tool. All selected settings will be considered during this bake. The bake tool for this type can be found in the Cache panel.

Important

"Replay" only works when the Playback Sync mode is set to "Play Every Frame". If you need to use "Frame Dropping" or "Sync to Audio", consider using the "Modular" or "All" options below.

Start
: Frame on which to start the simulation. This is the first frame that will be baked.

End
: Frame on which to stop the simulation. This is the last frame that will be baked.

Note

The simulation is only calculated for positive frames between the *Start* and *End* frames of the *Cache* panel. So if you want a simulation that is longer than the default frame range you have to change the *End* frame.

Offset

Frame offset that is used when loading the simulation from the cache. It is not considered when baking the simulation, only when loading it.

Use Resumable Cache
: Extra data will be saved so that you can resumed baking after pausing. Since more data will be written to drive it is recommended to avoid enabling this option when baking at high resolutions.

Volume File Format
: File format for volume based simulation data (i.e. grids and particles).

  Uni Cache
  : Blender's own caching format with some compression. Each simulation object is stored in its own `.uni` cache file.

  OpenVDB
  : Advanced and efficient storage format. All simulation objects (i.e. grids and particles) are stored in a single `.vdb` file per frame.

Mesh File Format *Liquids Only*
: File format for the mesh cache files.

  Binary Object
  : Mesh data files with some compression.

  Object
  : Simple, standard data format for mesh data.

Bake All, Free All
: This option is only available when using the Final cache type. *Bake All* will run the simulation considering all parameters from the settings (i.e. it will bake all steps that can be baked individually with the Modular cache type at once).

  The progress will be displayed in the status bar. Pressing `Esc` will abort the simulation.

Once the simulation has been baked, the cache can be deleted by pressing *Free All*. It is not possible to pause or resume a *Bake All* process as only the most essential cache files are stored on drive.

Advanced

Compression Volumes *OpenVDB Only*
: Compression method that is used when writing OpenVDB cache files.

  None
  : Cache files will be written without any compression.

  Zip
  : Cache files will be written with `Zip` compression. Effective but slower than `Blosc`.

  Blosc
  : Cache files will be written with `Blosc` compression. Multithreaded compression, similar in size and quality to `Zip` compression.

Precision Volumes *OpenVDB Only*
: Precision level that is used when writing OpenVDB cache files.

  Full
  : Volumetric data (e.g. grids, particles) will be written with full precision (32-bit).

  Half
  : Volumetric data (e.g. grids, particles) will be written with half precision (16-bit).

  Mini
  : Volumetric data (e.g. grids, particles) will be written with mini float precision (8-bit) where possible. For cache data where this is not possible, 16-bit floats will be used instead.

Export Mantaflow Script

Export the simulation as a standalone Mantaflow script when baking the scene (exported on “Bake Data”). Usually, only developers and advanced users who know how to use the Mantaflow GUI will make use of this functionality. Use a Debug Value of `3001` to enable.

Field Weights

Reference

Panel::	*Properties* ▸ *Physics* ▸ *Fluid* ▸ *Field Weights*
Type::	Domain

These settings determine how much gravity and Force Fields affect the fluid.

Effector Collection
: When set, fluid can only be influenced by force fields in the specified collection.

Gravity
: How much the fluid is affected by Gravity.

All
: Overall influence of all force fields.

The other settings determine how much influence individual force field types have.

Flow

Fluid *Flow* types are used to add or remove fluid to a domain object. Flow objects should be contained within the domain's Bounding Box in order to work.

To define any mesh object as a *Flow* object, add Fluid physics by clicking *Fluid* in *Properties* ▸ *Physics*. Then select *Flow* as the fluid *Type*. Now you should have a default fluid flow source object.

Settings

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Settings*
Type:: Flow

Flow Type
: Smoke
: Emit only smoke.

: Fire + Smoke
: Emit both fire and smoke.

: Fire
: Emit only fire. Note that the domain will automatically create some smoke to simulate smoke left by burnt fuel.

: Liquid
: Emit liquid.

Flow Behavior
: Controls if the Flow object either adds (*Inflow*), removes (*Outflow*), or turns the mesh itself into fluid (*Geometry*).

Inflow
: This object will emit fluid into the simulation, like a water tap or base of a fire.

Outflow
: Any fluid that enters the Bounding Box of this object will be removed from the domain (think of a drain or a black hole). This can be useful in combination with an inflow to prevent the whole domain from filling up. Outflow objects can be animated and the area where the fluid disappears will follow the object as it moves around.

Geometry
: All regions of this object that are inside the domain bounding box will be used as actual fluid in the simulation. You can place more than one fluid object inside the domain. Also make sure that the surface normals are pointing outwards or else they will not simulate properly. In contrast to domain objects, the actual mesh geometry is used for fluid objects.

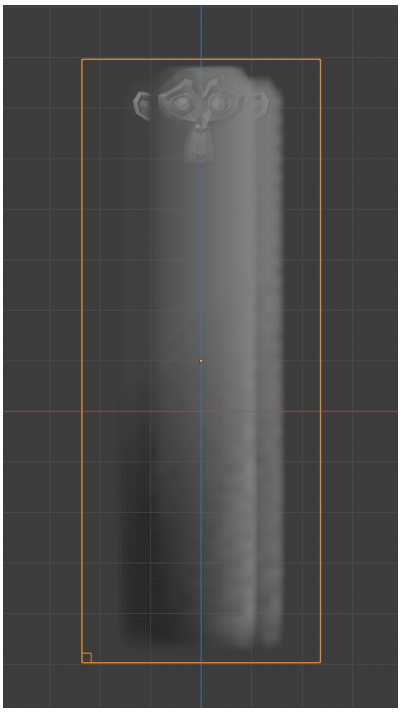
Use Flow
: Enables or disables the flow of fluid, this property is useful for animations to selectively enable and disable when fluid is being added to or removed from the domain.

Sampling Substeps
: Number of sub-steps used to reduce gaps in emission of fluid from fast-moving sources.

Comparison of smoke inflow quickly rising upwards at different sub-step rates.



Sampling sub-steps: 0.



Sampling sub-steps: 3.

Note that these emission sub-steps occur at every simulation step and not per frame. The simulation step count is controlled by the adaptive time stepping.

Smoke Color
: The color of emitted smoke. When smoke of different colors are mixed they will blend together, eventually settling into a new combined color.

Absolute Density
: If this checkbox is enabled, the emitter will only produce more smoke or fire if there is space for it in the emitter region. Otherwise smoke or fire will always be produced and add up.

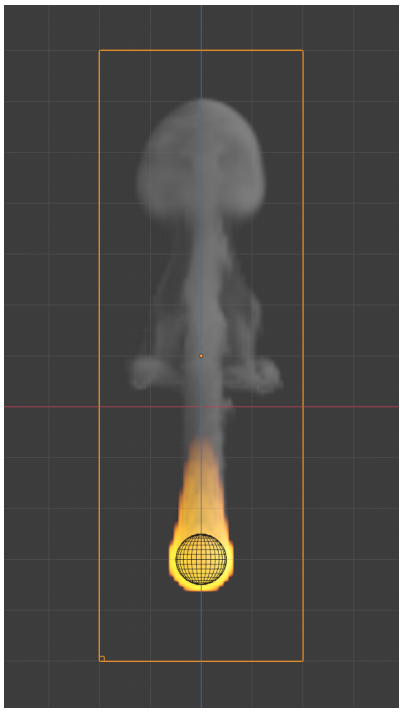
Initial Temperature
: Difference between the temperature of emitted smoke and the domain's ambient temperature. This setting's effect on smoke depends on the domain's Heat Buoyancy.

Density

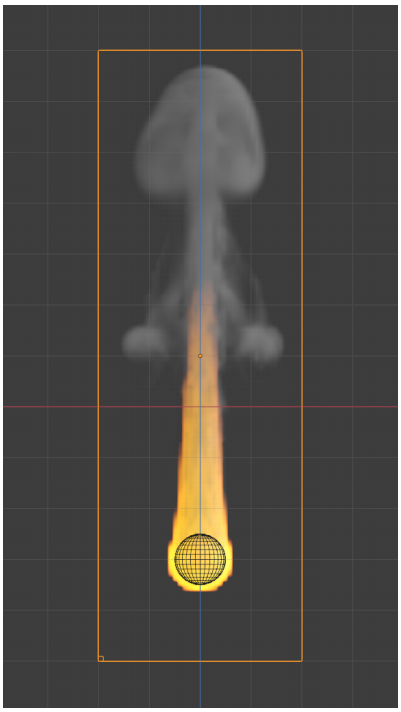
Amount of smoke to emit at once. Larger values result in more density being produced.

Fuel
: Amount of “fuel” being burned per second. Larger values result in larger flames, smaller values result in smaller flames:

Comparison of flames with varying fuel rates.



Fuel: 0.5.



Fuel: 1.0.

Vertex Group
: When set, use the specified Vertex Group to control where smoke is emitted.

Flow Source

Flow Source
: This setting defines the method used to emit fluid.

  Mesh

Emit fluid directly from the object's mesh.

Is Planar
: Defines the effector as either a single dimension object i.e. a plane or the mesh is Non-manifold. This ensures that the fluid simulator will give the most accurate results for these types of meshes.

Surface Emission
: Maximum distance in Voxels from the surface of the mesh in which fluid is emitted. Since this setting uses voxels to determine the distance, results will vary depending on the domain's resolution.

Volume Emission *Fire or Smoke Only*:
: Amount of fluid to emit inside the emitter mesh, where 0 is none and 1 is the full amount. Note that emitting fluid based on volume can have unpredictable results if your mesh is Non-manifold.

Particle System *Fire or Smoke Only*:
: Create smoke or fire from a particle system on the flow object. The particle system can be selected with a Data ID.

Note that only *Emitter* type particle systems can add smoke. See Particles for information on how to create a particle system.

Set Size
: When this setting is enabled, it allows the *Size* setting to define the maximum distance in voxels at which particles can emit smoke, similar to the *Surface Emission* setting for mesh sources.

When disabled, particles will fill the nearest Voxel with smoke.

Initial Velocity

When enabled, the fluid will inherit the momentum of the flow source.

Source
: Factor for the inherited velocity. A value of 1 will emit fluid moving at the same speed as the source.

Normal
: This option controls how much velocity fluid is given along a face Normal. Note that, initial velocities will always be applied along all face normals. Thus with a closed flow source mesh, fluid will always be emitted in more than one direction. To set initial velocities along only one direction all normals need to point in the same direction. This is can be achieved when using a plane as the flow object.

Initial X, Y, Z
: Initial velocity along X, Y, Z coordinates in world space. This can be used in addition to the initial velocity along the Normal.

Texture

Reference

Type:: Flow
Panel:: *Physics* ▸ *Fluid* ▸ *Settings* ▸ *Texture*

When enabled, use the specified texture and settings to control where on the mesh smoke or fire can be emitted from. These settings have no effect on *Outflow Flow Behavior*.

Texture
: A Data ID selector to choose the Texture.

Mapping
: Controls whether to use Generated UVs or manual UV mapping.

Size
: Overall texture scale.

Offset

Effector

Effector objects are used to deflect fluids and influence the fluid flow. To define any mesh object as an effector object, add fluid physics by clicking *Fluid* in *Properties* ▸ *Physics*. Then select *Effector* as the fluid *Type*.

Tip

Force Fields (such as wind or vortex) are supported, like in most physics systems. The influence individual force types have can be controlled per domain object.

Settings

Reference

Panel:: *Physics* ▸ *Fluid* ▸ *Settings*
Type:: Effector

Effector Type
: Collision
: Objects of this type will collide with fluid.

: Guide
: The velocity of objects of this type will be used when baking the guiding. So fluid guiding objects should move and have some velocity.

: Velocity Factor
: Multiply the guiding object velocities by this factor. This is useful when working with multiple guiding objects and some of them should have higher or smaller velocities.

Guide Mode

The mode describes how guiding velocities should be written into the global guiding velocity field of the domain.

Maximize

The guiding object will compare the existing velocity in the global velocity field with its own velocity. If its absolute value is greater than the absolute value in the velocity field the guiding velocity will be kept.

Minimize

A guiding object will compare the existing velocity in the global velocity field with its own velocity. If its absolute value is smaller than the absolute value in the velocity field the guiding velocity will be kept.

Override

The most intuitive option. A guiding object will always write its own current velocity into the global guiding velocity field. Values in the velocity field from a previous frame or guiding object will be overridden.

Averaged

A guiding object will write the average of its own current velocity and the existing guiding velocity at that cell into the global guiding velocity field.

Effector Substeps

Number of substeps used to reduce gaps in collision of fluid from fast-moving effectors.

Surface Thickness

Additional area around the effector that will be considered as an effector.

Use Effector

Enables or disables the effector object effect on the fluid, this property is useful for animations to selectively enable and disable when the

effector affects the fluid.

Is Planar

Defines the effector as either a single dimension object i.e. a plane or the mesh is Non-manifold. This ensures that the fluid simulator will give the most accurate results for these types of meshes.

A Manifold mesh can also be declared as planar. The fluid solver will then ignore the volume inside the mesh and just emit fluid from the mesh sides.

Materials

Smoke Material

Realistic smoke can be rendered with the Principled Volume shader.

Smoke Material Example Animation

Particle System

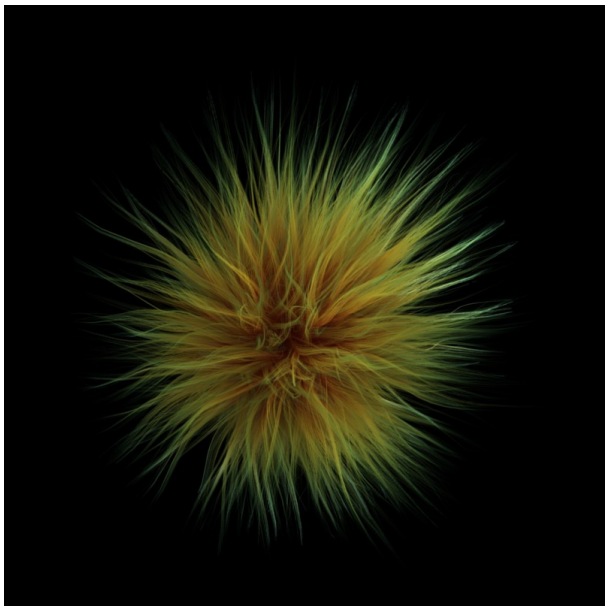
- Introduction
- Particle System Panel
  - Workflow
- Emitter
  - Emission
  - Cache
  - Velocity
  - Rotation
  - Physics
  - Render
  - Viewport Display
  - Children
  - Force Fields
  - Vertex Groups
- Hair
  - Introduction
  - Emission
  - Hair Dynamics
  - Render
  - Shape
  - Children
  - Viewport Display
- Texture Influence
  - General
  - Physics
  - Hair
- Particle Edit Mode
  - Usage
  - Selecting
  - Tools
  - Options
  - Editing

Introduction

Particles are lots of items emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Dynamic particles can represent fire, smoke, mist, and other things such as dust or magic spells.

Hair type particles are a subset of regular particles. Hair systems form curves that can represent hair, fur, grass and bristles.

You see particles as a Particle Modifier, but all settings are done in the *Particle tab*.



Some fur made from particles.

Particles generally flow out from their mesh into space. Their movement can be affected by many things, including:

- Initial velocity out from the mesh.
- Movement of the emitter (vertex, face or object) itself.

- Movement according to “gravity” or “air resistance”.
- Influence of force fields like wind, vortexes or guided along a curve.
- Interaction with other objects like collisions.
- Partially intelligent members of a flock (herd, school, …), that react to other members of their flock, while trying to reach a target or avoid predators.
- Smooth motion with soft body physics (only *Hair* particle systems).
- Or even manual transformation with Lattices.

Particles may be rendered as:

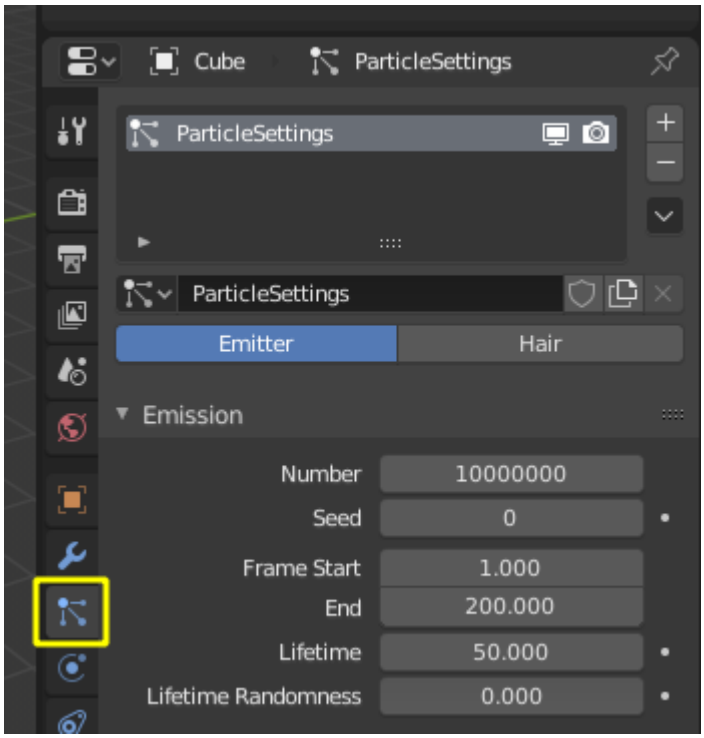
- Halos (for Flames, Smoke, Clouds).
- Meshes which in turn may be animated (e.g. fish, bees, …). In these cases, each particle “carries” another object.
- Hair curves, following the path of the particle. These hair curves can be manipulated in the 3D Viewport (combing, adding, cutting, moving, etc.).

Every object may carry many particle systems. Each particle system may contain up to 10,000,000 particles. Certain particle types (*Hair* and *Keyed*) may have up to 10,000 children for each particle (children move and emit more or less like their respective parents). The size of your memory and your patience are your practical boundaries.

Particle System Panel

Reference

Panel:: *Particle System ▸ Particle System*



Particle System panel.

These are the basic settings.

Active Particle System
The List View of the objects Particle Modifier(s).

Specials
Copy Active to Selected Objects
Copies the active particle system to all selected objects.

Copy All to Selected Objects

Copies all particle systems from the active object to all selected objects.

Duplicate Particle Systems
: Duplicates the particle system within the active object. The *Duplicate Settings* option (in the Adjust Last Operation panel) will duplicate settings as well, so the new particle system uses its own settings.

Particle Settings
: The Data-Block menu for settings.

Type
: Main selector of the system type.

Emitter
: In such a system, particles are emitted from the object.

Hair
: Use Hair type, rendered as strands.

Regrow
: Regrows the hair for each frame. This is useful when you are animating properties.

Advanced
: Enables advanced settings which reflect the same ones as working in Emitter mode.

Note

This manual assumes that this option is enabled.

Segments
: Controls the number of parts a hair is made of. Increasing this value will improve the quality of animations.

Workflow

The process for working with standard particles is:

1. Create the mesh which will emit the particles.
2. Create one or more Particle Systems to emit from the mesh. Many times, multiple particle systems interact or merge with each other to achieve the overall desired effect.
3. Tailor each Particle System's settings to achieve the desired effect.
4. Animate the base mesh and other particle meshes involved in the scene.
5. Define and shape the path and flow of the particles.
6. For Hair particle systems: Sculpt the emitter's flow (cut the hair to length and comb it for example).
7. Make final render and do physics simulation(s), and tweak as needed.

Creating a Particle System

To add a new particle system to an object, go to the *Particles* tab of the Properties editor and click the small + button. An object can have many Particle Systems.

Each particle system has separate settings attached to it. These settings can be shared among different particle systems, so one does not have to copy every setting manually and can use the same effect on multiple objects.

Types of Particle Systems

After you have created a particle system, the Properties fills with many panels and buttons. But do not panic! There are two different types of particle systems, and you can change between these two with the *Type* selector: Emitter and Hair.

The settings in the *Particle System* tab are partially different for each system type.

Emitter

- Emission
  - Source
- Cache
  - Hints
- Velocity
- Rotation
  - Angular Velocity
- Physics
  - Introduction
  - Newtonian
  - Keyed
  - Boids
  - Fluid
- Render
  - Common Settings
  - Render As
  - Extra
- Viewport Display
- Children
  - Common Options
  - Simple
  - Interpolated
  - Example
- Force Fields
  - Field Weights
  - Force Fields Settings
- Vertex Groups

Emission

Reference

Panel:: *Particle System* ▸ *Emission*

The *Emitter* system works just like its name says: it emits/produces particles for a certain amount of time. In such a system, particles are emitted from the selected object from the *Start* frame to the *End* frame and have a certain lifespan. These particles are rendered default as Halos, but you may also render this kind of particles as objects (depending on the particle system's render settings, see Visualization).

The buttons in the *Emission* panel control the way particles are emitted over time:

Number
: The maximum amount of parent particles used in the simulation.

Seed
: Blender uses this as starting point to produce random numbers during the simulation.

Frame Start
: The start frame of particle emission. You may set negative values, which enables you to start the simulation before the actual rendering.

End
: The end frame of particle emission.

Lifetime
: The lifespan (in frames) of the particles.

Lifetime Randomness

A random variation of the lifetime of a given particle. The shortest possible lifetime is *Lifetime* × (1 - *Random*). Values above 1.0 are not allowed. For example with the default *Lifetime* value of 50 a *Random* setting of 0.5 will give you particles with a live span ranging from 50 frames to \(50 × (1.0 - 0.5) = 25\) frames, and with a *Random* setting of 0.75 you will get particles with live spans ranging from 50 frames to \(50 × (1.0 - 0.75) = 12.5\) frames.

Source

Reference

Panel:: *Particle System* ▸ *Emission* ▸ *Source*

Emit From
: Defines how and where the particles are emitted, giving precise control over their distribution.

Tip

You may use vertex groups to confine the emission, that is done in the *Vertex Groups* panel.

Vertices
: Emits particles from the vertices of a mesh.

Faces
: Emits particles from the surface of a mesh's faces.

Volume
: Emits particles from the volume of an enclosed mesh.

Tip

Your mesh must be Manifold to emit particles from the volume. Some modifiers like the Edge Split Modifier break up the surface,

in which case volume emission will not work correctly!

Use Modifier Stack

Take any Modifiers above the Particle Modifier in the modifier stack into account when emitting particles, else it uses the original mesh geometry.

Note

Note that particles may differ in the final render if these modifiers generate different geometry between the viewport and render.

Distribution

These settings control how the emissions of particles are distributed throughout the emission locations when emitting from either *Faces* or *Volume*.

Jittered

Particles are placed at jittered intervals on the emitter elements.

Particles/Face

Number of emissions per face (0 = automatic).

Jittering Amount

Amount of jitter applied to the sampling.

Random

Particles are emitted from random locations in the emitter's elements.

Grid

Particles are set in a 3D grid and particles near/in the elements are kept.

Invert Grid

Invert what is considered the object and what is not.

Hexagonal
: Uses a hexagonal-shaped grid instead of a rectangular one.

Resolution
: Resolution of the grid.

Random
: Add a random offset to grid locations.

Random Order
: The emitter element indices are gone through in a random order instead of linearly (one after the other).

  Not available for *Grid* distribution.

Even Distribution
: Particle distribution is made even based on surface area of the elements, i.e. small elements emit less particles than large elements, so that the particle density is even.

Cache

Reference

Panel:: *Particle System* ▸ *Cache*

In order to improve real-time response and avoid unnecessary recalculation of particles, the particle data can be cached in memory or stored on a drive.

The *Emitter* particle system uses a unified system for caching and baking (together with Soft Body and Cloth).

See also

See the General Baking docs for more information.

Hints

- The simulation is only calculated for positive frames in between the *Start* and *End* frames of the *Cache* panel, whether you bake or not. So if you want a simulation that is longer than the default frame range, you have to change the *End* frame.
- When an animation is played, each physics system writes each frame to the cache. Note that for the cache to fill up, one has to start the playback before or on the frame that the simulation starts.
- The cache is cleared automatically on changes. But not on all changes, so it may be necessary to delete it manually, e.g. if you change a force field.
- The system is protected against changes after baking. If for example the mesh changes the simulation is not calculated anew.
- The bake result can be cleared by clicking on the *Delete Bake* button in the simulation cache settings.

- A simulation can only be edited in Particle Edit Mode when it has been baked in memory. And cannot be edited if the Disk Cache is used.
- If you are not allowed to write to the required subdirectory caching will not take place, e.g. if your blend-file path is very long and your operating system has a limit on the path length that is supported.
- Be careful with the sequence of modifiers in the modifier stack. You may have a different number of faces in the 3D Viewport and for rendering (e.g. when using subdivision surface), if so, the rendered result may be very different from what you see in the 3D Viewport.

Velocity

Reference

Panel:: *Particle System ▸ Velocity*

The initial velocity of particles can be set through different parameters, based on the type of the particle system. If the particle system type is Emitter or Hair, then the following parameters give the particle an initial velocity.

Normal
: The emitter's surface normals (i.e. let the surface normal give the particle a starting speed).

Tangent
: Let the tangent speed give the particle a starting speed.

Tangent Phase
: Rotates the surface tangent.

Object Align
: Give an initial velocity in the X, Y, and Z axes.

  X, Y, Z

Object Velocity
: The emitter objects movement (i.e. let the object give the particle a starting speed).

Randomize
: Gives the starting speed a random variation. You can use a texture to only change the value, see Controlling Emission, Interaction and Time.

Rotation

Reference

Panel:: *Particle System* ▸ *Rotation*

These parameters specify how the individual particles are rotated at the start of, and during, their lifetime. You can visualize their orientation by setting *Display As* to *Axis* in the Viewport Display panel.

Orientation Axis
: Aligns the X axis of new particles to:

  None
  : The global X axis.

  Normal
  : The emitter's surface normal.

  Normal-Tangent
  : The emitter's surface normal, additionally aligning the particle's Y axis to the positive V direction in the emitter's active UV map. This makes it possible to deform the emitter while keeping particle rotation consistent.

  Velocity / Hair
  : The particle's initial velocity vector/hair growth direction.

  Global X, Y, Z
  : One of the global axes.

  Object X, Y, Z
  : One of the emitter's local axes.

Randomize

How much to randomize the particle's initial rotation (along all axes).

Phase
: Initial rotation around the particle's X axis, going from -1 (-180°) to 1 (180°).

Randomize Phase
: Maximum random rotation to add to the *Phase*, going from 0 (0°) to 2 (360°).

Dynamic
: Whether the particles' rotation can change over time.

Angular Velocity

Reference

Panel:: *Particle System* ▸ *Rotation* ▸ *Angular Velocity*

Lets you configure if and how particles should spin over time. *Dynamic* needs to be enabled for this to work.

Axis
: The axis to spin around. If this is set to *Velocity*, *Horizontal*, or *Vertical*, particles will additionally spin to keep the same orientation relative to their direction of movement, even if *Amount* is zero.

  None
  : Spinning is disabled.

  Velocity
  : Spin around the particle's velocity vector.

  Horizontal
  : Spin around the axis that's horizontal (lying in the global XY plane) and perpendicular to the particle's velocity. Particles moving along

the global Z axis won't spin because no unique rotation axis exists in this case.

Vertical
: Spin around the axis that's perpendicular to both the particle's velocity and the above *Horizontal* axis. Particles moving along the global Z axis won't spin.

Global X, Y, Z
: Spin around the chosen global axis.

Random
: Spin around a random axis.

Hint

If you use a Curve Guide and want the particles to always point in the direction of the curve, you should set the *Orientation Axis* to *Velocity / Hair*, enable *Dynamic*, and set the *Angular Velocity Axis* to *Velocity*.

(For a regular object, you'd normally use the *Follow Curve* option of a Follow Path Constraint or the legacy Follow option of the curve itself, but these don't work for particles.)

Amount
: How fast to spin around the *Axis*.

Physics

- Introduction
  - Common Physics Settings
  - No Physics
- Newtonian
  - Forces
  - Integration
  - Deflection
- Keyed
  - Options
  - Relations
- Boids
  - Movement
  - Battle
  - Misc
  - Relations
  - Boid Brain
- Fluid
  - Options

Introduction

The movement of particles may be controlled in a multitude of ways. Here we will discuss only the particle physics in the narrower sense, i.e. the settings in the Physics panel.

Additional ways of moving particles are:

- By soft body animation (only for Hair particle systems).
- By force fields and along curves.
- By lattices.

Common Physics Settings

Size
: Sets the size of the particles.

Random Size
: Give the particles a random size variation.

Mass
: Specify the mass of the particles.

Multiply Mass with Particle Size
: Causes larger particles to have larger masses.

No Physics

The particles will be given no motion, which makes them belong to no physics system. At first a physics type that makes the particles to be static could seem a bit strange, but it can be very useful at times. None physics make the particles stick to their emitter their whole life time. The initial velocities here are for example used to give a velocity to particles that are

affected by a harmonic effector with this physics type when the effect of the effector ends.

Moreover, it can be very convenient to have particles at disposal (whose both Unborn and Died are visible on render) to groom vegetation and/or ecosystems using *Object* or *Group* types of visualization.

Newtonian

Reference

Panel:: *Particle System ▸ Physics*
Type:: Newtonian

The particles will move according to classical (Newtonian) mechanics. Particles start their life with the specified initial velocities and angular velocities, and move according to external forces. The response to environment and to forces is computed differently, according to the given integrator chosen by the animator.

Forces

Reference

Panel:: *Particle System ▸ Physics ▸ Forces*

Brownian
: Specify the amount of Brownian motion. Brownian motion adds random motion to the particles based on a Brownian noise field. This is nice to simulate small, random wind forces.

Drag
: A force that reduces particle velocity in relation to its speed and size (useful in order to simulate air drag or water drag).

Damp
: Reduces particle velocity (deceleration, friction, dampening).

Integration

Reference

Panel:: *Particle System ▸ Physics ▸ Integration*

Integrators are a set of mathematical methods available to calculate the movement of particles. The following guidelines will help to choose a proper integrator, according to the behavior aimed at by the animator.

Integration

Euler

Also known as "Forward Euler". Simplest integrator. Very fast but also with less exact results. If no dampening is used, particles get more and more energy over time. For example, bouncing particles will bounce higher and higher each time. Should not be confused with "Backward Euler" (not implemented) which has the opposite feature, the energy decrease over time, even with no dampening. Use this integrator for short simulations or simulations with a lot of dampening where speedy calculations are more important than accuracy.

Verlet

Very fast and stable integrator, energy is conserved over time with very little numerical dissipation.

Midpoint

Also known as "2nd order Runge-Kutta". Slower than Euler but much more stable. If the acceleration is constant (no drag for example), it is energy conservative. It should be noted that in example of the bouncing particles, the particles might bounce higher than they started once in a while, but this is not a trend. This integrator is a generally good integrator for use in most cases.

RK4

Short for "4th order Runge-Kutta". Similar to Midpoint but slower and in most cases more accurate. It is energy conservative even if the acceleration is not constant. Only needed in complex simulations where Midpoint is found not to be accurate enough.

Timestep
: The amount of simulation time (in seconds) that passes during each frame.

Subframes
: The number of simulation steps per frame. Subframes to simulate for improved stability and finer granularity in simulations. Use higher values for faster-moving particles.

The following options are only available for Fluid type physics:

Adaptive
: Automatically set the number of subframes.

  Threshold
  : A tolerance value that allows the number of subframes to vary. It sets the relative distance a particle can move before requiring more subframes.

    The number of steps per frame will be at least Subframes + 1. More subframes may be simulated if the fluid becomes turbulent, according to the Threshold.

Deflection

Reference

Panel:: *Particle System* ▸ *Physics* ▸ *Deflection*

Size Deflect
: Use the particle size in deflections.

Die on Hit
: Kill particle when it hits a deflector object.

Collision Collection
: If set, particles collide with objects from the collection.

Keyed

Reference

Panel:: *Particle System* ▸ *Physics*
Type:: Keyed

The path of Keyed particles is determined between particles of any two (or more) particle systems. This allows the creation of a chains of systems to create long strands or groovy moving particles. Basically the particles have no dynamics but are interpolated from one system to the next each frame.

To setup *Keyed* particles you need at least two particle systems in the *Keys* list.

Options

Loops
: Sets the number of times the entire *Keys* list is repeated. Disabled if *Use Timing* is enabled.

Use Timing
: Enabling this option allows you to specify the timing for each key independently, using the *Time* and *Duration* options. By default, the *Use Timing* option is deactivated, and the particles will pass through all keys for a time equal to its lifetime. A shorter lifetime means faster movement. The lifetime will be split equally between the keys, this may lead to varying particle speeds between the targets.

Relations

Reference

Panel:: *Particle System ▸ Physics ▸ Relations*

Key Targets
: The list view of keys (target particle systems).

Object
: The name of a target object for the selected key. If empty it uses the current particle system.

System
: Index of particle system on the target object.

Time
: The time (frame number) at which the particles will be at the position of the selected system. Note also that the *Start* frame of the Keyed system adds an offset to this time.

Duration
: How long (in frames) the particles stay on this system before they start moving to the next one.

Boids

Reference

Panel:: *Particle System ▸ Physics*
Type:: Boids

Boids particle systems are controlled by a limited artificial intelligence, which can be programmed to follow basic rules and behaviors. They are ideal for simulating flocks, swarms, herds and schools of various kind of animals, insects and fishes or predators vs. preys simulations. They can react on the presence of other objects and on the members of their own system. Boids can handle only a certain amount of information, therefore the sequence of the *Boid Brain* rules is very important. In certain situations only the first three parameter are evaluated.

Movement

Reference

Panel:: *Particle System ▸ Physics ▸ Movement*

Boids try to avoid objects with activated Collision. They try to reach goal objects, and fly from "predators" according to the *Boid Brain* settings.

Boids can have different physics depending on whether they are in the air, or on land (on collision object).

Allow Flight
: Allow boids to move in the air.

Allow Land
: Allow boids to move on land.

Allow Climbing
: Allow boids to climb goal objects.

Max Air Speed
: Set the Maximum velocity in the air.

Min Air Speed
: Set the Minimum velocity in the air.

Max Air Acceleration
: Lateral acceleration in air, percentage of the max velocity (turn). Defines how fast a boid is able to change direction.

Max Air Angular Velocity
: Tangential acceleration in air, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Air Personal Space
: Radius of boids personal space in air. Percentage of particle size.

Landing Smoothness
: How smoothly the boids land.

Max Land Speed
: Set the Maximum velocity on land.

Jump Speed
: Maximum speed for jumping.

Max Land Acceleration
: Lateral acceleration on land, percent of max velocity (turn). Defines how fast a boid is able to change direction.

Max Land Angular Velocity
: Tangential acceleration on land, percent 180 degrees. Defines how much the boid can suddenly accelerate in order to fulfill a rule.

Land Personal Space
: Radius of boids personal space on land. Percentage of particle size.

Land Stick Force
: How strong a force must be to start effecting a boid on land.

Collision Collection
: Only collide with objects in this collection.

Battle

Reference

Panel:: *Particle System ▸ Physics ▸ Battle*

Health
: Initial boid health when born.

Strength
: Maximum caused damage per second on attack.

Aggression
: Boid will fight this time stronger than enemy.

Accuracy
: Accuracy of attack.

Range
: Maximum distance of which a boid can attack.

Misc

Reference

Panel:: *Particle System ▸ Physics ▸ Misc*

Banking
: Amount of rotation around velocity vector on turns. Banking of 1.0 gives a natural banking effect.

Pitch
: Amount of rotation around side vector.

Height
: Boid height relative to particle size.

Relations

Reference

Panel:: *Particle System* ▸ *Physics* ▸ *Relations*

Target
: This list view allows you to set up other particle systems to react with the boids.

Target Object
: A data ID to select an object with a particle system set on.

System
: Index of the *Object*'s particle system as set in the list view in the particle panel.

Mode
: Enemy
  : Setting the type to *Enemy* will cause the systems to fight with each other.

  Friend
  : Will make the systems work together.

  Neutral
  : Will not cause them to align or fight with each other.

Deflection

Boids will try to avoid deflector objects according to the Collision rule's weight. It works best for convex surfaces (some work needed for concave surfaces).

Force Fields

As other physics types, Boids is also influenced by external force fields.

In addition, special *Boid* force fields can be used with the Boids physics. These effectors could be predators (positive Strength) that boids try to avoid, or targets (negative Strength) that boids try to reach according to the (respectively) *Avoid* and *Goal* rules of the *Boid Brain*.

Boid Brain

Reference

Panel:: *Particle System ▸ Physics ▸ Boid Brain*

The Boid Brain panel controls how the boids particles will react with each other. The boids' behavior is controlled by a list of rules. Only a certain amount of information in the list can be evaluated. If the memory capacity is exceeded, the remaining rules are ignored.

The rules are by default parsed from top-list to bottom-list (thus giving explicit priorities), and the order can be modified using the little arrows buttons on the right side.

Rule Evaluation
: There are three ways to control how rules are evaluated:

  Average
  : All rules are averaged.

  Random
  : A random rule is selected for each boid.

Fuzzy
: Uses fuzzy logic to evaluate rules. Rules are gone through top to bottom. Only the first rule that affect above the *Rule Fuzziness* threshold is evaluated. The value should be considered how hard the boid will try to respect a given rule (a value of 1 means the Boid will always stick to it, a value of 0 means it will never). If the boid meets more than one conflicting condition at the same time, it will try to fulfill all the rules according to the respective weight of each.

Note

A given boid will try as much as it can to comply to each of the rules it is given, but it is more than likely that some rule will take precedence on other in some cases. For example, in order to avoid a predator, a boid could probably "forget" about Collision, Separate and Flock rules, meaning that "while panicked" it could well run into obstacles, e.g. even if instructed not to, most of the time.

In Air
: The current rule affects boids while they are flying.

On Land
: The current rule affects boids while they are not flying.

Goal Rule

Seek goal.

Object
: Specifies the goal object. If not specified, Boid force fields with negative Strength are used as goals.

Predict
: Predict target's movements.

Avoid Rule

Avoid “predators”.

Object
: Specifies the object to avoid. If not specified, Boid force fields with positive Strength are used as predators.

Predict
: Predict target’s movements.

Fear Factor
: Avoid object if danger from it is above this threshold.

Avoid Collision Rule

Avoid objects with activated Deflection.

Boids
: Avoid collision with other boids.

Deflectors
: Avoid collision with deflector objects.

Look Ahead
: Time to look ahead in seconds.

Separate Rule

Boids move away from each other.

Flock Rule

Copy movements of neighboring boids, but avoid each other.

Follow Leader Rule

Follows a leader object instead of a boid.

Distance
: Distance behind leader to follow.

Line
: Follow the leader in a line.

  Queue Size
  : How many boids that are allowed to follow in a line.

Average Speed Rule

Maintain average velocity.

Speed
: Percentage of maximum speed.

Wander
: How fast velocity's direction is randomized.

Level
: How much velocity's Z component is kept constant.

Fight Rule

Move toward nearby boids.

Fight Distance
: Attack boids at a maximum of this distance.

Flee Distance
: Flee to this distance.

Fluid

Reference

Panel::	*Particle System* ▸ *Physics*
Type::	Fluid

Fluid particles are similar to Newtonian ones but this time particles are influenced by internal forces like pressure, surface tension, viscosity, springs, etc. From liquids to slime, goo to sand and wispy smoke the number of possible use cases is endless.

Blender particle fluids use the SPH techniques to solve the particles fluid equations. Smoothed-particle hydrodynamics (SPH) is a computational method used for simulating fluid flows. It has been used in many fields of research, including astrophysics, ballistics, vulcanology, and oceanography. It is a mesh-free Lagrangian method (where the coordinates move with the fluid), and the resolution of the method can easily be adjusted with respect to variables such as the density.

Options

Fluid physics share options with Newtonian Physics. These are covered on that page.

Fluid Properties

Stiffness
: How incompressible the fluid is.

Viscosity
: Linear viscosity. Use lower viscosity for thicker fluids.

Buoyancy

Artificial buoyancy force in negative gravity direction based on pressure differences inside the fluid.

Advanced

Reference

Panel:: *Particle System ▸ Physics ▸ Advanced*

Repulsion Factor
: How strongly the fluid tries to keep from clustering (factor of stiffness). Checkbox sets repulsion as a factor of stiffness.

Stiff Viscosity
: Creates viscosity for expanding fluid. Checkbox sets this to be a factor of normal viscosity.

Interaction Radius
: Fluid's interaction radius. Checkbox sets this to be a factor of 4 × *particle size*.

Rest Density
: Density of fluid when at rest. Checkbox sets this to be a factor of default density.

Springs

Reference

Panel:: *Particle System ▸ Physics ▸ Springs*

Force
: Spring force.

Rest Length
: Rest length of springs. Factor of particle radius. Checkbox sets this to be a factor of 2 × *particle size*.

Viscoelastic Springs

Use viscoelastic springs instead of Hooke's springs.

Elastic Limit

How much the spring has to be stretched/compressed in order to change its rest length.

Plasticity

How much the spring rest length can change after the elastic limit is crossed.

Initial Rest Length

Use initial length as spring rest length instead of 2 × *particle size*.

Frames

Create springs for this number of frames since particle's birth (0 is always).

Render

Reference

Panel:: *Particle System* ▸ *Render*

The Render Panel controls how particles appear when they are rendered.

Note

Cycles supports only Object and Collection render types.

Common Settings

Scale
: Todo.

Scale Randomness
: Todo.

Material
: Set which of the object's materials is used to shade the particles.

Coordinates System
: Use a different object's coordinates to determine the birth of particles.

Show Emitter
: When disabled, the emitter is no longer rendered. Activate the button *Emitter* to also render the mesh.

Render As

None

When set to *None*, particles are not rendered. This is useful if you are using the particles to duplicate objects.

Halo

Halos are rendered as glowing dots or a little cloud of light. Although they are not really lights because they do not cast light into the scene like a light object. They are called *Halos* because you can see them, but they do not have any substance.

Path

The *Path* visualization needs a Hair particle system or Keyed particles.

B-Spline
: Interpolate hair using B-splines. This may be an option for you if you want to use low *Render* values. You loose a bit of control but gain smoother paths.

Steps
: Set the number of subdivisions of the rendered paths (the value is a power of 2). You should set this value carefully, because if you increase the render value by two you need four times more memory to render. Also the rendering is faster if you use low render values (sometimes drastically). But how low you can go with this value depends on the waviness of the hair (the value is a power of 2). This means 0 steps give 1 subdivision, 1 give 2 subdivisions, $2 \rightarrow 4$, $3 \rightarrow 8$, $4 \rightarrow 16$, … $n \rightarrow 2^n$.

Timing

Reference

Panel::	*Particle System* ▸ *Render* ▸ *Timing*
Type::	Hair

Absolute Path Time
: Path timing is in absolute frames.

End
: End time of the practical path.

Random
: Give the path length a random variation.

Object

Reference

Panel:: *Particle System ▸ Render ▸ Object*

Instance Object
: The specified object is instanced in place of each particle.

Global Coordinates
: Use object's global coordinates for instancing.

Object Rotation
: Use the rotation of the object.

Object Scale
: Use the size of the object.

Collection

Reference

Panel:: *Particle System ▸ Render ▸ Collection*

Instance Collection
: The objects that belong to a collection are instanced sequentially in the place of the particles.

Whole Collection
: Use the whole group at once, instead of one of its elements, the group being displayed in place of each particle.

Pick Random
: The objects in the group are selected in a random order, and only one object is displayed in place of a particle. Please note that this mechanism fully replaces old Blender particles system using parentage and *Instancing Vertices* to replace particles with actual geometry. This method is fully deprecated and does not work anymore.

Global Coordinates
: Use object's global coordinates for instancing.

Object Rotation
: Use the rotation of the objects.

Object Scale
: Use the size of the objects.

Use Count

Reference

Panel:: *Particle System ▸ Render ▸ Collection ▸ Use Count*

Use objects multiple times in the same groups. Specify the order and number of times to repeat each object with the list view that appears. You can duplicate an object in the list with the + button, or remove a duplicate with the – button.

Extra

Reference

Panel:: *Particle System ▸ Render ▸ Extra*

Parents Particles

Render also parent particles if child particles are used. Children have a lot of different deformation options, so the straight parents would stand between their curly children. So by default *Parents* are not rendered if you activate *Children*. See Children.

Unborn

Render particles before they are born.

Dead

Render particles after they have died. This is very useful if particles die in a collision *Die on hit*, so you can cover objects with particles.

Viewport Display

Reference

Panel:: *Particle System ▸ Viewport Display*

The Display Panel controls how particles are displayed in the 3D Viewport. This does not necessarily determine how they will appear when rendered.

Display As

None
: The particles are not shown in the 3D Viewport and are not rendered. The emitter may be rendered though.

Rendered
: Particles are displayed the way they are rendered.

Point
: Particles are displayed as square points. Their size is independent of the distance from the camera.

Circle
: Particles are displayed as circles that face the view. Their size is independent of the distance from the camera.

Cross
: Particles are displayed as 6-point crosses that align to the rotation of the particles. Their size is independent of the distance from the camera.

Axis
: Particles are displayed as 3-point axes. This is useful if you want to see the orientation and rotation of particles in the viewport. Increase the *Display Size* until you can clearly distinguish the axis.

Note

Particles visualized like Point, Circle, Cross and Axis do not have any special options, but can be very useful when you have multiple particle systems at play, if you do not want to confuse particles of one system from another (e.g. in simulations using *Boids* physics).

Color

The Color Menu allows you to display particle's color according to certain particle properties.

None

Particles are black.

Material

Particles are colored according to the material they are given.

Velocity

Color particles according to their speed. The color is a ramp from blue to green to red, Blue being the slowest, and Red being velocities approaching the value of *Max* or above. Increasing *Max* allows for a wider range of particle velocities.

Acceleration

Color particles according to their acceleration.

Amount

Specifies the percentage of all particles to show in the viewport (all particles are still rendered).

Show Emitter

Make instancer visible in viewport.

Children

Reference

Panel:: *Particle System* ▸ *Children*

Children are *Hair* or *Emitter* particles originating from individual particles. They make it possible to work primarily with a relatively low amount of Parent particles, for whom the physics are calculated. The children are then aligned to their parents. The number and visualization of the children can be changed without a recalculation of the physics.

If you activate children, the parents are no longer rendered. This can be enabled in the Render panel Parent Particles. By default, parent particles are not rendered because the shape of the children can be quite different from that of their parents.

Common Options

Child Type
: None
  : No children are generated.

  Simple
  : Children are emitted from the parent position.

  Interpolated
  : Children are emitted between the *Parent* particles on the faces of a mesh. They interpolate between adjacent parents. This is especially useful for fur, because you can achieve an even distribution. Some of the children can become virtual parents, which are influencing other particles nearby.

Display Amount

The number of children in the 3D Viewport.

Render Amount
The number of children to be rendered.

Length
Length of child paths.

Threshold
Amount of particles left untouched by child path length.

Seed
Offset in the random number table for child particles, to get a different randomized result.

Clumping

Reference

Panel:: *Particle System* ▸ *Children* ▸ *Clumping*

Use Clump Curve
Use Curve Widget instead of parameters.

Clump
Clumping amount along child strands. The children may meet at their tip (1.0) or start together at their root (-1.0).

Shape
Form of *Clump*. Either inverse parabolic (0.99) or exponentially (-0.99).

Twist
Todo.

Use Twist Curve
Todo.

Clump Noise

Creates random clumps around the parent hair.

Clump Noise Size
: The size of the clumps.

Roughness

Reference

Panel:: *Particle System ▸ Children ▸ Roughness*

Use Roughness Curve
: Use Curve Widget instead of parameters.

Uniform, Size
: It is based on children location so it varies the paths in a similar way when the children are near.

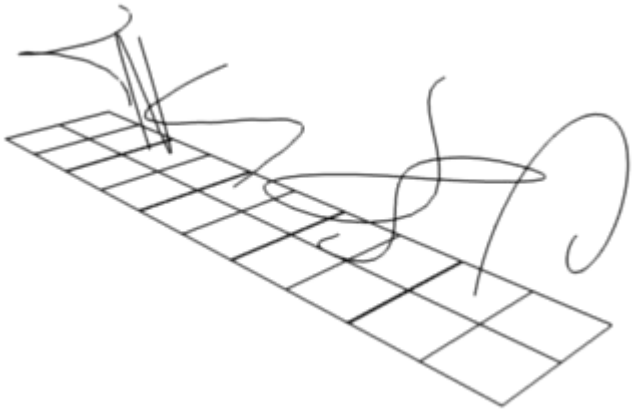
Endpoint, Shape
: “Rough End” randomizes path ends (a bit like random negative clumping). Shape may be varied from <1 (parabolic) to 10.0 (hyperbolic).

Random, Size, Threshold
: It is based on a random vector so it is not the same for nearby children. The threshold can be specified to apply this to only a part of children. This is useful for creating a few stray children that will not do what others do.

Kink

Reference

Panel:: *Particle System ▸ Children ▸ Kink*



Child particles with Kink.

From left to right: Curl, Radial, Wave, Braid, Spiral.

With *Kink* you can rotate the children around the parent. See Fig. Child particles with Kink. above picture for the different types of *Kink*.

Kink Type

Nothing

Deactivated.

Curl

Children grow in a spiral around the parent hairs.

Radial

Children form around the parent a wave shape that passes through the parent hair.

Wave

Children form a wave, all in the same direction.

Braid

Children braid themselves around the parent hair.

Spiral
: Generates a spiral at the end of each hair.

  Radius, Resolution
  : Define the overall size.

  Shape
  : Makes the spiral grow in- or outward.

Note

Alignment Limitations

When hair is pointing straight up (along the chosen spiral axis, default Z), spirals may not show up! This is a limitation of the projection method used. Giving a slight tilt or random orientation to hairs fixes this.

Amplitude
: The amplitude of the offset.

Clump
: How much clump effects kink amplitude.

Flatness
: How flat the hairs are.

Frequency
: The frequency of the offset (1/total length). The higher the frequency the more rotations are done.

Shape
: Where the rotation starts (offset of rotation).

Simple

Size

A multiplier for children size.

Random Size
: Random variation to the size of child particles.

Radius
: The radius in which the children are distributed around their parents. This is 3D, so children may be emitted higher or lower than their parents.

Roundness
: The roundness of the children around their parents. Either in a sphere (1.0) or in-plane (0.0).

Interpolated

Virtual Parents
: Relative amount of virtual parents.

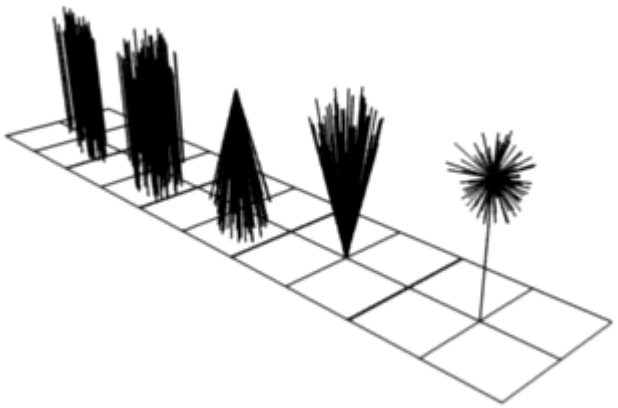
Long Hair
: Calculate children that suit long hair well.

Parting

Parting
: Creates parting in the children based on parent strands.

Min/Max
: The minimum/maximum root to tip angle (tip distance/root distance for long hair).

Example



From left to right: Round: 0.0, Round: 1.0, Clump: 1.0, Clump: -1.0, Shape: -0.99.

Force Fields

Field Weights

Reference

Panel:: *Particle System ▸ Field Weights*

The Field Weight panel allows you to control how much influence each type of external force field, or effector, has on the particle system. Force fields are external forces that give dynamic system's motion. The force fields types are detailed on the Force Field Page.

Effector Group
: Limit effectors to a specified group. Only effectors in this group will have an effect on the current system.

Gravity
: Control how much the Global Gravity has an effect on the system.

All
: Scale all of the effector weights.

Force Fields Settings

Reference

Panel:: *Particle System ▸ Force Fields Settings*

The Force Field Settings panel allows you to make each individual act as a force field, allowing them to affect other dynamic systems, or even, each other.

Self Effect
: Causes the particle force fields to have an effect on other particles within the same system.

Effector Amount
: Set how many of the particles act as force fields. 0 means all of them are effectors.

You can give particle systems up to two force fields. By default they do not have any. Choose an effector type from the selector to enable them. Settings are described in the Common Settings section.

Vertex Groups

Reference

Panel:: *Particle System ▸ Vertex Groups*

The Vertex groups panel allows you to specify vertex groups to use for several child particle settings. You can also negate the effect of each vertex group with the checkboxes. You can affect the following attributes:

Density
: Defines the density of the particle distribution.

Length
: Defines the length of the hair.

Clump
: Controls the amount of clumping. The weight of 1.0 gives current *Clump* value, weight of 0.0 completely removes effect.

Kink
: Controls the frequency of the children Kink.

Roughness 1
: Adjusts the *Uniform* roughness parameter.

Roughness 2
: Adjusts the *Random* roughness parameter.

Roughness End
: Adjusts the *Endpoint* roughness parameter.

Twist
: Vertex group to control the children's *Twist* effect. Gives control over the direction of the twist, as well as the amount. The weight of 0.5 is

neutral, i.e. there is no twist effect.

Hair

This page is about the end of life hair system. Read about the new hair system on the Hair Nodes page.

- Introduction
  - Growing
  - Styling
  - Animating
  - Rendering
- Emission
- Hair Dynamics
  - Collisions
  - Structure
  - Volume
- Render
- Shape
- Children
- Viewport Display

Introduction

Hair type particle system can be used for strand-like objects, such as hair, fur, grass, quills, etc.



Particle hair systems example. Used for the grass and fur.

Growing

The first step is to create the hair, specifying the amount of hair strands and their lengths.

The complete path of the particles is calculated in advance. So everything a particle does a hair may do also. A hair is as long as the particle path would be for a particle with a lifetime of 100 frames. Instead of rendering every frame of the particle animation point by point there are calculated control points with an interpolation, the segments.

Styling

The next step is to style the hair. You can change the look of base hairs by changing the Physics Settings.

A more advanced way of changing the hair appearance is to use Children. This adds child hairs to the original ones, and has settings for giving them different types of shapes.

You can also interactively style hairs in Particle Edit Mode. In this mode, the particle settings become disabled, and you can comb, trim, lengthen, etc. the hair curves.

Animating

Hair can be made dynamic using the cloth solver. This is covered in the Hair Dynamics page.

Rendering

With Cycles you can render hair with specialized hair BSDFs Hair BSDF or Principled Hair BSDF.

Hair can also be used as a basis for the Particle Instance Modifier, which allows you to have a mesh be deformed along the curves, which is useful for thicker strands, or things like grass, or feathers, which may have a more specific look.

Emission

Reference

Panel:: *Particle System* ▸ *Emission*

Number
: Sets the amount of hair strands. Use as few particles as possible (especially if you plan to use soft body animation later), but still enough to cover the surface and have good control. A few thousand particles is generally enough for a regular haircut. The hair will be made denser later on using Children.

Hair Length
: Controls the length of the hair.

See also

Emitter particles Emission panel

Hair Dynamics

Reference

Panel:: *Particle System ▸ Hair Dynamics*

Hair particles can have dynamic properties using physics. To enable hair physics, click the checkbox beside *Hair Dynamics*.

Quality Steps
: Quality of the simulation in steps per frame (higher is better quality but slower).

Pin Goal Strength
: Spring stiffness of the vertex target position.

Warning

If you use motion blur in your animation, you will need to bake one extra frame past the last frame which you will be rendering.

Collisions

Quality
: A general setting for how fine and good a simulation you wish. Higher numbers take more time but ensure less tears and penetrations through the hair.

Distance
: The distance another object must get to the hair for the simulation to repel the hair out of the way. Smaller values might cause errors but provide some speed-up while larger will give unrealistic results if too large and can be slow. It is best to find a good in between value.

Impulse Clamping
: Prevents explosions in tight and complicated collision situations by restricting the amount of movement after a collision.

Collision Collection
: Only objects that are a part of this Collection can collide with the hair. Note that these objects must also have Collision physics enabled.

Structure

Reference

Panel:: *Particle System ▸ Hair Dynamics ▸ Structure*

Vertex Mass
: Value for the mass of the hair.

Stiffness
: Controls the bending stiffness of the hair strands.

  Random
  : Random stiffness of hair.

Damping
: Damping of bending motion.

Volume

Reference

Panel:: *Particle System ▸ Hair Dynamics ▸ Volume*

Some phenomena of real-world hair can be simulated more efficiently using a volumetric model instead of the basic geometric strand model. This means

constructing a regular grid such as those used in fluid simulations and interpolating hair properties between the grid cells.

Air Drag
: Controls how thick the air is around the hair causing the hair to flow slower.

Internal Friction
: Amount of friction between individual hairs.

Voxel Grid Cell Size
: Size of the voxel grid cells for interaction effects.

Density Target
: Maximum density of the hair.

Density Strength
: The influence that the *Density Target* has on the simulation.

Render

Reference

Panel:: *Particle System ▸ Render*

Hair can be rendered as a Path, Object, or Group. See Particle Visualization for descriptions and the Hair Shape settings.

See also

Blender Hair Basics, a thorough overview of all of the hair particle settings.

Shape

Reference

Panel:: *Particle System ▸ Hair Shape*

These settings control the shape of hair curves for rendering.

Strand Shape
: A shape parameter that controls the transition in thickness between the root and tip. Negative values make the primitive rounded more towards the top, the value of zero makes the primitive linear, and positive values make the primitive rounded more towards the bottom.

Diameter Root
: Multiplier of the hair width at the root.

Tip
: Multiplier of the hair width at the tip.

Radius Scale
: Multiplier for the *Root* and *Tip* values. This can be used to change the thickness of the hair.

Close Tip
: Sets the thickness at the tip to zero, even when using a nonzero tip multiplier.

Children

Reference

Panel:: *Particle System ▸ Children*

See Children.

Viewport Display

Reference

Panel:: *Particle System ▸ Display*

Rendered
: Display hair as curves.

Path
: Display just the end points of the hairs.

Steps
: The number of segments (control points minus 1) of the hair strand. In between the control points the segments are interpolated. The number of control points is important:

  - For the soft body animation, because the control points are animated like vertices, so more control points mean longer calculation times.
  - For the interactive editing, because you can only move the control points (but you may recalculate the number of control points in *Particle Edit Mode*).

  Hint

  Segments

  Ten Segments should be sufficient even for very long hair, five Segments are enough for shorter hair, and two or three segments should be enough for short fur.

Texture Influence

Reference

Panel:: *Texture ▸ Influence*

Defines the settings of a Particle system spatial with a texture.

General

Time
: Affect the emission time of the particles.

Lifetime
: Affect the life time of the particles.

Density
: Affect the density of the particles.

Size
: Affect the particles size.

Physics

Velocity
: Affect the particles initial velocity.

Damp
: Affect the particles velocity damping.

Gravity
: Affect the particles gravity.

Force Fields

Affect the particles force fields.

Hair

Length
: Affect the child hair length.

Clump
: Affect the child clumping.

Kink
: Affect the child kink.

Rough
: Affect the child roughness.

Particle Edit Mode

Using *Particle Edit Mode* you can edit the keyed points (keyframes) and paths of Hair, Particle, Cloth, and Soft Body simulations. (You can also edit and style hair before baking.)

Since working in Particle Edit Mode is pretty easy and very similar to working with vertices in the 3D Viewport, we will show how to set up a particle system and then give a reference of the various functions.

Important

Particle Edit Mode, specifically for hair is deprecated; please use the new Empty Hair object with its associated Sculpt Mode instead.

Important

Editing a cached cloth simulation is not currently working, see: blender/blender#77114 for details.

Usage

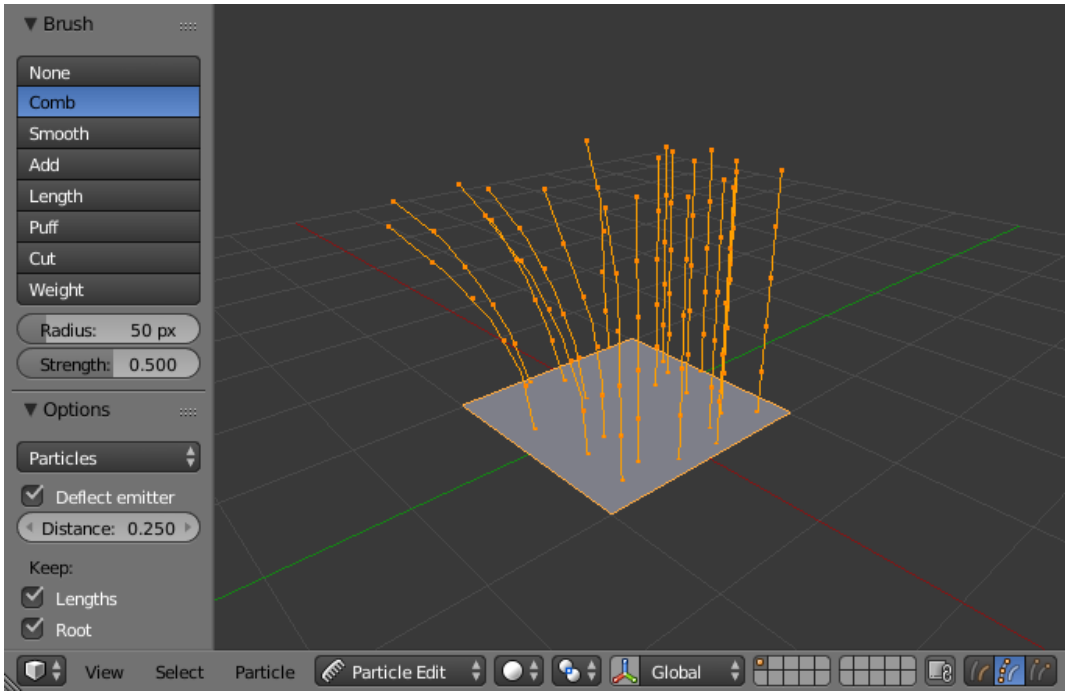
Tip

Only Frames Baked to Memory are Editable!

If you cannot edit the particles, check that you are not baking to a Disk Cache.

Setup for Hair Particles

1. Create a *Hair* particle system.
2. Give it an initial velocity in the *Normal* direction.
3. Create a simulation.
4. Check the *Hair Dynamics* box.



Editing hair strands in Particle Edit Mode.

Setup for Particle, Cloth, and Soft Body Simulations

1. Use *Emitter* particles, or a cloth/soft body simulation.
2. Create a simulation by setting up objects and or emitters, set your time range (use a small range if you are just starting out and experimenting), set up the simulation how you want it, using `Alt-A` to preview it.

Bake the Simulation

Once you are happy with the general simulation, bake the simulation from Object Mode. The simulation must be baked to enable editing.

Edit the Simulation

Switch to *Particle Edit* from the *Mode* select menu in the header of the 3D Viewport to edit the particle's paths/Keyframes. You may need to press `T` from within the 3D Viewport to see the *Particle Edit* toolbox. Move to the frame you want to edit and use the various tools to edit your simulation.

Selecting

Tip

Switch to the *Point select mode* (see below) in the header of the 3D Viewport to be able to see and select the keypoints.

- Select single: `LMB`.
- Add to/remove from selection: `Shift-LMB`.
- All: `A`.
- None: `Alt-A`.
- Invert: `Ctrl-I`.
- Box select: `B`.
- Circle Select: `C`.
- Lasso Select: `Ctrl-Alt-LMB`.
- Select Linked: Move the mouse over a path and press `L` to add all its points to the selection.
- Unselect Linked: Move the mouse over a path and press `Shift-L` to remove all its points from the selection.
- Root/Tips: *Select ▸ Roots / Tips*.

Select Random

Randomly selects particles.

Percent
: Percent of particles to randomly select.

Random Seed
: Seed value to use for the selection.

Action
: Select random can be either used to select or deselect particles.

Type
: Selects either hair or points. Here these terms can be confusing because hair/point does not refer to the particle type but the path/points of the hair/particle.

Select Modes



Select Modes.

Path:: No keypoints are visible, you can select/deselect only all particles.
Point:: You see all of the keypoints.
Tip:: You can see and edit (including the brushes) only the tip of the particles, i.e. the last keypoint.

Tools

Reference

Mode:: Particle Edit Mode
Tool:: *Toolbar*

Comb

Moves the keypoints (similar to the Proportional Editing tool).

Deflect Emitter
: Hair particles only – Do not move keypoints through the emitting mesh.

  Distance
  : The distance to keep from the Emitter.

Smooth

Parallels visually adjacent segments.

Add

Adds new particles.

Count
: The number of new particles per step.

Interpolate
: Interpolate the shape of new hairs from existing ones.

Steps
: Amount of brush steps.

Keys
: How many keys to make new particles with.

Length

Scales the segments, so it makes the hair longer with *Grow* or shorter with *Shrink*.

Grow/Shrink
: Sets the brush to add the effect or reverse it.

Puff

Rotates the hair around its first keypoint (root). So it makes the hair stand up with *Add* or lay down with *Sub*.

Puff Volume
: Apply puff to unselected end points, (Helps to maintain the hair volume when puffing the root.)

Cut

Scales the segments until the last keypoint reaches the brush.

Weight

This is especially useful for soft body animations, because the weight defines the soft body *Goal*. A keypoint with a weight of 1 will not move at all, a keypoint with a weight of 0 subjects fully to soft body animation. This value is scaled by the Strength *Min* to *Max* range of soft body goals…

Common Options

Below the brush types, their settings appear:

Radius `F`
: Set the radius of the brush.

Strength `Shift-F`
: Set the strength of the brush effect (not for Add brush).

Options

Reference

Mode::	Particle Edit Mode
Panel::	*Tool Settings* ‣ *Options*

Auto-Velocity *Emitter*
: Recalculate velocities of particles according to their edited paths. Otherwise, the original velocities values remains unchanged regardless of the actual distance that the particles moves.

Mirror X
: Enable mirror editing across the local X axis.

Preserve

Strand Length

Keep the length of the segments between the keypoints when combing or smoothing the hair. This is done by moving all the other keypoints.

Root Positions

Keep first key unmodified, so you cannot transplant hair.

Cut Particles to Shape

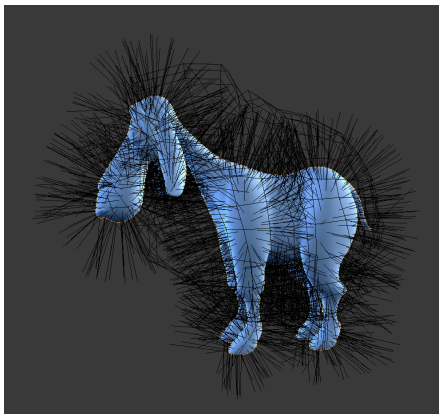
Shape Object

A mesh object which boundary is used by the *Shape Cut* tool.

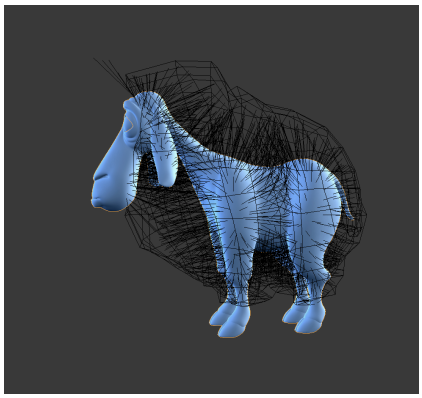
Cut

This grooming tool trims hairs to a shape defined by the *Shape Object*. This is a quicker way of avoiding protruding hair sections from lengthening than using the Cutting tool. It works especially well for characters with extensive fur, where working in a single plane with the Cutting tool becomes tedious.

Shape Cut example.



Before.



After.

Viewport Display

Path Steps
: The number of steps used to draw the path; improves the smoothness of the particle path.

Children *Hair*
: Displays the children of the particles too. This allows to fine-tune the particles and see their effects on the result, but it may slow down your system if you have many children.

Particles *Emitter*
: Displays the actual particles on top of the paths.

Fade Time
: Fade out paths and keys further away from current time.

  Frames
  : How many frames to fade.

Editing

Moving Keypoints or Particles

- To move selected keypoints press `G`, or use one of the various other methods to move vertices.
- To move a particle root you have to turn off Keep *Root* in the Toolbar.
- You can do many of the things like with vertices, including scaling, rotating and removing (complete particles or single keys).
- You may not duplicate or extrude keys or particles, but you can subdivide particles which adds new keypoints *Particle* ▸ *Subdivide*.
- Alternatively you can re-key a particle *Particle* ▸ *Rekey*.

How smoothly the hair and particle paths are displayed depends on the *Path Steps* setting in the Toolbar. Low settings produce blocky interpolation between points, while high settings produce a smooth curve.

Mirror

Reference

Mode::	Particle Edit Mode
Menu::	*Particle ‣ Mirror*

If you want to create an X axis symmetrical haircut you have to do following steps:

1. Select all particles with A.
2. Mirror the particles with *Particle ‣ Mirror*.
3. Turn on *X Mirror* in *Sidebar Region ‣ Tool ‣ Options*.

It may happen that after mirroring two particles occupy nearly the same place. Since this would be a waste of memory and render time, you can use *Merge by Distance* from the *Particle* menu.

Unify Length

Reference

Mode::	Particle Edit Mode
Menu::	*Particle ‣ Unify Length*

This tool is used to make all selected hair uniform length by finding the average length.

Show/Hide

Reference

Mode::	Particle Edit Mode
Menu::	*Particle ‣ Show/Hide*

Hiding and unhiding of particles works similar as with vertices in the 3D Viewport. Select one or more keypoints of the particle you want to hide and press H. The particle in fact does not vanish, only the key points.

Hidden particles (i.e. particles whose keypoints are hidden) do not react on the various brushes. But:

If you use *Mirror Editing* even particles with hidden keypoints may be moved, if their mirrored counterpart is moved.

To unhide all hidden particles press `Alt-H`.

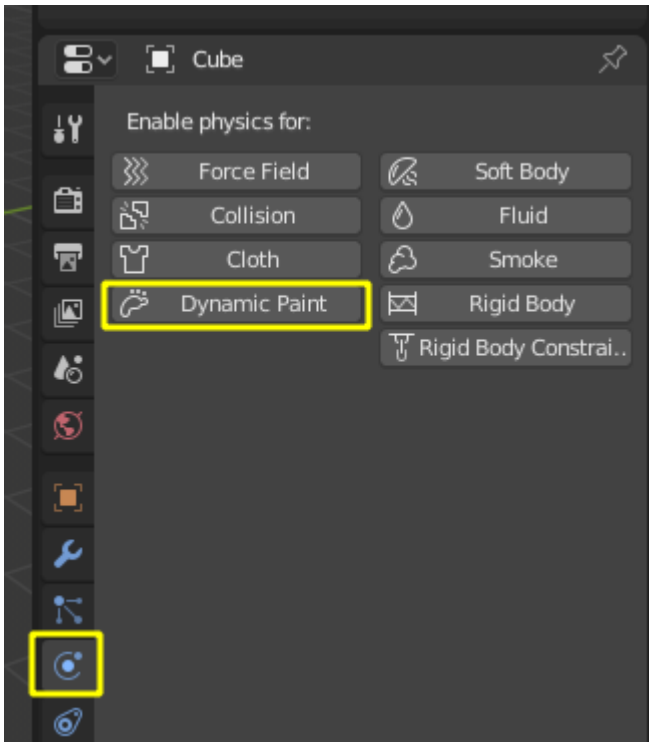
Dynamic Paint

- Introduction
  - Activating the Modifier
  - Types
- Brush
  - Source
  - Velocity
  - Waves
- Canvas
  - Settings
  - Surface
  - Cache
  - Effects
  - Initial Color
  - Output

Introduction

Dynamic paint is a modifier and physics system that can turn objects into paint canvases and brushes, creating Color Attributes, image sequences, or displacement. This makes many effects possible like, for example footsteps in the snow, raindrops that make the ground wet, paint that sticks to walls, or objects that gradually freeze.

Activating the Modifier



How to activate the Dynamic Paint.

Dynamic Paint can be activated from the “Physics” tab of the “Properties” editor.

Types

Modifier itself has two different types Canvas and Brush.

Note

You can also enable brush and canvas simultaneously. In that case same object's "brush" does not influence its "canvas", but can still interact with other objects in the scene.

See also

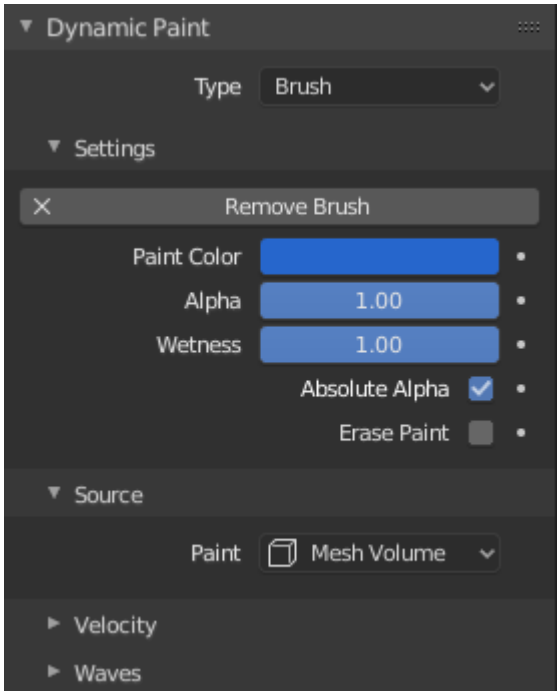
- A step-by step introduction.
- A detailed guide that covers every setting with images and examples (currently not up-to-date).

Brush

Reference

Panel:: *Physics ▸ Dynamic Paint*
Type:: Brush

The Brush type makes object apply paint on the canvas.



Brush main panel.

From the first brush panel you can define how brush affects canvas color surfaces.

Paint Color
: Color of the paint.

Alpha
: Defines brush alpha or visibility. Final wetness is also affected by alpha.

Wetness
: Defines how “wet” new paint is. Wetness is visible on “Paint” surface “wetmap”. Speed of “Drip” and “Spread” effects also depends on how wet the paint is.

Absolute Alpha
: This setting limits brush alpha influence. Without it, brush is “added” on surface over and over again each frame, increasing alpha and therefore influence of brush on canvas. In many cases however, it is preferred to not increase brush alpha if it already is on brushes level.

Erase Paint
: Makes brush dissolve existing paint instead of adding it.

Source

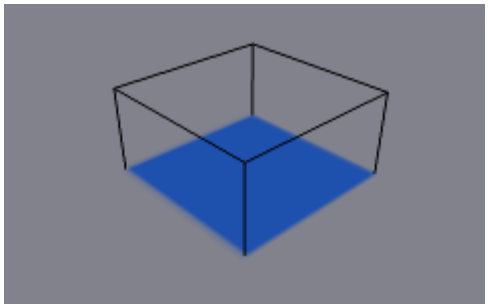
Reference

Type::	Brush
Panel::	*Physics* ▸ *Dynamic Paint* ▸ *Source*

Paint source setting lets you define how brush influence/intersection is defined.

Mesh Volume

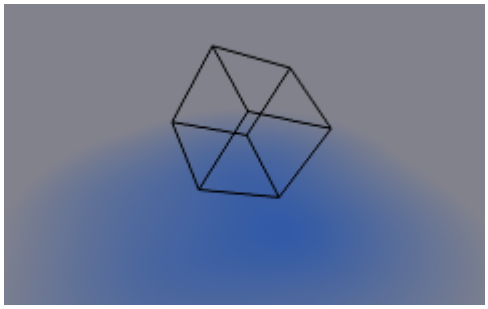
The Brush affects all surface point inside the mesh volume.



Source: Mesh Volume.

Proximity

Only uses defined distance to the closest point on brush mesh surface. Note that inside of the volume is not necessarily affected because it is not close to the surface.



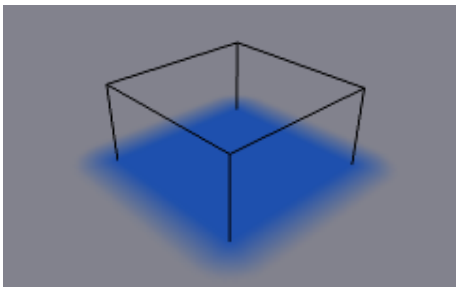
Source: Proximity. Brush affects all canvas pixels around it.

Mesh Volume + Proximity

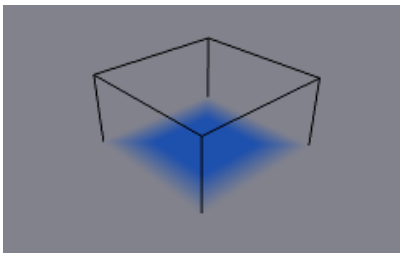
Same as volume type, but also has influence over defined distance.

Inner Proximity
: Applies proximity inside the mesh volume.

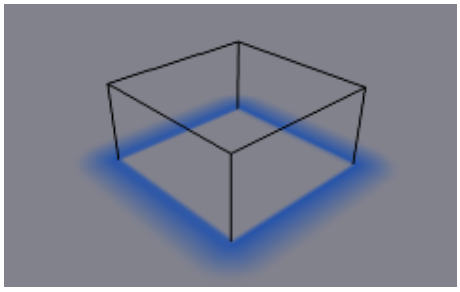
Negate Volume
: Negates brush alpha within mesh volume.



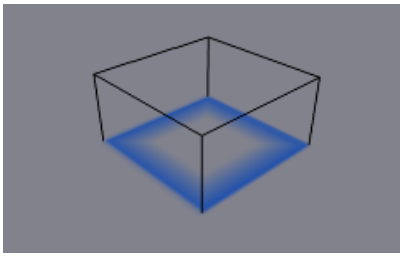
The Volume + Proximity brush with no additional settings.



Inner Proximity. Proximity falloff is now visible inside the volume.



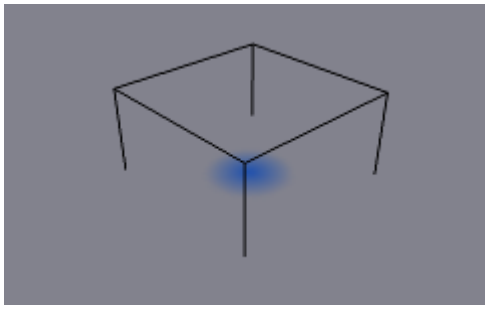
Negate Volume. Inner side of the volume has become completely transparent.



Inner Proximity and Negate Volume enabled together.

Object Center

Instead of calculating proximity to the brush object mesh, which can be quite slow in some cases, only distance to only center is calculated. This is much faster and often good enough.



Source: Object Center.

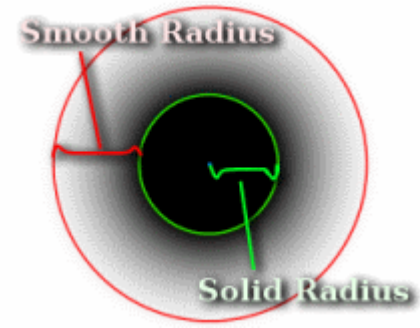
Particle System

Brush influence is defined by particles from a selected particle system.

Effect Solid Radius
: Defines the distance, inside which paint is solid color.

Use Particle Radius

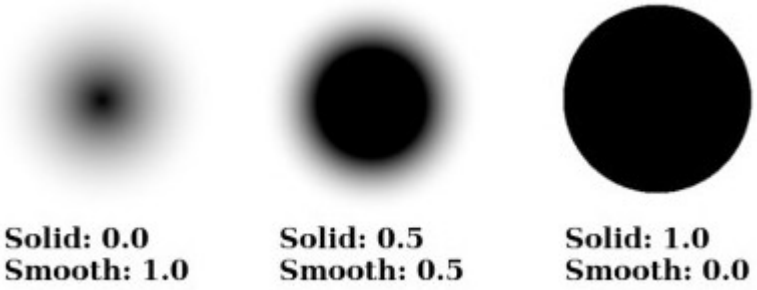
Uses the settings in the particle panel to determine solid radius size. Solid Radius size disabled while Particle Radius enabled.

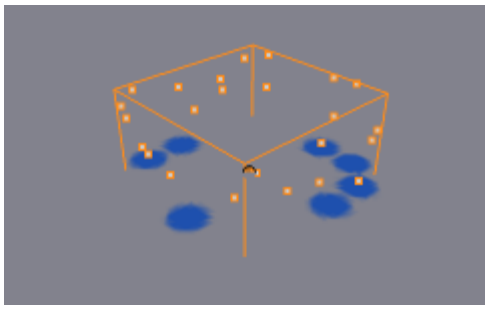


Smooth Radius

An additional radius outside Solid Radius to add a smooth falloff.

If you set “Smooth Radius” to zero, particle will be painted as a solid sphere. If you set “Solid Radius” to zero, it gets painted as a smooth halo.





Source: Particle System.

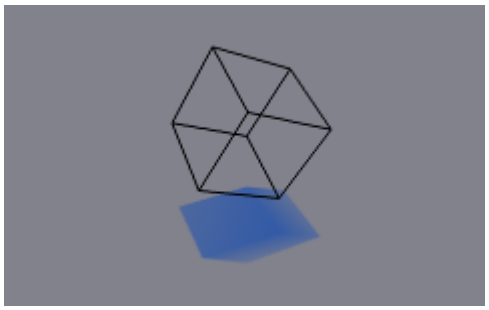
Common Options

Paint Distance

The maximum distance to mesh surface to affect paint.

Project

Projects brush to the canvas from a defined direction. Basically this can be considered as “direction aligned” proximity.



The Project option enabled. See how brush only affects canvas in normal direction.

Falloff

Sharp:: Paints solid paint within the defined distance.

Smooth:: Makes paint to linearly fade out until becoming completely invisible when it reaches the maximum distance.

Color Ramp:: Allows you to manually make a custom falloff behavior.

Velocity

Reference

Type:: Brush

Panel:: *Physics* ▸ *Dynamic Paint* ▸ *Velocity*

This panel shows brush options that are based on object velocity.

On top you have a color ramp and several related settings. Basically the color ramp represents brush velocity values: left side being zero velocity and right side being the “Max velocity”. Speed is measured in “units per frame”.

Checkboxes above can be used to define color ramp influence.

Multiply Alpha
: Uses color ramp's alpha value depending on current velocity and multiplies brush alpha with it.

Replace Color
: Replaces the brush color with the values from the Color Ramp Widget.

Multiply Depth
: Multiplies brushes "depth intersection" effect. Basically you can adjust displace and wave strength depending on brush speed.

Do Smudge
: Enabling Smudge makes the brush "smudge" (or "smear") existing colors on the surface as it moves. The strength of this effect can be defined from the *Smudge Strength* property.

  Even when smudge is enabled brush still does its normal paint effect. If you want a purely smudging brush use zero alpha. It is also possible to have *Erase* option enabled together with smudge.

Waves

Reference

Type:: Brush
Panel:: *Physics* ▸ *Dynamic Paint* ▸ *Waves*

This panel is used to adjust brush influence to "Wave" surfaces.

Wave Type
: Select what effect the brush creates in the wave simulation.

  Depth Change:: The brush create waves when the intersection depth with the surface is *changed* on that point. If the brush is not moved, it will have no effect.

Using a negative “Factor” with this type can create a nice looking “wake” for moving objects like ships.

Obstacle:: Constantly affects surface whenever intersecting. Waves are also reflected off this brush type. However, due the nature of wave simulation algorithm this type creates an unnatural “dent” in the surface if the brush is not moved.

Force:: Directly affects the velocity of wave motion. Therefore the effect is not one-to-one with brush intersection depth, yet the force strength depends on it.

Reflect Only:: This type has no visible effect on the surface alone but reflects waves that are already on the surface.

Factor

Adjusts how strongly brush “depth” affects the simulation. You can also use negative values to make brush pull water up instead of down.

Clamp Waves

In some cases the brush goes very deep inside the surface messing whole simulation up. You can use this setting to “limit” influence to only certain depth.

Canvas

Reference

Panel:: *Physics* ▸ *Dynamic Paint*
Type:: Canvas

The Canvas type makes object receive paint from Dynamic Paint brushes.

Settings

Paint Surface
: A list of Dynamic Paint surfaces. These surfaces are basically layers of paint, that work independently from each other. You can define individual settings for them and bake them separately.

  Is Active
  : The checkbox toggles whether surface is active at all. If not selected no calculations are done.

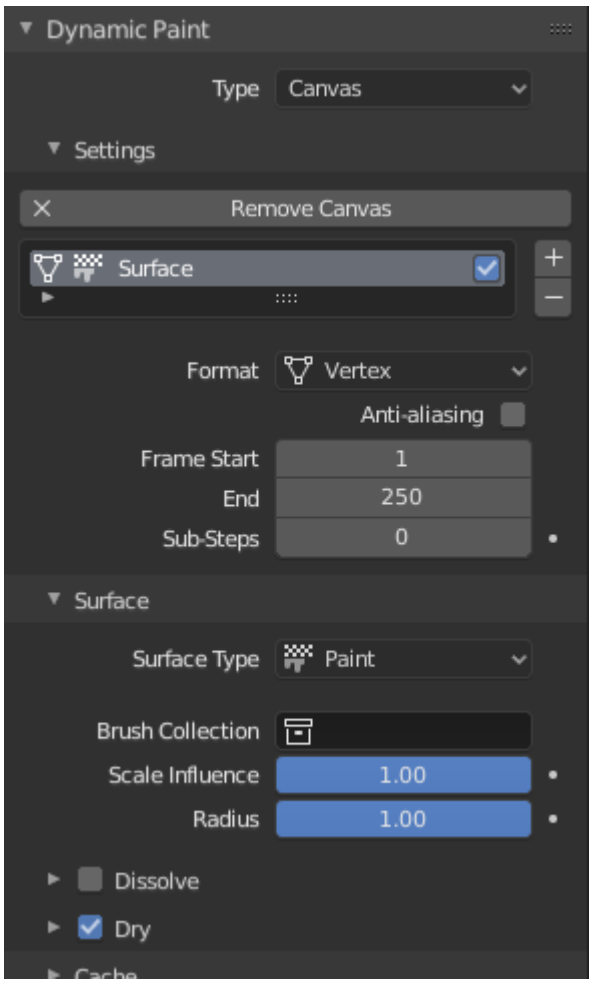
Below you can set surface type and adjust quality and timing settings.

Format
: Each surface has a certain format and type. Format determines how data is stored and outputted.

  Dynamic Paint operates directly on mesh vertex data. Results are stored by point cache and can be displayed in viewports. However, using vertex level also requires a highly subdivided mesh to work.

  Dynamic Paint generates UV wrapped image files of defined resolution as output.
  Resolution

Vertex::
Image Sequences:: You can adjust the output image dimensions for the *Image Sequences* surface type. For example using 256 will lead to 256×256 image output.



Canvas main panel.

Doubling the resolution will likely quadruple the baking time and vice versa.

Anti-Aliasing
: Anti-Aliasing to smooth paint edges using a 5× multisampling method.

Frame Start, End
: Defines surface processing start and end frame.

Sub-Steps
: Extra samples between frames. They are usually required when there is a very fast brush.

Surface

Reference

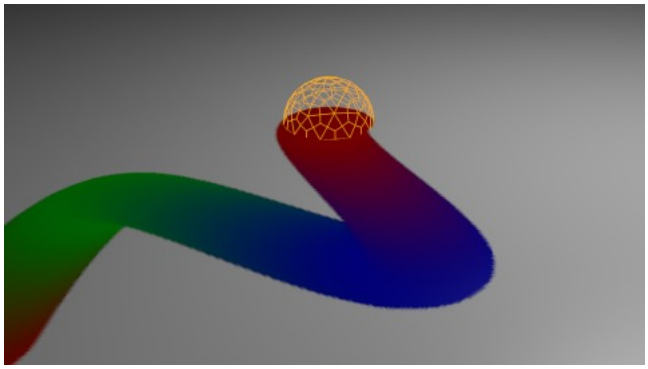
Type:: Canvas
Panel:: *Physics* ▸ *Dynamic Paint* ▸ *Surface*

From *Surface* panel you can adjust surface type and related settings.

Surface Type

Each surface has a “type” that defines what surface is used for.

Paint



Paint Surface.

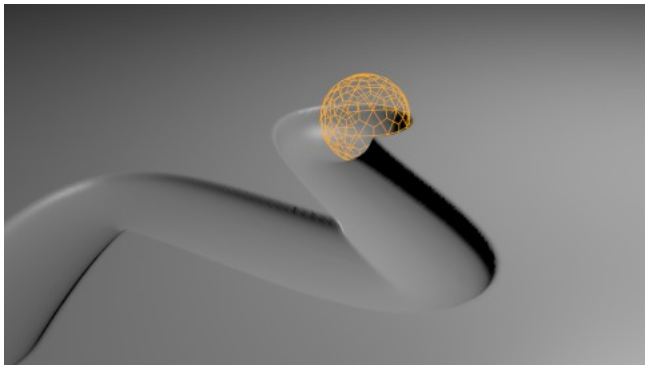
Paint is the basic surface type that outputs color and wetness values. In case of vertex surfaces, results are outputted as Color Attributes.

A wetmap is a black-and-white output that visualizes paint wetness. White being maximum wetness, black being completely dry. It is usually used as mask for rendering. Some “paint effects” affect wet paint only.

Dry
: Completely disable drying is useful for indefinitely spreading paint.

Color Dry
: It can be used to define wetness level when paint colors start to shift to surface “background”. Lower values can be useful to prevent spreading paint from becoming transparent as it dries, while higher values give better results in general.

Displace



Displace Surface.

This type of surface outputs intersection depth from brush objects.

Incremental
: A new displace is added cumulatively on top of an existing displace.

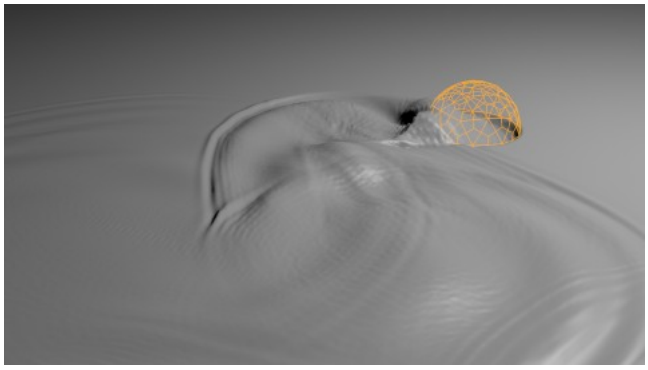
Max Displace
: The maximum level of intersection depth, larger values will be clamped to this value.

Displace Factor
: The multiplier for the intersection depth. You can use it to adjust final displace strength or use negative values to paint bumps.

Tip

If the displace output seems too rough it usually helps to add a Smooth Modifier after Dynamic Paint in the modifier stack.

Waves



Waves Surface.

This surface type produces simulated wave motion. Like displace, wave surface also uses brush intersection depth to define brush strength.

Open Borders

Allows waves to pass through mesh “edges” instead of reflecting from them.

Timescale

Directly adjusts simulation speed without affecting simulation outcome. Lower values make simulation go slower and otherwise.

Speed

Affects how fast waves travel on the surface. This setting is also corresponds to the size of the simulation. Half the speed equals surface double as large.

Damping

Reduces the wave strength over time. Basically adjusts how fast wave disappears.

Spring

Adjusts the force that pulls water back to “zero level”.

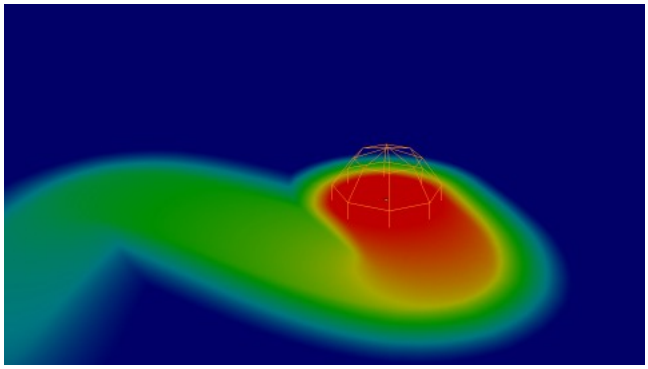
Smoothness

Limits the maximum steepness of the wave slope between simulation points. This greatly helps getting rid of the “noise” that occurs when using objects with sharp edges (like cubes) as a brush. The default value should be enough to only get rid of the sharpest spikes, in order to get even smoother waves use higher values at the expense of reduced detail.

Tip

In some cases the wave motion gets very unstable around brush. It usually helps to reduce wave speed, brush “wave factor” or even the resolution of mesh/surface.

Weight



Weight Surface.

This is a special surface type only available for vertex format. It outputs vertex weight groups that can be used by other Blender modifiers and tools.

Tip

It is usually preferred to use “proximity” based brushes for weight surfaces to allow smooth falloff between weight values.

Common Options

For each surface type there are special settings to adjust. Most types have the settings *Dissolve* and *Brush*:

Dissolve
: Used to make the surface smoothly return to its original state during a defined *Time* period.

Brush Collection
: Used to define a specific collection to pick brush objects from.

Influence Scale, Radius Scale
: For tweaking brush settings individually for each surface.

Cache

Reference

Type:: Canvas
Panel:: *Physics ▸ Dynamic Paint ▸ Cache*

This panel is currently only visible for *Vertex* format surfaces. You can use it to adjust and bake point cache.

Effects

Reference

Type:: Canvas
Panel:: *Physics ▸ Dynamic Paint ▸ Effects*

This is a special feature for “Paint” type surface. It generates animated movement on canvas surface.

Effects
: Spread

Paint slowly spreads to surrounding points eventually filling all connected areas.

Drip
: Paint moves in specific direction specified by Blender force fields, gravity and velocity with user-defined influences.

Shrink
: Painted area slowly shrinks until disappears completely.

For spread and drip effects, only “wet paint” is affected, so as the paint dries, movement becomes slower until it stops.

Initial Color

Reference

Type::	Canvas
Panel::	*Physics* ▸ *Dynamic Paint* ▸ *Initial Color*

Allows you to define the initial color of the canvas. (Todo 2.62)

- None
- Color
- UV Texture
- Vertex Color

Output

Reference

Type::	Canvas
Panel::	*Physics* ▸ *Dynamic Paint* ▸ *Output*

From Output panel you can adjust how surface outputs its results.

Vertex

For *Vertex* format surfaces, you can select a mesh data layer (color/weight depending on surface type) to generate results to. You can use the “+”/”-” icons to add/remove a data layers of given name. If layer with given name is not found, it is shown as red.

Image Sequence

For *Image Sequence* surfaces, you can define used UV maps and output file saving directory, filenames and image format.

Forces

- Gravity
- Force Fields
  - Introduction
  - Types

Gravity

Reference

Panel:: *Scene ▸ Gravity*

Gravity is a global setting that is applied to all physics systems in a scene. It can be found in the scene tab. This value is generally fine left at its default, -9.810 on the Z axis, which is the force of gravity in the real world. Changing this value would simulate a lower or higher force of gravity. Gravity denoted g, measurement $m \times s^{-2}$.

Gravity is applied in the same way to all physics systems.

Gravity is practically the same around the entirety of planet *Earth*. For rendering scenes on The Moon, use -1.622 $m \times s^{-2}$ on the Z axis. Another popular gravity value might be for Mars which has a gravitation acceleration of -3.69 $m \times s^{-2}$ on the Z Axis.

Note

The gravity value per physics system can be scaled down in the *Field Weights* tab.

Force Fields

- Introduction
  - Creating a Force Field
  - Common Field Settings

Types

- Boid
- Charge
- Curve Guide
- Drag
- Fluid Flow
- Force
- Harmonic
- Lennard-Jones
- Magnetic
- Texture
- Turbulence
- Vortex
- Wind

Introduction

Force fields offer a way to influence a simulation, in example to add extra movement. Particles, Soft Bodies, Rigid Bodies, and Cloth objects can all be affected by forces fields. Force fields automatically affect everything. To remove a simulation or particle system from their influence, simply turn down the influence of that type of force field in its Field Weights panel.

- All types of objects and particles can generate fields, but only curve object can bear a Curve Guide field.
- Force fields can also be generated from particles. See Particle Physics.
- The objects need to share at least one common layer to have an effect.

You may limit the effect on particles to a group of objects (see the Particle Physics page).

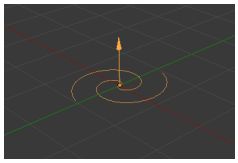
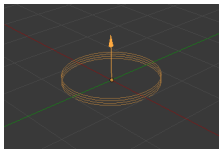
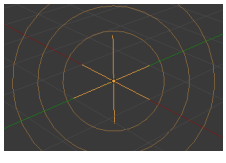
Creating a Force Field

Reference

Mode::	Object Mode
Menu::	*Add* ▸ *Force Field*
Panel::	*Physics* ▸ *Force Field*

To create a single force field, you can select *Add* ▸ *Force Field* and select the desired force field. This method creates an empty with the force field attached.

Examples of an empty with the force field attached.

Vortex force field.	Wind force field.	Force force field.

To create a field from an existing object you have to select the object and change to the *Physics* tab. Select the field type in the *Fields* menu.

Note

After changing the fields *Fields* panel or deflection *Collision* panel settings, you have to recalculate the particle, soft body or cloth system by *Free Cache*, this is not done automatically.

Particles react to all kinds of force fields, soft bodies only to *Force*, *Wind*, *Vortex* (they react on *Harmonic* fields but not in a useful way).

Common Field Settings

Most fields have the same settings, even though they act very differently. Settings unique to a field type are described below. Curve Guide and Texture fields have very different options.

Shape
: Sets the direction which is used to calculate the effector force. For force fields from an empty object only *Point*, *Line* and *Plane* shapes are available, as for a field from a 3D object there are additional *Surface* and *Every Point* options, and *Curve* for a field from a curve.

Point::	Point with omni-directional influence. Uses the object origin as the effector point.
Line::	The force only acts in the local XY plane, using the Z axis line as the effector.
Plane::	The force only acts in the local Z direction, using the XY axis plane as the effector.
Surface::	The force field acts on a 3D object's surface. In this case, the Z axis is the surface normal.

Every Point:: Uses every vertex in the mesh object as an effector point.

Strength
: The strength of the field effect. This can be positive or negative to change the direction that the force operates in. A force field's strength is scaled with the force object's scale, allowing you to scale up and down the scene, keeping the same effects.

Flow
: If nonzero, this adds a drag force proportional and opposite to the point velocity.

  This effectively re-interprets the force field so that the *Strength* to *Flow* ratio at a certain point defines the velocity of an "air flow" field, and objects are encouraged to follow the flow by the resistance caused by the *Flow* drag force.

Affect
: Location
  : Influence the location of particles and other physics entities.

  Rotation
  : Influence the rotation of particles with Dynamic Rotation. The option is not relevant for other types of physics systems.

  Disabling both options completely deactivates the force field.

Noise Amount
: Adds noise to the strength of the force.

Seed
: Changes the seed of the random noise.

Absorption
: Force gets absorbed by collision objects.

Wind Factor

Specifies how much the force is reduced when acting parallel to a surface, e.g. cloth. If set to 1, only the normal component of the force is taken into account.

Falloff

Here you can specify the shape of the force field (if the falloff *Power* is greater than 0).

Shape

Cone:: The falloff results in a cone-shaped force field. Additional options are the same as those of *Tube* options.

Sphere:: The falloff is uniform in all directions, as in a sphere.

Tube:: The falloff results in a tube-shaped force field. The field's *Radial Power* can be adjusted, as well as the *Minimum* and *Maximum* distances of the field.

Z Direction

The direction the force affects on the Z axis.

+Z:: The force only has an affect on the positive Z axis.

-Z:: The force only has an affect on the negative Z axis.

Both Z:: The force has an affect on the positive and negative Z axis.

Power

How the power of the force field changes with the distance from the force field. If *r* is the distance from the origin of the object, the force changes with $1/(r - min + 1)^{power}$. A falloff of 2 changes the force field with $1/(r - min + 1)^2$, which is similar to the falloff of gravitational pull.

Min Distance

The distance from the object's origin, up to where the force field is effective with full strength. If you have a falloff of 0, this parameter will have no effect, because the field is effective with full strength up to *Max Distance* (or infinite). Shown by an additional circle around the object.

Max Distance

Specifies the maximum radius in which the force field affects other objects (shown by an additional circle around the object).

Boid

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Boid

The *Boid* force fields do not affect physics, and are used together with the Boids Particles to define boid predators and goals for the *Boid Brain* rules.

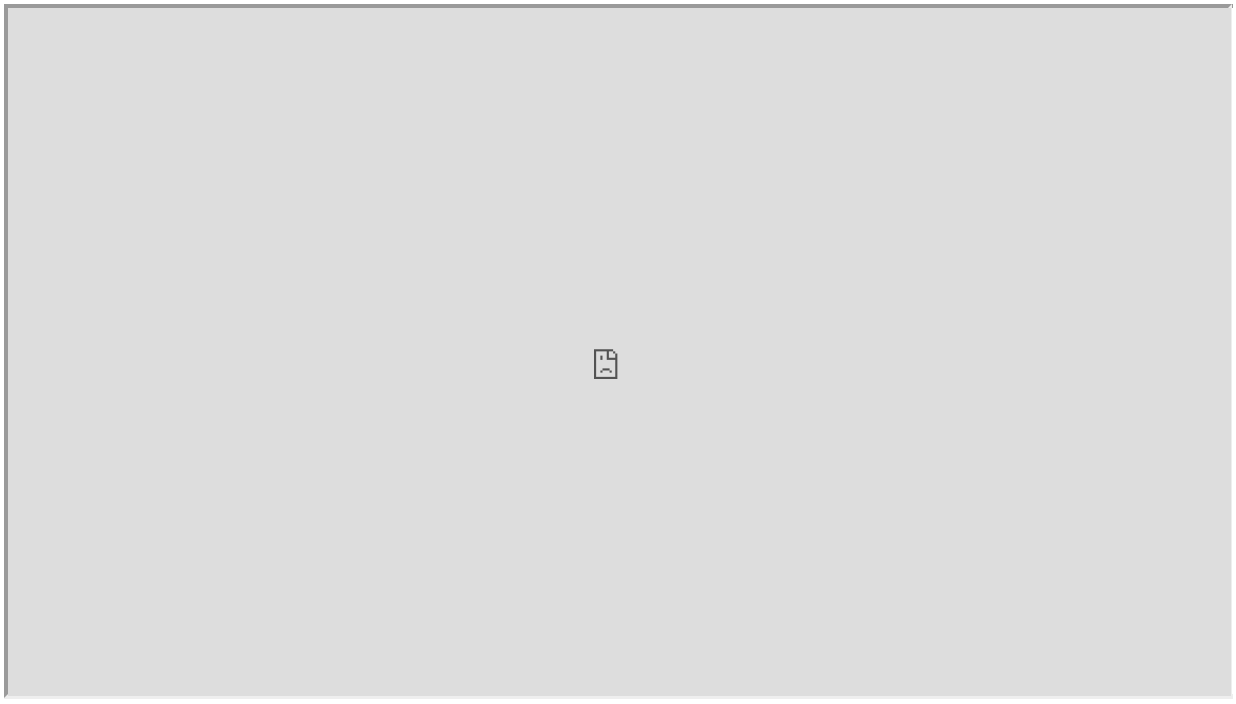
Charge

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Charge

A *Charge* force field is similar to a *Force* field except it changes it's behavior (attract/repulse) based on the effected particles charge field (negative/positive), like real particles with a charge. Which means that this field has only an effect on particles that have also a *Charge* field (else, they have no "charge", and hence are unaffected)!

Example



Curve Guide

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Curve Guide

The *Curve Guide* is used to force particles to follow a certain path defined by a Curve Object. A typical scenario would be to move a red blood cell inside a vein, or to animate the particle flow in a motor. You can also use *Curve Guide* to shape certain hair strands.

Note

You can also use the Particle Edit Mode to define a path.

Since you can animate curves as a soft body or any other usual way, you may build very complex animations while keeping great control and keeping the simulation time to a minimum.

To make particles point in the direction of the curve, you need to set their *Orientation Axis* to *Velocity / Hair*, enable *Dynamic*, and set their *Angular Velocity Axis* to *Velocity*, all in the the Rotation settings of the particle system. The Follow Path Constraint and the curve's legacy Follow option won't work for this.

A *Curve Guide* force affects all particles on the same layer, independently from their distance to the curve. If you have several guides in a layer, their fields add up to each other (the way you may have learned it in your physics course). But you can limit their influence radius by changing the *Minimum Distance* (see below).

A particle follows a *Curve Guide* during its lifetime, the velocity depends on its lifetime and the length of the path.

Note

The Curve Guide does not affect soft bodies.

Options

Free
: Fraction of particle life time, that is not used for the curve.

Falloff Power
: This setting governs the strength of the guide between *Min Distance* and *Max Distance*. A falloff of 1 means a linear progression.

Additive
: If you use *Additive*, the speed of the particles is also evaluated depending on the falloff.

Weights
: Use Curve weights to influence the particle influence along the curve.

Clumping Amount
: The particles come together at the end of the curve (1) or they drift apart (-1).

Shape
: Defines the form in which the particles come together. +0.99: the particles meet at the end of the curve. 0: linear progression along the curve. -0.99: the particles meet at the beginning of the curve.

Min Distance
: The distance from the curve, up to where the force field is effective with full strength. If you have a falloff of 0, this parameter will have no effect, because the field is effective with full strength up to *Max Distance* (or the infinity). *Min Distance* is shown with a circle at the endpoints of the curve in the 3D Viewport.

Max Distance
: The maximum influence radius. Shown by an additional circle around the curve object.

Kink

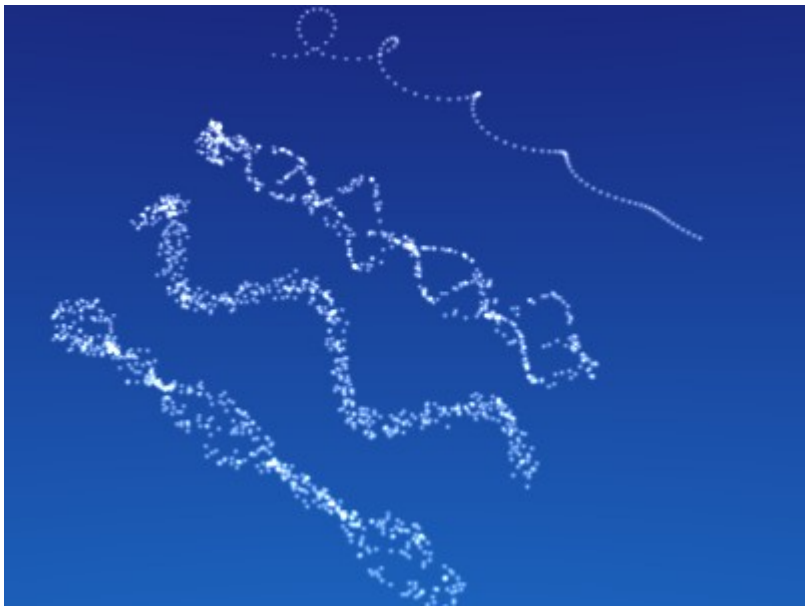
Warning

This feature is broken in the current version, see Bug Report #46776.

Type
: Changes the shape that the particles can take.

None::	Todo.
Braid::	Braid.
Curl::	The radius of the influence depends on the distance of the curve to the emitter.
Radial::	A three-dimensional, standing wave.
Roll::	A one-dimensional, standing wave.
Rotation::	Todo.
Wave::	A two-dimensional, standing wave.

It is not so easy to describe the resulting shapes, so have a look at the example below.



Kink options of a curve guide. From left to right: Radial, Wave, Braid, Roll. Animation.

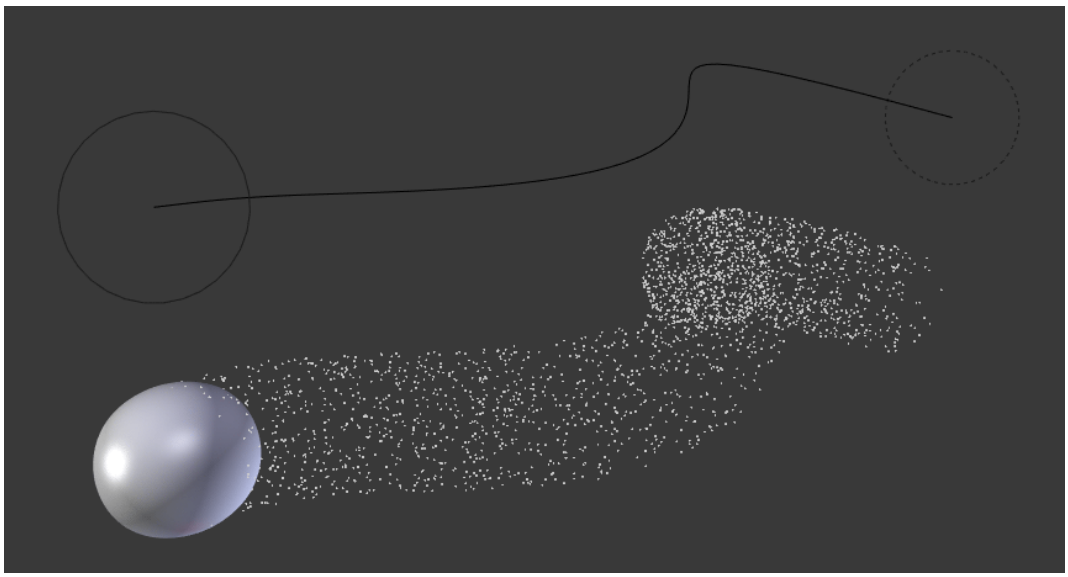
Axis
: Which axis to use for the offset.

Frequency
: The frequency of the offset.

Shape
: Adjust the offset to the beginning/end.

Amplitude
: The Amplitude of the offset.

Examples



Curve Guide force field.

Drag

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Drag

A *Drag* force field resists particle motion by slowing it down.

Options

Linear
: Drag component proportional to velocity.

Quadratic
: Drag component proportional to the square of the velocity.

Fluid Flow

Reference

Panel::	*Physics* ▸ *Force Fields*
Type::	Fluid Flow

The *Fluid Flow* force field creates a force based on a Fluid simulation air flow. It applies the smoke simulation air flow velocity as a force to other simulations that use force fields. To use it you need to add a *Fluid Flow* force field and select a domain object for it. For example fire sparkles can realistically flow along the air turbulence near the simulated fire.

Options

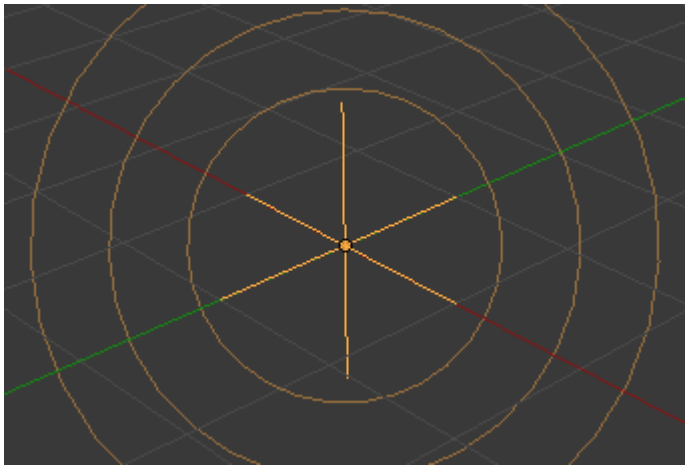
Domain Object
: An object that is used as a domain for the smoke simulation.

Apply Density
: Adjust the force strength based on the smoke density.

Force

Reference

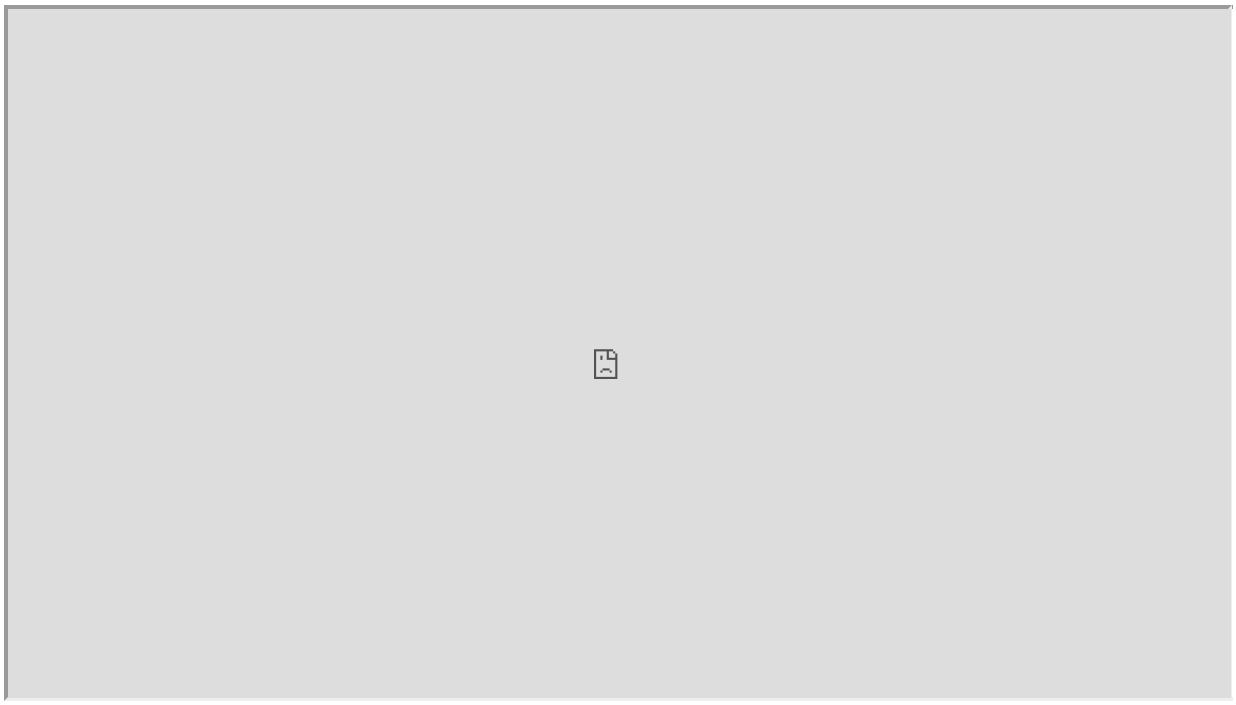
Panel:: *Physics* ▸ *Force Fields*
Type:: Force



Force force field visualization.

The *Force* field is the simplest of the fields. It gives a constant force towards (positive strength) or away from (negative strength) the object's origin. Newtonian particles are attracted to a field with negative strength, and are blown away from a field with positive strength.

Example



Harmonic

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Harmonic

In a *Harmonic* force field, the source of the force field is the zero point of a harmonic oscillator (spring, pendulum). If you set the *Damping* parameter to 1, the movement is stopped in the moment the object is reached. This force field is really special if you assign it to particles.

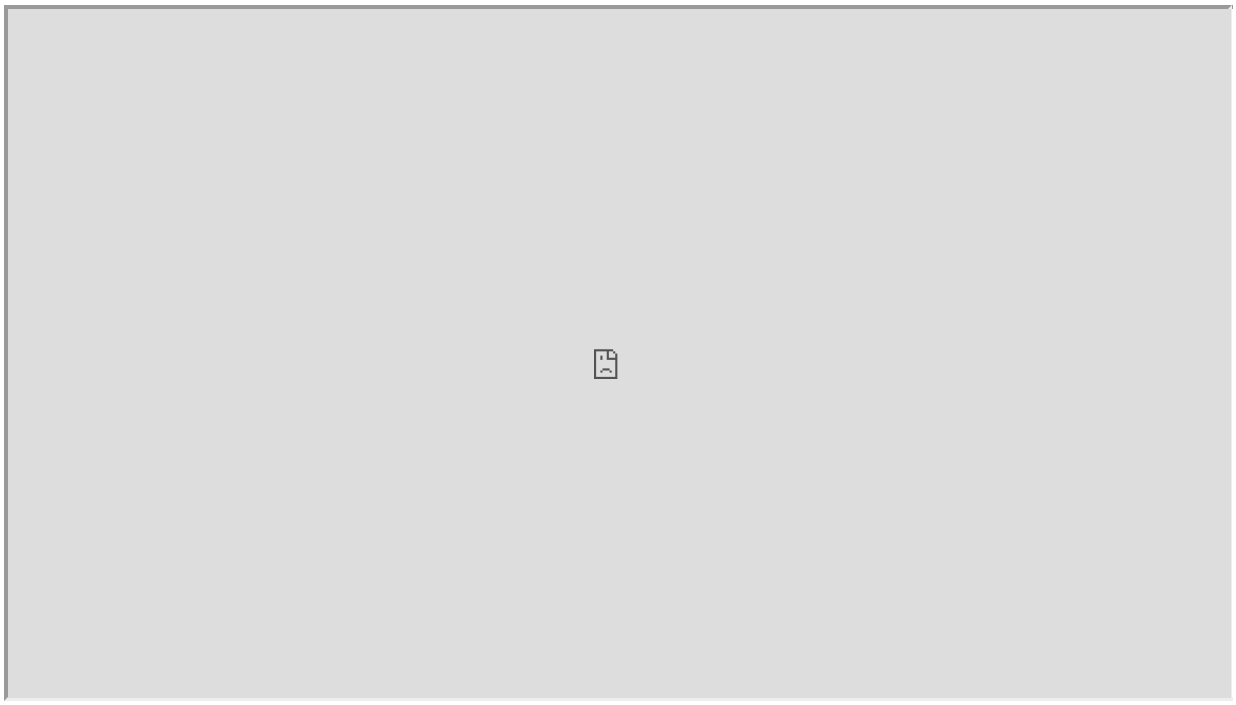
Options

Rest Length
: Controls the rest length of the harmonic force.

Multiple Springs
: Causes every point to be affected by multiple springs.

Normally every particle of the field system influences every particle of the target system. Not with *Harmonic*! Here every target particle is assigned to a field particle. So particles will move to the place of other particles, thus forming shapes.

Example



Lennard-Jones

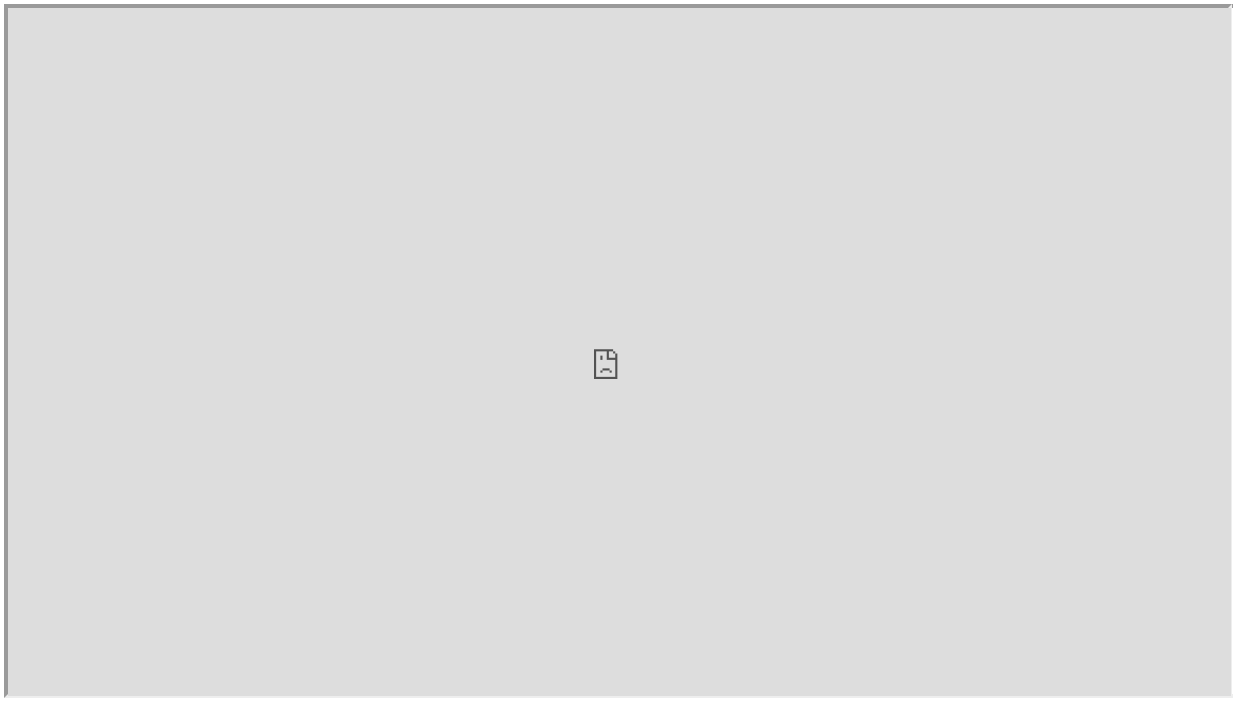
Reference

Panel::	*Physics ▸ Force Fields*
Type::	Lennard-Jones

The *Lennard-Jones* force field is a very short range force with a behavior determined by the sizes of the effector and effected particle. At a distance smaller than the combined sizes, the field is very repulsive and after that distance it is attractive. It tries to keep the particles at an equilibrium distance from each other. Particles need to be at a close proximity to each other to be effected by this field at all.

Particles can have for example both a charge and a Lennard-Jones potential, which is probably something for the nuclear physicists among us.

Example



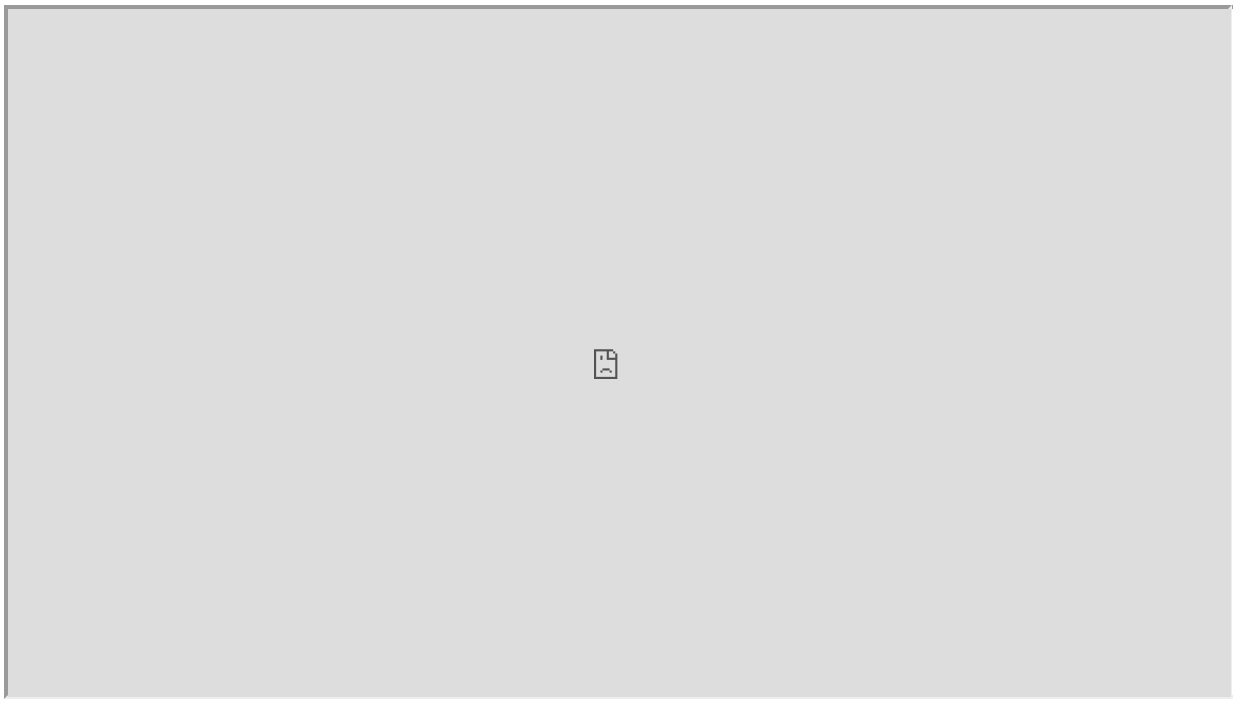
Magnetic

Reference

Panel::	*Physics ▸ Force Fields*
Type::	Magnetic

This field depends on the speed of the particles. It simulates the force of magnetism on magnetized objects.

Example



Texture

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Texture

You can use a *Texture* force field to create an arbitrarily complicated force field, which force in the three directions is color coded. Red is coding for the X axis, green for the Y axis and blue for the Z axis (like the color of the coordinate axes in the 3D Viewport). A value of 0.5 means no force, a value larger than 0.5 acceleration in negative axis direction (like -Z), a value smaller than 0.5 acceleration in positive axis direction (like +Z).

Options

Texture Mode
: This sets the way a force vector is derived from the texture.

Curl:: Calculates the force vector from the curl of the 3D-RGB texture (rotation of RGB vectors). This also works only with a color texture. It can be used for example to create a nice looking turbulence force with a color clouds texture with Perlin noise.

Gradient:: Calculates the force vector as the 3D gradient of the intensity (grayscale) of the texture. The gradient vector always points to the direction of increasing brightness.

RGB:: Uses the color components directly as the force vector components in the color encoded directions. You need an RGB texture for this, e.g. an image or a color ramp. So a *Blend* texture without a color ramp would not suffice.

Nabla
: It is the offset used to calculate the partial derivatives needed for *Gradient* and *Curl* texture modes.

Use Coordinates
: Uses the emitter object coordinates (and rotation & scale) as the texture space the particles use. Allows for moving force fields, that have their coordinates bound to the location coordinates of an object.

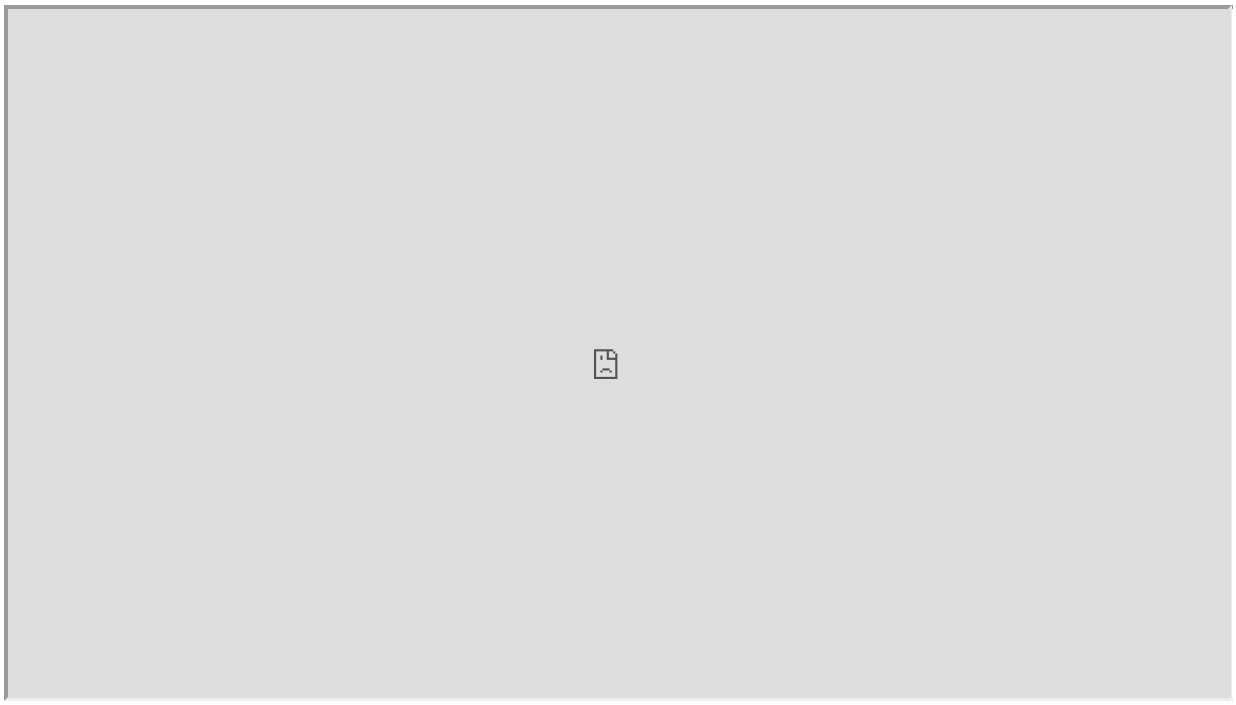
Root Texture Coordinates
: This is useful for hair as it uses the texture force calculated for the particle root position for all parts of the hair strand.

2D
: Disregards the particles Z coordinate and only uses particles X & Y as the texture coordinates.

Remember that only procedural texture are truly 3D.

Examples

- A single colored texture (0.5, 0.0, 0.5) creates a force in the direction of the positive Y axis, e.g. hair is orientated to the Y axis.
- A blend texture with color ramp can be used to created a force “plane”. E.g. on the left side (0.5, 0.5, 0.5), on the right side (1.0, 0.5, 0.5) you have a force plane perpendicular to XY (i.e. parallel to Z). If you use an object for the coordinates, you can use the object to push particles around.
- An animated wood texture can be used to create a wave like motion.



Turbulence

Reference

Panel:: *Physics ▸ Force Fields*
Type:: Turbulence

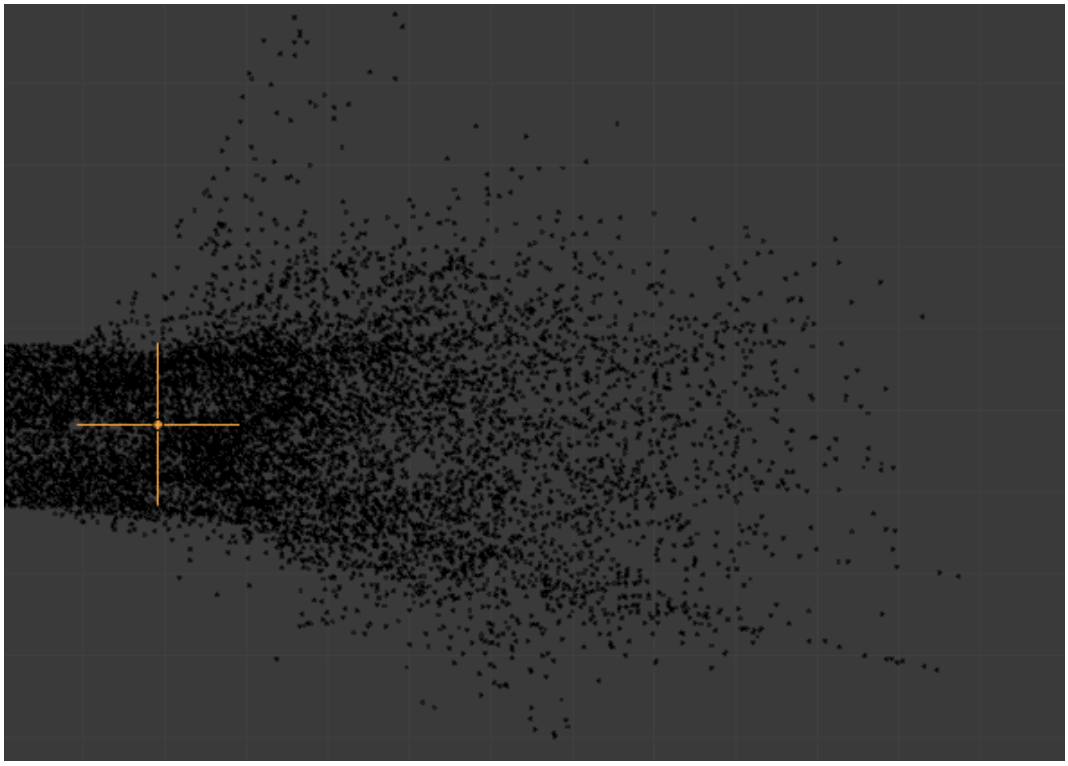
A *Turbulence* force field creates a random & chaotic 3D noise effect, similar to jets of water or geysers under the ocean.

Options

Size
: Indicates the scale of the noise.

Global
: Makes the size and strength of the noise relative to the world, instead of the object it is attached to.

Example

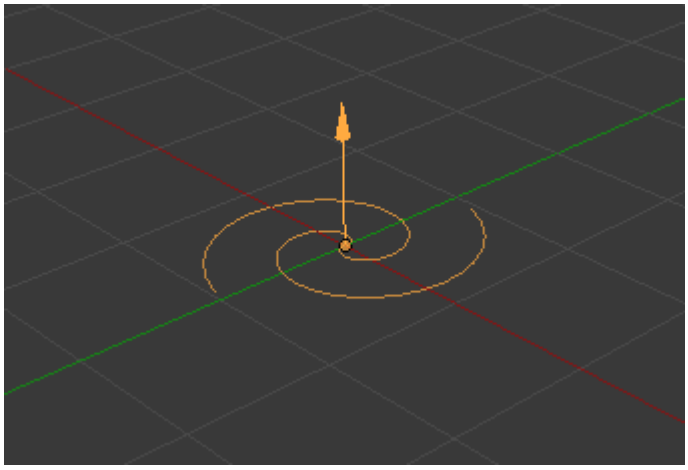


Turbulence force field affecting a particle system.

Vortex

Reference

Panel:: *Physics* ▸ *Force Fields*
Type:: Vortex



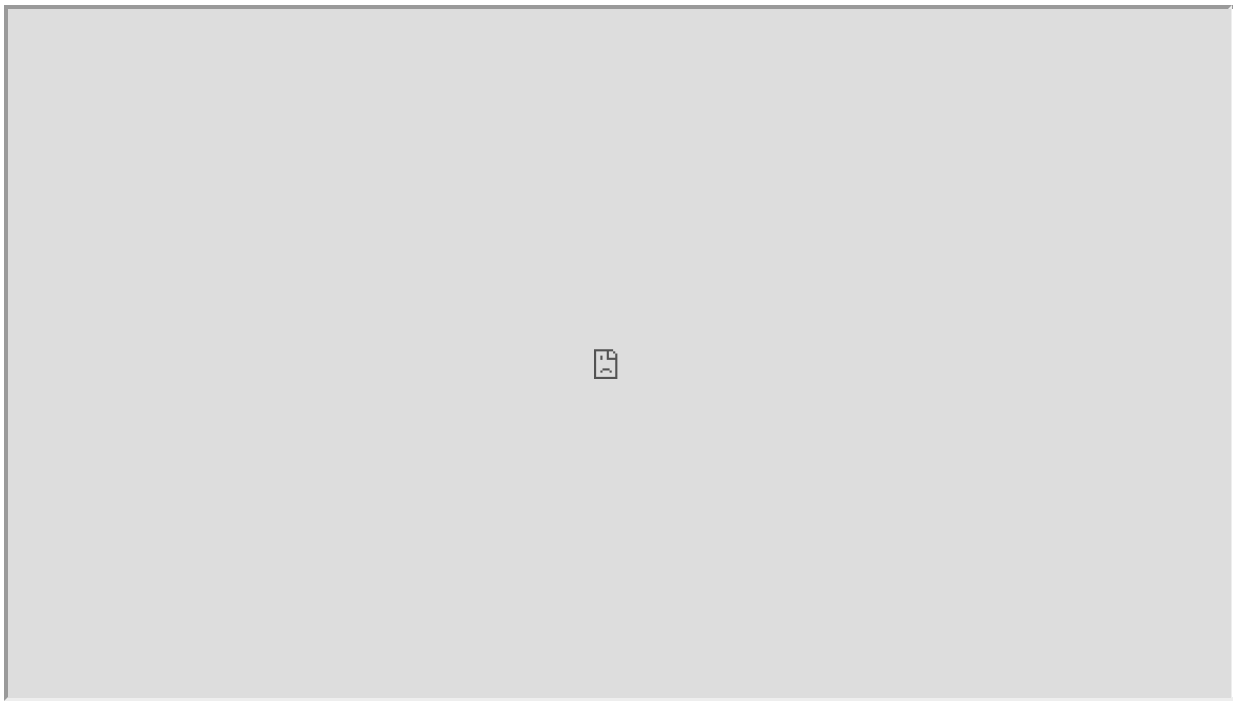
Vortex force field visualization.

The *Vortex* force field gives a spiraling force that twists the direction of points around the force object's local Z axis. This can be useful for making a swirling sink, or tornado, or kinks in particle hair.

Options

Inflow
: Inwards component of the vortex force.

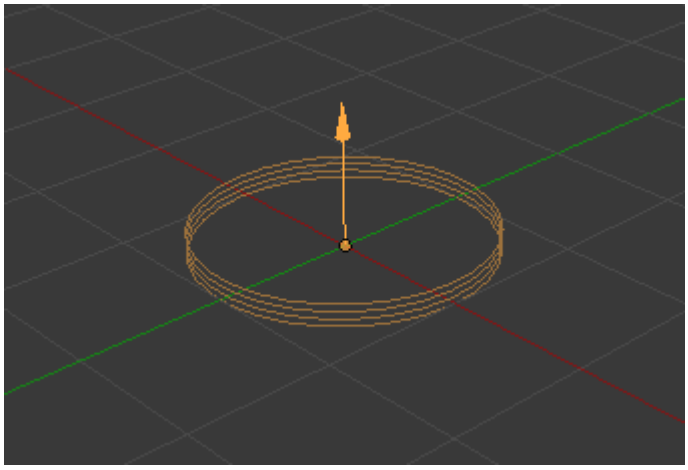
Example



Wind

Reference

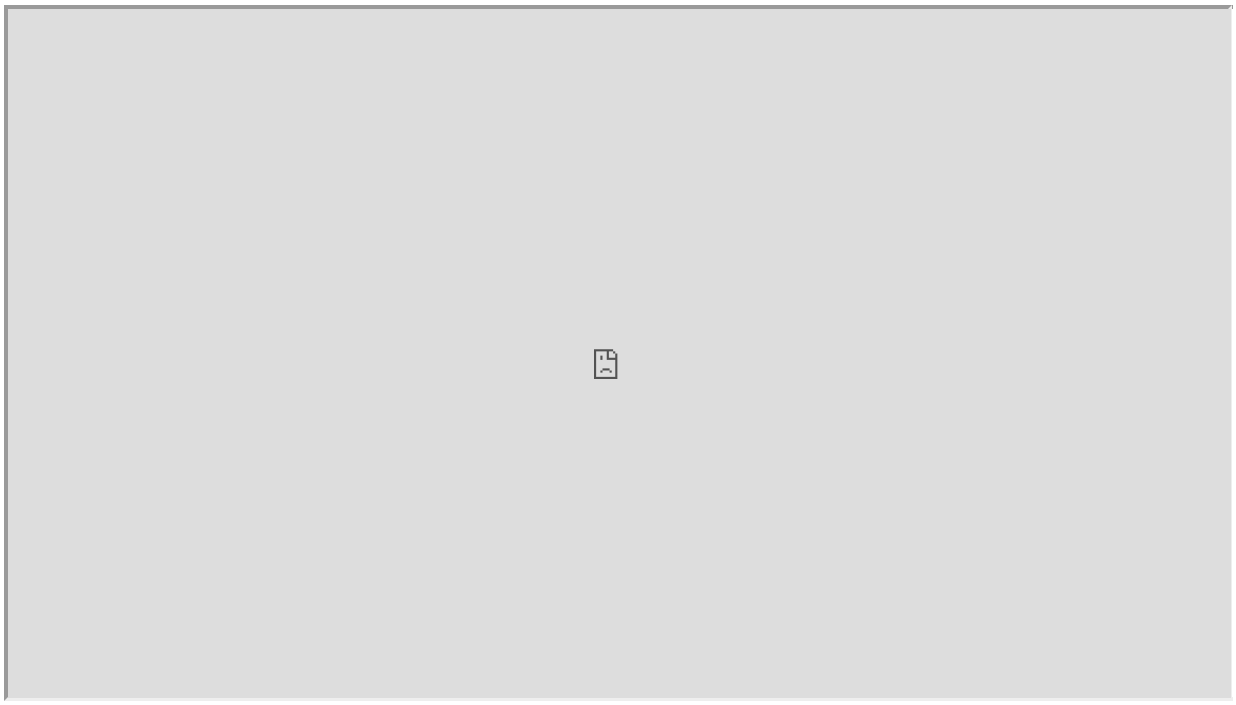
Panel:: *Physics* ▸ *Force Fields*
Type:: Wind



Wind force field visualization.

The *Wind* force field gives a constant force in a single direction, along the force object's local Z axis. The strength of the force is visualized by the spacing of the circles shown.

Example



Collision

Reference

Mode::	Object Mode
Panel::	*Physics* ▸ *Collision*

Particles, Soft Bodies and Cloth objects may collide with mesh objects. Boids try to avoid *Collision* objects.

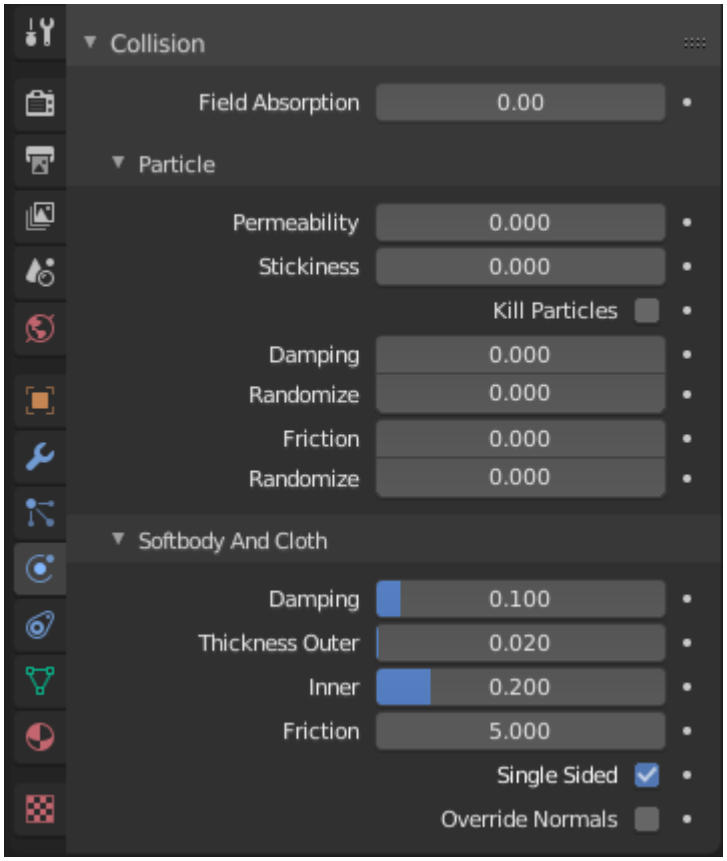
- You may limit the effect on particles to a group of objects (in the Field Weights panel).
- *Deflection* for soft body objects is difficult, they often penetrate the colliding objects.
- Hair particles ignore deflecting objects (but you can animate them as soft bodies which take deflection into account).

If you change the deflection settings for an object you have to recalculate the particle, soft body or cloth system by *Delete Bake*, this is not done automatically.

A collider object can be temporarily disabled via an animatable toggle to the right of the button that permanently activates or deactivates it.



Options



Collision panel.

Collision

Field Absorption
: A deflector can also deflect effectors. You can specify some collision/deflector objects which deflect a specific portion of the effector force using the *Field Absorption* value. 100% absorption results in no force getting through the collision/deflector object at all. If you have three collision object behind each other with e.g. 10%, 43% and 3%, the absorption ends up at around 50% \(100 × (1 - 0.1) × (1 - 0.43) × (1 - 0.03)\).

Particle

Permeability
: Fraction of particles passing through the mesh.

Stickiness
How much particles stick to the object.

Kill Particles
Deletes Particles upon impact.

Damping
Damping during a collision (independent of the velocity of the particles).

Randomize
Random variation of damping.

Friction
Friction during movements along the surface.

Randomize
Random variation of friction.

Soft Body and Cloth

It is also important to note that this collision panel is used to tell all simulations that this object is to participate in colliding/deflecting other objects on a shared layer (particles, soft bodies, and cloth).

Note

The object's shape deforms the cloth, so the cloth simulation must be inputted the "true" shape of that mesh object at that frame. This true shape is the basis shape as modified by shape keys or armatures. Therefore, the Collision Modifier must be **after** any of those. The image to the right shows the *Modifiers* panel for the Character mesh object (not the cloth object).

Damping

Damping during a collision. The amount of bounce that the surfaces will have.

- 0.0 - No damping, soft bodies will have a maximum bounciness.
- 1.0 - Maximum damping, soft bodies will not bounce at all.

Thickness

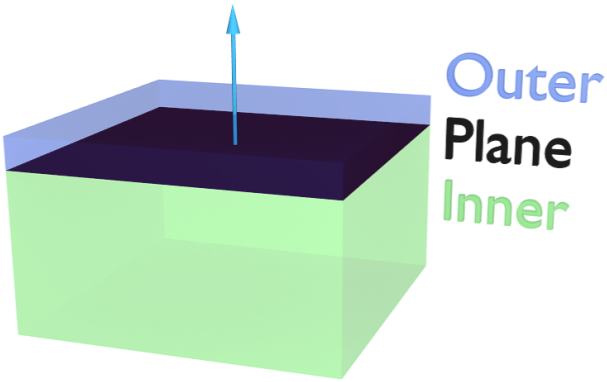
A padding distance is added to the inside and outside of each face, to help to prevent intersections. The soft body will come to rest at this distance away from the face of the colliding object. Outside and inside is defined by the face normal, depicted as blue arrow in Fig. A soft body vertex colliding with a plane..

Outer

Size of the outer collision zone.

Inner

Size of the inner collision zone (padding distance).



A soft body vertex colliding with a plane.

Friction

A coefficient for how slippery the cloth is when it collides with itself. For example, silk has a lower coefficient of friction than cotton.

Single Sided

When enabled, the collider is considered to represent the boundary of a solid object rather than a thin surface, and ejects intersecting cloth in the direction of its normal.

Override Normals
: When enabled, cloth collision impulses act in the direction of the collider normals.

Note

Soft body collisions are difficult to get perfect. If one of the objects move too fast, the soft body will penetrate the mesh. See also the section about Soft Bodies.

Examples



Deflected particles.

Here is a *Meta* object, using Instancing Vertices to a particle system emitting downwards, and deflected by a mesh cube.

Hints

- Make sure that the normals of the mesh surface are facing towards the particles/points for correct deflection. Negative scales on the object can have a similar effect. Make sure to recalculate the normals after applying the scale.
- Hair particles react directly to force fields, so if you use a force field with a short range you do not need necessarily collision.
- Hair particles avoid their emitting mesh if you edit them in *Particle Edit Mode*. So you can at least model the hair with collision.

Baking Physics Simulations

Baking refers to the act of storing or caching the results of a calculation. The result of a simulation is automatically cached to memory when the animation is played, so that the next time it runs, it can be replayed more quickly by reading the results from the memory.

If you bake the simulation the cache is protected, and you will be unable to change the simulation settings until you clear the baked frames by clicking *Delete Bake*.

It is generally recommended to bake your physics simulations before rendering. Aside from no longer needing to go through the time-consuming process of simulating again, baking can help prevent potential glitches and ensure that the outcome of the simulation remains exactly the same every time.

Note

Most physics simulators in Blender use a similar system, but not all have exactly the same settings available. All the settings are covered here, but individual physics types may not provide all these options.

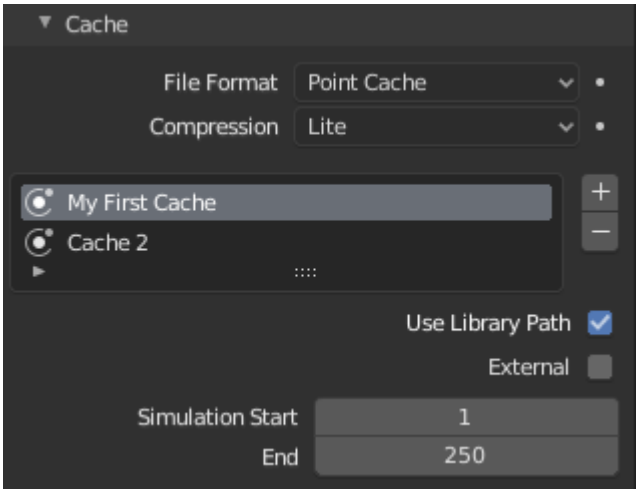
Options

Caches List

Blender allows for storing and managing multiple caches at once for the same physics object. You can manage the caches with this list view. Double-click the cache entry to give it a name.

Each cache can have a name. Double-click the cache entry to give it a name. If this name is given, any disk cache will be stored in files

starting with that name. For example, a cache named `MyCache` will be stored in `MyCache_xxxxxx_yy.bphys`.



Two different caches stored simultaneously.

If the cache does not have a name (which is the default), the filename of the cache will depend on the object it is attached to, although this is not immediately obvious. For example, a cache on an object `Cube` will be stored in `43756265_xxxxxx_yy.bphys`, where `43756265` is determined by the object name.

Warning

When there are multiple caches on one object, **always specify a Cache Name**. As described above, the filename of an unnamed cache is determined by the name of the object it is attached to. As a result, an object with multiple physics systems that all have an unnamed cache will cause conflict and **can result in losing cache files**.

External

Allows you to read the cache from a drive using a user-specified file path.

Note

The cache name in *Caches List* and the *Index* has to exactly match the external cache files name in order to work. The cache files name format is `name_frame_index.bphys`.

Index

The index number of cache files. (The last two digits of the files name.)

Path
: Select the directory path to the cache files.

Disk Cache
: The cache of a baked simulation will be stored inside the blend-file when you save it. When *Disk Cache* is checked, Blender will save the cache separately to the drive in a folder named `blendcache_[filename]` alongside the blend-file. (The blend-file must be saved first.)

Note

When using Library Overrides, data-blocks only support *Disk Cache* storage.

Use Library Path
: Share the disk cache when the physics object is linked into another blend-file. When this option is enabled, linked versions of the object will reference the same disk cache. Otherwise linked versions of the object will use independent caches.

Compression
: The compression level for cache files. Some physics caches can be very large, Blender can compress these caches in order to save space.

None
: Do not compress the cache.

Light
: Compression will optimize the speed of compressing/decompressing operations over file size.

Heavy
: Compression will result in smaller cache files more than *Light*, however, requires more CPU time to compress/decompress.

Start
: Frame on which to start the simulation.

End
: Frame on which to stop the simulation.

> **Note**
>
> The simulation is only calculated for positive frames in between the *Start* and *End* frames of the *Cache* panel, whether you bake or not. So if you want a simulation that is longer than the default frame range you have to change the *End* frame.

Cache Step
: Interval for storing simulation data.

> **Note**
>
> Some physics systems (such as particles) allow for positions to be stored only on every nth frame, letting the positions for in-between frames be interpolated. Using a cache step greater than one will result in a smaller cache, but the result may differ from the original simulation.

Baking

Bake
: Start baking. Blender will become unresponsive during most baking operations. The cursor will display as a number representing the progress of the baking. You need to be in Object Mode to bake.

Delete Bake
: Mark the baked cache as temporary. The data will still exist, but will be removed with the next object modification and frame change. This button is only available when the physics system has been baked.

Calculate to Frame

Bake only up to the current frame. Limited by *End* frame set in the cache settings.

Current Cache to Bake

Store any temporarily cached simulation data as a bake. Note that playing the animation will try to simulate any visible physics simulations. Depending on the physics type, this data may be temporarily cached. Normally such temporary caches are cleared when an object or setting is modified, but converting it to a bake will "save" it.

Bake All Dynamics

Bake all physics systems in the scene, even those of different types. Useful for baking complex setups involving interactions between different physics types.

See Bake.

Delete All Bakes

Delete bakes of all physics systems in the scene, even those of different types.

See Delete Bake for more information.

Update All to Frame

Bake all physics systems in the scene to the current frame.

See Calculate To Frame.

Simulation Nodes

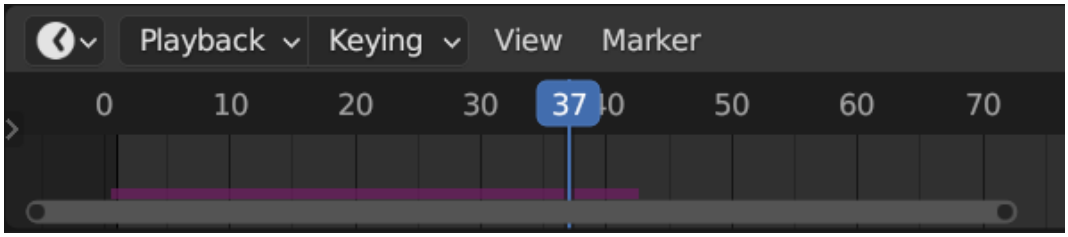
Through the use of Simulation Zones, Geometry Nodes can be used to create custom physic simulations through nodes. Simulation zones allow the result of one frame to influence the next one. That way even a set of simple rules can lead to complex results, with the passing of time. The most common type of them is physics simulation, with specific solvers for physical phenomena.

See also

Read more about Simulation Zones

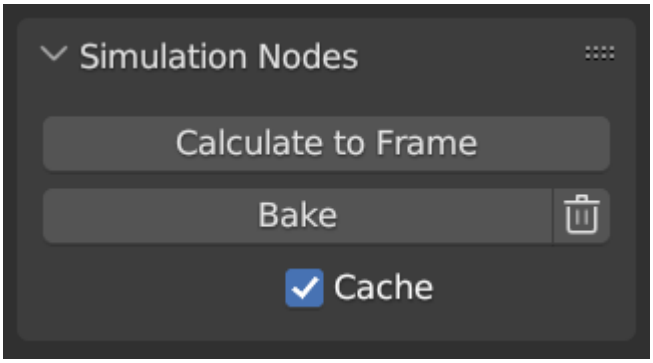
Baking

The simulation is automatically cached during playback. The valid cache can be seen as a strong yellow line in the timeline editor. This allows for animators to quickly inspect all the previous frames of a simulation.



Cached frames in the Timeline.

When the result is ready to be sent to a render-farm, it can be baked to disk. This allows for the simulation to be rendered in a non-sequential order.



Simulation and Physics, Geometry Nodes user interface.

Note

Baking the simulation will bake all the simulations in all modifiers for the selected objects.

Calculate to Frame
: Calculate simulations in geometry nodes modifiers from the start to current frame.

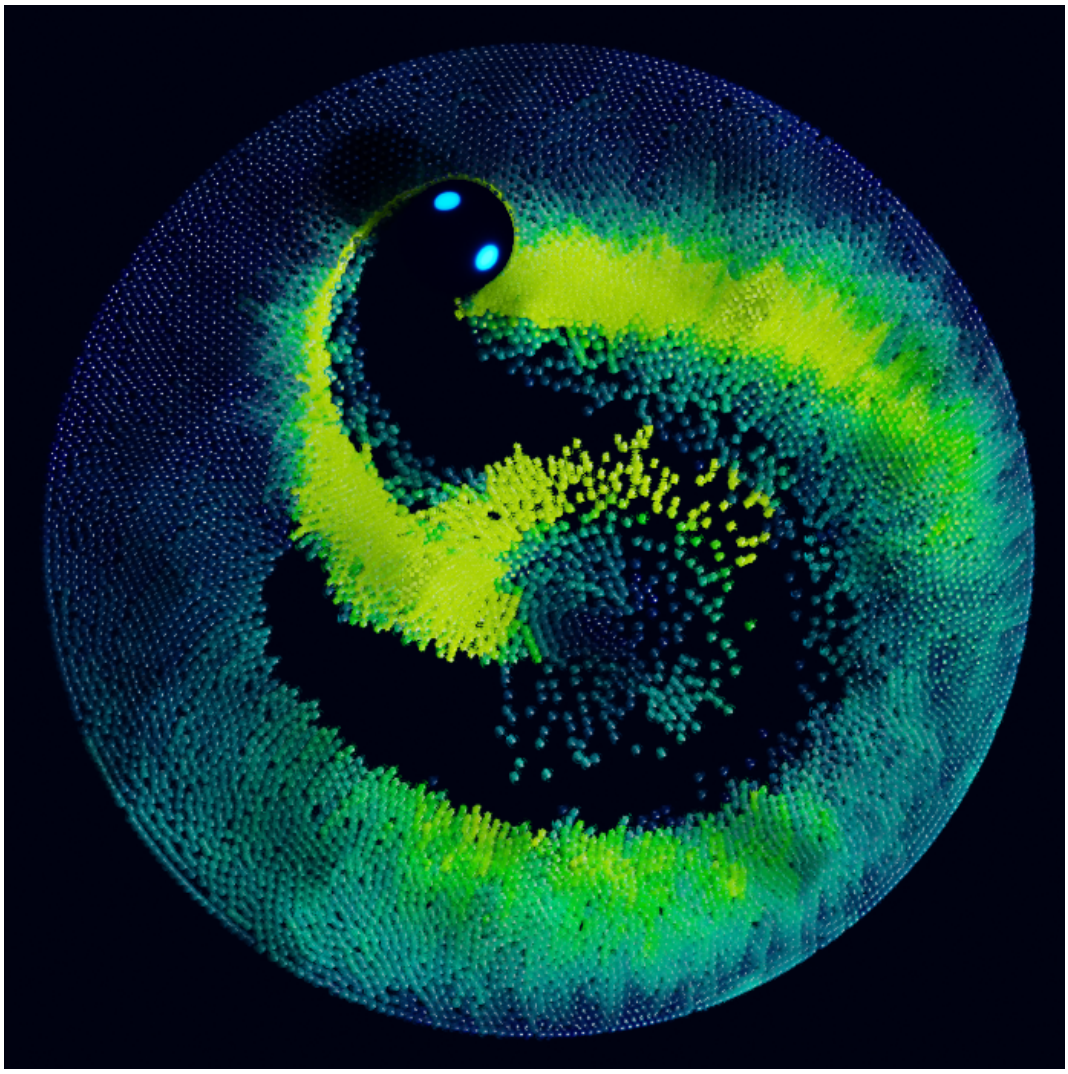
Bake
: Bake simulations in geometry nodes modifiers. In order to bake the simulation, the blend-file must be saved to your computer. The location the file is saved determines where the baked data is also saved. The directory the baked data is saved to can be changed per modifier in the Internal Dependencies.

  Delete Cached Simulation
  : Delete cached/baked simulations in geometry nodes modifiers

Cache
: For the cases where the current frame is the only one relevant, users can opt-out of caching the results to save memory.

Examples

Combined with the Index of Nearest, this can be used for a number of sphere-based simulations.



Index of Nearest sample file CC-BY Sean Christofferson.

Rendering

- Introduction
- EEVEE
  - Introduction
  - Render Settings
  - Scene Settings
  - World Settings
  - Object Settings
  - Materials
  - Light Settings
  - Light Probes
  - Limitations
- Cycles
  - Introduction
  - Render Settings
  - World Settings
  - Object Settings
  - Material Settings
  - Light Settings
  - GPU Rendering
  - Experimental Features
  - Render Baking
  - Optimizing Renders
- Workbench
  - Introduction
  - Performance
  - Sampling
  - Lighting
  - Color
  - Options
  - Grease Pencil
  - Viewport Display
- Cameras
  - Properties

- Lights
  - Light Objects
  - Light Linking
  - World Environment
- Materials
  - Introduction
  - Components
  - Assignment
  - Preview
  - Settings
  - Line Art
  - Legacy Textures
- Shader Nodes
  - Introduction
  - Input
  - Output
  - Shader
  - Texture
  - Color
  - Vector
  - Converter
  - Group
  - Open Shading Language
- Color Management
  - Workflow
  - Render Settings
  - Image Files
  - OpenColorIO Configuration
  - Default OpenColorIO Configuration
- Freestyle
  - Introduction
  - Render Properties
  - View Layer Properties
  - Material Properties
  - Python Scripting
- Layers & Passes
  - Introduction

    - View Layer
    - Passes
- Render Output
    - Introduction
    - Output Properties
    - Audio Rendering
    - Rendering Animations
    - Animation Player

Introduction

Rendering is the process of turning a 3D scene into a 2D image. Blender includes three render engines with different strengths:

- EEVEE is a physically based realtime renderer.
- Cycles is a physically based path tracer.
- Workbench is designed for layout, modeling and previews.

More renderers from third-party developers are available as add-ons. Each renderer has its own render settings to control render quality and performance.

What the render looks like is defined by cameras, lights and materials. These are shared between EEVEE and Cycles, however some features are only supported in one or the other.

Renders can be split up into layers and passes, which can then be composited together for creative control, or to combine with real footage. Freestyle can be used to add non-photorealistic line rendering.

Blender supports interactive 3D viewport rendering for all render engines, for quick iteration on lighting and shading. Once this is done, the final quality image or animation can be rendered and output.

EEVEE

- Introduction
- Render Settings
  - Sampling
  - Clamping
  - Raytracing
  - Volumes
  - Curves
  - Depth of Field
  - Motion Blur
  - Film
  - Performance
  - Grease Pencil
- Scene Settings
- World Settings
- Object Settings
  - Object Properties
- Materials
- Light Settings
- Light Probes
  - Sphere
  - Plane
  - Volume
- Limitations
  - Limitations
  - Supported Nodes

Introduction

EEVEE is Blender's realtime render engine focused on speed and interactivity while achieving the goal of rendering PBR materials. EEVEE can be used interactively in the 3D Viewport but also produce high quality final renders.



EEVEE in the 3D Viewport – "Tiger" by Daniel Bystedt.

EEVEE materials are created using the same shader nodes as Cycles, making it easy to render existing scenes. For Cycles users, this makes EEVEE work great for previewing materials in realtime.

EEVEE is based on rasterization and is not a path tracer. Instead of computing each ray of light, rasterization determines what surface is visible from the camera. It then estimates the way light interacts with these surfaces and materials using numerous algorithms. While EEVEE is designed to use PBR principles, it is not perfect and Cycles will always provide more physically accurate renders. For these reasons, EEVEE has a set of limitations.



EEVEE final render – “Temple” by Dominik Graf.

Render Settings

- Sampling
- Clamping
- Raytracing
- Volumes
- Curves
- Depth of Field
- Motion Blur
- Film
- Performance
- Grease Pencil

Sampling

EEVEE uses a process called Temporal Anti-Aliasing (TAA) which reduces Aliasing. TAA is sample based so the more samples the more aliasing is reduced at the cost of performance.

Reference

Panel:: *Render ▸ Sampling*

Viewport

Samples
: The number of samples to use in the 3D Viewport. When setting this to zero the viewport will be resampled continuously.

Temporal Reprojection
: Reduces noise while moving the viewport or during animation playback. Can leave some ghosting.

Jittered Shadows
: Enable jittered shadows on the viewport. Jittered shadows are always enabled for final renders. This also affects shadows casted by transparent shadows.

Render

Samples
: The number of samples to use in the final render.

Shadows

Rays

Number of rays to trace for each light. Higher values reduces the noise caused by random shadow sampling.

Steps
: Number of shadow map sample per shadow ray. Higher step count results in softer shadows but have a higher cost.

Volumetric Shadows
: Approximate light absorption of the surrounding volume objects. This makes the volumes more opaque to light. This is a very computationally expensive option and has limitations.

  Steps
  : Number of steps to compute volumetric shadowing.

  See also

  Volume Limitations.

Resolution
: Resolution percentage of shadow maps.

Advanced

Light Threshold
: Minimum light intensity for a light to contribute to the lighting. Used to compute the distance at which to cut-off lights influence. Lower values improve performance.

  See also

  Custom Distance overrides this setting.

Clamping

Reference

Panel:: *Render ▸ Clamping*

Surface

Direct Light
: This option limits the maximum light intensity a surface can reflect. It reduces Aliasing noise and Fireflies at the cost of accuracy. Setting this option to 0.0 disables clamping altogether. Lower values have a greater effect on the resulting image than higher values.

Indirect Light
: Similar to **Direct Light** but limits the maximum light intensity reflected using ray-tracing and light-probes.

Note

These options provide a way to limit Fireflies and Aliasing of highly reflective surfaces and dense volumes. However, note that as you clamp out such values, other bright lights will be dimmed as well.

Care must be taken when using this setting to find a balance between mitigating fireflies and losing intentionally bright parts.

Volume

Direct Light
: The same as *Surface Direct Light* but for volume direct lighting.

Indirect Light

The same as *Surface Direct Light* but for volume indirect lighting.

Raytracing

Reference

Panel:: *Render ▸ Raytracing*

The ray-tracing pipeline goal is to increase the accuracy of surface indirect lighting. This is done by generating ray from each BSDF and finding their intersection with the scene individually.

When disabled, it is replaced by a faster pipeline that uses pre-filtered light-probes. This fallback mode offers a more visually stable and optimized alternative when visual fidelity is not the primary goal.

See also

Limitations.

Method
: Determine the tracing method used to find scene-ray intersections and indirect lighting.

Light Probe:: Use light-probe spheres and planes to find scene intersection. This option has the lowest tracing cost but relies on manually placed light-probes.

Screen-Trace:: Trace ray against the screen depth buffer. Fallback to light-probes if ray exits the view.

Resolution
: Resolution at which the ray-tracing is performed. Lower options will be faster and use less memory but will produce blurrier results.

Screen Tracing

These settings control the behavior of the screen space ray-tracing. They are only visible if *Screen-Trace* is the active tracing *Method*.

Precision
: Higher values increase precision of the screen space ray-tracing but lower the maximum trace distance. Increased precision also increases performance cost.

Thickness
: How thick to consider the pixels of the depth buffer during the tracing. Higher values will stretch the reflections and add flickering. Lower values may make the ray miss surfaces.

Denoising

Denoising can be enabled to reduce the amount of noise from the raw ray-traced output. This can help image stability but will also over-blur the final ray-traced output.

Spatial Reuse
: Reuse the rays from neighbor pixels. Can introduce some light leaks across surfaces.

Temporal Accumulation
: Accumulate samples by re-projecting the last ray tracing results. This removes Fireflies but also introduces color bias. Useful for viewport temporal stability or making renders converge faster.

Bilateral Filter
: Blur the resolved ray-traced output using a bilateral filter.

Fast GI Approximation

Fast GI Approximation is a fallback to the ray-tracing pipeline for BSDF with high roughness. It produces a less noisy output and captures bounce lighting more efficiently than individually traced rays.

This is currently implemented as a screen space effect and will inherit all associated limitations.

Threshold

Maximum roughness a BSDF can have to use ray-tracing. BSDFs with higher roughness will progressively use the *Fast GI Approximation*. A value of 1 will raytrace every surfaces and disable the Fast GI.

Method

Determine the method used to compute the fast GI approximation.

Ambient Occlusion::

Use scene intersections to shadow the distant lighting from light-probes. This is the fastest option.

Global Illumination::

Compute global illumination taking into account light bouncing off surrounding objects.

Resolution

Resolution at which the fast GI is computed. Lower options will be faster and use less memory but will produce blurrier results.

Rays

Number of GI rays per pixel at the specified *Resolution*. Higher values will reduce noise.

Steps

Number of screen samples per GI ray. Higher values will reduce the noise amount and increase the quality.

Tip

With a higher step count, there is less chance to miss other surfaces that could reflect or block the light. This means that the Fast GI *Thickness* parameters can be tweaked to lower values without losing too much light bounce energy.

Precision

Higher values increase the precision of the scene intersections with the GI rays. Increased precision also increases performance cost.

Distance

If non-zero, the maximum distance at which other surfaces will contribute to the fast GI approximation.

Thickness Near

Geometric thickness of the surfaces when computing fast GI and ambient occlusion. Reduces light leaking and missing contact occlusion. The effectiveness decreases proportionally to the distance from the shading point, following the inverse square law.

Far

Angular thickness of the surfaces when computing fast GI and ambient occlusion. Reduces energy loss and missing occlusion of far geometry. Higher values will make the very thin objects block or reflect too much light.

Bias

Bias the shading normal to reduce self intersection artifacts.

Volumes

Reference

Panel:: *Properties ▸ Render ▸ Volumes*

EEVEE simulates volumetric scattering by evaluating all volume objects inside the view frustum.

To achieve this, EEVEE uses several 3D textures which have a high video memory usage. The texture dimensions can be tweaked using the *Resolution* and *Steps* parameters.

Resolution
: Controls the quality of the volumetric effects. Lower resolution increases video memory usage and quality.

Steps
: Number of steps to compute volumetric effects. Higher count increases video memory usage and quality. These samples are distributed along the view depth (view Z axis).

Distribution
: Blend between linear and exponential sample distribution. Higher values put more samples near the camera.

Max Depth
: Maximum surface intersection count used by accurate volume intersection method. Will create artifacts if it is exceeded.

Custom Range

When working with volume objects, EEVEE automatically computes the best depth range where to compute the volume sampling and lighting. In

certain situations, this isn't enough and produces sub-optimal sampling which increases noise. This is particularly the case when using a volume shader inside the *World* or when working with large number of volume objects. The custom depth range can be enabled to restrict the computation of volumes to a certain range along the camera depth and thus increase precision.

Start
: Start distance of the volumetric effect.

End
: End distance of the volumetric effect.

See also

Limitations.

Curves

Reference

Panel:: *Render ▸ Curves*

Shape

Strand:: Render curves as a thin strand roughly a pixel wide. Curve diameter parameters are ignored with this setting.

Strip:: Render curves as a flat ribbon with rounded normals.

Additional Subdivisions

Additional subdivisions to be applied on top of the curve resolution set in the hair system settings. Increasing this value will smooth out the curves of the strands.

Depth of Field

To render a scene, EEVEE uses a pinhole camera model which produces a perfectly focused image of the scene. For an enhanced realism, EEVEE can simulate the optical Depth of Field using a post-process filter, and a sample-based method. The optical settings are located in the camera settings properties. Whereas the quality of the effect can be controlled by the settings found in the present section.

Note

In the 3D Viewport, depth of field only works while in Camera View.

The post-process method is computed in two passes. The first pass is using a blur that fails to produce quality bokeh for highlights but works for the general case. Followed by a second pass which is sprite-based and improves only the quality of very bright highlights. That is because it is too slow to be applied on every part of the image. So it just includes very bright isolated parts of the image such that are different from their surroundings. Which pixels are being processed by second pass can be control with the *Sprite Threshold* and *Neighbor Rejection* options.

Secondly the sample-based method works by randomizing the camera position for every sample. It is more accurate but needs many samples to achieve a smooth result. Accordingly the post-process blurring radius is scaled down to remove undersampling. Yet some scenes might still need more post-process blur in order to remove the noticeable sample pattern. This is exactly what the *Overblur* option does, but it will also reduce the bokeh shape sharpness.

Reference

Panel:: *Render ▸ Depth of Field*

Max Size

Maximum size in pixels of the depth of field post-process effect (lower is faster). A value of 0 will disable the post-process effect but not the sample-based method.

Sprite Threshold

Minimum brightness a pixel needs to have to be considered by the sprite-based depth of field. Higher values will improve the performance but will also reduce the quality of highlights. Brightness is in the scene's referred color space.

Neighbor Rejection

Maximum intensity to consider when doing sprite neighborhood rejection. This should be set to a brightness value above which there is small visual differences to be noticeable after color management. Lower values will improve the performance but will also reduce the quality of highlights. Brightness is in the scene's referred color space.

Jitter Camera

Randomize the camera position for every scene render sample to increase precision. Enabling this option can change the scene's actual sample count.

Note

Be aware that the actual sample count can grow quite rapidly.

Hint

The actual number of samples is computed by the following formula:

$$sample_count = (ring_count^{2} + ring_count) * 3 + 1$$

where $ring_count$ is the number of ring in the hexaweb pattern. The $ring_count$ is chosen so that the entire pattern contains at least the number of samples set in the Render Settings.

Over-blur

Scales the post-process depth of field radius to reduce artifacts. Higher values will soften the bokeh shape.

See also

Limitations.

Motion Blur

Reference

Panel:: *Render ▸ Motion Blur*

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie frame and in a photograph from a real-world camera.

Note

Motion blur is only visible in the viewport during animation playback and uses a simpler algorithm than final render. Same thing applies to Viewport Renders.

Position
: Controls at what point the shutter opens in relation to the current frame.

Start on Frame:: Shutter is starting to open at the current frame.
Center on Frame:: Shutter is fully opened at the current frame.
End on Frame:: Shutter is fully closed at the current frame.

Shutter
: Time (in frames) taken between shutter open and close.

Bleeding Bias
: Used by the post-process blur to avoid blurring the background over the foreground. Lower values will reduce background bleeding onto foreground elements.

Max Blur

Max Blur is intended to act as an optimization tool by limiting the number of pixels across which the blur is calculated.

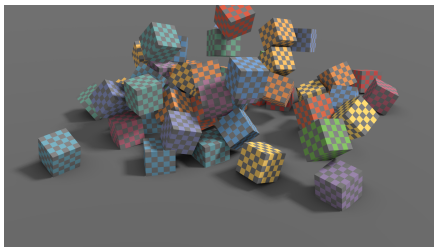
Steps
: This controls the number of steps used by the accumulation blur and thus its accuracy. More steps means longer render time.

> **Note**
>
> When using multiple time steps, the render sample count is rounded up to the next multiple of steps to ensure even distribution of samples across steps.

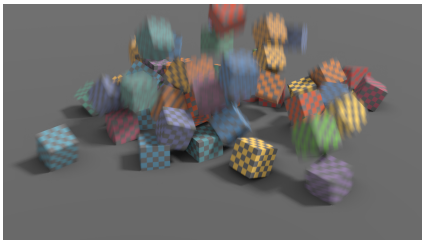
EEVEE splits the render into multiple time steps and accumulates the result which is known as Accumulation Motion Blur. This technique is precise but requires many steps for clean gradients. This is used in combination with the post-process blur to handle the inter-step gaps. Each step corresponds to a full scene re-evaluation and can add a lot of overhead to the render time. By adding more steps you can also reduce the *Max Blur* options because the post-process blur has to cover a smaller distance.

Shutter Curve
: Use a custom shutter curve.

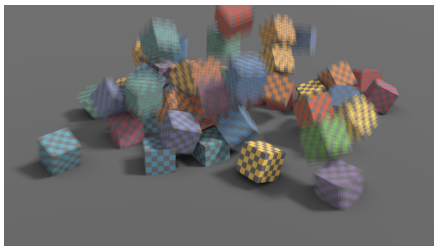
Example



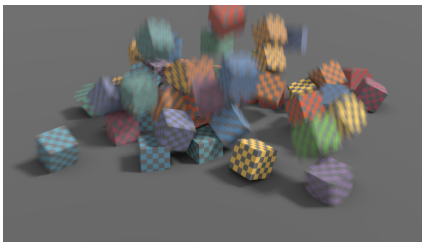
No motion blur.



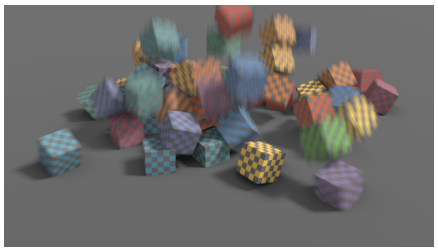
Only post-process blur.



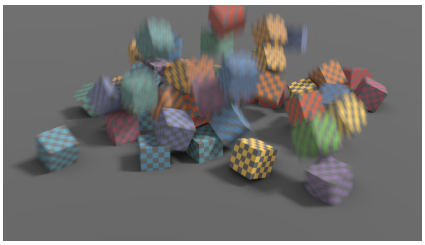
4 time steps without post-process blur.



4 time steps with post-process blur.



32 time steps without post-process blur.



32 time steps with post-process blur.

Film

Filter Size

Due to limited resolution of images and computer screens, pixel filters are needed to avoid Aliasing. This is achieved by slightly blurring the image to soften edges.

This Setting controls how much the image is softened; lower values give more crisp renders, higher values are softer and reduce aliasing.

Transparent

Render the background transparent, for compositing the image over another background after rendering.

Overscan

Percentage of the render size to add to the internal render buffer. This will have a serious impact on performance but can fix render glitches around the perimeter of the rendered image.

Performance

Reference

Panel:: *Properties* ▸ *Render* ▸ *Performance*

High Quality Normals
: Uses higher precision normals and tangents which can improve visual quality for dense meshes with high frequency textures at the cost of memory.

Memory

Shadow Pool
: A bigger pool size allows for more shadows in the scene but might not fit into GPU memory and decreases performance. Increasing the size might fix the *Shadow buffer full* error.

See also

Shadow documentation

Light Probes Volume Pool
: A bigger pool size allows for more irradiance grids in the scene but might not fit into GPU memory and decreases performance.

Viewport

Pixel Size
: Option to control the resolution for viewport rendering. Allows you to speed up viewport rendering, which is especially useful for displays with high DPI.

Grease Pencil

Reference

Panel:: *Render ▸ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold
: Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Scene Settings

Light Probes

Light Probe Spheres Resolution
: Defines the resolution of every light probe sphere in the scene.

World Settings

The world environment can emit light, ranging from a single solid color to arbitrary textures.

In EEVEE, the world lighting contribution is stored into an internal Light Probe. This makes the lighting less precise than Cycles.

Mist Pass

Reference

Panel:: *World ▸ Mist Pass*

Note

The mist pass must be enabled in the View Layer tab of the Properties Editor before the settings below are available in the World tab.

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender generates a render layer with a depth map ranging between 0.0 and 1.0 that can be used in the Compositor to generate a mist effect.

Start
: The distance from the camera at which the mist starts to fade in.

Depth
: The distance from *Start* of the mist, that it fades in over. Objects further from the camera than *Start* + *Depth* are completely hidden by the mist.

Falloff
: The curve function that controls the rate of change of the mist's strength further and further into the distance.

Quadratic:: Uses the same calculation as light falloff (\(1\over{x^2}\)) and provides the smoothest transition from transparent (0.0) to opaque (1.0).

Linear:: Has a steeper start than quadratic (\(1\over{x}\)).

Inverse Quadratic:: Has the steepest start (\(1\over{\sqrt{x}}\)) and approaches 1.0 faster than the other two functions.

Tip

A visualization can be activated in the *Camera* ‣ *Viewport Display* panel.



Mist example (blend-file).

Settings

Reference

Panel:: *World* ‣ *Light Probe*

Light Probe

Resolution
: The resolution used to store the light from the world. This is equivalent to the resolution for light probe spheres.

See also

Light Probe Sphere.

Sun

EEVEE can separate the light from intense light sources (e.g. a sun from an outdoor HDRI) and replace them with a sun light. This increases the quality of the lighting as the internal light probes alone cannot reproduce this type of lighting with enough precision.

Threshold
: If non-zero, the maximum value for world contribution that will be recorded inside the world light probe. The excess contribution is converted to a sun light. This reduces the light bleeding caused by very bright light sources. A value of zero will disable this feature and all lighting will be stored inside the internal light probes.

Angle
: Angular diameter of the extracted sun light as seen from the Earth.

Use Shadow
: Enable shadow casting on the extracted sun light.

See also

The shadow properties control the extracted sun shadows. They are exactly the same as for a sun light object. Light Properties.

Object Settings

Settings for objects and object data.

- Object Properties
  - Shading
  - Visibility

Object Properties

Shading

Reference

Panel:: *Properties* ▸ *Object Properties* ▸ *Shading*

Light Linking

Limit light influence to specified objects, with Light Linking.

Receiver Collection
: Collection of objects that will receive light emitted from the object.

Shadow Linking

Limit shadows to specified objects, with Light Linking.

Shadow Blocker Collection
: Collection of objects that will act as shadow blockers for light emitted from the object.

Visibility

Reference

Panel:: *Object Properties* ▸ *Visibility*

Ray Visibility

Objects can be set to be invisible to particular ray types. This can be used, for example, to make an emitting mesh invisible to camera rays. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect.

Camera
: Makes the object visible to the Camera; this includes the viewport's perspective in viewport rendering.

Shadow
: Enables the object to cast shadows. The object will not be capture inside the shadow maps.

Light Probes

Objects can be set to not be captured by certain light probe. This can be used, for example, to avoid animated object being recorded into static light probes. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

Volume
: Makes the object visible during light probe volumes baking.

Sphere
: Makes the object visible during light probe sphere capture.

Plane
: Makes the object visible during light probe plane capture.

Materials

See also

While EEVEE shares the same material node system as Cycles, not all features are supported. See Shader nodes limitations.

Thickness

Reference

Panel:: *Properties* ▸ *Material* ▸ *Thickness*

Used to approximate the inner geometry structure of the object without heavy computation. This is currently used for Subsurface Scattering, Translucent BSDF, Refraction BSDF, and the nodes containing these effects.

If no value is plugged into the output node, a default thickness based on the smallest dimension of the object is used. If a value is connected it will be used as object space thickness (i.e. scaled by object transform). A value of zero will disable the thickness approximation and treat the object as having only one interface.

This output is only used by the EEVEE render engine.

Note

- The thickness is used to skip the inner part of the object.
- Refraction will not refract objects inside the thickness distance.

- Shadow casting object will not cast shadow within the thickness distance.

Tip

- For large or compound meshes (e.g. vegetation), the thickness should be set to the thickness of individual parts (e.g. leaves, grass blades).
- Thickness can be baked to textures or custom attributes for more accurate result.

See also

Thickness Mode – controls how the thickness value is used.

Material Settings

Reference

Panel:: *Properties* ▸ *Material* ▸ *Settings*

Pass Index
: Index number for the *Material Index* render pass. This can be used to give a mask to a material which then can be read with the ID Mask Node in the Compositor.

Note

Volume Objects do not support the pass index.

Surface

Backface Culling

Backface Culling hides the back side of faces. This option should be turned on whenever it is possible, as it has an impact on performance.

Camera
: Use back face culling to hide the back side of the face.

Shadow
: Use back face culling when casting shadows.

Light Probe Volume
: Use back face culling when baking Light Probe Volumes. Additionally helps rejecting capture point inside the object to avoid light leaking

Displacement
: Controls how the displacement output from the shader node tree is used.

Bump Only:: Use Bump Mapping to simulated the appearance of displacement. This only modifies the shading normal of the object. Vertex position is not affected.

Displacement Only:: This mode is not supported and falls back to *Displacement and Bump*.

Displacement and Bump:: Combination of true displacement and bump mapping for finer details. Vertex position is modified.

Note

This type of displacement is not precomputed. It has a performance impact multiplied by the render sample count. However, the evaluation is much faster than doing it using geometry nodes or a displacement modifier.

Note

Displacing flat shaded geometry will split adjacent faces. This can be worked around by passing the vertex normals as a custom attribute.

Max Distance

The maximum distance a vertex can be displaced when using true displacement. Displacements over this threshold may cause visibility issues. These visibility issues can be observed when the object is out of view at the edge of screen with parts being displaced inside the view. The object would then disappear because of camera culling. This can also produce missing shadow updates where the displaced geometry is.

Transparent shadows

Use transparent shadows for this material if it contains a Transparent BSDF. Disabling will render faster but not give accurate shadows.

Render Method

Controls the blending and the compatibility with certain features.

Dithered:: Allows for grayscale hashed transparency, and compatible with render passes and raytracing. Also know as deferred rendering.
When using *Dithered* render method, the materials are rendered in layers. Each layer can only transmit (e.g. refract) light emitted from previous layers. If no intersection with the layers below exists, the transmissive BSDFs will fallback to light probes.

Raytraced Transmission

Use raytracing to determine transmitted color instead of using only light probes. This prevents the surface from contributing to the lighting of surfaces not using this setting.

Blended:: Allows the colored transparency, but incompatible with render passes and raytracing. Also known as forward rendering.

Sorting Problem
When using *Blended* render method, the order in which the color blending happens is important as it can change the final output color. EEVEE does not support per-fragment (pixel) sorting or per-triangle sorting. Only per-object sorting is available and is automatically done on all transparent surfaces based on object origin. Opaque surfaces (i.e. that have no transparency) will still have correct sorting regardless of the render method.

Tip
Face order can be adjusted in edit mode by using sort element or using a geometry node.

Note
Per-object sorting has a performance cost and having thousands of objects in a scene will greatly degrade performance.

Transparency Overlap
: If enabled, all transparent fragments will be rendered. If disabled, only the front-most surface fragments will be rendered. This option can be disabled to fix sorting issues caused by blending order. Only available for the *Blended* render method.

Thickness
: Determines what model to use to approximate the object geometry.

Sphere:: Approximate the object as a sphere whose diameter is equal to the thickness defined by the node tree. This is more suited to objects with rounder edges (e.g. a monkey head), and is perfectly suited to spheres.

Slab::

Approximate the object as an infinite slab of thickness defined by the node tree. This is more suited to very flat or thin objects (e.g. glass panels, grass blades).

From Shadow
: Use the shadow maps from shadow casting lights to refine the thickness defined by the material node tree. This takes the minimum thickness between the shadow map and the material node tree value. This is useful for objects where pre-computation is difficult (e.g. complex meshes), impossible (e.g. procedural geometry with displacement) or just impractical. However, this will have a performance impact that scale with the number of render samples.

Volume

Intersection
: Determines which inner part of the mesh will produce volumetric effect.

Fast:: Each face is considered as a medium interface. Gives correct results for manifold geometry that contains no inner part.

Accurate:: Faces are considered as medium interface only when they have different consecutive facing. Gives correct results as long as the max ray depth is not exceeded. Has significant memory overhead compared to the fast method.

Light Settings

Reference

Panel:: *Properties* ▸ *Light Shader Editor* ▸ *Sidebar* ▸ *Options*

Besides lighting from the background and materials with emission shaders, lights are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

See Light settings for settings common to all renderers.

Shadow

EEVEE uses a technique called *Virtual Shadow Mapping* along with *Shadow Map Raytracing*. *Virtual Shadow Mapping* produces more accurate results than traditional shadow mapping by putting resolution only where it is needed. It also includes a very efficient caching mechanism. This technique offers better performance than ray tracing and is compatible with any Render Method.

Tip

- The error message *Shadow buffer full* means that the system cannot allocate enough shadow memory. Increasing the Shadow Pool Size or the Resolution Limit on some lights can fix the issue. Otherwise, the only workaround is to disable shadow casting on some lights.
- *Shadow Map Raytracing* can be tweaked in the Render Settings.
- Turning on Jitter can reduce the light leaking artifacts caused by large lights and *Shadow Map Raytracing*.

See also

Limitations.

Jitter

Enable jittered soft shadows to increase shadow precision. Has a high performance impact as the shadow map cannot be cached and needs to be updated for each render sample.

Note

The effect isn't visible by default in the viewport. See render settings.

Overblur

Apply shadow tracing to each jittered sample to reduce under-sampling artifacts.

Note

Any value higher than zero will result in a blurrier shadow and is not physically correct.

Filter

Blur shadow aliasing using PCF with a circular kernel. The effective world scale of the filter depends on the shadow map resolution at the shadowed pixel position.

Note

Any value bigger than 1px will increase the chances of light leaking artifacts.

Resolution Limit

Minimum size of a shadow map pixel. Higher values use less memory at the cost of shadow quality. Higher values also speed-up rendering of heavy scenes. Each shadow is scaled depending on the shadowed pixel on screen. This can create very sharp shadows but also requires a lot of memory if the shadowed pixel is close to the camera. This property limits the maximum amount of detail that the shadow map can capture.

Note

Reducing the shadow map resolution will increase the chances of light leaking artifacts.

Absolute Resolution Limit

Limit the resolution at 1 unit from the light origin instead of relative to the shadowed pixel. This makes Resolution Limit act as a regular shadow map pixel size.

Hint

With this option enabled, the following equation can be used to set the *Resolution Limit* with a desired resolution:

\[resolution_limit = 2 * \sqrt{2} / resolution\]
The \(2 * \sqrt{2}\) refers to the unit cube diagonal and \(resolution\) refers to the desired resolution (e.g. 1024px).

Note

The setting Absolute Resolution Limit does not exist for Sun Light.

Influence

These parameters modulate the intensity of the light depending on the shader type. These are meant for artistic control, and any value other than 1.0 breaks PBR rules.

Diffuse
: Diffuse reflection intensity multiplier.

Glossy
: Glossy light intensity multiplier.

Transmission
: Transmission light intensity multiplier.

Volume Scatter
: Volume light intensity multiplier.

Custom Distance

If enabled, uses Distance as the custom attenuation distance instead of global Light Threshold. In order to avoid long setup times, this distance is first computed automatically based on a light threshold. The distance is computed at the light origin and using the inverse square falloff.

Distance
: Specifies where light influence will be set to 0.

Note

The setting Custom Distance does not exist for Sun Light.

See also

Global Light Threshold.

Light Probes

Light probe objects are used by EEVEE as support objects.

There are three different types of light probes. Each type of light probe records the lighting at a different resolution and frequency. Probes are used together to recover incoming light information when using ray tracing is not possible (either for performance or for technical limitations).

These types of objects are only useful for EEVEE (and by extension, the Material Preview mode).

Types

- Sphere
  - Properties
- Plane
  - Placement
  - Properties
- Volume
  - Properties

Light Probe Sphere

A light probe sphere records the light incomming from many directions at a single location.

They are used for smooth and semi-rough reflections. Sphere probes smoothly blend to light probe volume lighting for completely diffuse reflections.

If *Raytracing* is turned on, they are used as a fallback if a ray misses.

Note

In both usages, the light probe spheres are shadowed by light probe volume. This is done in order to reduce light leaking in shadowed areas and reduce the need to setup more light probe spheres.

Adjusting their resolution is done inside the Scene data panel.

The world also has an internal light probe sphere with a resolution that can be adjusted in the World data panel.

Properties

Reference

Panel:: *Object Data ▸ Probe*

Type
: Shape of the influence volume. Can be set to Sphere or Box.

Radius

A probe object only influences the lighting of nearby surfaces. This influence zone is defined by the size parameter and object scaling.

Falloff
: Percentage of the influence distance in which the influence of a probe fades linearly.

Capture

Note

In the viewport, capture only happens if an update is detected on the light probe data or position. For renders, the capture happens at the start of each frame.

Clipping
: Define the near and far clip distances when capturing the scene.

Custom Parallax

Reference

Panel:: *Object Data ▸ Custom Parallax*

By default, the influence volume is also the parallax volume. The parallax volume is a volume on which the recorded light is projected. It should roughly fit it surrounding area. In some cases it may be better to adjust the parallax volume without touching the influence parameters. In this case, enable the *Custom Parallax* and change the shape and radius of the parallax volume independently.

Viewport Display

Data
: Show the captured light using a reflective sphere of the given size.

Clipping
: Show the clipping distance in the 3D Viewport.

Influence
: Show the influence bounds in the 3D Viewport. The inner sphere is where the falloff starts.

Parallax
: Show the *Custom Parallax* shape in the 3D Viewport.

Light Probe Plane

A light probe plane records the light incoming from a single direction for all visible points on a plane. The specular reflection direction is the only one currently available.

This type of light probe is suited to smooth planar surfaces.

Each visible planar light probe increases the render time as the scene needs to be rendered for each of them.

Light probe planes only work when the ray tracing method is set to `Screen-Trace`. When enabled, they accelerate the tracing process and complete the missing data from the screen space ray tracing.

Note

Reflections and volumetrics are not supported inside Light probe planes.

Placement

If Backface Culling is not enabled, snapping the light probe plane to the planar surface will effectively capture the underside of the surface.

You can manually move the light probe plane above the surface enough for it to not appear in the capture. Alternatively you can disable the light probe visibility in the object visibility panel.

Properties

Reference

Panel:: *Object Data ▸ Probe*

Distance
: A probe object only influences the lighting of surfaces inside its influence zone. This influence zone is defined by the distance parameter and the object's scale.

  For light probe planes, the influence distance is the distance from the plane. Only surfaces whose normals are aligned with the Reflection Plane will receive the captured reflection.

Capture

Clipping Offset
: Define how far below the plane the near clip is when capturing the scene. Increasing this can fix reflection contact problems.

Viewport Display

Reference

Panel:: *Object Data ▸ Viewport Display*

Arrow Size
: Size of the arrow showing the reflection plane normal.

Capture
: Show the captured reflected image onto a fully reflective plane in the 3D Viewport.

Influence
: Show the influence bounds in the 3D Viewport.

Light Probe Volume

A volume probe records the light incomming from all directions at many locations inside a volume.

The light is then filtered and only the diffuse light is recorded. The capture point positions are visible as an overlay when the Irradiance Volume object is selected.

If an object is not inside any Irradiance Volume, or if the indirect lighting has not been baked, the world's diffuse lighting will be used to shade it.

Tip

- When lighting indoor environments, try to align grids with the room shape.
- Try not to put too much resolution in empty areas or areas with a low amount of lighting variation.
- Bad samples can be fixed by adding a smaller grid near the problematic area.
- Large scenes may require using many volumes with different level of details.

Properties

Reference

Panel:: *Object Data ▸ Probe*

Intensity
: Intensity factor of the recorded lighting. Making this parameter anything other than 1.0 is not physically correct. To be used for tweaking, animating or artistic purposes.

Sampling Bias

Normal Bias

Offset sampling of the irradiance grid in the surface normal direction to reduce light bleeding. Can lead to specular appearance of diffuse surface if set too high.

View Bias

Offset sampling of the irradiance grid in the viewing direction to reduce light bleeding. Can lead to view dependant result if set too high. Prefer this if camera is static.

Facing Bias

When set to zero, avoids capturing points behind the shaded surface to bleed light onto the shaded surface. This produces non-smooth interpolation when the capture resolution is high. Increasing this bias will make the interpolation smoother but also introduce some light bleeding.

Validity & Dilation

During the baking process, a validity score is assigned to each capture point. This score is based on the number of back-faces hit when capturing the incoming lighting. Only materials with *Single Sided* turned on for Light Probe Volumes will reduce the validity score.

Validity Threshold

Capture points with validity below this threshold will be ignored during lighting interpolation. This remove the influence of capture points trapped inside closed geometry, reducing the artifacts they produced.

Dilation Threshold

Capture points with validity below this threshold will have their data replaced using valid neighbors.

Radius

Radius in capture points in which to search for a valid neighbor.

Bake

Light probe volume light data is static and needs to be manually baked. Once baked, the data is stored inside the object data-block and can be moved, animated and linked between blender files.

Note

Baking uses the render visibility of the objects in the scene.

During baking, the scene is converted into a different representation to accelerate light transport. This representation can be very memory intensive and prevents baking if it cannot fit inside the GPU memory. There are a few way to deal with this issue:

- Larger scenes should be divided into smaller sections or use different level of details.
- Reduce *Surfel Resolution*.
- Turn off the light probe volume visibility option on objects that have little to no effect in the bake.

Tip

The internal scene representation can be inspected using the `Debug Value` 3, 4 and 5.

Resolution

Resolution X, Y, Z

Spatial resolution for volumetric light probes is determined per probe. The local volume is divided into a regular grid of the specified dimensions. The lighting will be captured for each cell in this grid.

Bake Samples

Number of ray directions to evaluate when baking. This increases the baking time proportionally to the size of the scene representation.

Surfel Resolution
: Number of surfels to spawn in one local unit distance. Higher values increase quality, but have a huge impact on memory usage.

Tip

A good value is twice the maximum *Resolution*.

Capture

Capture Distance
: Distance around the light probe volume that will be captured during the bake. A distance of 0 will only considered the inside of the volume.

World Contribution
: Bake incoming light from the world instead of just visibility for more accurate lighting, but lose correct blending to surrounding irradiance volumes.

Indirect Light Contribution
: Capture light bounces from light source.

Emission Contribution
: Capture emissive surfaces when baking.

Clamping

Direct Light
: Clamp incoming direct light. 0.0 disables direct light clamping. Here direct light refers to the light that bounces only once (from the light object) or light coming from emissive materials.

Indirect Light

Clamp incoming indirect light. 0.0 disables indirect light clamping. Here indirect light refers to the light that bounces off a surface after the first bounce (from the light object) or during the first bounce if the light comes from emissive materials.

Tip

Setting *Clamp Indirect* to a very non-zero value will effectively only record the first light bounce leading.

Offset

In order to reduce artifacts caused by bad capture point positioning, the bake process adjusts their location before capturing light. It moves the capture points slightly away from surrounding surfaces and tries to move them out of objects if they are not too far bellow the surface.

Surface Offset
: Distance to move the capture points away from surfaces.

Search Distance
: Distance to search for valid capture positions if the capture point is near the back-face of a single-sided object.

Note

Only materials with *Single Sided* turned on for Light Probe Volumes will move capture point position.

Viewport Display

Data
: Show the captured light using small diffuse spheres of the given size.

Influence

Show the influence bounds in the 3D Viewport. The inner sphere is where the falloff starts.

Clipping

Show the clipping distance in the 3D Viewport.

Limitations

- Limitations
  - Attributes and Properties
  - Cameras
  - Lights
  - Light Probes
  - Indirect Lighting
  - Shadows
  - Volumetrics
  - Depth of Field
  - Screen Space Effects
  - Shader Nodes
  - Memory Management
  - CPU Rendering
  - Multiple GPU Support
  - Headless Rendering
- Supported Nodes
  - EEVEE only Nodes
  - Other Nodes Support

Limitations

EEVEE's goal is to be an interactive render engine. Some features may not be there yet or may be impossible to implement into EEVEE's architecture without compromising performance.

Here is a rather exhaustive list of all the limitations you can expect while working with EEVEE.

Attributes and Properties

- Only 14 attributes from Geometry Nodes are supported in a material
- Only 8 custom object properties are supported in a material

Cameras

- Only perspective and orthographic projections are currently supported.

Lights

- Lights can only have one color and do not support light node trees.
- Unlike in Cycles, the Size of spot lights does not change the softness of the cone.
- The area light Beam spread option is not supported.

Light Probes

- EEVEE supports up to 128 active light probe spheres.
- EEVEE supports up to 16 active light probe planes inside the view frustum.
- Active light probe volumes must fit inside the Light Probes Volume Memory Pool.

Indirect Lighting

- Light probe capture does not support specular reflections. Specular energy is treated as diffuse.

Shadows

- *Shadow Map Raytracing* can produce light leaking because of overlapping shadow casters. This can be mitigated by using lower step count, enabling jitter, or reducing the light shape size.
- Thin objects (e.g. walls without thickness) might have light leaking on the shadowed side. This can be mitigated by making the object have some thickness or lowering Resolution Limit.

Volumetrics

- Only single scattering is supported.
- Volumetrics are rendered only for the camera "rays". They don't appear in reflections/refractions and probes.
- Volumetric shadowing only work in volumetrics. They won't cast shadows onto solid objects in the scene.
- Volumetric shadowing only work for volumes inside the view frustum.

Depth of Field

- Blended materials cannot be correctly handled by the post-processing blur, but will be correctly handled by the sample-based method. For this, you need to disable the post-process depth of field by setting the *Max Size* to 0.

Screen Space Effects

Ray-triangle intersection is not currently supported. Instead of this, EEVEE uses the depth buffer as an approximated scene representation. This reduces

the complexity of scene scale effects and enables a higher performance. However, only what is in inside the view can be considered when computing these effects. Also, since it only uses one layer of depth, only the front-most pixel distance is known.

These limitations creates a few problems:

- The screen space effects disappear when reaching the screen border. This can be partially fixed by using the *overscan* feature.
- Screen space effects lack deep information (or the thickness of objects). This is why most effects have a thickness parameter to control how to consider potential intersected pixels.
- Objects behind other objects (occluded) are not considered by these effects.
- Blended surfaces are not considered by these effects. They are not part of the depth prepass and do not appear in the depth buffer.
- Objects that a part of Holdout Collections will not be rendered with screen space effects.

Raytracing

- Blended materials and materials using raytrace refractions will not appear in dithered materials reflections.
- Blended materials are not compatible with raytracing.
- Only one refraction event is correctly modeled. An approximation of the second refraction event can be achieved using the Thickness workflow.
- Only dithered materials *not* using Raytrace Refractions can be refracted.

Shader Nodes

- All BSDF's are using approximations to achieve realtime performance so there will always be small differences between Cycles and EEVEE.
- Some utility nodes are not yet compatible with EEVEE.
- Certain combinations of BSDF's will result in more noise than others. This is the case when mixing Diffuse BSDF and Refraction BSDF.

- Displacement of flat shaded surfaces will split the mesh into triangles. See Displacement for a workaround.

See also

For a full list of unsupported nodes see Nodes Support.

Memory Management

In EEVEE, GPU Memory management is done by the GPU driver. In theory, only the needed textures and meshes (now referred as "the resources") for a single draw call (i.e. one object) needs to fit into the GPU memory.

So if the scene is really heavy, the driver will swap things in and out to make sure all objects are rendered correctly.

In practice, using too much GPU memory can make the GPU driver crash, freeze, or kill the application. So be careful of what you ask.

There is no standard way of estimating if the resources will fit into the GPU memory and/or if the GPU will render them successfully.

CPU Rendering

Being a rasterization engine, EEVEE only uses the power of the GPU to render. There is no plan to support CPU (software) rendering as it would be very inefficient. CPU power is still needed to handle high scene complexity as the geometry must be prepared by the CPU before rendering each frame.

Multiple GPU Support

There is currently no support for multiple GPU systems.

Headless Rendering

Headless rendering is not supported on headless Windows systems.

Supported Nodes

Most nodes are taken from Cycles. However, some features are missing and may (or may not) be implemented in EEVEE in the future.

See also

Shader Nodes.

EEVEE only Nodes

These nodes are only available if EEVEE is the active render engine. These nodes will not work in Cycles.

Shader to RGB

EEVEE supports the conversion of BSDF outputs into color inputs to make a wide variety of custom shading. This is supported using the Shader to RGB node. This node evaluates the lighting of the BSDFs connected to it just like a *Blended* material and inherits its limitation.

Specular BSDF

This node implements the specular workflow found in other render engines.

Other Nodes Support

If something is not listed here, it is supported.

Shader Nodes

In the general case, shader nodes should behave more or less like in Cycles. So be sure to check out the Cycles section of this manual for that.

See also

Materials.

Although most BSDFs are supported, many of them are approximations and are not feature complete.

Diffuse BSDF
: Roughness is not supported. Only Lambertian diffusion is supported.

Glass / Refraction BSDF
: Only supports GGX and Multiscatter GGX distribution. See Raytracing limitations.

Glossy BSDF
: Only supports GGX and Multiscatter GGX distributions.

Subsurface Scattering
: Random Walk sampling, IOR and Anisotropic are not supported.

Transparent BSDF
: Colored and additive transparency are only compatible with blended modes.

Translucent BSDF
: Does not diffuse the light inside the object. It only lights the object with reversed normals.

Principled BSDF
: Cumulative limitations from Diffuse BSDF, Glossy BSDF, Refraction BSDF and Subsurface Scattering. Anisotropy is not supported. The Sheen layer is a crude approximation.

Volume Absorption

See Volume Limitation.

Volume Scatter
: The anisotropy parameter will be mixed and averaged for all overlapping volumetric objects, which is not physically correct and differs from Cycles. Also see Volume Limitation.

Principled Volume
: Same as Volume Scatter. See Volume Limitation.

Holdout
: Partially supported, using dithered mode may give incorrect results.

Anisotropic BSDF
: Not supported.

Toon BSDF
: Not supported.

Hair BSDF
: Not supported.

Sheen BSDF
: Not supported.

Principled Hair BSDF
: Not supported.

Input Nodes

Ambient Occlusion
: The *Only Local* option is not supported.

Geometry
: Pointiness is not supported.

Random per Island
: Random per Island is not supported.

Attribute

Defaults to active UV layer. Only “density”, “color”, “flame” and “temperature” built-in Geometry attributes are supported. UVs and Color Attributes are supported. Only up to 8 Object or Instancer attributes per material (both types share the same limit), and 512 View Layer attributes per scene are supported.

Bevel

Not supported.

Curves Info

The Random output uses a different RNG algorithm. Range and statistical distribution of the values should be the same but the values will be different.

Light Path

EEVEE has no real concept of rays. But in order to ease the workflow between Cycles and EEVEE some of the outputs are only supported in particular cases. This node makes it possible to tweak indirect lighting in the shader.

- *Is Camera*: Supported.
- *Is Shadow*: Supported.
- *Is Diffuse*: Set to 1.0 when baking light probe volume. Otherwise is set to 0.0.
- *Is Glossy*: Set to 1.0 when baking light probe sphere or plane. Otherwise is set to 0.0.
- *Is Singular*: Not supported. Same as Is Glossy.
- *Is Reflection*: Not supported. Same as Is Glossy.
- *Is Transmission*: Not supported. Same as Is Glossy.
- *Ray Length*: Not supported. Defaults to 1.0.
- *Ray Depth*: Not supported. Defaults to 0.0.
- *Diffuse Depth*: Partially supported. Set to 1.0 when baking light probe volume. Otherwise is set to 0.0.
- *Glossy Depth*: Partially supported. Set to 1.0 when baking light probe sphere or plane. Otherwise is set to 0.0.
- *Transparent Depth*: Not supported. Defaults to 0.

- *Transmission Depth*: Not supported. Same as Glossy Depth.

Note

Is Glossy does not work with Screen Space Reflections/Refractions but does work with reflection planes (whether used with SSR or not).

Particle Info
: Not supported.

Texture Coordinate
: *From Instancer* is not supported.

UV Map
: *From Instancer* is not supported.

Wireframe
: Pixel size option does not give exactly the same output as Cycles. The width can be a bit different.

Texture Nodes

Most texture nodes are supported except for the exceptions listed below:

IES Texture
: Not supported.

Image Texture
: Smart Interpolation always uses Cubic interpolation. Artifact present using Tube or Sphere projection with linear interpolation. This is due to hardware mip-mapping and Anisotropic filtering. This kind of artifact will be also visible if the texture coordinates provided are not continuous. Using Box projection with *Extend type* set to Clip or Extend is not supported. Instead, it will always use Repeat.

Point Density
: Not supported.

Sky Texture
: In Nishita mode, the *Sun Disc* property is not supported.

Other Nodes

Light Falloff
: Not supported.

Cycles

- Introduction
- Render Settings
  - Grease Pencil
  - Sampling
  - Light Paths
  - Volumes
  - Subdivision
  - Curves
  - Simplify
  - Motion Blur
  - Film
  - Performance
- World Settings
- Object Settings
  - Object
  - Adaptive Subdivision
  - Cameras
- Material Settings
- Light Settings
- GPU Rendering
- Experimental Features
- Render Baking
- Optimizing Renders
  - Reducing Noise
  - Shader Nodes

Introduction



Cycles is Blender's physically-based path tracer for production rendering. It is designed to provide physically based results out-of-the-box, with artistic control and flexible shading nodes for production needs.

To use Cycles, select it as the *Render Engine* in the Render properties. For GPU accelerated rendering, enable compatible devices in *Preferences* ▸ *System* ▸ *Cycles Render Devices*.

See also

The Cycles website with more information and a gallery.

Render Settings

- Grease Pencil
- Sampling
  - Adaptive Sampling
  - Denoising
  - Path Guiding
  - Lights
  - Advanced
- Light Paths
  - Ray Types
  - Bounce Control
  - Transparency
  - Settings
- Volumes
- Subdivision
- Curves
  - Viewport Display
- Simplify
  - Viewport
  - Render
  - Culling
  - Grease Pencil
- Motion Blur
  - Shutter Curve
  - Limitations
- Film
  - Pixel Filter
  - Transparent
- Performance
  - Threads
  - Memory
  - Acceleration Structure
  - Final Render
  - Viewport

Grease Pencil

Reference

Panel:: *Render ▸ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold
: Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Sampling

Reference

Panel:: *Render ▸ Sampling*

The integrator is the rendering algorithm used to compute the lighting. Cycles currently supports a path tracing integrator with direct light sampling. It works well for various lighting setups, but is not as suitable for caustics and some other complex lighting situations.

Rays are traced from the camera into the scene, bouncing around until they find a light source such as a light, an object emitting light, or the world background. To find lights and surfaces emitting light, both indirect light sampling (letting the ray follow the surface BSDF) and direct light sampling (picking a light source and tracing a ray towards it) are used.

Viewport Samples
: Number of samples for viewport rendering. Setting this value to zero enables indefinite sampling of the viewport.

Render Samples
: Number of paths to trace for each pixel in the final render. As more samples are taken, the solution becomes less noisy and more accurate.

Time Limit
: Renders scene until time limit or sample count is reached. When the time is set to 0, the sample count is used to determine when the render stops.

> **Note**
>
> The time limit does not include pre-render processing time, only render time.

Adaptive Sampling

With adaptive sampling Cycles automatically reduces the number of samples in areas that have little noise, for faster rendering and more even noise distribution. For example hair on a character may need many samples, but the background may need very few.

With adaptive sampling it is also possible to render images with a target amount of noise. This is done by settings the *Noise Threshold*, typical values are in the range from 0.1 to 0.001. Then render samples can then be set to a high value, and the renderer will automatically choose the appropriate amount of samples.

Noise Threshold
: The error threshold to decide whether to continue sampling a pixel or not. Typical values are in the range from 0.1 to 0.001, with lower values meaning less noise. Setting it to exactly 0 lets Cycles guess an automatic value for it based on the total sample count.

Min Samples
: The minimum number of samples a pixel receives before adaptive sampling is applied. When set to 0 (default), it is automatically set to a value determined by the *Noise Threshold*.

Denoising

Denoising removes noise while previewing scenes in *Rendered* mode in the 3D Viewport or for final renders.

Render
: Denoising for the final render can be enabled or disabled with the checkbox. For denoising the image after rendering with the Denoising node, the Data Render Passes also adapt to the selected denoiser.

  OpenImageDenoise::

Uses Intel's Open Image Denoise, an AI denoiser. Typically provides the highest quality, and is the default.

OptiX:: Uses NVIDIA's OptiX AI denoiser. Supports GPU acceleration on some older NVIDIA GPUs where OpenImageDenoise does not.
Only available on NVIDIA GPUs when configured in the Cycles Render Device user preferences.

Viewport

Denoising for the *Rendered* mode in the 3D Viewport can be enabled or disabled for with the checkbox.

Automatic:: Uses GPU accelerated denoising if supported, for best performance. Prefers OpenImageDenoise over OptiX.

OpenImageDenoise::
Uses Intel's Open Image Denoise, an AI denoiser. Typically provides the highest quality.

OptiX:: Uses NVIDIA's OptiX AI denoiser. Supports GPU acceleration on some older NVIDIA GPUs where OpenImageDenoise does not.
Only available on NVIDIA GPUs when configured in the Cycles Render Device user preferences.

Passes

Controls which Render Pass the denoiser should use as input, which can have different effects on the denoised image. Generally, the more passes the denoiser has to denoise the better the result. It is recommended to at least use *Albedo* as *None* can blur out details, especially at lower sample counts.

None:: Denoises the image using color data.
Albedo:: Denoises the image using color and albedo data.

Albedo + Normal:: Denoises the image using color, albedo, and normal pass data.

Prefilter *OpenImageDenoise*

Controls whether or not prefiltering is applied to *Input Passes* for use when denoising. Visible only when using *OpenImageDenoise*.

None:: Does not apply any prefiltering to the input passes. This option retains the most detail and is the fastest, but assumes the input passes are noise free which may require a high sample count. If the input passes aren't noise free, then noise will remain in the image after denoising.

Fast:: Assumes the input passes are not noise free, yet does not apply prefiltering to the input passes. This option is faster than *Accurate* but produces a blurrier result.

Accurate:: Prefilters the input passes before denoising to reduce noise. This option usually produces more detailed results than *Fast* with increased processing time.

Quality *OpenImageDenoise*

Overall denoising quality. Visible only when using *OpenImageDenoise*.

High:: Produces the highest quality output at the cost of time.

Balanced:: Balanced between performance and quality.

Fast:: Produces an output fast at the cost of quality (ideal for viewport rendering).

Start Sample

Sample to start denoising in the 3D Viewport.

Use GPU

Perform denoising on the GPU. This is significantly faster than on CPU, but requires additional GPU memory. When large scenes need more GPU memory, this option can be disabled.

See GPU Rendering for details on supported GPU.

Path Guiding

Path guiding helps reduce noise in scenes where finding a path to light is difficult for regular path tracing, for example when a room is lit through a small door opening. Important light directions are learned over time, improving as more samples are taken. Guiding is supported for surfaces with diffuse BSDFs and volumes with isotropic and anisotropic scattering.

Note

- Path guiding is only available when rendering on a CPU.
- While path guiding helps render caustics in some scenes, it is not designed for complex caustics as they are harder to learn and guide.

Training Samples
: The maximum number of samples to use for training. A value of 0 will keep training until the end of the render. Usually 128 to 256 training samples is enough for accurate guiding. Higher values can lead to a minor increases in guiding quality but with increased render times.

Surface
: Enable path guiding for the diffuse and glossy components of surfaces.

Volume
: Enable path guiding inside volumes.

Lights

Light Tree
: Use a light tree to more effectively sample lights in the scene, taking into account distance and estimated intensity. This can significantly reduce noise, at the cost of a somewhat longer render time per sample.

Certain lighting properties are not accounted for in the light tree. This include custom falloff, ray visibility, and complex shader node setups including textures. This can result in an increase in noise in some scenes that make use of these features.

Note, this feature is currently disabled for AMD GPUs on macOS.

Light Threshold
: Probabilistically terminates light samples when the light contribution is below this threshold (more noise but faster rendering). Zero disables the test and never ignores lights. This is useful because in large scenes with many light sources, some lights might only contribute a small amount to the final image, and increase render times. Using this setting can decrease the render times needed to calculate the rays which in the end have very little effect on the image.

Advanced

Pattern
: The random sampling pattern used by the integrator.

Automatic::	Uses *Blue-Noise* (see below), but for viewport rendering, it optimizes for first sample quality for an interactive preview.
Classic::	Use pre-computed tables of Owen-scrambled Sobol for random sampling.
Blue-Noise::	Use a blue-noise pattern, which optimizes the frequency distribution of noise, for random sampling. This results in an output that appears smoother despite not being less noisy overall.

Seed
: Seed value for integrator to get different noise patterns.

  Use Animated Seed (clock icon)
  : Changes the seed for each frame. It is a good idea to enable this when rendering animations because a varying noise pattern is less

noticeable.

Sample Offset
: The number of samples to skip when starting render. This can be used to distribute a render across multiple computers then combine the images with `bpy.ops.cycles.merge_images`

Scrambling Distance
: These properties are not compatible with *Blue-Noise* sampling patterns.

  Automatic
  : Uses a formula to adapt the scrambling distance strength based on the sample count.

  Viewport
  : Uses the *Scrambling Distance* value for the viewport rendering. This will make the rendering faster but may cause flickering.

  Multiplier
  : Lower values Reduce randomization between pixels to improve GPU rendering performance, at the cost of possible rendering artifacts if set too low.

Min Light Bounces
: Minimum number of light bounces for each path, after which the integrator uses Russian Roulette to terminate paths that contribute less to the image. Setting this higher gives less noise, but may also increase render time considerably. For a low number of bounces, it is strongly recommended to set this equal to the maximum number of bounces.

Min Transparent Bounces
: Minimum number of transparent bounces (more specifically "passthroughs"). Setting this higher reduces noise in the first bounces, but can also be less efficient for more complex geometry like hair and volumes.

Layer Samples

When render layers have per layer number of samples set, this option specifies how to use them.

Use::	The render layer samples will override the set scene samples.
Bounded::	Bound render layer samples by scene samples.
Ignore::	Ignore render layer sample settings.

Light Paths

Reference

Panel:: *Render ▸ Light Paths*

Ray Types

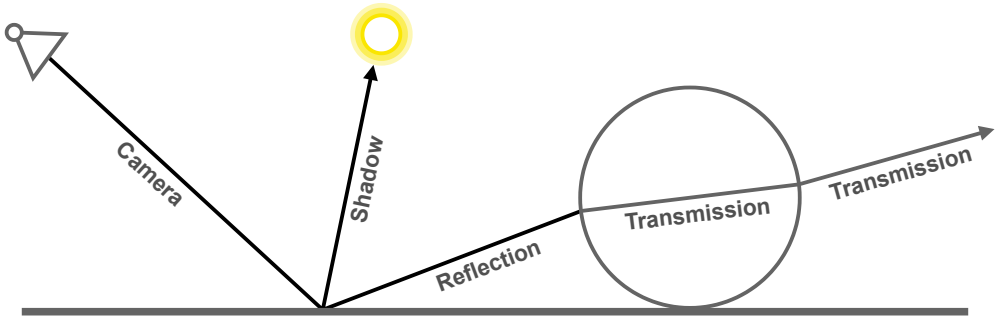
Ray types can be divided into four categories:

1. Camera: the ray comes straight from the camera.
2. Reflection: the ray is generated by a reflection off a surface.
3. Transmission: the ray is generated by a transmission through a surface.
4. Shadow: the ray is used for (transparent) shadows.

Reflection and transmission rays can further have these properties:

- Diffuse: the ray is generated by a diffuse reflection or transmission (translucency).
- Glossy: the ray is generated by a glossy specular reflection or transmission.
- Singular: the ray is generated by a perfectly sharp reflection or transmission.

The Light Path node can be used to find out the type of ray the shading is being computed for.



See also

The object ray visibility settings.

Bounce Control

The maximum number of light bounces can be controlled manually. While ideally this should be infinite, in practice a smaller number of bounces may be sufficient, or some light interactions may be intentionally left out for faster convergence. The number of diffuse reflection, glossy reflection and transmission bounces can also be controlled individually.

Light paths are terminated probabilistically when specifying a minimum number of light bounces lower than the maximum. In that case, paths longer than minimum will be randomly stopped when they are expected to contribute less light to the image. This will still converge to the same image, but renders faster while possibly being noisier.

Transparency

The Transparent BSDF shader is given special treatment. Rays pass straight through it, changing neither direction nor type as if there were no geometry at all.

Alpha pass output is also different for the transparent BSDF. Other transmission BSDFs are considered opaque, because they change the light

direction. As such they cannot be used for alpha-over compositing, while this is possible with the transparent BSDF.

Note that, while semantically the ray passes through as if no geometry was hit, rendering performance is affected as each transparency step requires executing the shader and tracing a ray.

Settings

Max Bounces

Total
: Maximum number of light bounces. For best quality, this should be set to the maximum. However, in practice, it may be good to set it to lower values for faster rendering. A value of 0 bounces results in direct lighting only.

Diffuse
: Maximum number of diffuse bounces.

Glossy
: Maximum number of glossy bounces.

Transmission
: Maximum number of transmission bounces.

Volume
: Maximum number of volume scattering bounces.

Transparent
: Maximum number of transparency bounces.

  Note, the maximum number of transparent bounces is controlled separately from other bounces. It is also possible to use probabilistic termination of transparent bounces, which might help rendering many layers of transparency.

Clamping

Direct Light

This option limits the maximum intensity a sample from rays which have not yet bounced can contribute to a pixel. It reduces noise at the cost of accuracy. Setting this option to 0.0 disables clamping altogether. Lower have a greater affect (dimmer samples) on the resulting image than higher values.

Note

This option provides a way to limit Fireflies. However, note that as you clamp out such values, other bright lights/reflections will be dimmed as well.

Care must be taken when using this setting to find a balance between mitigating fireflies and losing intentionally bright parts. It is often useful to clamp indirect bounces separately, as they tend to cause more fireflies than direct bounces. See the *Clamp Indirect* setting.

Indirect Light

The same as *Direct Light*, but for rays which have bounced multiple times.

Caustics

A common source of noise is Caustics.

See also

See Reducing Noise for examples of the clamp settings in use.

Filter Glossy

When using a value higher than 0.0, this will blur glossy reflections after blurry bounces, to reduce noise at the cost of accuracy. 1.0 is a

good starting value to tweak.

Some light paths have a low probability of being found while contributing much light to the pixel. As a result these light paths will be found in some pixels and not in others, causing Fireflies. An example of such a difficult path might be a small light that is causing a small specular highlight on a sharp glossy material, which is observed through a rough glossy material. In fact in such a case there practically occurs a caustic.

With path tracing it is difficult to find the specular highlight, but if you increase the roughness on the material, the highlight gets bigger and softer, and so easier to find. Often this blurring will hardly be noticeable, because it is blurred by the material anyway, but there are also cases where this will lead to a loss of detail in lighting.

Caustics

Reflective

While in principle path tracing supports rendering of caustics with a sufficient number of samples, in practice it may be inefficient to the point that there is just too much noise. This option can be unchecked, to disable reflective caustics.

Refractive

The same as above, but for refractive caustics.

Fast GI Approximation

Reference

Panel:: *Render ▸ Light Paths ▸ Fast GI Approximation*

Approximate diffuse indirect light with background tinted ambient occlusion. This provides fast alternative to full global illumination (GI), for interactive viewport rendering or final renders with reduced quality.

Method

Fast GI approximation method.

Replace:: Replace global illumination with ambient occlusion after a specified number of bounces.

Add:: Add ambient occlusion to diffuse surfaces.

AO Factor

The strength of the ambient occlusion.

AO Distance

Distance from shading point to trace rays. A shorter distance emphasizes nearby features, while longer distances make it also take objects farther away into account.

This option can also be overridden per object in the Object Properties, which is useful when you have both small and large scale objects in the same scene.

Viewport Bounces

Replace global illumination with ambient occlusion after the specified number of bounces when rendering in the 3D Viewport. This can reduce noise in interior scenes with little visual difference.

Render Bounces

Number of bounces when rendering final renders.

Volumes

Reference

Panel:: *Render ▸ Volumes*

Volume *Step* size is the distance between volume shader samples. Cycles automatically estimates this distance based on voxel size in volume objects and smoke simulations.

Render time can be reduced by increasing the step size, at the cost of potentially losing some volume detail. For procedural volume shaders that add detail, step size can be increased per object, material or world.

Step Rate Render
: Global multiplier on the step size for all volumes in renders. Increase to reduce render time, at the cost of losing detail.

Viewport
: Global multiplier on the step size for all volumes in the viewport. Increase for more responsive viewport rendering.

Max Steps
: Maximum number of steps through the volume before giving up, to protect from extremely long render times with big objects or small step sizes.

Subdivision

Reference

Panel:: *Render ▸ Subdivision*

Note

These settings are only available if Experimental Feature Set is turned on.

These settings are used to control Adaptive Subdivision.

Dicing Rate Render, Viewport
: Size of Micropolygons in pixels for the final/viewport render.

Offscreen Scale
: Multiplier for dicing rate of geometry outside of the camera view. The dicing rate of objects is gradually increased the further they are outside the camera view. Lower values provide higher quality reflections and shadows for off screen objects, while higher values use less memory.

Max Subdivisions
: Stop subdividing when this level is reached even if the dicing rate would produce finer Tessellation.

Dicing Camera
: Camera to use as reference point when subdividing geometry, useful to avoid crawling artifacts in animations when the scene camera is moving.

Curves

Reference

Panel:: *Render ▸ Curves*

These are global settings that apply to all instances of particle hair systems. The resolution of the strands is controlled by the step values in particle settings. Each hair system uses the material identified in the particle settings.

Shape

Rounded Ribbons:: Render curves as flat ribbon with rounded normals, for fast rendering. Curves are subdivided with a fixed number of specified subdivisions.

Curve Subdivisions

Number of subdivisions used in cardinal curve intersection (power of 2).

3D Curves:: Render curves as circular 3D geometry, for accurate results when viewing curves close up. Curves are automatically subdivided until the curve is smooth.

Viewport Display

These settings control the curve rendering settings used when the 3D viewport is set to Material Preview

Shape

Strand:: Render curves as a thin strand roughly a pixel wide. Curve diameter parameters are ignored with this setting.

Strip:: Render curves as a flat ribbon with rounded normals.

Additional Subdivisions

Additional subdivisions to be applied on top of the curve resolution set in the hair system settings. Increasing this value will smooth out the curves of the strands.

Simplify

Reference

Menu:: *Render ▸ Simplify*

Common Settings

Max Subdivision
: Maximum number of subdivision by the Subdivision Surface modifiers.

Child Particles
: Show only a subset of all child hairs and particles.

Texture Limit
: Automatically scales textures down so that they are no larger than the values chosen. This can help reduce computer memory resources when rendering large scenes with huge textures.

Viewport

See Common Settings above.

Volume Resolution
: Resolution percentage of volume objects in the viewport. This mostly affects memory usage rather than computation times.

Normals
: Skip computing custom normals and face corner normals for displaying meshes in the viewport.

Render

See Common Settings above.

Culling

Camera Cull
: Automatically culls objects based on the camera frustum defined by the *Margin*.

Distance Cull
: Automatically culls objects based on their distance from the active camera. This is set via the *Distance* property.

Grease Pencil

Playback Only
: Activates the simplification process only during animation playback.

Fill
: Shows the fill component in Grease Pencil materials.

Modifiers
: Shows Grease Pencil modifiers.

Shader Effects
: Shows Grease Pencil visual effects.

Layer Tinting
: Shows layers tint overrides.

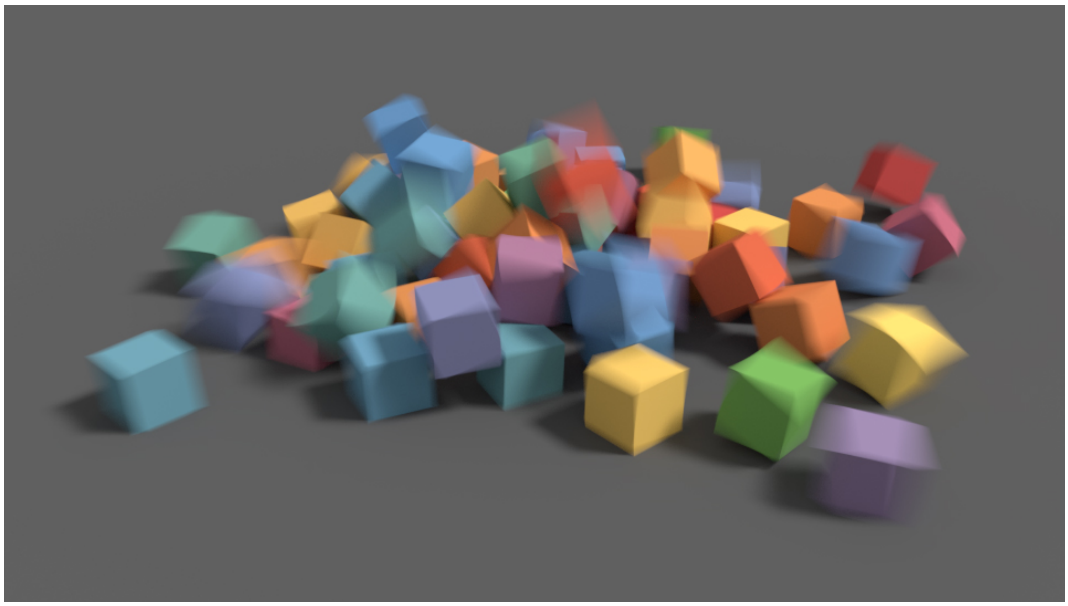
Anti-Aliasing
: Use Anti-Aliasing to smooth stroke edges. The amount of anti-aliasing can be adjusted by the Anti-Aliasing Threshold.

Motion Blur

Reference

Panel:: *Render ▸ Motion Blur*

Blender's animations are by default rendered as a sequence of *perfectly still* images. While great for stop-motion and time-lapses, this is unrealistic, since fast-moving objects do appear to be blurred in the direction of motion, both in a movie frame and in a photograph from a real-world camera.



Motion blur example. (blend-file)

Position

Controls at what point the shutter opens in relation to the current frame.

Start on Frame:: Shutter is starting to open at the current frame.
Center on Frame:: Shutter is fully opened at the current frame.
End on Frame:: Shutter is fully closed at the current frame.

Shutter

Time (in frames) between when the shutter starts to open and fully closed. For example, shutter time 1.0 blurs over the length of 1 frame.

Rolling Shutter
: Creates a Rolling Shutter effect.

None::	No rolling shutter effect.
Top-Bottom::	Renders rolling shutter from the top of the image to the bottom.

Rolling Shutter Duration
: Controls balance between pure rolling shutter effect (if the value is zero) and pure motion blur effect (if the value is one).

Note

If there are particles or other physics system in a scene, be sure to bake them before rendering, otherwise you might not get correct or consistent motion blur.

See also

Each object has its own settings to control motion blur. These options can be found in the Object tab of the Properties. See object setting for more information.

Shutter Curve

Curve defining how the shutter opens and closes. The X axis is time, Y values of 0 mean fully closed shutter, Y values of 1 mean fully opened shutter. The default mapping is set to when shutter opens and closes instantly.

Limitations

- Camera motion blur does not work for Orthographic Cameras.
- Motion blur does not take into account the movement of Lights.

Film

Reference

Panel:: *Render ▸ Film*

Exposure
: This can be used to change the brightness of an image. Different than the *Exposure* option found in the Color management panel, this exposure option works *on the data* while the Color management exposure is on the *view transform*.

Pixel Filter

Due to limited resolution of images and computer screens, pixel filters are needed to avoid Aliasing. This is achieved by slightly blurring the image to soften edges.

Type
: Pixel Filtering algorithm to use.

Box::	No filter.
Gaussian::	Smooth filter.
Blackman-Harris::	Default filter with a better balance between smoothness and detail preservation.

Width
: Lower values give more crisp renders, higher values are softer and reduce aliasing.

Transparent

Render the background transparent, for compositing the image over another background after rendering.

Transparent Glass

Render transmissive surfaces as transparent, for compositing glass over another background.

Roughness Threshold

For transparent glass, keep surfaces with roughness above the threshold opaque.

Performance

Reference

Panel:: *Render ▸ Performance*

Properties that affect render speeds or memory consumption. There are several presets available to help choose between different trade offs:

Default:: Balances memory saving and faster rendering settings.

Faster Render:: Uses settings to render faster at the cost of higher memory consumption.

Lower Memory:: Uses settings to decrease memory considered at the cost of slower renders.

Threads

Threads Mode

Method to determine the maximum number of CPU cores to use while rendering.

Auto-Detect:: Automatically chooses the amount of threads to match the number of logical processors on your computer.

Fixed:: Manually choose the maximum number threads to use for rendering. This can be useful for example, if you want to use your computer while rendering, you can set the property to a thread count lower the amount of logical processors on your computer.

Threads

The maximum number of CPU cores to use simultaneously while rendering.

Memory

Use Tiling
: Render high resolution images in tiles to reduce memory usage. Tiles are cached to disk while rendering to save memory.

Tile Size
: This value is used to control the size of the tile used for rendering; decreasing the size reduces memory usage.

> **Note**
>
> In some cases changing the *Tile Size* can result in increased performance. For example when a small object renders slowly compared to other objects, using a small *Tiles Size* can lead to an increase in performance.

Acceleration Structure

Use Spatial Splits
: Spatial splits improve the rendering performance in scenes with a mix of large and small polygons. The downsides are longer BVH build times and slightly increased memory usage.

Use Curves BVH
: Use a special type of BVH for rendering curves. The bounding boxes are not axis aligned allowing a spatially closer fit to the curve geometry. Disabling this option will reduce memory, at the cost of increasing curve render time.

BVH Time Steps
: Split BVH primitives by this number of time steps to speed up render time at the expense of memory.

Use Compact BVH
: Use a more compact BVH structure, which can reduce RAM usage but render slower.

Final Render

Persistent Data
: Keep render data in memory after rendering for faster re-renders and animation renders at the cost of extra memory usage while performing other tasks in Blender.

  When using multiple View Layers, only data from a single view layer is preserved to keep memory usage within bounds; however, objects shared between view layers are preserved.

Viewport

Pixel Size
: Option to control the resolution for viewport rendering. Allows you to speed up viewport rendering, which is especially useful for displays with high DPI.

World Settings

Mist Pass

Reference

Panel:: *World ▸ Mist Pass*

Note

The mist pass must be enabled in the View Layer tab of the Properties Editor before the settings below are available in the World tab.

Mist can greatly enhance the illusion of depth in your rendering. To create mist, Blender generates a render layer with a depth map ranging between 0.0 and 1.0 that can be used in the Compositor to generate a mist effect.

Start
: The distance from the camera at which the mist starts to fade in.

Depth
: The distance from *Start* of the mist, that it fades in over. Objects further from the camera than *Start* + *Depth* are completely hidden by the mist.

Falloff
: The curve function that controls the rate of change of the mist's strength further and further into the distance.

Quadratic:: Uses the same calculation as light falloff (\(1\over{x^2}\)) and provides the smoothest transition from transparent (0.0) to opaque (1.0).
Linear:: Has a steeper start than quadratic (\(1\over{x}\)).
Inverse Quadratic::

Has the steepest start (\(1\over{\sqrt{x}}\)) and approaches 1.0 faster than the other two functions.

Tip

A visualization can be activated in the *Camera* ▸ *Viewport Display* panel.



Mist example (blend-file).

Ray Visibility

Reference

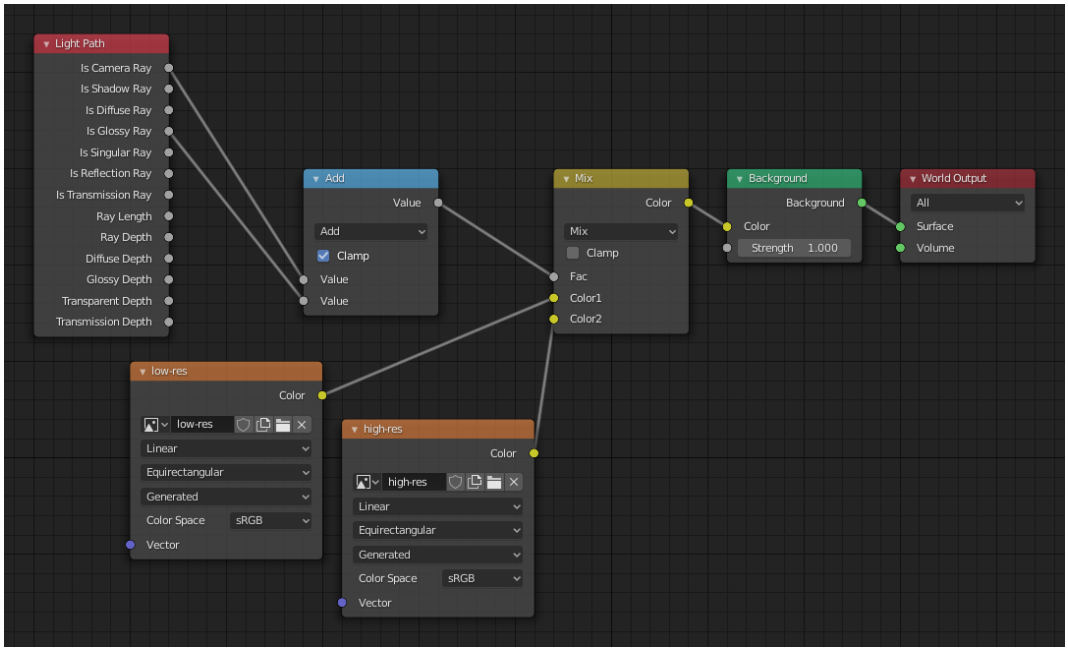
Panel:: *World* ▸ *Ray Visibility*

As with other objects, *Ray Visibility* allows you to control which other shaders can “see” the environment.

Tricks

Sometimes it may be useful to have a different background that is directly visible versus one that is indirectly lighting the objects. A simple solution to this is to add a Mix node, with the *Blend Factor* set to *Is Camera Ray*. The first input color is then the indirect color, and the second the directly visible color. This is useful when using a high-res image for the background and a low-res image for the actual lighting.

Similarly, adding the *Is Camera* and *Is Glossy* rays will mean that the high-res image will also be visible in reflections.



Nodes for the trick above.

Settings

Reference

Panel:: *World ▸ Settings*

Surface

Sampling

Controls the sampling method for the world material. Selecting Auto or Manual enables *Multiple Importance Sampling* while None disables it. *Multiple Importance Sampling* is a method to sample the background texture such that lighter parts are favored, creating an importance map. It will produce less noise in the render in trade of artifacts (Fireflies). Enable this when using an image texture with small area lights (like the sun), otherwise noise can take a long time to converge.

Below is a comparison between *Multiple Importance Sample* off and on. Both images are rendered for 25 seconds (Off: 1,500 samples, On: 1,000 samples).



Multiple Importance Sample off.



Multiple Importance Sample on.

Map Resolution

Sets the resolution of the importance map. A higher resolution will better detect small features in the map and give more accurate sampling but conversely will take up more memory and render slightly slower. Higher values also may produce less noise when using high-res images.

Max Bounces

Maximal number of bounces the background light will contribute to the render.

See also

See Reducing Noise for more information on how to reduce noise.

Shadow Caustics

Mark the World Shader as a refractive caustic caster. This setting can be used in conjunction with the Cast and Receive caustics object settings to selectively speed up refractive caustic rendering of select objects.

Volume

Sampling Method

Distance:: For dense volumes lit from far away *Distance* sampling is more efficient in most cases. Usually this shouldn't be used for World volumes.

Equiangular:: If you have got a light inside or near the volume then *equiangular* sampling is better.

Multiple Importance:: If you have a combination of both, then the multiple importance sampling will be better.

Interpolation

Interpolation method to use for the volume.

Linear:: Simple interpolation which gives good results for thin volumes.

Cubic:: Smoothed high-quality interpolation needed for more dense volumes, but slower.

Homogeneous

Assume volume has the same density everywhere (not using any textures), for faster rendering. Usually this is automatically determined by the renderer. This settings provides a manual control for cases where it is not detected.

Step Size

Distance between volume shader samples for world volume shaders. See Volume Render Settings for more information.

Light Group

Light Group *Cycles only*

Select the Light Group to add the current *World Surface Shader* too.

Add Light Group

If the name input into the *Light Group* field does not align with an existing Light Group, then pressing this button will create a *Light Group* with that name and assign this *World Shader* to it.

Object Settings

Settings for objects and object data.

- Object
  - Visibility
  - Motion Blur
  - Shading
- Adaptive Subdivision
  - Known Limitations
- Cameras
  - Panoramic Cameras

Object

Visibility

Reference

Panel:: *Object Properties* ▸ *Visibility*

See also

There are several other general visibility properties.

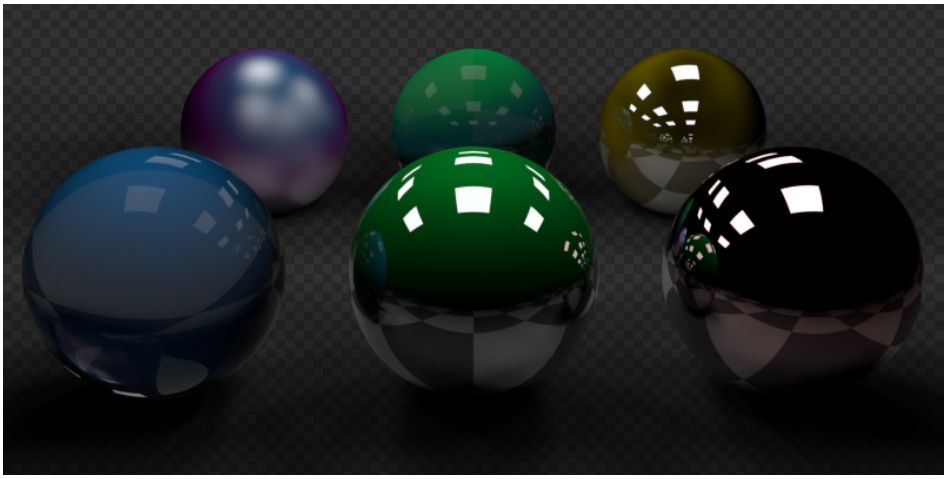
Mask

Shadow Catcher

Enables the object to only receive shadow rays. It is to be noted that, shadow catcher objects will interact with other CG objects via indirect light interaction. This simplifies compositing CGI elements into real-world footage.

Note

The *Shadow Catcher* outputs different results depending on if the *Shadow Catcher* pass is enabled in Render Layer settings. With the *Shadow Catcher* pass enabled, all indirect light interactions are captured. With it disabled, a simple approximation is used instead. The simple approximation is used in viewport rendering.



Example of the shadow catcher. Note how the material of the plane can still be viewed in the spheres.

Ray Visibility

Objects can be set to be invisible to particular ray types. This can be used, for example, to make an emitting mesh invisible to camera rays. For instanced objects, visibility is inherited; if the parent object is hidden for some ray types, the children will be hidden for these too.

In terms of performance, using these options is more efficient that using a shader node setup that achieves the same effect. Objects invisible to a certain ray will be skipped in ray traversal already, leading to fewer ray casts and shader executions.

Camera
: Makes the object visible to the Camera; this includes the viewport's perspective in viewport rendering.

Diffuse
: Makes the object visible in diffuse rays.

Glossy
: Makes the object visible in glossy rays.

Transmission
: Makes the object visible in transmission rays.

Volume Scatter
: Makes the object visible in volumetric scatter rays.

Shadow
: Enables the object to cast shadows.

Culling

In order to activate these options the respectively camera cull options have to be enabled in the scene simplify panel.

Use Camera Cull
: Ignore and this way make objects invisible to rays outside of the camera frustum.

Use Distance Cull
: Will cull any objects further from the camera than a given distance. When used in combination with camera frustum culling, this can be used to avoid culling nearby objects that are outside the camera frustum, but still visible in reflections. It is also useful to cull small objects far from the camera.

Motion Blur

Reference

Panel:: *Properties ▸ Object Properties ▸ Motion Blur*

Each object has its own motion blur settings along with the Scene Level Motion Blur. These settings can be found in the Object Properties tab of the Properties.

Steps
: Controls accuracy of deformation motion blur, more steps uses more memory. The actual number of time steps is \(2^{steps -1}\).

Deformation

Enables motion blur for deformed meshes such as animated characters, including hair.

Warning

An object modifier setup that changes mesh topology over time can not render deformation motion blur correctly. Deformation blur should be disabled for such objects. Common examples of this are animated Booleans, Deformation before Edge Split, Remesh, Skin or Decimate modifiers.

Shading

Reference

Panel:: *Properties* ▸ *Object Properties* ▸ *Shading*

Shadow Terminator

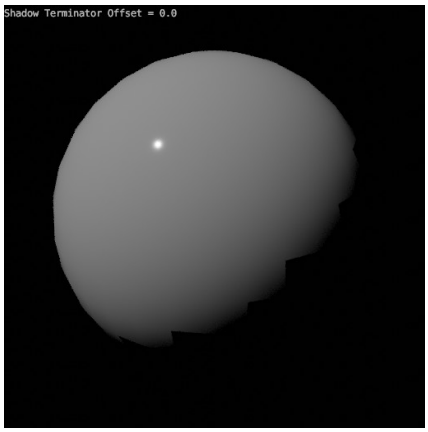
Geometry Offset

Offset rays from the surface to reduce shadow terminator artifacts on low-poly geometry. Higher values affect more triangles, a value of one affecting all triangles and zero having no affect. The default value only affects triangles at grazing angles to light and should eliminate most artifacts.

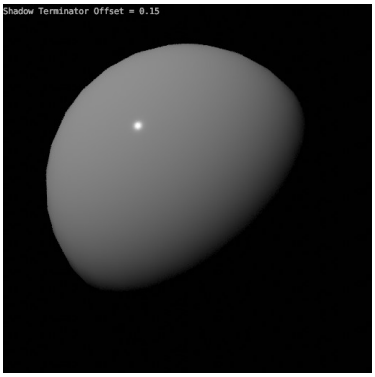
Unlike the *Shading Offset*, this option has little affect on the lighting making it the preferable method to handle shadow terminator artifacts.

Shading Offset

Pushes the shadow terminator (the line that divides the light and dark) towards the light to hide artifacts on low-poly geometry such as the ones below:



Shadow Terminator Artifacts.



Result of using an offset of 0.15.

Note

This property artificially alters the scene's lighting and is not energy conserving and consequently not physically accurate (see *Geometry Offset* instead).

Fast GI Approximation

AO Distance

Override for the world's AO Distance, if the value is zero the world's distance is used.

Caustics

Mark objects as caustic casters or receivers. This is used in conjunction with a Light or World Shader with *Shadow Caustics* enabled to selectively speed up caustic rendering of objects in your scene.

Note

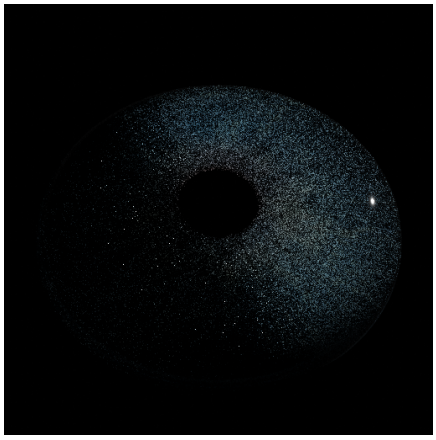
The rendering technique used to speed up the rendering of caustics is based on MNEE. There are a number of limitations with this technique

and it's implementation in Cycles:

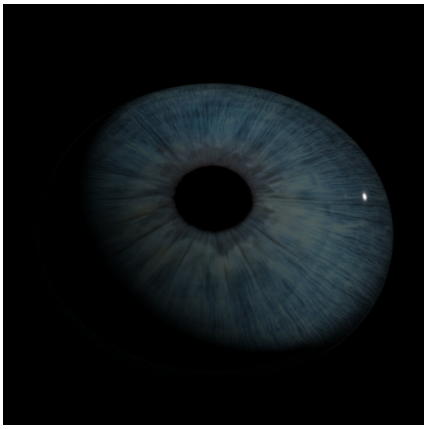
- Only refractive caustics in the shadows of objects work. Caustics from reflections or caustics that fall outside shadows are not rendered with this technique.
- MNEE Caustics are an approximation of caustics and will produce physically inaccurate results in many situations. Examples include incorrect brightnesses and the incorrect representation of caustics caused by rough or curved surfaces.
- In complex materials with multiple refractive BSDFs, MNEE will only generate caustics for one of the BSDFs.
- Filter Glossy settings are ignored when using MNEE for refractive caustics.
- MNEE Caustic rays can pass through up to 6 Caustic Caster surfaces between a Caustic Receiver and a Shadow Caustic light before the ray is terminated and caustics are ignored.
- The Ambient Occlusion and Bevel nodes will not produce a valid result on objects that are a Caustic caster or Caustic receiver while the scene contains a active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- MNEE Caustics only work if the caustic caster has smooth normals.
- Volumetric materials are not considered when calculating MNEE caustics.
- Bump and normal maps are ignored when calculating caustics.

Cast Shadow Caustics
: Mark an object as a caustic caster.

Receive Shadow Caustics
: Mark an object as a caustic receiver.



Rendering caustics inside an eye without MNEE at 32 samples per pixel.



Rendering caustics inside an eye using MNEE at 32 samples per pixel.

Light Group

Light Group
: Select the Light Group to add the current *Object* or *Light* too.

  Add Light Group
  : If the name input into the *Light Group* field does not align with an existing Light Group, then pressing this button will create a *Light Group* with that name and assign the selected *Object* or *Light* to it.

Light Linking

Limit light influence to specified objects, with Light Linking.

Receiver Collection
: Collection of objects that will receive light emitted from the object.

Shadow Linking

Limit shadows to specified objects, with Light Linking.

Shadow Blocker Collection

Collection of objects that will act as shadow blockers for light emitted from the object.

Adaptive Subdivision

Reference

Panel:: *Modifier ▸ Subdivision Surface*

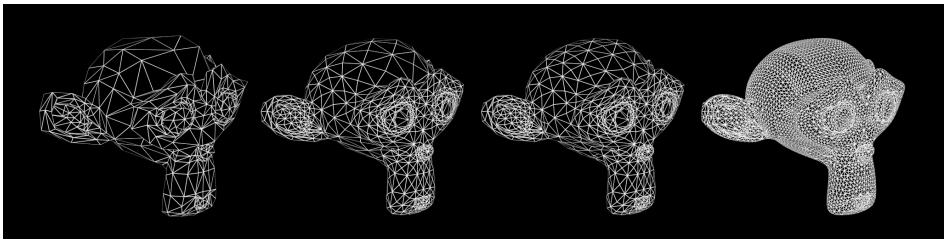
Note

Implementation not finished yet, marked as an Experimental Feature Set.

When using the *Experimental Feature Set* the Subdivision Surface Modifier gets changed to control the subdivision of a mesh at the time of rendering. For this, all the other settings are the same except the *View* and *Render* settings. These before mentioned settings get removed/renamed and the following settings are added:

Adaptive Subdivision
: Use OpenSubdiv to give different subdivision levels to near and far objects automatically. This allows nearer objects to get more subdivisions and far objects to get less.

  Dicing Scale
  : Multiplier of the scene dicing rate to determine the final size of Micropolygons in pixels.



Subdivision off/on, Dicing Scale: 1.0 - 0.3 - 0.05
(monkeys look identical in viewport, no modifiers).

Known Limitations

- Multi-user object data are currently made single users, leading to increased memory usage. For those it is better to use non-adaptive subdivision still.
- Multi-view renders can have some inconsistencies between views.

Warning

Instances are not tessellated individually. Instead, the original object is tessellated and then duplicated on all instances. To take advantage of both adaptive subdivision and instancing you should place the original object at the position of the instance that is closest from the camera.

Cameras

Panoramic Cameras

Cycles supports several types of panoramic cameras which are described in detail below. Note that these cannot be displayed in non-rendered modes in the viewport, i.e. *Solid* mode; they will only work for the final render.

Equirectangular

Render a panoramic view of the scenes from the camera location and use an equirectangular projection, always rendering the full 360° over the X axis and 180° over the Y axis.

This projection is compatible with the environment texture as used for world shaders, so it can be used to render an environment map. To match the default mapping, set the camera object rotation to (90, 0, -90) or pointing along the positive X axis. This corresponds to looking at the center of the image using the default environment texture mapping.

Latitude Min, Max
: Limits of the vertical field of view angles.

Longitude Min, Max
: Limits of the horizontal field of view angles.

Equiangular Cubemap Face

Improves on *Equirectangular* by providing a more uniform distribution of rendered pixel of the spherical environment. This results in an image that has little variation in visual resolution for the entire spherical projection. This is in contrast to *Equirectangular* which can lose detail in the poles of the image. This is also in contrast to cube map projections which lose detail near the edges of each face.

This panorama type is great for virtual reality use cases where providing as much visual detail for a limited resolution is important.

A limitation over *Equirectangular* is that this method does not have longitude or latitude limits.

Fisheye

Fisheye lenses are typically wide angle lenses with strong distortion, useful for creating panoramic images for e.g. dome projection, or as an artistic effect.

The *Fisheye Equisolid* lens will best match real cameras. It provides a lens focal length and field of view angle, and will also take the sensor dimensions into account.

The *Fisheye Equidistant* lens does not correspond to any real lens model; it will give a circular fisheye that does not take any sensor information into account but rather uses the whole sensor. This is a good lens for full-dome projections.

Lens
: Lens focal length in millimeters.

Field of View
: Field of view angle, going to 360 and more to capture the whole environment.

Fisheye Lens Polynomial

Match a real world camera by specifying the coordinates of a 4th degree polynomial.

The projection works as follows. Pixels in the image are mapped to positions (x, y) on the camera sensor in mm. A position on the sensor is mapped to a direction with spherical coordinates $(1, \theta, \phi)$ in radians as follows:

$$\begin{split}& r = \sqrt{x^2 + y^2}\\ & \theta = k_0 + k_1 r + k_2 r^2 + k_3 r^3 + k_4 r^4\\ & \phi = acos(x/r)\end{split}$$

Incoming light from this direction is then projected onto the corresponding pixel.

This can be used to model both fisheye and perspective cameras.

Mirror Ball

Render as if taking a photo of a reflective mirror ball. This can be useful in rare cases to compare with a similar photo taken to capture an environment.

Material Settings

Reference

Panel:: *Material* ▸ *Settings* and *Shader Editor* ▸ *Sidebar* ▸ *Settings*

Surface

Displacement
: Method used to perform Displacement on materials.

  Displacement Only:: Mesh vertices will be displaced before rendering, modifying the actual mesh. This gives the best quality results, if the mesh is finely subdivided. As a result, this method is also the most memory intensive.

  Bump Only:: When executing the surface shader, a modified surface normal is used instead of the true normal. This is a less memory intensive alternative to actual displacement, but only an approximation. Surface silhouettes will not be accurate and there will be no self-shadowing of the displacement.

  Displacement and Bump:: Both methods can be combined, to do displacement on a coarser mesh, and use bump mapping for the final detail.

Emission Sampling
: The method used for sampling the emissive component of the material. This option will only have an influence if the material contains an emissive material node, otherwise it will be ignored.

None:: Do not use this surface as a light for sampling.

Auto:: Automatically determine if the surface should be treated as a light for sampling based on emission intensity.

Front:: Treat only the front side of the surface as a light, useful for closed meshes whose interior is not visible.

Back:: Treat only the back side of the surface as a light for sampling.

Front and Back:: Treat surface as a light for sampling, emitting from both the front and back side.

Transparent Shadows

Use transparent shadows if it contains a Transparent BSDF, disabling will render faster but will not give accurate shadows.

Bump Map Correction

Applies corrections to solve shadow terminator artifacts caused by bump mapping.

Volume

Sampling Method

Distance:: For dense volumes lit from far away *Distance* sampling is usually more efficient.

Equiangular:: If you have got a light inside or near the volume then *equiangular* sampling is better.

Multiple Importance:: If you have a combination of both, then the multiple importance sampling will be better.

Interpolation

Interpolation method to use for the volume objects and smoke simulation grids.

Linear:: Simple interpolation which gives good results for thin volumes.

Cubic:: Smoothed high-quality interpolation needed for more dense volumes, but slower.

Homogeneous

Assume volume has the same density everywhere (not using any textures), for faster rendering. For example absorption in a glass object would typically not have any textures, and so the renderer can be set to avoid taking small steps to sample the volume shader. Usually this is automatically determined by the renderer. This setting provides a manual control for cases where it is not detected.

Step Rate

Adjust distance between volume shader samples for volume shaders. This is typically used to reduce the step size for procedural shaders that add more detail with procedural textures, when it is not captured by the default step size. See Volume Render Settings for more information.

Light Settings

Reference

Panel:: *Properties* ▸ *Light* and *Shader Editor* ▸ *Sidebar* ▸ *Options* ▸ *Light*

Next to lighting from the background and any object with an emission shader, lights are another way to add light into the scene. The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

Common

Light settings for all renderers.

Cycles

Max Bounces
: Maximum number of times light from the light is allowed to Bounce. Limited by scene-wide bounce settings.

Cast Shadow
: By disabling this option, light from lights will not be blocked by objects in between. This can speed up rendering by not having to trace rays to the light source.

Multiple Importance Sample
: By default lights use only direct light sampling. For area lights and sharp glossy reflections, however, this can be noisy, and enabling this option will enable indirect light sampling to be used in addition to reduce noise.

Shadow Caustics

Mark a light as a refractive caustic caster. This setting can be used in conjunction with the Cast and Receive caustics object settings to selectively speed up refractive caustic rendering of select objects.

Area Lights

Portals

Area lights can also function as light portals to help sample the environment light, and significantly reduce noise in interior scenes. Note that rendering with portals is usually slower, but as it converges more quickly, less samples are required.

Light portals work by enabling the *Portal* option, and placing areas lights in windows, door openings, and any place where light will enter the interior.

In outdoor scenes most rays do not bounce much and just fly off into the sky and therefore, light portals are not helpful for outdoor scenes.



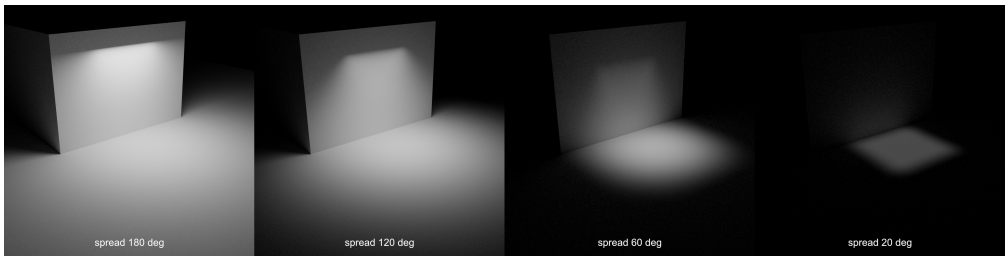


White Room model by Jay Hardy.

Beam Shape

Spread

How wide the emitted light fans out controlling how diffused the light is. Larger values create soft shadows while smaller values create sharper light simulating a gridded softbox.



Example of Spread at different angles.

GPU Rendering

GPU rendering makes it possible to use your graphics card for rendering, instead of the CPU. This can speed up rendering because modern GPUs are designed to do quite a lot of number crunching. On the other hand, they also have some limitations in rendering complex scenes, due to more limited memory, and issues with interactivity when using the same graphics card for display and rendering.

To enable GPU rendering, go into the *Preferences* ▸ *System* ▸ *Cycles Render Devices*, and select either *CUDA*, *OptiX*, *HIP*, *oneAPI*, or *Metal*. Next, you must configure each scene to use GPU rendering in *Properties* ▸ *Render* ▸ *Device*.

Rendering Technologies

Blender supports different technologies to render on the GPU depending on the particular GPU manufacturer and operating system.

CUDA – NVIDIA

CUDA is supported on Windows and Linux and requires a Nvidia graphics cards with compute capability 3.0 and higher. To make sure your GPU is supported, see the list of Nvidia graphics cards with the compute capabilities and supported graphics cards.

OptiX – NVIDIA

OptiX is supported on Windows and Linux and requires a Nvidia graphics cards with compute capability 5.0 and higher and a driver version of at least 495.89. To make sure your GPU is supported, see the list of Nvidia graphics cards.

OptiX takes advantage of hardware ray-tracing acceleration in RTX graphics cards, for improved performance.

GPU acceleration for OpenImageDenoise is available for compute capability 7.0 and higher, which includes all NVIDIA RTX cards.

HIP – AMD

HIP is supported on Windows and Linux and requires a AMD graphics card with the RDNA1 architecture or newer. Both discrete GPUs and APUs are supported.

Supported GPUs include:

- Radeon RX 5000 Series
- Radeon RX 6000 Series
- Radeon RX 7000 Series
- Radeon Pro W6000 Series

Minimum driver versions:

- Windows: Radeon Software 21.12.1 or Radeon PRO Software 21.Q4
- Linux: Radeon Software 22.10 or ROCm 5.3

Please refer to AMD's website for more information about AMD graphics cards and their architectures.

Experimental hardware ray-tracing support is available with the most recent drivers. This can be enabled in the preferences. However there are currently known issues regarding motion blur, hair and point cloud rendering, and degenerate triangle shapes.

GPU accelerated denoising is available on discrete Radeon RX 6000 and Radeon RX 7000 GPUs.

oneAPI – Intel

oneAPI is a computation library that is supported on Windows and Linux and requires a Intel® Arc™ graphics card with the Xe HPG architecture. Hardware acceleration for ray-tracing and denoising is supported.

Supported GPUs include:

- Intel® Arc™ A-Series

Minimum driver versions:

- Windows: Intel Graphics Driver XX.X.101.5518
- Linux: `intel-level-zero-gpu` package 1.3.27642, typically available through the `intel-compute-runtime` package XX.XX.27642.38

Please refer to Intel's website for more information about Intel graphics cards and their architectures.

GPU accelerated denoising is available on all supported GPUs.

Metal – Apple (macOS)

Metal is supported on Apple computers with Apple Silicon. macOS 13.0 or newer is required to support all features.

GPU accelerated denoising is available on Apple Silicon.

Limitations

- Path Guiding is not supported on any GPU.
- Open Shading Language is only supported for OptiX, with some limitations listed in the documentation.

Frequently Asked Questions

Why is Blender unresponsive during rendering?

On older GPU generations, graphics cards can only either render or draw the user interface. This can make Blender unresponsive while it is rendering. Heavy scenes can also make Blender unresponsive on newer GPUs, when using a lot of memory or executing expensive shaders, however this is generally less of a problem.

The only complete solution for this is to use a dedicated GPU for rendering, and another for display.

Why does a scene that renders on the CPU not render on the GPU?

There may be multiple causes, but the most common one is that there is not enough memory on your graphics card. Typically, the GPU can only use the amount of memory that is on the GPU (see Would multiple GPUs increase available memory? for more information). This is usually much smaller than the amount of system memory the CPU can access. With CUDA, OptiX, HIP and Metal devices, if the GPU memory is full Blender will automatically try to use system memory. This has a performance impact, but will usually still result in a faster render than using CPU rendering.

Can multiple GPUs be used for rendering?

Yes, go to *Preferences* ▸ *System* ▸ *Compute Device Panel*, and configure it as you desire.

Would multiple GPUs increase available memory?

Typically, no, each GPU can only access its own memory.

The exception is NVIDIA GPUs connected with NVLink, where multiple GPUs can share memory at a small performance cost. This is can be enabled with Distributed Memory Across Devices in the preferences.

What renders faster?

This varies depending on the hardware used. Different technologies also have different compute times depending on the scene tested. For the most up to date information on the performance of different devices, browse the Blender Open Data resource.

Error Messages

In case of problems, be sure to install the official graphics drivers from the GPU manufacturers website, or through the package manager on Linux. The graphics drivers provided by the computer manufacturer can sometimes be outdated or incomplete.

Error: Out of memory

This usually means there is not enough memory to store the scene for use by the GPU.

Note

One way to reduce memory usage is by using smaller resolution textures. For example, 8k, 4k, 2k, and 1k image textures take up respectively 256MB, 64MB, 16MB and 4MB of memory.

The NVIDIA OpenGL driver lost connection with the display driver

If a GPU is used for both display and rendering, Windows has a limit on the time the GPU can do render computations. If you have a particularly heavy scene, Cycles can take up too much GPU time. Reducing Tile Size in the Performance panel may alleviate the issue, but the only real solution is to use separate graphics cards for display and rendering.

Another solution can be to increase the time-out, although this will make the user interface less responsive when rendering heavy scenes. Learn More Here.

Unsupported GNU version

On Linux, depending on your GCC version you might get this error. See the Nvidia CUDA Installation Guide for Linux for a list of supported GCC versions. There are two possible solutions to this error:

Use an alternate compiler

If you have an older GCC installed that is compatible with the installed CUDA toolkit version, then you can use it instead of the default compiler. This is done by setting the `CYCLES_CUDA_EXTRA_CFLAGS` environment variable when starting Blender.

Launch Blender from the command line as follows:

```
CYCLES_CUDA_EXTRA_CFLAGS="-ccbin gcc-x.x" blender
```

(Substitute the name or path of the compatible GCC compiler).

Remove compatibility checks

If the above is unsuccessful, delete the following line in `/usr/local/cuda/include/host_config.h`:

```
#error -- unsupported GNU version! gcc x.x and up are not supported!
```

This will allow Cycles to successfully compile the CUDA rendering kernel the first time it attempts to use your GPU for rendering. Once the kernel is built successfully, you can launch Blender as you normally would and the CUDA kernel will still be used for rendering.

CUDA Error: Kernel compilation failed

This error may happen if you have a new NVIDIA graphics card that is not yet supported by the Blender version and CUDA toolkit you have installed. In this case Blender may try to dynamically build a kernel for your graphics card and fail.

In this case you can:

1. Check if the latest Blender version (official or experimental builds) supports your graphics card.
2. If you build Blender yourself, try to download and install a newer CUDA developer toolkit.

Normally users do not need to install the CUDA toolkit as Blender comes with precompiled kernels.

Experimental Features

Reference

Panel:: *Render*

Experimental features are disabled / hidden by default, but can be enabled by setting *Feature Set* to *Experimental* in the Render properties. Enabling the *Experimental Feature Set* will use experimental and incomplete features that might be broken or change in the future.

Adaptive subdivision is currently the only experimental feature.

Render Baking

Cycles shaders and lighting can be baked to image textures. This has a few different purposes, most commonly:

- Baking textures like base color or normal maps for export to game engines.
- Baking ambient occlusion or procedural textures, as a base for texture painting or further edits.
- Creating light maps to provide global illumination or speed up rendering in games.

Setup

Baking requires a mesh to have a UV map, and either a Color Attribute or an Image Texture node with an image to be baked to. The Active Image Texture node or Color Attribute is used as the baking target.

Use Render Bake in intensive light/shadow solutions, such as AO or soft shadows from area lights. If you bake AO for the main objects, you will not have to enable it for the full render, saving render time. Cycles uses the render settings (samples, bounces, …) for baking. This way the quality of the baked textures should match the result you get from the rendered scene.

Settings

Reference

Panel:: *Render ▸ Bake*

Bake
: Perform the baking operation.

Bake from Multires

Bake directly from multires object.

Bake Type
Type of pass to bake.

Combined:: Bakes all materials, textures, and lighting except specularity. The passes that contribute to the combined pass can be toggled individually to form the final map.

Ambient Occlusion:: Bakes ambient occlusion as specified in the World panels. Ignores all lights in the scene.

Shadow:: Bakes shadows and lighting.

Normal:: Bakes normals to an RGB image.

UV:: Mapped UV coordinates, used to represent where on a mesh a texture gets mapped too. This is represented through the red and green channels of the image, the blue channel is encoded with a constant value of 1 but does not hold any information.

Roughness:: Bakes the roughness pass of a material.

Emit:: Bakes Emission, or the Glow color of a material.

Environment:: Bakes the environment (i.e. the world surface shader defined for the scene) onto the selected object(s) as seen by rays cast from the world origin.

Diffuse:: Bakes the diffuse pass of a material.

Glossy:: Bakes the glossiness pass of a material.

Transmission:: Bakes the transmission pass of a material.

View From
Source of reflection ray directions.

Above Surface:: Cast rays from above the surface.

Active Camera:: Use the active camera's position to cast rays.

Influence

Combined

Lighting

Direct

Add direct lighting contribution.

Indirect

Add indirect lighting contribution.

Contributions

Diffuse

Add diffuse contribution.

Glossy

Add glossy contribution.

Transmission

Add transmission contribution.

Ambient Occlusion

Add ambient occlusion contribution.

Emit

Add emission contribution.

Diffuse, Glossy, Transmission

Contributions

Direct

See above.

Indirect

See above.

Color

Colorize the pass.

- If only *Color* is selected you get the pass color, which is a property of the surface and independent of sampling refinement.
- If *Color* is not selected, you get the direct and/or indirect contributions in gray-scale.
- If *Color* and either *Direct* or *Indirect* are selected, you get the direct and/or indirect contributions colored.

Normal

Space
: Normals can be baked in different spaces:

  For materials, the same spaces can be chosen in the image texture options next to the existing *Normal Map* setting. For correct results, the setting here should match the setting used for baking.

  Object:: Normals in object coordinates, independent of object transformation, but dependent on deformation.

  Tangent:: Normals in tangent space coordinates, independent of object transformation and deformation. This is the default, and the right choice in most cases, since then the normal map can be used for animated objects too.

Swizzle R, G, B
: Axis to bake into the red, green and blue channel.

Selected to Active

Bake shading on the surface of selected objects to the active object. The rays are cast from the low-poly object inwards towards the high-poly object. If the high-poly object is not entirely involved by the low-poly object, you can tweak the rays start point with *Max Ray Distance* or *Extrusion* (depending on whether or not you are using cage). For even more control you can use a *Cage Object*.

Note

There is a CPU fixed memory footprint for every object used to bake from. In order to avoid crashes due to lack of memory, the high-poly objects can be joined before the baking process.

Cage
: Cast rays to active object from a cage. A cage is a ballooned-out version of the low-poly mesh created either automatically (by adjusting the ray distance) or manually (by specifying an object to use). When not using a cage the rays will conform to the mesh normals. This produces glitches on the edges, but it is a preferable method when baking into planes to avoid the need of adding extra loops around the edges.

  Cage Object
  : Object to use as cage instead of calculating the cage from the active object with the *Cage Extrusion*.

Cage Extrusion
: Distance to use for the inward ray cast when using *Selected to Active* and *Cage*. The inward rays are cast from a version of the active object with disabled Edge Split Modifiers. Hard splits (e.g. when the Edge Split Modifier is applied) should be avoided because they will lead to non-smooth normals around the edges.

  Note

  When the base mesh extruded does not give good results, you can create a copy of the base mesh and modify it to use as a *Cage*. Both meshes need to have the same Topology (number of faces and face order).

Max Ray Distance
: Distance to use for the inward ray cast when using *Selected to Active*. Ray distance is only available when not using *Cage*.

Output

Target
: Where to output the baked map.

Image Textures:: Bake to the image data-block associated with the Active Image Texture node.
: Clear Image
: If selected, clears the image before baking render.

Active Color Attribute:: Bake to the Active Color Attributes layer on the active mesh. Note, the active object must be a mesh as other object types do not have Color Attributes.

Margin

When baking to images, by default a margin is generated around UV "islands". This is important to avoid discontinuities at UV seams, due to texture filtering and mip-mapping.

Type
: Method to generate the margin.

Extend:: Extend border pixels outwards.
Adjacent Faces:: Fill margin using pixels from adjacent faces across UV seams.

Size
: Size of the margin in pixels.

Optimizing Renders

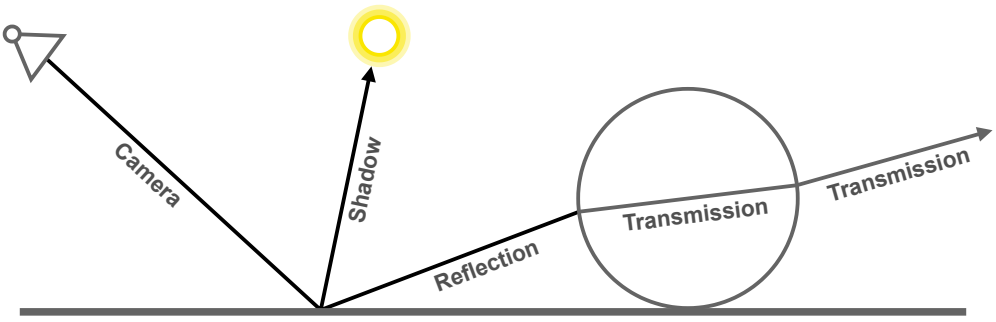
- Reducing Noise
  - Path Tracing
  - The Source of the Noise
  - Bounces
  - Caustics and Filter Glossy
  - Light Falloff
  - Multiple Importance Sampling
  - Glass and Transparent Shadows
  - Light Portals
  - Denoising
  - Clamp Fireflies
- Shader Nodes
  - Node Optimizations
  - Run-Time Optimizations
  - Open Shading Language

Reducing Noise

When performing a final render, it is important to reduce noise as much as possible. Here we will discuss a number of tricks that, while breaking the laws of physics, are particularly important when rendering animations within a reasonable time. Click to enlarge the example images to see the noise differences well.

Path Tracing

Cycles uses path tracing with next event estimation, which is not good at rendering all types of light effects, like caustics, but has the advantage of being able to render more detailed and larger scenes compared to some other rendering algorithms. This is because we do not need to store, for example, a photon map in memory, and because we can keep rays relatively coherent to use an on-demand image cache, compared to e.g. bidirectional path tracing.



We do the inverse of what reality does, tracing light rays from the camera into the scene and onto lights, rather than from the light sources into the scene and then into the camera. This has the advantage that we do not waste light rays that will not end up in the camera, but also means that it is difficult to find some light paths that may contribute a lot. Light rays will be sent either according to the surface BSDF, or in the direction of known light sources.

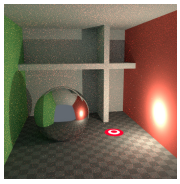
See also

For more details, see the Light Paths and Sampling documentation.

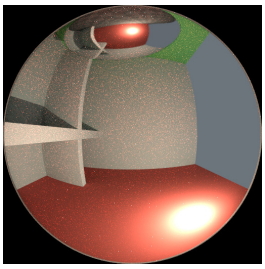
The Source of the Noise

To understand where noise can come from, take for example the scene below. When we trace a light ray into the location marked by the white circle on a red dot, the second image below gives an impression of what the diffuse shader “sees”.

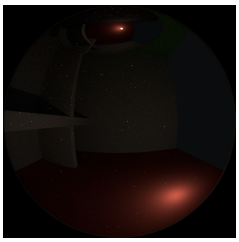
To find the light that is reflected from this surface, we need to find the average color from all these pixels. Note the glossy highlight on the sphere, and the bright spot the light casts on the nearby wall. These hot-spots are much brighter than other parts of the image and will contribute significantly to the lighting of this pixel.



The scene.



Irradiance at the shading point.

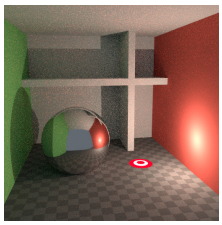


The detected highlights.

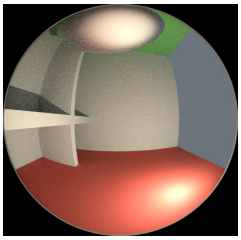
The light is a known light source, so its location is already known, but the glossy highlight(s) that it causes are a different matter. The best we can do with path tracing is to distribute light rays randomly over the hemisphere, hoping to find all the important bright spots. If for some pixels we miss some bright spot, but we do find it for another, that results in noise. The

more samples we take, the higher the probability that we cover all the important sources of light.

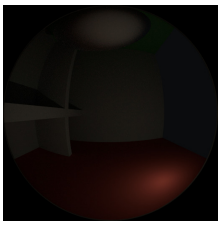
With some tricks we can reduce this noise. If we blur the bright spots, they become bigger and less intense, making them easier to find and less noisy. This will not give the same exact result, but often it's close enough when viewed through a diffuse or soft glossy reflection. Below is an example of using Glossy Filter and Light Falloff.



Using Glossy Filter & Light Falloff.



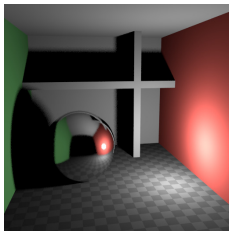
Irradiance at the shading point.



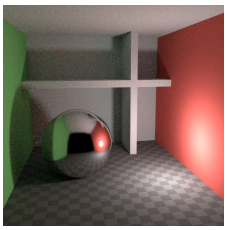
The detected highlights.

Bounces

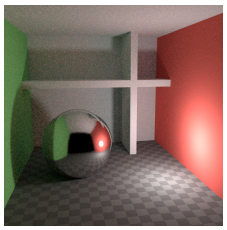
In reality light will bounce a huge number of times due to the speed of light being very high. In practice more bounces will introduce more noise, and it might be good to use something like the Limited Global Illumination preset in the Light Paths Section that uses *fewer* bounces for different shader types. Diffuse surfaces typically can get away with fewer bounces, while glossy surfaces need a few more, and transmission shaders such as glass usually need the most.



No bounces.



Two bounces at max.

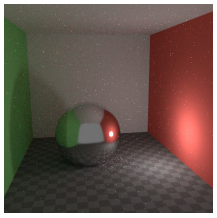


Four bounces at max.

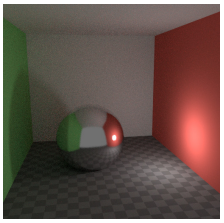
Also important is to use shader colors that do **not** have components of value 1.0 or values near that; try to keep the maximum value to 0.8 or less and make your lights brighter. In reality, surfaces are rarely perfectly reflecting all light, but there are of course exceptions; usually glass will let most light through, which is why we need more bounces there. High values for the color components tend to introduce noise because light intensity then does not decrease much as it bounces off each surface.

Caustics and Filter Glossy

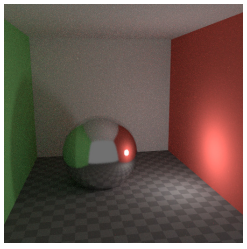
Caustics are a well-known source of noise, causing Fireflies. They happen because the renderer has difficulty finding specular highlights viewed through a soft glossy or diffuse reflection. There is a No Caustics option to disable glossy behind a diffuse reflection entirely. Many renderers will typically disable caustics by default.



Default settings.



Caustics disabled.



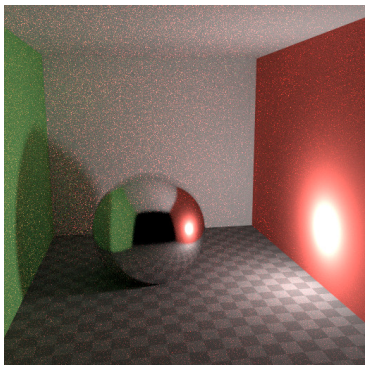
Filter Glossy greater than zero.

However, using *No Caustics* will result in missing light, and it still does not cover the case where a sharp glossy reflection is viewed through a soft glossy reflection. There is a Filter Glossy option to reduce the noise from such cases at the cost of accuracy. This will blur the sharp glossy reflection to make it easier to find, by increasing the shader Roughness.

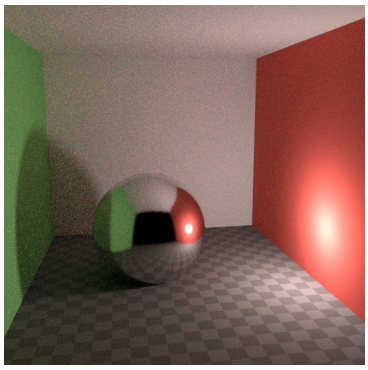
The above images show default settings, no caustics, and filter glossy set to 1.0.

Light Falloff

In reality light in a vacuum will always fall off at a rate of 1/(distance^2). However, as distance goes to zero, this value goes to infinity and we can get very bright spots in the image. These are mostly a problem for indirect lighting, where the probability of hitting such a small but extremely bright spot is low and so happens only rarely. This is a typical recipe for Fireflies.



Hard Falloff.



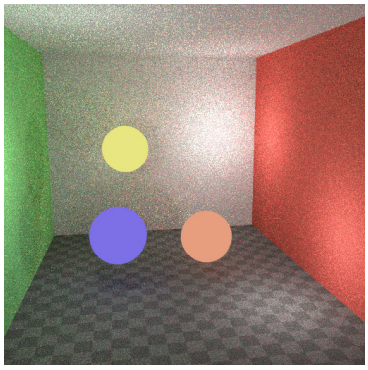
Soft Falloff.

To reduce this problem, the Light Falloff node has a *Smooth factor*, that can be used to reduce the maximum intensity a light can contribute to nearby surfaces. The images above show default falloff and smooth value 1.0.

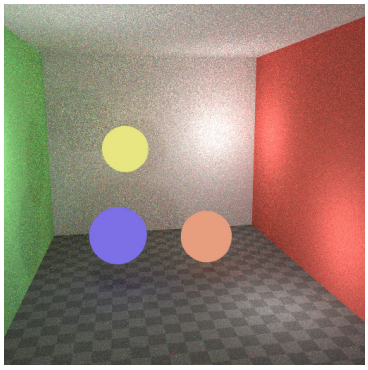
Multiple Importance Sampling

Materials with emission shaders can be configured to use Multiple Importance Sampling (Material Settings). This means that they will get rays sent directly towards them, rather than ending up there based on rays randomly bouncing around. For very bright mesh light sources, this can reduce noise significantly. However, when the emission is not particularly bright, this will take samples away from other brighter light sources for which it is important to find them this way.

The optimal setting here is difficult to guess; it may be a matter of trial and error, but often it is clear that a somewhat glowing object may be only contributing light locally, while a mesh light used as a light would need this option enabled. Here is an example where the emissive spheres contribute little to the lighting, and the image renders with slightly less noise by disabling Multiple Importance on them.



Multiple Importance off.

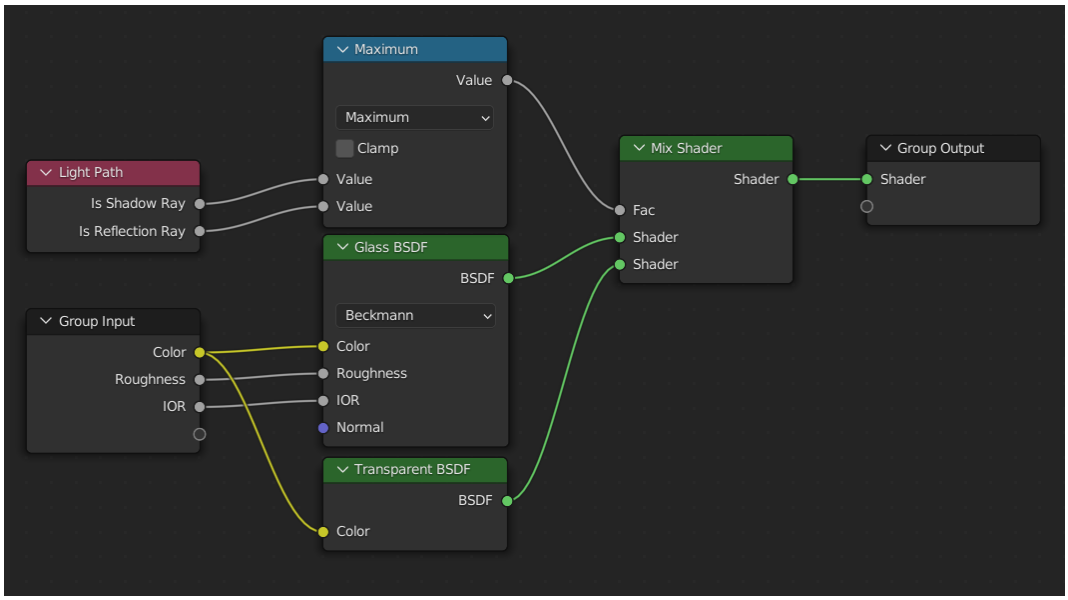


Multiple Importance on.

The world background also has a *Multiple Importance* (Settings) option. This is mostly useful for environment maps that have small bright spots in them, rather than being smooth. This option will then, in a preprocess, determine the bright spots, and send light rays directly towards them. Again, enabling this option may take samples away from more important light sources if it is not needed.

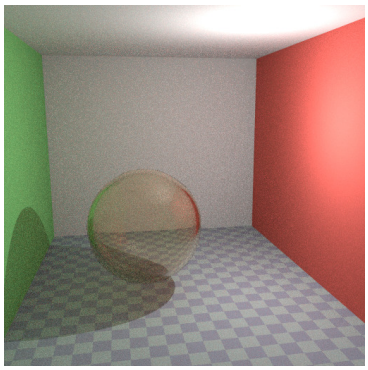
Glass and Transparent Shadows

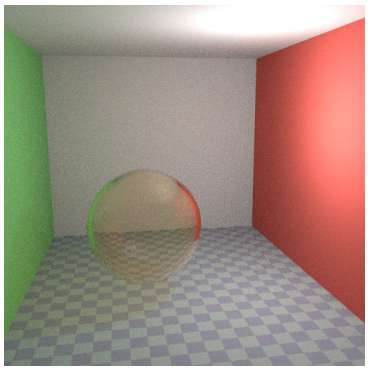
With caustics disabled, glass shadows may appear too dark, and with filter glossy the caustics might be too soft. We can make a glass shader that will use a Glass BSDF when viewed *directly*, and a Transparent BSDF when viewed *indirectly*. The Transparent BSDF can be used for transparent shadows to find light sources straight through surfaces, and will give properly-colored shadows, but without the caustics. The Light Path node is used to determine when to use which of the two shaders.



Optimized glass shader.

Above we can see the node setup used for the glass transparency trick; on the left the render has dark shadows due to missing caustics, and on the right the render with the trick.





Default Glass BSDF. Optimized Glass Shader.

Light Portals

When rendering a daylight indoor scene where most of the light is coming in through a window or door opening, it is difficult for the integrator to find its way to them. To fix this, use Light Portals. You then will need to modify its shape to match that of the opening that you are trying to fill.





Denoising

Even with all the settings described above there will always end up being some render noise no matter how many samples you use. To fix this there is a post-processing technique to cleanup the final bit of noise. To use this enable Denoising in the *Render* tab of the Properties.

Below is an example render by The Pixelary.



Example render before denoising.

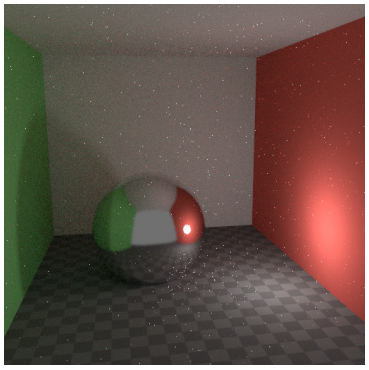


Example render after denoising.

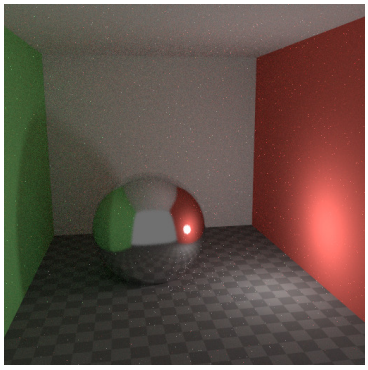
Clamp Fireflies

Ideally with all the previous tricks, Fireflies would be eliminated, but they could still happen. For that, the *intensity* that any individual light ray sample will contribute to a pixel can be *clamped* to a maximum value with the integrator Clamp setting.

If set too low this can cause missing highlights in the image, which might be useful to preserve for camera effects such as bloom or glare. To mitigate this conundrum it's often useful to clamp only indirect bounces, leaving highlights directly visible to the camera untouched.



No Clamp (0).



With Clamp set to 4.

Shader Nodes

Cycles applies a number of shader node optimizations both at compile time and run-time. By exploiting them it is possible to design complicated “Uber Shader” style node groups that incur minimal render time overhead for unused features.

Node Optimizations

As the first step in preparing a node shader for execution, Cycles expands all node groups, as if using the Ungroup tool, and discards UI only features like frames and reroute nodes.

After that, it applies some obvious transformations, for example, it can (the list is not exhaustive):

- Replace the following nodes with the constant result of their evaluation, if all their inputs are determined to be constant:

  RGB, Value, Mix RGB, Math, Vector Math, RGB to BW, Gamma, Bright Contrast, Invert, Separate/Combine RGB/XYZ/HSV, Blackbody, RGB Curves, Vector Curves, Color Ramps.

- Detect Mix RGB, Math and Vector Math nodes that become no-op (without Clamp) or evaluate to 0 as a result of addition, subtraction, multiplication, division or dot/cross product with a known constant 0 or 1 input, and replace with the appropriate input link or constant result.

- Eliminate Mix RGB Mix (without Clamp) and Mix Shader nodes when Factor is known to be 0 or 1 by replacing with the appropriate input value or link.

- Eliminate no-op Mix RGB (except Color Burn, Color Dodge, Lighten, or enabled Clamp), Invert, RGB Curves and Vector Curves nodes with

known zero Factor.

- Eliminate Emission and Background shader nodes that do not emit any light, and Add Shader nodes with one or both input arguments missing.
- Eliminate Bump with constant Height input, using its Normal input or Geometry Normal instead. This is useful for implementing node group inputs that default to normal via routing through a no-op Bump before doing math.
- Replace Attribute nodes of the View Layer type with the evaluated attribute value (it is constant within the whole Render Layer).
- Combine multiple copies of the same node with the same inputs into only one instance.

Finally, any nodes that end up not connected either directly or indirectly to the Output node are removed.

Run-Time Optimizations

When executing shaders, a special optimization is applied to Mix Shader nodes. If Factor evaluates to 0 or 1, any nodes that are only reachable via the unused branch of the mix are not evaluated.

This can substantially reduce the performance cost of combining multiple materials in one shader with a Color Attribute, texture, or other input used as a switch.

Open Shading Language

If Open Shading Language is chosen as the rendering backend, node shaders are translated to OSL code and then compiled and executed by the OSL runtime. In the process it applies its own extensive set of optimizations, both at compile time and run-time.

Open Shading Language can optimize out Script nodes if their outputs are unused or constant, even if their OSL shaders have side effects like debug tracing and message passing, which may be confusing. For that reason message passing with `setmessage` and `getmessage` should generally not be used for passing information forward in the graph; explicitly passing information through sockets should be preferred.

Workbench

- Introduction
- Performance
- Sampling
- Lighting
- Color
- Options
- Grease Pencil
- Viewport Display

Introduction

The Workbench Engine is a render engine optimized for fast rendering during modeling and animation preview. It is not intended to be a render engine that will render final images for a project. Its primary task is to display a scene in the 3D Viewport when it is being worked on.

Note

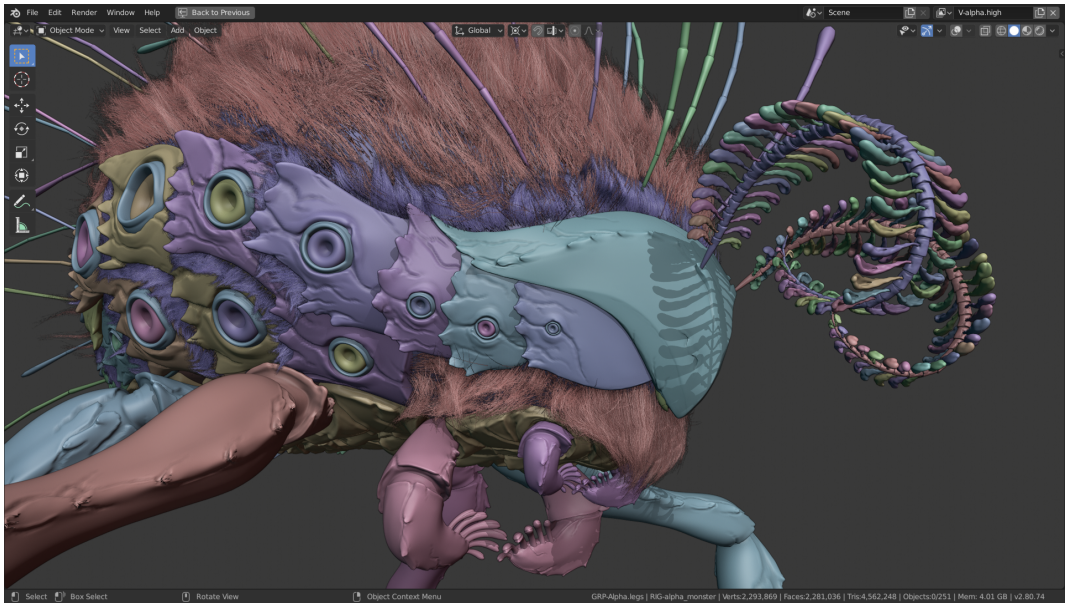
While it is not intended to be used for final renders, the Workbench render engine can be selected as the *Render Engine* in the Render properties.

By default the 3D Viewport uses Workbench to shade and light objects. Unlike other render engines such as EEVEE or Cycles, the Workbench engine does not use shader nodes. Instead, shading settings can be tweaked in the 3D Viewport's Shading popover or the render properties when doing final renders.

Workbench supports assigning random colors to objects to make each visually distinct. Other coloring mechanisms also exist, including; materials, Color Attributes, and textures.

Workbench also has an X-ray mode to see through objects, along with cavity and shadow shading to help display details in objects. Workbench supports several lighting mechanisms including studio lighting and MatCaps.

The image below is an excellent example of the Workbench engine's capabilities using random coloring and shadows to show the details of the model.



Workbench example.

Performance

Reference

Panel:: *Properties ▸ Render ▸ Performance*

High Quality Normals
: Uses higher precision normals and tangents which can improve visual quality for dense meshes with high frequency textures at the cost of memory.

Sampling

The quality of the renders can be adjusted by changing the Anti-Aliasing method. A different one can be selected for the 3D Viewport, viewport rendering and for final rendering.

The setting for the 3D Viewport is a user preference to specify the anti-aliasing method that runs best on the used system. The setting for viewport rendering and final rendering is saved per scene.

Reference

Panel:: *Render ▸ Sampling Preferences ▸ Viewport*

No Anti-Aliasing
: With this option selected no anti-aliasing will be applied.

Single Pass Anti-Aliasing
: Scene will be rendered with a post-process anti-aliasing pass.

Multisample
: The scene will be rendered multiple times with a slight offset. The anti-aliasing will be gathered from the multiple renders. The number of samples are predefined so it uses the best distribution of the samples.

  5, 8, 11, 16, 32

  Tip

  Multisample anti-aliasing is well suited for rendering small details like hair.

Progressive Viewport Rendering

For the 3D Viewport, one sample is rendered at a time. When there are no changes to the scene or viewport the next sample will be rendered.

Lighting

The Workbench engine does not use the lights of the scene. The lighting conditions that will be used can be set in the Lighting panel.

Reference

Panel:: *Properties* ▸ *Render* ▸ *Lighting*

Flat:: Objects are “shaded” in a flat color, without any hilights or shadows.

Studio:: Use a predefined studio lighting setup, such as a key light shining from the front and a rim light shining from the back. Click the sphere to choose a different setup.
The studio lights can be configured in the Preferences. By default, they follow the viewport camera around, but this can be changed:

World Space Lighting
: Keep the lights fixed in place rather than following the viewport camera.

Rotation
: The rotation of the lights on the Z axis.

MatCap:: Use a Material Capture, which is an image with texturing, lighting and even reflections baked into it. Objects are shaded by simply picking colors from this image based on the direction of the normal in relation to the camera.
Click the sphere to choose a different MatCap, or the double arrow button to flip it horizontally.
Custom MatCaps can be loaded in the Preferences.

Color

The colors that the Workbench uses to render objects can be changed.

Reference

Panel:: *Render ▸ Color*

Material:: Use the color that can be set per material in the Viewport Display Material panel.

Object:: Use the color that can be set per object in the Viewport Display Object panel.

Attribute:: Display the active Color Attribute of an object. When an object has no active Color Attribute it will be rendered in the color set in the Viewport Display Object panel.

Single:: Render the whole scene using a single color. The color can be chosen.

Random:: A random color will be selected for every object in the scene.

Texture:: Show the texture from the active Image Texture Node using the active UV map. If there is no active texture, the object will be rendered with the settings in the Material's Viewport Display panel.

Options

Reference

Menu:: *Properties* ▸ *Render* ▸ *Options* panel.

Backface Culling
: Use backface culling to hide backsides of faces.

X-Ray
: Render the scene transparent. With the slider you can control how transparent the scene should appear.

Shadow
: Renders a sharp shadow in the scene.

  Darkness
  : Defines how dark the shadow should be rendered. This slider can be adjusted between 0 (shadow not visible) and 1 (shadow is black).

  Light Direction
  : Controls the direction of the light source that casts the shadows.

  Shadow Shift
  : Controls the Shadow termination angle. It can be used to limit self shadowing artifacts.

  Shadow Focus
  : Controls the falloff near the edge of the shadow.

Cavity
: Highlight ridges and valleys in the scene geometry.

  Type
  : How to calculate cavities.

World:: More precise but is slower to calculate.
Screen:: Fast but does not take the size of the ridges and valleys into account.
Both:: Use both methods.

Ridge
: Control the visibility of ridges.

Valley
: Control the visibility of valleys.

Depth of Field
: Use the Depth of Field settings of the active camera in the viewport. Only visible when looking through the camera.

  The settings are located on *Properties* ▸ *Camera* ▸ *Depth of Field* panel.

Outline
: Render the outline of objects in the viewport. The color of the outline can be adjusted.

Specular Highlighting
: Render specular highlights.

  Note

  Only available when Lighting is set to *Studio* lighting or when a MatCap has been selected that contains a specular pass.

Grease Pencil

Reference

Panel:: *Render ▸ Grease Pencil*

This panel is comprised of settings to control the rendering of Grease Pencil Lines.

Anti-Aliasing Threshold
: Threshold for the edge detection algorithm used to correct aliasing, higher values might over blur some part of the image.

Viewport Display

The Workbench engine does not work with shader trees. In various tabs of the Properties are Viewport Display panels where settings can be adjusted that the Workbench engine uses.

Object

The Viewport Display panel in the Object Properties has several settings that are used by the Workbench Engine.

Reference

Panel:: *Properties* ▸ *Object* ▸ *Viewport Display*

Shadow
: When the *Shadow* in the Options is enabled this object will cast a shadow.

In Front
: When checked the object will be rendered in front of the other objects in the scene.

Color
: The color to render the object in when object color needs to be rendered. The alpha channel can be used to render the object transparent.

Material

The Viewport Display panel in the Material Properties has several settings that are used by the Workbench Engine.

Reference

Panel:: *Properties ▸ Material ▸ Viewport Display*

Color
: The color when rendering the material. The alpha channel can be used to render the object transparent.

Metallic
: Changes the amount of specular lighting. This is only available when *Specular Lighting* in the Options is enabled.

Roughness
: Changes the amount of roughness for specular lighting. This is only available when *Specular Lighting* in the Options is enabled.

World

The Viewport Display panel in the World Properties has several settings that are used by the Workbench Engine.

Reference

Panel:: *Properties ▸ World ▸ Viewport Display*

Color
: The color of the world background. This color will be rendered in the background of the scene.

Cameras

A camera is an object that provides a means of rendering images from Blender. It defines which portion of a scene is visible in the rendered image.

Cameras are invisible in renders, so they do not have any material or texture settings. However, they do have *Object* and *Editing* setting panels available which are displayed when a camera is the active object.

See also

3D Viewport Camera Navigation for documentation about managing cameras in the viewport.

Properties

Reference

Mode:: Object Mode
Editor:: *Properties* ▸ *Camera*

Lens

Type

The camera lens options control the way 3D objects are represented in a 2D image.

Perspective
: This matches how you view things in the real world. Objects in the distance will appear smaller than objects in the foreground, and parallel

lines (such as the rails on a railroad) will appear to converge as they get farther away.

Focal Length/Field of View

The Focal Length controls the amount of zoom, i.e. the amount of the scene which is visible all at once. Longer focal lengths result in a smaller FOV (more zoom), while short focal lengths allow you to see more of the scene at once (larger FOV, less zoom).



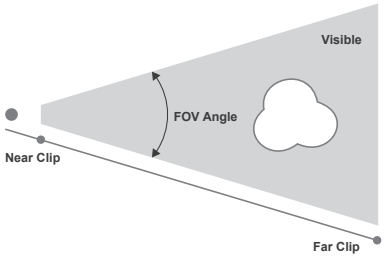
Perspective camera with 35 mm focal length.



Perspective camera with 210 mm focal length instead of 35 mm.

Lens Unit

The focal length can be set either in terms of millimeters or the actual Field of View as an angle.



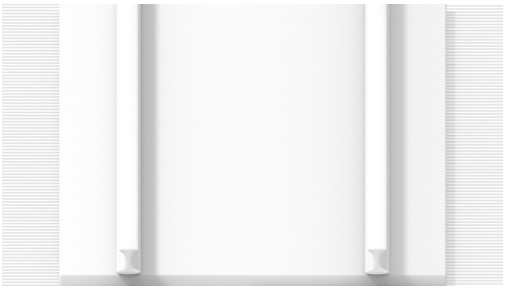
Hint

While the camera is moving towards an object the *Focal Length* property can be decreased to produce a *Dolly Zoom* camera effect, or vice versa.

This video demonstrates the *Dolly Zoom* camera effect.

Orthographic

With *Orthographic* perspective objects always appear at their actual size, regardless of distance. This means that parallel lines appear parallel, and do not converge like they do with *Perspective*.

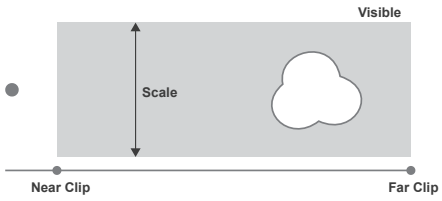


Render from the same camera angle as the previous examples, but with orthographic perspective.

Orthographic Scale

This controls the apparent size of objects projected on the image.

Note that this is effectively the only setting which applies to orthographic perspective. Since parallel lines do not converge in orthographic mode (no vanishing points), the lens shift settings are equivalent to translating the camera in the 3D Viewport.



Panoramic

Panoramic cameras only work in Cycles. See the Cycles panoramic camera settings for more information.

Shift

Allows for the adjustment of *vanishing points*. *Vanishing points* refer to the positions to which parallel lines converge. In these render examples, the most obvious vanishing point is at the end of the railroad.



Horizontal lens shift of 0.330.



Rotation of the camera object instead of a lens shift.

Notice how the horizontal lines remain perfectly horizontal when using the lens shift, but do get skewed when rotating the camera object.

Note

Using lens shift is equivalent to rendering an image with a larger FOV and cropping it off-center.

Clip Start and End

The interval in which objects are directly visible. Any objects outside this range still influence the image indirectly, as further light bounces are not clipped.

Note

For viewport rendering, setting clipping distances to limited values is important to ensure sufficient rasterization precision. Ray tracing renders do not suffer from this issue so much, and as such more extreme values can safely be set.

Tip

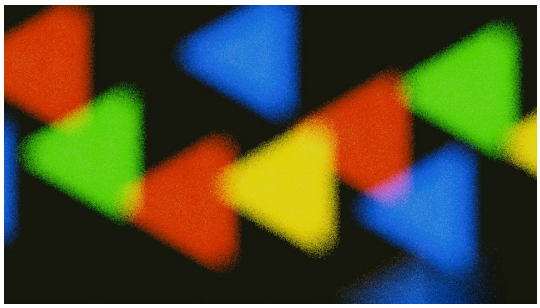
When *Limits* in the *Viewport Display* panel is enabled, the clip bounds will be visible as two yellow connected dots on the camera's line of sight.

See also

- 3D Viewport clipping

Depth of Field

Real-world cameras transmit light through a lens that bends and focuses it onto the sensor. Because of this, objects that are a certain distance away are in focus, but objects in front and behind that are blurred.



Example of DOF bokeh effect.

The area in focus is called the *focal point* and can be set using either an exact value, or by using the distance between the camera and a chosen object:

Focus Object
: Choose an object which will determine the focal point. Linking an object will deactivate the distance parameter.

Focal Distance
: Sets the distance to the focal point when no *Focus Object* is specified. This distance can be visualized in the 3D Viewport by enabling *Limits* in the camera's Viewport Display panel.

Hint

Use the eyedropper icon or hover the mouse over the *Focal Distance* property and press `E` to enable the depth *Depth Picker*. Then `LMB` on a point in the 3D Viewport to sample the distance from that point to the camera.

Aperture

F-Stop
: F-Stop ratio that defines the amount of blurring. Lower values give a strong depth of field effect.

Blades
: Total number of polygonal blades used to alter the shape of the blurred objects in the render, and render preview. As with the viewport, the minimum amount of blades to enable the bokeh effect is 3, resulting in a triangular-shaped blur.

Rotation
: Rotate the polygonal blades along the facing axis, and will rotate in a clockwise, and counter-clockwise fashion.

Ratio
: Change the amount of distortion to simulate the anamorphic bokeh effect. A setting of 1.0 shows no distortion, where a number below 1.0 will cause a horizontal distortion, and a higher number will cause a vertical distortion.

Camera

These settings adjusts properties that relate to a physical camera body. Several Presets can be chosen to match real-world cameras.

Sensor Fit

Adjusts how the camera's sensor fits within the outputs dimension adjusting the angular field of view.

Auto::	Calculates a square sensor size based on the larger of the Resolution dimensions.
Horizontal::	Manually adjust the *Width* of the sensor, the *Height* is calculated based on the aspect ratio of the output's Resolution.
Vertical::	Manually adjust the *Height* of the sensor, the *Width* is calculated based on the aspect ratio of the output's Resolution.

Size / Width, Height
: This setting is an alternative way to control the field of view, as opposed to modifying the focal length. It is useful to match a camera in Blender to a physical camera and lens combination, e.g. for motion tracking.

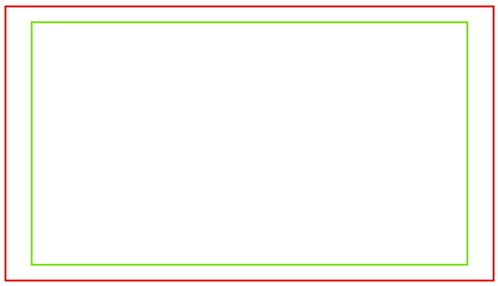
Safe Areas

Safe areas are guides used to position elements to ensure that the most important parts of the content can be seen across all screens.

Different screens have varying amounts of Overscan (especially older TV sets). That means that not all content will be visible to all viewers, since parts of the image surrounding the edges are not shown. To work around this problem TV producers defined two areas where content is guaranteed to be shown: action safe and title safe.

Modern LCD/plasma screens with purely digital signals have no Overscan, yet safe areas are still considered best practice and may be legally required for broadcast.

In Blender, safe areas can be set from the Camera and Sequencer views.



Red line: Action safe. Green line: Title safe.

The Safe Areas can be customized by their outer margin, which is a percentage scale of the area between the center and the render size. Values are shared between the Video Sequence editor and camera view.

Title Safe Margins X/Y
: Also known as *Graphics Safe*. Place all important information (graphics or text) inside this area to ensure it can be seen by the majority of viewers.

Action Safe Margins X/Y
: Make sure any significant action or characters in the shot are inside this area. This zone also doubles as a sort of "margin" for the screen which can be used to keep elements from piling up against the edges.

Tip

Each country sets a legal standard for broadcasting. These include, among other things, specific values for safe areas. Blender defaults for safe areas follow the EBU (European Union) standard. Make sure you are using the correct values when working for broadcast to avoid any trouble.

Center-Cut Safe Areas

Center-cuts are a second set of safe areas to ensure content is seen correctly on screens with a different aspect ratio. Old TV sets receiving `16:9` or `21:9` video will cut off the sides. Position content inside the center-cut areas to

make sure the most important elements of your composition can still be visible in these screens.

Blender defaults show a `4:3` (square) ratio inside `16:9` (widescreen).



Cyan line: action center safe. Blue line: title center safe.

Background Images

A background picture in your camera can be very helpful in many situations: modeling is obviously one, but it is also useful when painting (e.g. you can have reference pictures of faces when painting textures directly on your model…), or animation (when using a video as background), etc.

Background Source

The source of the background image.

Image:: Use an external image, image sequence, video file or generated texture.

Movie Clip:: Use one of the Movie Clip data-blocks.

Active Clip

Display a Movie Clip from the scene's Active Clip.

Render Undistorted

Display the background image using undistorted proxies when available.

Proxy Render Size

Select between full (non-proxy) display or a proxy size to draw the background image.

See also
To build a proxy, the Movie Clip Editor Proxy settings have to be used. Otherwise the proxy settings here have no effect.

Color Space
: The color space the image or video file uses within Blender.

View as Render
: Apply the color management settings when displaying this image on the screen.

Opacity
: Controls the transparency of the background image.

Depth
: Choose whether the image is shown behind all objects, or in front of everything.

Frame Method
: Controls how the image is placed in the camera view.

Stretch:: Forces the image dimensions to match the camera bounds (may alter the aspect ratio).

Fit:: Scales the image down to fit inside the camera view without altering the aspect ratio.

Crop:: Scales the image up so that it fills the entire camera view, but without altering the aspect ratio (some of the image will be cropped).

Offset X, Y
: Positions the background image using these offsets.

In orthographic views, this is measured in the normal scene units. In the camera view, this is measured relative to the camera bounds (0.1 will

offset it by 10% of the view width/height).

Rotation
: Rotates the image around its center.

Scale
: Scales the image up or down from its center.

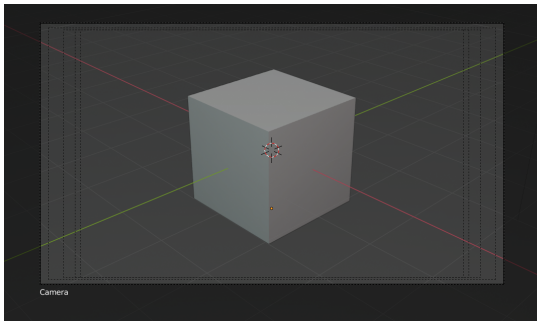
Flip
: X
: Swaps the image around, such that the left side is now on the right, and the right now on the left.

: Y
: Swaps the image around, such that the top side is now on the bottom, and the bottom now on the top.

Note

Movie Clips or images with view as render are only visible behind objects when film transparency is enabled or the scene world is disabled in the viewport.

Viewport Display



Camera view displaying safe areas, sensor and name.

Size

Size of the camera visualization in the 3D Viewport. This setting has **no** effect on the render output of a camera. The camera visualization can also be scaled using the standard Scale `S` transform key.

Show

Limits

Shows an orange line indicating the *Clip Start* and *End* values, as well as a yellow cross indicating the *Focus Distance*. If the *Focus Distance* gizmo is enabled in the 3D Viewport's gizmo settings, this cross can also be dragged with the mouse to adjust the distance.

Mist

Toggles viewing of the mist limits on and off. The limits are shown as two connected white dots on the camera line of sight. The mist limits and other options are set in the *World* panel, in the Mist section.

Sensor

Displays a dotted frame in camera view.

Name

Toggle name display on and off in camera view.

Composition Guides

Composition Guides enable overlays onto the camera display that can help when framing a shot.

Thirds

Adds lines dividing the frame in thirds vertically and horizontally.

Center

Center

Adds lines dividing the frame in half vertically and horizontally.

Diagonal

Adds lines connecting opposite corners.

Golden

Ratio

Divides the width and height into Golden proportions (about 0.618 of the size from all sides of the frame).

Triangle A

Displays a diagonal line from the lower left to upper right corners, then adds perpendicular lines that pass through the top left and bottom right corners.

Triangle B

Same as A, but with the opposite corners.

Harmony

Triangle A

Displays a diagonal line from the lower left to upper right corners, then lines from the top left and bottom right corners to 0.618 the lengths of the opposite side.

Triangle B

Same as A, but with the opposite corners.

Passepartout

This option darkens the area outside of the camera's field of view. The opacity of the passepartout can be adjusted using the value slider.

Tip

If the Passepartout is fully opaque, Blender will make optimizations to speed up the rendering of areas inside the camera view.

Lights

- Light Objects
  - Common Settings
  - Renderer Settings
  - Point Light
  - Spot Light
  - Area Light
  - Sun Light
  - Power of Lights
- Light Linking
  - Setup
  - Include & Exclude
  - Performance
- World Environment
  - Surface
  - Volume
  - Viewport Display

Light Objects

Reference

Panel:: *Properties* ▸ *Light* and *Shader Editor* ▸ *Sidebar* ▸ *Settings*

Common Settings

Type
: Defines the type of light.

Color
: Color tint of the emitted light.

Renderer Settings

- EEVEE specific settings
- Cycles specific settings

Point Light

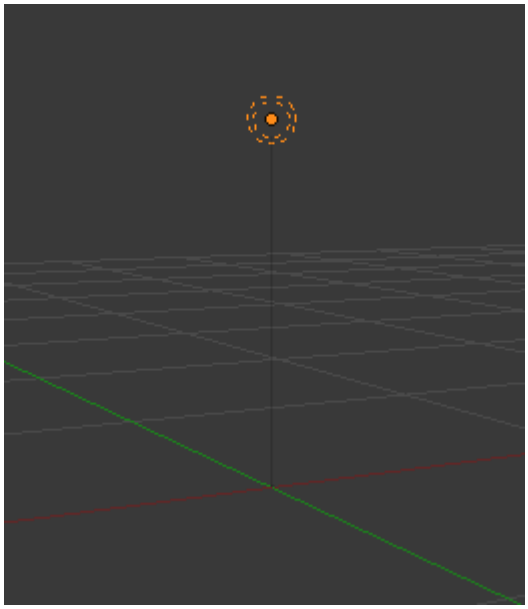
The point light is an omni-directional point of light, that is, a point radiating the same amount of light in all directions. It's visualized by a plain, circled dot. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the light and the point on the surface of the object itself. It can be used as simple model of e.g. a light bulb.

Light intensity/energy decays based on (among other variables) distance from the point light to the object. In other words, surfaces that are further away will be rendered darker.

Power
: Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

Soft Falloff
: Apply falloff to avoid sharp edges when the light geometry intersects with other objects.

Radius
: When larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows and specular highlights, and they will also appear dimmer because their power is distributed over a larger area.



Point light.

Spot Light

A spot light emits a cone-shaped beam of light from the tip of the cone, in a given direction.

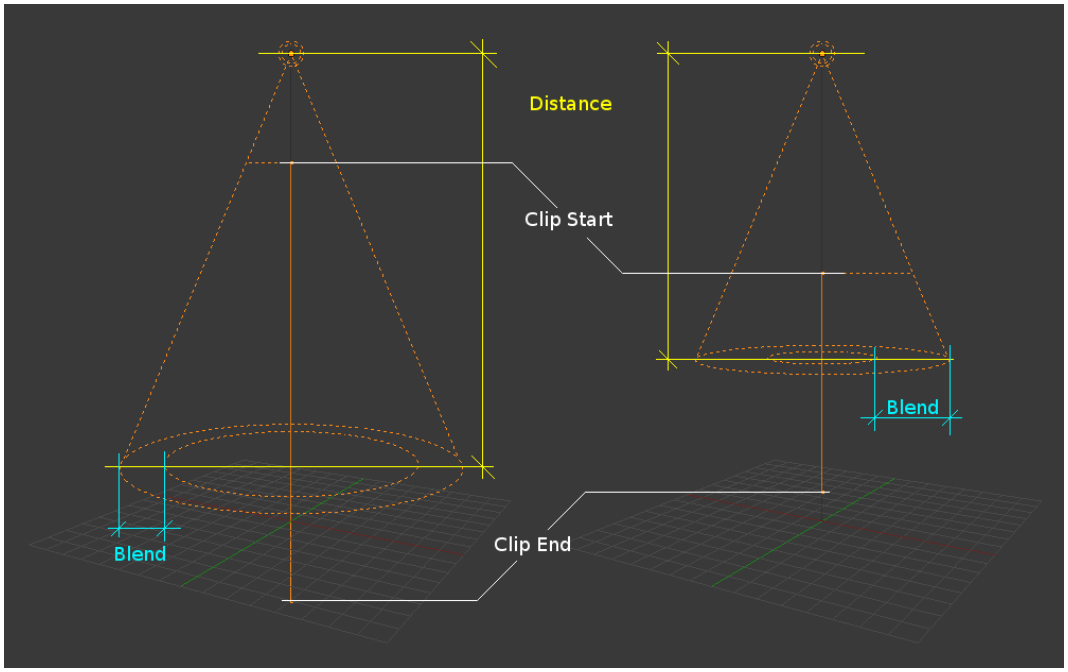
Power
: Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

Soft Falloff
: Apply falloff to avoid sharp edges when the light geometry intersects with other objects.

Radius
: When larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows

and specular highlights.

Beam/Spot Shape

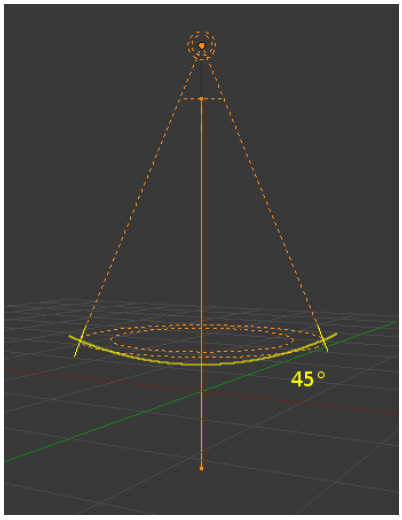


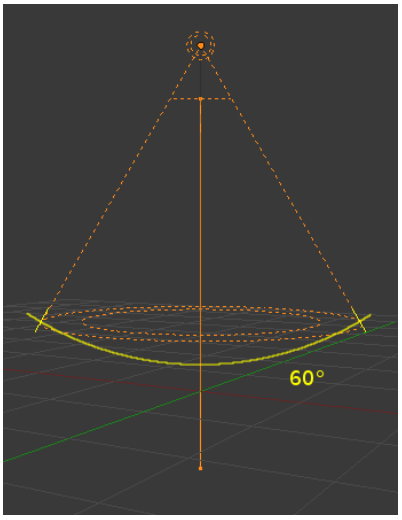
Changing the spot options also changes the appearance of the spotlight as displayed in the 3D Viewport.

Size

The size of the outer cone of a spot, which largely controls the circular area a spot light covers. This slider in fact controls the angle at the top of the lighting cone, and can be between (1.0 to 180.0).

Changing the spot *Size* option.





Blend

The *Blend* slider controls the inner cone of the spot. The *Blend* value can be between (0.0 to 1.0). The value is proportional and represents that amount of space that the inner cone should occupy inside the outer cone *Size*.

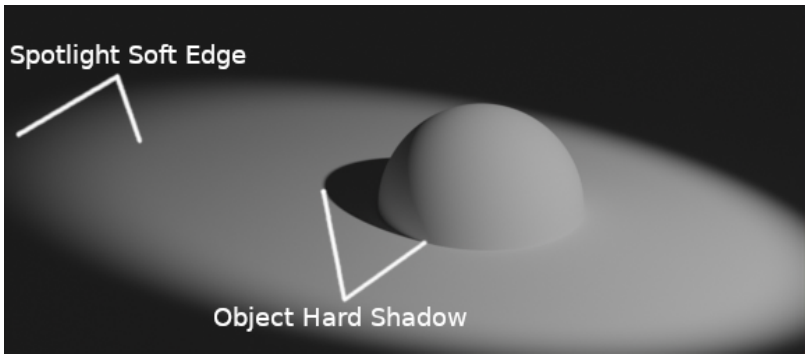
The inner cone boundary line indicates the point at which light from the spot will start to blur/soften; before this point its light will mostly be full strength. The larger the value of *Blend* the more blurred/soft the edges of the spotlight will be, and the smaller the inner cone's circular area will be (as it starts to blur/soften earlier).

To make the spot have a sharper falloff rate and therefore less blurred/soft edges, decrease the value of *Blend*. Setting *Blend* to 0.0 results in very sharp spotlight edges, without any transition between light and shadow.

The falloff rate of the spot light is a ratio between the *Blend* and *Size* values; the larger the circular gap between the two, the more gradual the light fades between *Blend* and *Size*.

Blend and *Size* only control the spot light cone's aperture and softness ("radial" falloff); they do not control the shadow's softness as shown

below.



Render showing the soft edge spotlighted area and the sharp/hard object shadow.

Notice in the picture above that the object's shadow is sharp as a result of the ray tracing, whereas the spotlight edges are soft. If you want other items to cast soft shadows within the spot area, you will need to alter other shadow settings.

Show Cone
: Displays a transparent cone in 3D Viewport to visualize which objects are contained in it.

Area Light

The area light simulates light originating from a surface (or surface-like) emitter. For example, a TV screen, office neon lights, a window, or a cloudy sky are just a few types of area light. The area light produces shadows with soft borders by sampling a light along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders.

Power
: Power of the light in Watts. Higher values increase the intensity of the light. Negative values can be set, but should be avoided for predictable and physically based result.

Shape

Shape of the light.

Rectangle::	The shape of the light can be represented as a rectangle and changed with the “X” and “Y” values.
Square::	The shape of the light can be represented as a square and changed with the *Size* property.
Disk::	The shape of the light can be represented as a disk and changed with the *Size* property.
Ellipse::	The shape of the light can be represented as an ellipse and changed with the X and Y values.

Tip

Choosing the appropriate shape for your area light will enhance the believability of your scene. For example, you may have an indoor scene and would like to simulate light entering through a window. You could place a *Rectangular* area light in a window (vertical) or from neon lights (horizontal) with proper ratio for *Size X* and *Size Y*. For the simulation of the light emitted by a TV screen, a vertical *Square* area light would be better in most cases.

Size / Size X / Size Y

Dimensions for the *Square* or *Rectangle*.

Sun Light

A sun light provides light of constant intensity emitted in a single direction from infinitely far away. It can be very handy for a uniform clear daylight open-space illumination. In the 3D Viewport, the sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light.

Note

This direction can be changed by rotating the sun light, like any other object, but because the light is emitted from a location considered infinitely far away, the location of a sun light does not affect the rendered result.

Strength
: Strength of the lights in Watts per square meter. Typical values are around 250 for an overcast day and 1000 or more for direct sunlight. See more details at Power of Lights.

Angle
: The size of the sun light according to its angular diameter as seen from earth.

Power of Lights

The power of sun lights is specified in Watts per square meter. The power of point lights, spot lights, and area lights is specified in Watts. But this is not the electrical Watts that consumer light bulbs are rated at. It is Radiant Flux or Radiant Power which is also measured in Watts. It is the energy radiated from the light in the form of visible light.

If you want to set the power to real world values, you have to convert the wattage of consumer bulbs or LED lights to radiant flux, but it is not a straightforward process. The wattage of bulbs means the electrical power required to power them. LED lights have a "Watt equivalent" which is neither the electrical power they require nor the amount of light they put out. Some consumer lights specify lumens or luminous flux which is the radiant flux weighted with the wavelengths perceived by the human eye.

To save you from doing the conversion, here is a table of typical power values for point, spot, and area lights:

Real world light	Power	Suggested Light Type
Candle	0.05 W	Point
800 lm LED bulb	2.1 W	Point
1000 lm light bulb	2.9 W	Point
1500 lm PAR38 floodlight	4 W	Area, Disk
2500 lm fluorescent tube	4.5 W	Area, Rectangle
5000 lm car headlight	22 W	Spot, size 125 degrees

And a table of typical Strength values for sun lights:

Sun type	Strength
Clear sky	1000 W/m^2
Cloudy sky	500 W/m^2
Overcast sky	200 W/m^2
Moonlight	0.001 W/m^2

Light Linking

With light linking, lights can be set to affect only specific objects in the scene. Shadow linking additionally gives control over which objects acts as shadow blockers for a light.

This adds more artistic control for lighting by breaking the laws of physics. For example the environment and characters in a shot might have different light setups. A character could have a dedicated linked rim light to make it stand out, and shadow linking could be used to ensure no objects from the environment block it.

Setup

- Select the light or emissive mesh object and go to the Cycles Shading panel or EEVEE Shading panel.
- Create a new light or shadowing linking collection.
- Drag & drop objects or collection from the outliner.

Note

Light linking emissive mesh object is only available for Cycles.

Links can also be set up in the 3D viewport, with the Link Data operator. The active light object is linked to selected receiver or blocker object.

A light or shadow linking collection can be assigned to and shared between multiple light objects. While a scene collection can be directly assigned as a light or shadow linking collection, it is recommended to instead create a dedicated collection and link any scene collection inside it instead. This way it's easy to include or exclude additional objects without affecting the scene layout.

Include & Exclude

Light receiver objects can be set to be either included or excluded. The behavior is as follows:

- If only included objects are specified, the light only affects those objects.
- If only excluded objects are specified, the light affects all objects in the scene except those specified.
- If both included and excluded objects are specified, the light affects only included objects minus the excluded objects. This can be used to for example set a character collection to be included, and then exclude specific objects part of the character.

Performance

Sampling for light linking is most efficient with the light tree enabled, where a specialized acceleration structure is built for light linking.

When using shadow linking, renders can be slower and trace additional rays, as direct and indirect lighting take different paths.

World Environment

The world defines the environment that the scene is in. The surface shader sets the background and environment lighting, either as a fixed color, sky model or HDRI texture. With volume shaders the entire scene can be covered in mist or other volumetric effects.



Lighting with an HDR image.

Surface

Reference

Panel:: *World ▸ Surface*

The surface shader defines the light emission from the environment into the scene. The world surface is rendered as if it is very distant from the scene, and as such there is no two-way interacting between objects in the scene and the environment, only light coming in. The only shader accepted is the Background node with a color input and strength factor for the intensity of the light.

Image Based Lighting

For image based lighting, use the Environment Texture node rather than the Image Texture node for correct mapping. This supports *Equirectangular* (also known as latitude/longitude) for environment maps, and *Mirror Ball* mapping for converting photos of mirror balls to environment maps.

Volume

Reference

Panel:: *World ▸ Volume*

A volume shader can be applied to the entirely world, filling the entire space.

Currently this is most useful for night time or other dark scenes, as the world surface shader or sun lights will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Viewport Display

Reference

Panel:: *World ▸ Viewport Display*

Color
: The color to render the 3D Viewport background when choosing World Background.

Materials

- Introduction
  - Setting up Materials
  - Components
  - Physically Based Shading
- Components
  - Surfaces
  - Volumes
  - Displacement
- Assignment
  - Material Slots
  - Reusing Existing Materials
  - Deleting a Material
  - Multiple Materials
- Preview
- Settings
  - Renderer Settings
  - Pass Index
  - Viewport Display
- Line Art
- Legacy Textures
  - Introduction
  - Colors
  - Types

Introduction

Materials control the appearance of meshes, curves, volumes and other objects. They define the substance that the object is made of, its color and texture, and how light interacts with it.

Physically based materials can be created using the Principled BSDF, Principled Hair, and Principled Volume shaders. With these uber shaders, a wide variety of materials including plastic, glass, metal, cloth, skin, hair, smoke and fire can be created.

A flexible shading nodes system is used to set up textures and create entirely different types of materials like toon shading.

Setting up Materials

Materials can be created in either the Material properties, or in the Shader Editor. These provide a different view of the same shader nodes and material settings.

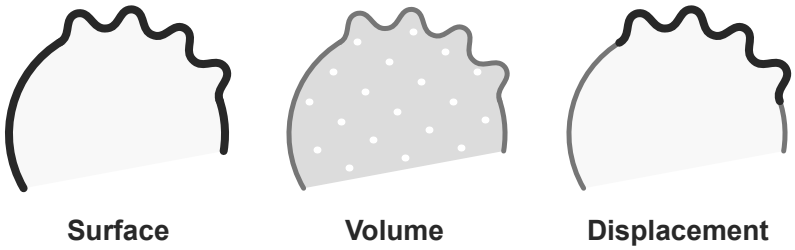
The default Shading workspace has a Shader Editor and a 3D Viewport that can be set to Material Preview or Rendered shading, to interactively preview how the material interacts with objects and lights in the scene.

Materials are data-blocks that can be assigned to one or more objects, and different materials can be assigned to different parts of meshes.

Image textures can be created from scratch in Texture Paint Mode, or by loading in existing images with the Image Texture node. A variety of procedural texture nodes is also available.

Components

Materials consist of three shaders, defining the appearance of the surface, the volume inside the object, and the displacement of the surface.



Surface Shader

The surface shader controls the textures and light interaction at the surface of the mesh.

Volume Shader

The volume shader defines the interior of the mesh. A material can have just a volume shader for cases like smoke and fire, or it can be combined with a surface shader for materials like cloudy glass.

Displacement

The shape of the surface and the volume inside it may be altered by displacement. This way, textures can then be used to make the mesh surface more detailed.

Depending on the settings, the displacement may be virtual, only modifying the surface normals to give the impression of displacement, which is known as bump mapping, or a combination of real and virtual displacement.

Physically Based Shading

The material system is built with physically-based rendering in mind, separating how a material looks and which rendering algorithm is used to render it. This makes it easier to achieve realistic results and balanced lighting, though there are a few things to keep in mind.

In order for materials to work well with global illumination, they should be energy conserving. That means they cannot reflect more light than comes in. This property is not strictly enforced, but if colors are in the range 0.0 to 1.0, and BSDFs are only mixed together with the Mix Shader node, this will automatically be true.

It is however, possible to break this, with color values higher than 1.0 or using the Add Shader node, but one must be careful when doing this to keep materials behaving predictably under various lighting conditions.

Components

- Surfaces
  - Terminology
  - BSDF Parameters
- Volumes
  - Shading
  - Mesh Volumes
  - World Volume
  - Multiple Scattering
- Displacement
  - Bump Only
  - Displacement Only
  - Displacement and Bump

Surfaces

The surface shader defines the light interaction at the surface of the mesh. One or more BSDFs specify if incoming light is reflected back, refracted into the mesh, or absorbed.

Emission defines how light is emitted from the surface, allowing any surface to become a light source.

Terminology

BSDF
: Stands for Bidirectional Scattering Distribution Function. It defines how light is reflected and refracted at a surface.

Reflection
: BSDFs reflect an incoming ray on the same side of the surface.

Transmission
: BSDFs transmit an incoming ray through the surface, leaving on the other side.

Refraction
: BSDFs are a type of *Transmission*, transmitting an incoming ray and changing its direction as it exits on the other side of the surface.

BSDF Parameters

A major difference from non-physically-based renderers is that direct light reflection from lights and indirect light reflection of other surfaces are not decoupled, but rather handled using a single BSDF. This limits the possibilities a bit, but we believe overall it is helpful in creating consistent-looking renders with fewer parameters to tune.

Roughness

For the glossy BSDFs, the *roughness* parameter controls the sharpness of the reflection, from 0.0 (perfectly sharp) to 1.0 (very soft).

Volumes

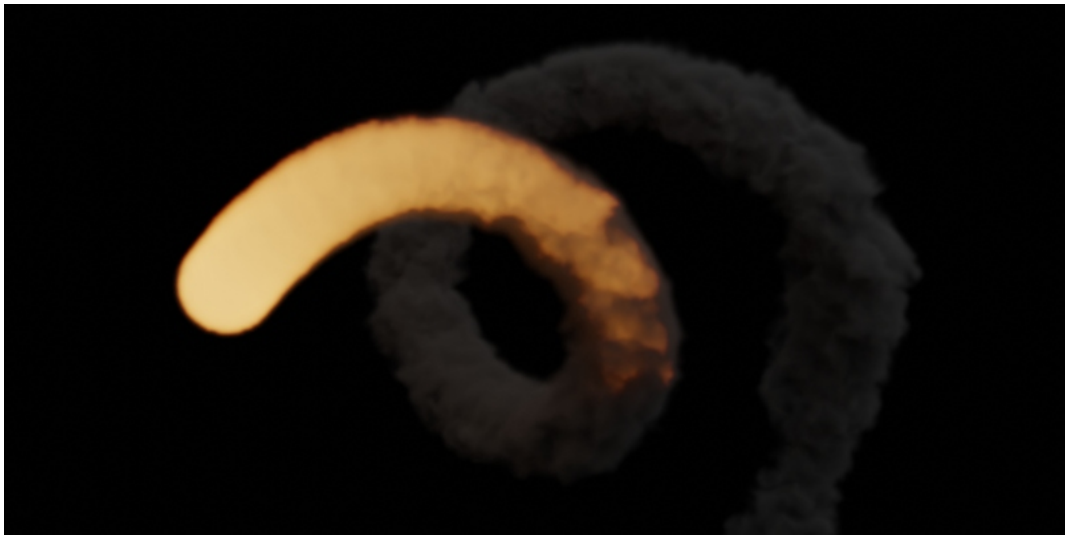
Volume rendering is used to render various effects that cannot be represented by hard surfaces alone.

- Smoke, fire or clouds are set up using a volume object or fluid simulation, with only a volume shader.
- Meshes can also be used to create such shapes by removing the default surface shader and using a volume shader with the mesh shape defining the volume bounds and textures defining the volume density.
- Mist is created with a volume shader for the world, or with a large mesh object encompassing the scene.
- Absorption in glass is simulated by combining a glass surface shader with refraction and a volume absorption shader for the interior of the object.

Shading

Principled Volume

Principled Volume is a physically-based volume shader that can be used to create a wide range of volume materials. It supports scattering, absorption and emission in one easy to use node. Fire can be rendered with blackbody emission.

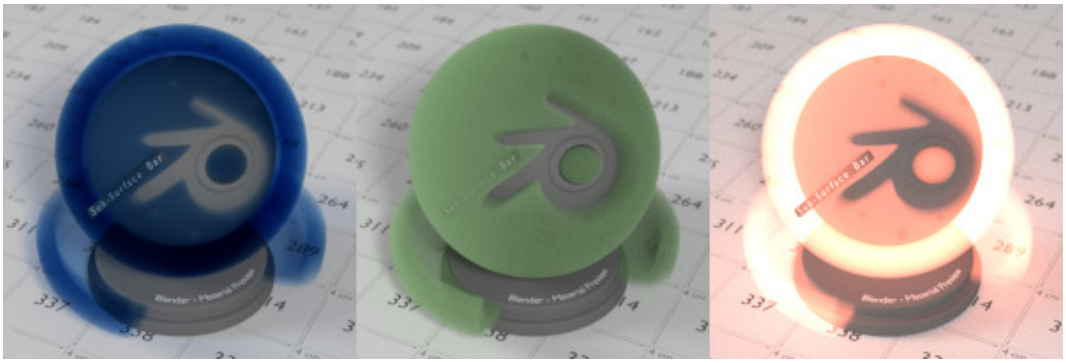


Smoke and fire rendered with Principled Volume shader.

Volume Components

For more control, volume shading components can be manually combined into a custom shader setup.

- Volume Absorption will absorb part of the light as it passes through the volume. This can be used to shade for example black smoke or colored glass objects, or mixed with the Volume Scatter node. This node is similar to the transparent BSDF node, it blocks part of the light and lets other light pass straight through.
- Volume Scatter lets light scatter in other directions as it hits particles in the volume. The anisotropy defines in which direction the light is more likely to scatter. A value of 0 will let light scatter evenly in all directions (similar to the diffuse BSDF node), negative values let light scatter mostly backwards, and positive values let light scatter mostly forward. This can be used to shade white smoke or clouds for example.
- Emission will emit light from the volume, for example for fire.



Volume Absorption, Scatter and Emission

Attributes

When rendering smoke and fire, volume attributes are used to define the shape and shading of the volume. The Principled Volume shader will use them by default, while custom volume shaders can use the Attribute node to get attributes such as density, color and temperature.

Density

All volume shaders have a density input. The density defines how much of the light will interact with the volume, getting absorbed or scattered, and how much will pass straight through. For effects such as smoke you would specify a density field to indicate where in the volume there is smoke and how much (density bigger than 0), and where there is no smoke (density equals 0).

Volumes in the real world consist of particles, a higher density means there are more particles per unit volume. More particles means there is a higher chance for light to collide with a particle and get absorbed or scattered, rather than passing straight through.

Mesh Volumes

Meshes used for volume render should be closed and Manifold. That means that there should be no holes in the mesh. Each edge must be connected to

exactly two faces such that there are no holes or T-shaped faces where three or more faces are connected to an edge.

Normals must point outside for correct results. The normals are used to determine if a ray enters or exits a volume, and if they point in a wrong direction, or there is a hole in the mesh, then the renderer is unable to decide what is the inside or outside of the volume.

These rules are the same as for rendering glass refraction correctly.

World Volume

A volume shader can also be applied to the world, filling the entire space.

Currently, this is most useful for night time or other dark scenes, as the world surface shader or sun lights will have no effect if a volume shader is used. This is because the world background is assumed to be infinitely far away, which is accurate enough for the sun for example. However, for modeling effects such as fog or atmospheric scattering, it is not a good assumption that the volume fills the entire space, as most of the distance between the sun and the earth is empty space. For such effects it is be better to create a volume object surrounding the scene. The size of this object will determine how much light is scattered or absorbed.

Multiple Scattering

Real-world effects such as scattering in clouds or subsurface scattering require many scattering bounces. However, unbiased rendering of such effects can be noisy, so by default the number of bounces is zero in Cycles, and no support is available in EEVEE. The effect you get when rendering with zero volume bounces is what is known as “single scattering”, the effect from more bounces is “multiple scattering”.

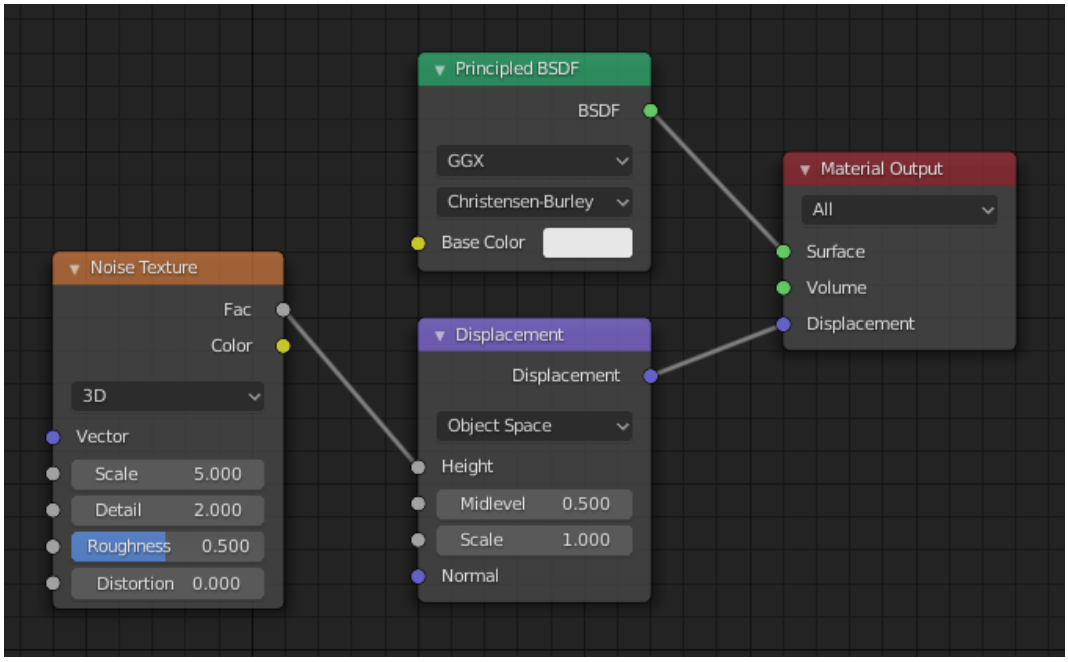
For rendering materials like skin or milk that require multiple scattering, subsurface scattering is more efficient and easier to control. Particularly the random walk method can accurately render such materials.

For materials such as clouds or smoke that do not have a well-defined surface, volume rendering is required. These look best with many scattering bounces, but in practice one might have to limit the number of bounces to keep render times acceptable.

Displacement

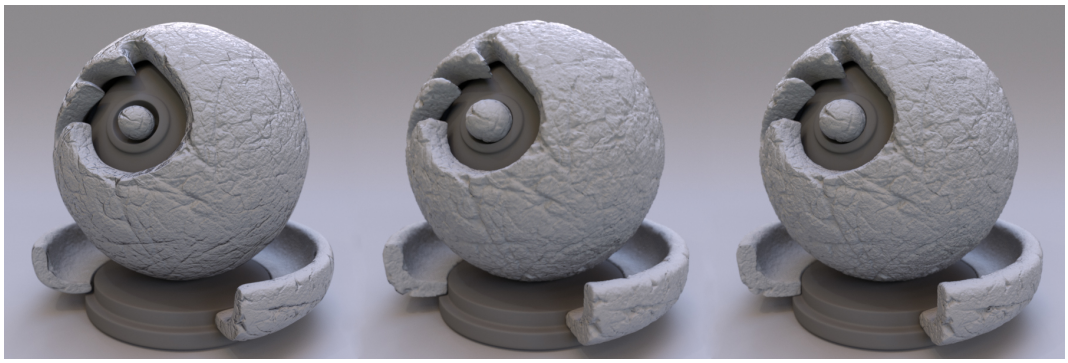
Detail can be added to the shape of a surface with displacement shaders.

To create displacement, connect a Displacement or Vector Displacement node to the displacement input of the Material Output node. Procedural, painted or baked textures can then be connected to these nodes.



Typical displacement node setup.

Three displacement methods exist, with varying accuracy, performance and memory usage. The displacement method can be set per material in the Material Settings.



Bump only, displacement only, and displacement and bump combined.

Bump Only

The least accurate but most memory efficient method is bump mapping. This method does not actually alter the mesh surface, but merely changes the shading to make it seem so.

Bump maps are often used to add smaller details on a model, for example pores or wrinkles on skin.

For baked bump maps, 8-bit images are commonly used. However, 16 or 32-bit float maps can provide better looking results. When using image textures use Cubic interpolation to avoid stepping artifacts, these are more visible for bump maps than other types of textures.

Important

Because bump mapping is a fake effect, it can cause artifacts if the actual shape of the geometry is too different from the bump mapped shape. If this happens the strength of bump mapping should be reduced or actual displacement should be used.

Displacement Only

Cycles Only

The most accurate and memory intensive displacement method is to apply true displacement to the mesh surface.

It requires the mesh to be finely subdivided, which can be memory intensive. Adaptive Subdivision is the best way to subdivide the mesh, so that exactly the right amount of subdivision is used depending on the distance of the object to the camera.

For baked displacement maps, best results are achieved with 16 or 32-bit float maps, as 8-bit images often can not represent all the necessary detail.

See also

The Displace Modifier can also be used to displace a mesh.

Displacement and Bump

Both methods can be combined to use actual displacement for the bigger displacement and bump for the finer details. This can provide a good balance to reduce memory usage.

Once you subdivide the mesh very finely, it is better to use only actual displacement in Cycles. Keeping bump maps will then only increase memory usage and slow down renders.

Assignment

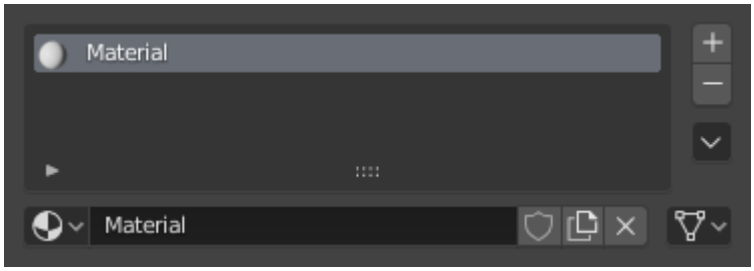
Reference

Panel:: *Material ▸ Material Slots*

Materials are data-blocks that can be created and then assigned to one or more objects. An object can also have multiple materials assigned in different material slots, which correspond to different parts of an object. If a smooth transition between materials is desired, then mixing shader nodes with a Mix shader is a better solution.

Material Slots

Material slots link materials to objects and meshes. By default objects only have a single material slot, which assigns a material to the entire object. If different parts of the mesh need different materials, multiple material slots can be created.



Material slots panel.

Slot List

The object's material slots and active material displayed in a List View.

Add Material Slot
: Add a new material slot on the object.

Remove Material Slot
: Remove a material slot from the object.

Copy Material
: Copy material shader nodes and settings to clipboard.

Paste Material
: Paste material shader nodes and settings from clipboard.

Copy Material to Selected
: Copy the same material assignment from the active to other selected objects.

Remove Unused Slots
: Removes all material slots not assigned to the object.

Data-Block

Material
: The Material Data-Block Menu for the selected material slot. Here new materials can be created, or existing materials can to the material slot.

Link
: Specifies whether the material is to be linked to the object or to the object data.

  The Link selector has two choices, Data and Object. These two menu choices determine whether the material is linked to the object or to the data (e.g. a mesh or curve). The Data menu item determines that this material will be linked to the mesh's data-block which is linked to the object's data-block. The Object menu item determines that the material will be linked to the object's data-block directly.

  This has consequences of course. For example, different objects may share the same mesh data-block. Since this data-block defines the shape of the object any change in Edit Mode will be reflected on all of those objects. Moreover, anything linked to that mesh data-block will be

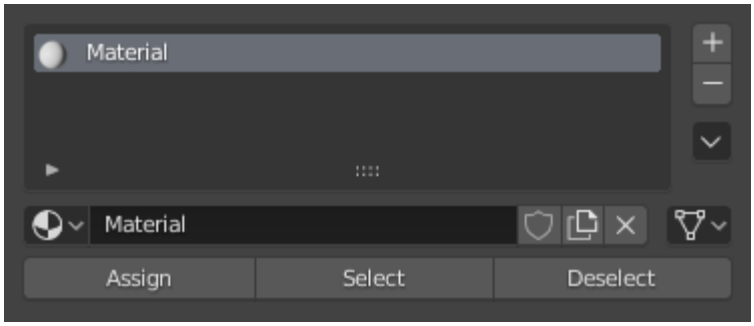
shared by every object that shares that mesh. So, if the material is linked to the mesh, every object will share it.

On the other hand, if the material is linked directly to the object data-block, the objects can have different materials and still share the same mesh.

Short explanation: If connected to the object, you can have several instances of the same Object Data using different materials. If linked to mesh data, you cannot. See Data System for more information.

Edit Mode

To assign materials to different parts of a mesh, enter Edit Mode on the mesh. Additional buttons will then appear in the material slots panel.



Material slots panel in Edit Mode.

Assign
: Assign active material slot and material to the selected faces in the mesh, strokes in a Grease Pencil, and similar for other object types.

Select
: Select faces assigned to the active material slot.

Deselect
: Deselect faces assigned to the active material slot.

Reusing Existing Materials

Blender is built to allow you to reuse *anything*, including material settings, between many objects. Instead of creating duplicate materials, you can simply reuse an existing material. There are several ways to do this using the Material's data-block menu:

Single Object – With the object selected, click the sphere located to the left of the Material name. A pop-up appears showing all the materials available in the current blend-file. To use one, just click on it.

Tip

Searching for Materials

The search field at the bottom of the material list allows you to search the names in the list. For example, by entering "wood" all existent materials are filtered so that only materials containing "wood" are displayed in the list.

Multiple Objects – In the 3D Viewport, with `Ctrl-L` you can quickly link all selected objects to the material (and other aspects) of the active object. Very useful if you need to set a large number of objects to the same material; just select all of them, then the object that has the desired material, and `Ctrl-L` links them to that "parent".

Deleting a Material

To delete a material, select the material and click X in the Available Materials List entry.

Although the material will seem to disappear immediately, the Delete action can depend on how the material is used elsewhere.

If the material is linked to the object and there are other objects which use this material, then the material will be removed from that object (but remain on all its other objects).

If the “Fake User” button has been lit in the Available Materials list, then the material will be retained when the file is saved, even if it has no users.

Only if it has 0 “real” users, and no “Fake” user, will the material be permanently deleted. Note that it will still remain in the Materials list until the blend-file is saved, but will have disappeared when the file is reloaded.

Multiple Materials

Normally, different colors or patterns on an object are achieved by adding textures to your materials. However, in some applications you can obtain multiple colors on an object by assigning different materials to the individual faces of the object.

To apply several materials to different faces of the same object, you use the Material Slots options in the Materials header panel.

The workflow for applying a second material to some faces of an object covered by a base material is as follows:

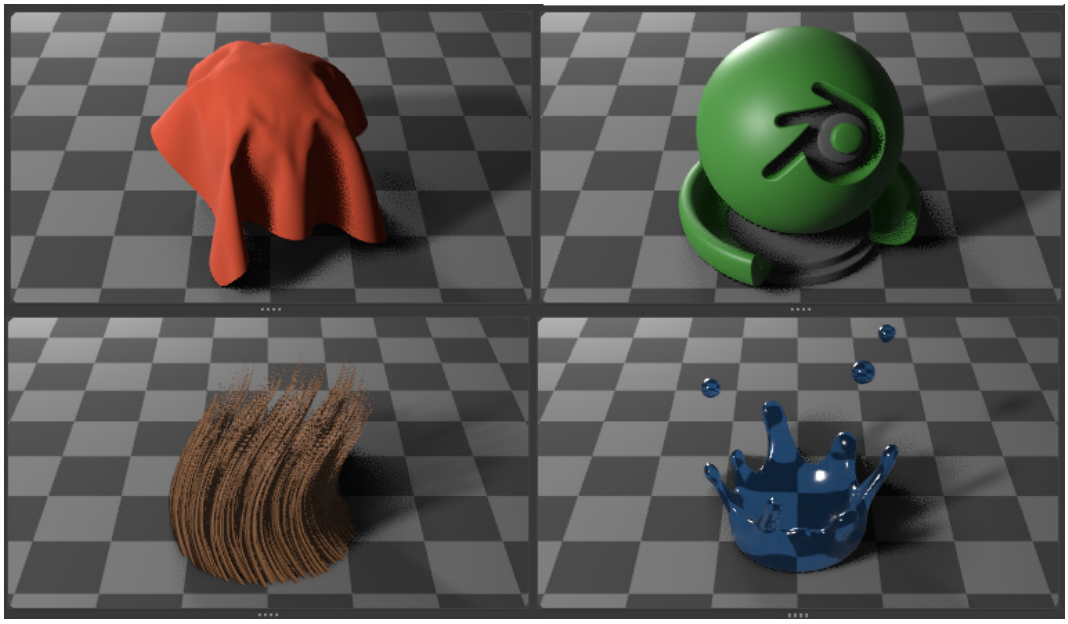
1. In Object Mode, create a base material.
2. Go into Edit Mode and Face Select (a new list will appear below the Active Material list with *Assign*, *Select*, *Deselect* buttons).
3. Select the faces to be colored with the second material.
4. In the Object Material Slots list, click the + button to create a new slot or select an existing material.
5. Click the *Assign* button, and the material will appear on the selected object faces.

Preview

The Preview panel gives a quick visualization of the active material applied in a simple scene.

Shape
: Preview the material on a Plane, Sphere, Cube, Hair, Shader Ball, Cloth or Fluid object. This shape is also used for previews when linking and appending materials.

Preview World *Cycles Only*
: Use the world from the current scene for lighting in the material preview.



Preview shapes.

Settings

Reference

Panel:: *Material* ▸ *Settings*

Renderer Settings

While shading nodes control the appearance, these settings control the quality and algorithms that each renderer uses to render the material.

- EEVEE specific settings
- Cycles specific settings

Pass Index

Pass Index
: Index number for the *Material Index* render pass. This can be used to give a mask to a material and then be read with the ID Mask Node in the Compositor.

Note

Volume Objects are not supported.

Viewport Display

These settings control the 3D Viewport display in solid shading. They provide a faster alternative to full shader nodes, which may be too heavy or distracting for tasks like modeling, layout or sculpting.

Color

Diffuse or metal surface color.

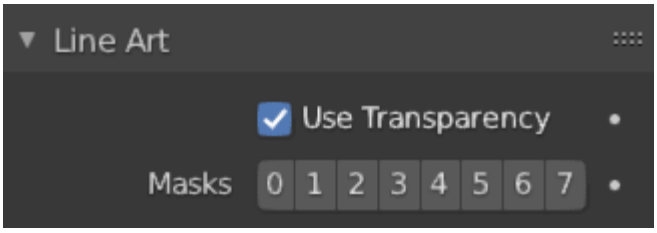
Metallic
: Blends between a non-metallic and metallic material model. A value of 1.0 gives a fully specular reflection tinted with the base color, without diffuse reflection or transmission. At 0.0 the material consists of a diffuse or transmissive base layer, with a specular reflection layer on top.

Roughness
: Specifies microfacet roughness of the surface for metal and specular reflection.

Line Art

Reference

Panel:: *Material ▸ Line Art*



Line Art material properties.

Material Mask
: Material masks are a way to provide Line Art extra information about faces that caused the occlusion. So edges occluded by those faces can be selected to have different styles.

Masks
: The layer to include faces of the current material.

Levels
: Faces with this material will behave as if it has set number of layers in occlusion.

Intersection Priority
: Assigns an intersection priority value for this material. Note that this priority takes precedent over Object or Collection priority values. The intersection line will be included into the object with the higher intersection priority value.

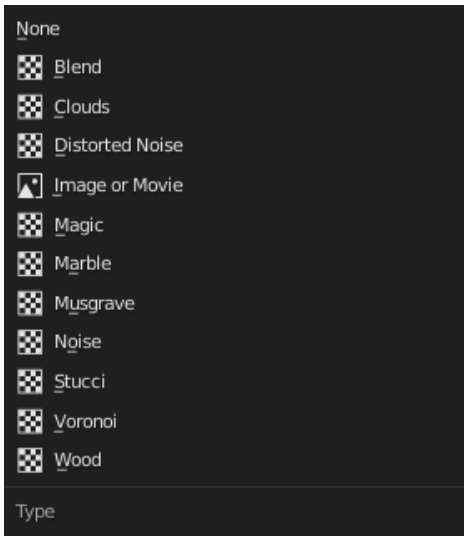
Legacy Textures

- Introduction
- Colors

Types

- Blend
- Clouds
- Distorted Noise
- Image or Movie
- Magic
- Marble
- Musgrave
- Noise
- Stucci
- Voronoi
- Wood

Introduction



The Texture Type list in the Texture panel of the Texture buttons.

Procedural textures are textures that are defined mathematically. They are generally relatively simple to use, because they do not need to be mapped in a special way. This does not mean that procedural textures cannot become very complex.

These types of textures are ‘real’ 3D. By that we mean that they fit together perfectly at the edges and continue to look like what they are meant to look like even when they are cut; as if a block of wood had really been cut in two. Procedural textures are not filtered or anti-aliased. This is hardly ever a problem: the user can easily keep the specified frequencies within acceptable limits.

Common Options

Noise Basis

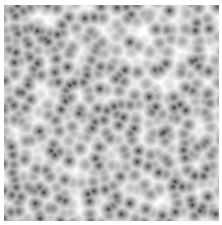
Each noise-based Blender texture (except Voronoi and Simple Noise) has a *Noise Basis* setting that allows the user to select which algorithm is used to

generate the texture. This list includes the original Blender noise algorithm. The *Noise Basis* settings makes the procedural textures extremely flexible (especially *Musgrave*).

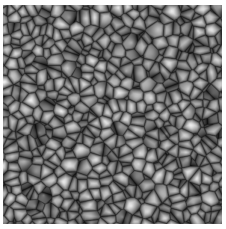
The *Noise Basis* governs the structural appearance of the texture:



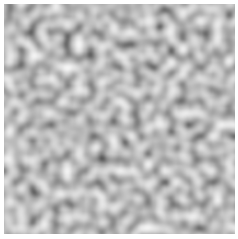
Blender Original.



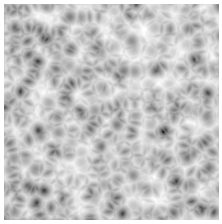
Voronoi F1.



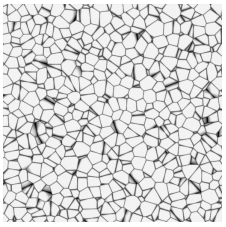
Voronoi F2-F1.



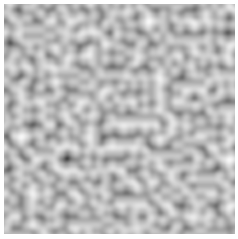
Original Perlin.



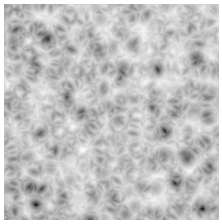
Voronoi F2.



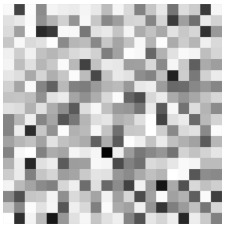
Voronoi Crackle.



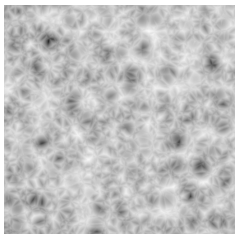
Improved Perlin.



Voronoi F3.



Cell Noise.



Voronoi F4.

There are two more possible settings for *Noise Basis*, which are relatively similar to *Blender Original*: Improved Perlin and Original Perlin.

Nabla

Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (except *Blend* and *Magic*). This is important for Normal and Displacement Maps. The strength of the effect is controlled with the *Nabla* number field.

Hints

Procedural textures can either produce colored textures, intensity only textures, textures with alpha values and normal textures. If intensity only ones are used the result is a black-and-white texture, which can be greatly enhanced by the use of ramps. If on the other hand you use ramps and need an intensity value, you have to switch on *No RGB* in the *Mapping* panel.

Colors

The color of a texture can be modified with the *Brightness*, *Contrast*, and *Saturation* buttons. All textures with RGB values, including *Images* and *Environment Maps*, may be modified with the RGB sliders.

Clamp
: Set negative texture RGB and intensity values to zero, for some uses like displacement this option can be disabled to get the full range.

Multiply R, G, B
: Tint the color of a texture by brightening each red, green and blue channel.

Brightness
: Change the overall brightness/intensity of the texture.

Contrast
: Change the contrast of the texture.

Saturation
: Change the saturation of the texture.

Color Ramp

Activates a color ramp which allows you to remap the colors of a texture to new ones.

Blend

The Blend texture generates a smoothly interpolated progression. This is one of the most frequently used procedural textures. You can use blend textures to blend other textures together (with *Stencil*), or to create nice effects (especially with the *Mapping: Normal* trick).

Note

Remember that if you use a ramp to create a custom blending, you may have to use *No RGB*, if the *Mapping* value needs an intensity input.



Blend Texture panels.

Options

Progression
: Profile of blend.

Linear
: A linear progression.

Quadratic
: A quadratic progression.

Easing
: A flowing, nonlinear progression.

Diagonal
: A diagonal progression.

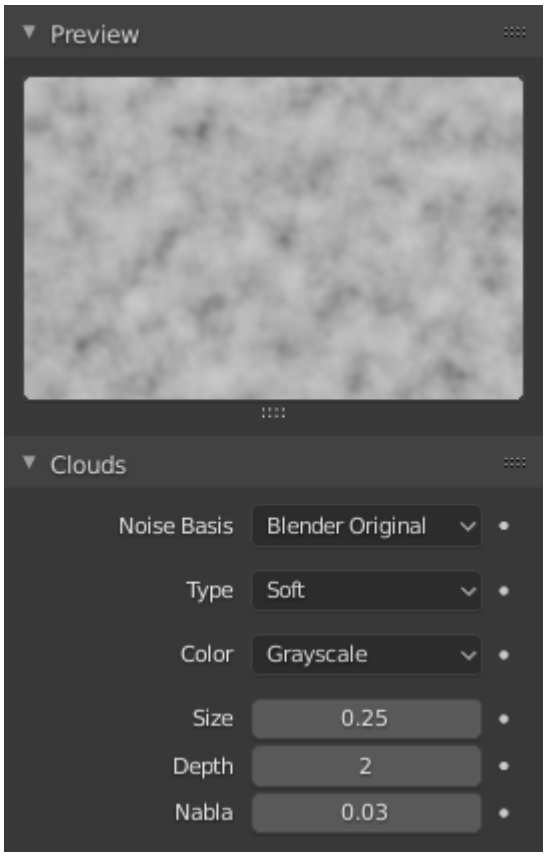
Spherical
: A progression with the shape of a three-dimensional ball.

Quadratic Sphere
: A quadratic progression with the shape of a three-dimensional ball.

Radial
: A radial progression: *Horizontal* / *Vertical*. The direction of the progression is flipped a quarter turn.

Clouds

Clouds represent Perlin noise. In addition, each noise-based Blender texture (with the exception of Voronoi and simple noise) has a *Noise Basis* setting that allows the user to select which algorithm is used to generate the texture. This is often used for Clouds, Fire, Smoke. Well-suited to be used as a Bump map, giving an overall irregularity to the material.



Clouds Texture panels.

Options

Grayscale
: The standard noise, gives an intensity.

Color
: The noise gives an RGB value.

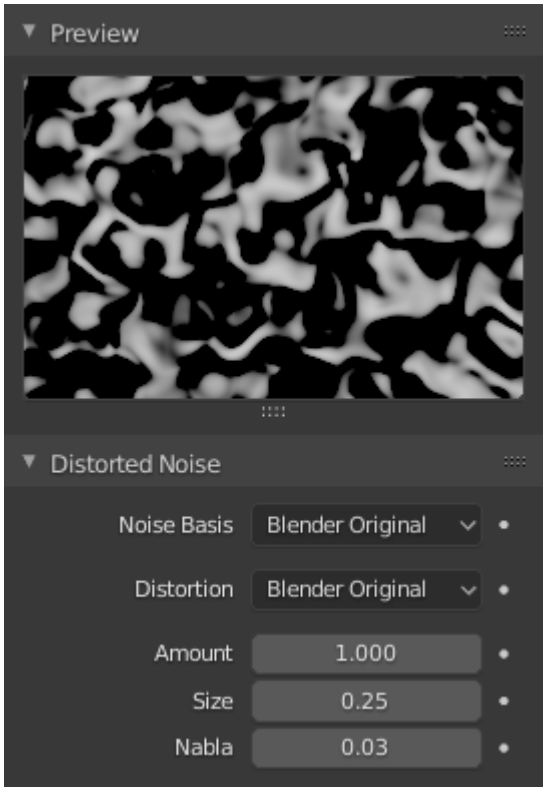
Noise
: *Soft* or *Hard*, changes contrast and sharpness.

Size
: The dimension of the Noise table.

Depth
: The depth of the *Clouds* calculation. A higher number results in a long calculation time, but also in finer details.

Distorted Noise

Distortion Noise takes the option that you pick from *Noise Basis* and filters it, to create hybrid pattern. It is often used for grunge but is also very complex and versatile.



Distorted Noise Texture panels.

Options

Noise Basis
: The texture to be distorted.

Distortion
: The texture to use to distort another.

Amount
: The amount that *Distortion Noise* affects *Basis*.

Size
: The size of the noise generated.

Nabla
: Almost all procedural textures in Blender use derivatives for calculating normals for texture mapping (except *Blend* and *Magic*). This is important for Normal and Displacement Maps. The strength of the effect is controlled with the *Nabla* number field.

Image or Movie

The term *Image Texture* simply means that a graphic image, which is a pixel grid composed of R, G, B, and sometimes Alpha values. It is used as the input source to the texture. As with other types of textures, this information can be used in a number of ways, not only as a simple “decal”.

Video textures are a some kind of Image textures and based on movie file or sequence of successive numbered separate images. They are added in the same way that image textures are.

When the Texture Type *Image or Movie* is selected, three new panels present themselves allowing to control most aspects of how image textures are applied: *Image*, *Image Sampling*, and *Image Mapping*.

About Image-Based Texturing

Texture images take up precious memory space, often being loaded into a special video memory bank that is very fast and very expensive, so it is often very small. So, keep the images as small as possible. A 64×64 image takes up only one fourth the memory of a 128×128 image.

For photorealistic rendering of objects in animations, often larger image textures are used, because the object might be zoomed in on in camera moves. In general, you want to use a texture sized proportionally to the number of pixels that it will occupy in the final render. Ultimately, you only have a certain amount of physical RAM to hold an image texture and the model and to provide workspace when rendering your image.

For the most efficient memory usage, image textures should be square, with dimensions as powers of 2, such as 32×32, 64×64, 128×128, 256×256, 1024×1024, 2048×2048, and 4096×4096.

If you can reuse images across different meshes, this greatly reduces memory requirements. You can reuse images if you map those areas of the

meshes that “look alike” to a layout that uses the common image.

When using file textures, it is very important that you have Mapped the UVs of the mesh, and they are laid out appropriately.

You do not have to UV map the *entire* mesh. The sphere above on the left has some faces mapped, but other faces use procedural materials and textures. Only use UV textures for those portions of your mesh where you want very graphic, precise detail. For example, a model of a vase only needs UV texture for the rim where decorative artwork is incorporated. A throw pillow does not need a different image for the back as the front; in fact many throw pillows have a fabric (procedural material) back.

As another example, you should UV map both eyes of a head to the same image (unless you want one bloodshot and the other clear). Mapping both sides of a face to the same image might not be advisable, because the location of freckles and skin defects are not symmetrical. You could of course change the UV map for one side of the face to slightly offset, but it might be noticeable. Ears are another example where images or section of an image can be mapped to similar faces.

Options

Image
: The Image Data-Block Menu.

Alpha

Use the alpha channel information stored in the image. Where the alpha value in the image is less than 1.0, the object will be partially transparent and things behind it will be visible. Works with image formats that store transparency information.

Calculate
: Calculate an alpha based on the RGB values of the Image. Black (0, 0, 0) is transparent, white (1, 1, 1) opaque. Enable this option if the image

texture is a mask. Note that mask images can use shades of gray that result in semi-transparency, like ghosts, flames, and smoke/fog.

The image with various alpha and gray-scale values.

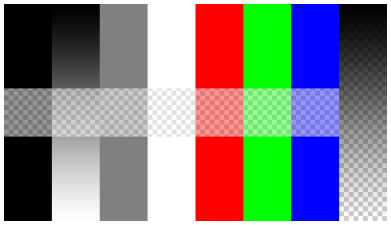


Image with *Use* alpha. The alpha values of the pixels are evaluated.

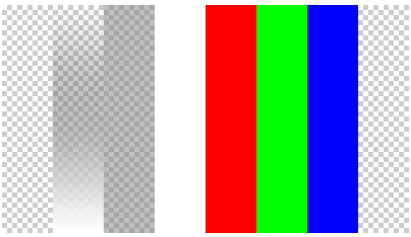


Image with *Calculate* alpha only, *Use Alpha* in the *Image* panel is disabled.

Invert
: Reverses the alpha value. Use this option if the mask image has white where you want it transparent and vice versa.

Mapping

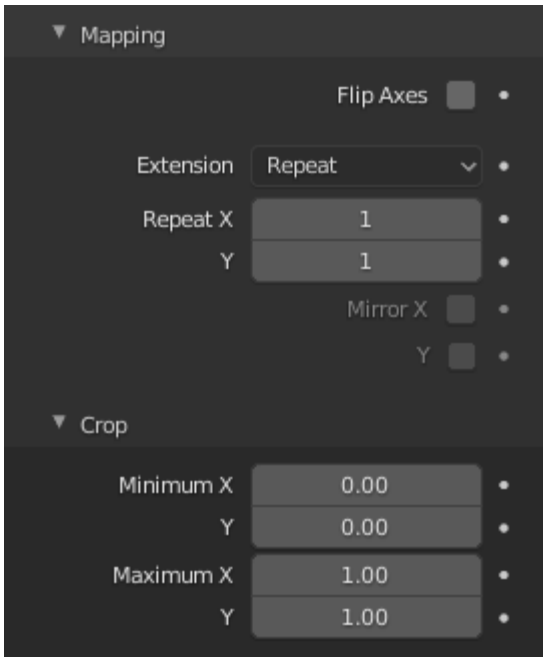


Image Mapping panel.

In the *Mapping* panel, you can control how the image is mapped or projected onto the 3D model.

Flip Axes
: Rotates the image 90 degrees counterclockwise when rendered.

Extension
: How the image is extrapolated beyond its original bounds.

Extend:: Outside the image the colors of the edges are extended.

Clip:: Clip to image size and set exterior pixels as transparent. Outside the image, an alpha value of 0.0 is returned. This allows you to 'paste' a small logo on a large object.

Clip Cube:: Clips to cubic-shaped area around the images and sets exterior pixels as transparent. The same as Clip, but now the 'Z' coordinate is calculated as well. An alpha value of 0.0 is returned outside a cube-shaped area around the image.

Repeat:: The image is repeated horizontally and vertically.

Repeat X, Y
: X/Y repetition multiplier.

Mirror X, Y
: Mirror on X/Y axes. These buttons allow you to map the texture as a mirror, or automatic flip of the image, in the corresponding X and/or Y direction.

Checker:: Checkerboards quickly made. You can use the option *size* on the *Mapping* panel as well to create the desired number of checkers.

Tiles Even/Odd
: Set even/odd tiles.

Distance

Governs the distance between the checkers in parts of the texture size.

Crop

Minimum X, Y / Maximum X, Y
: The offset and the size of the texture in relation to the texture space. Pixels outside this space are ignored. Use these to crop, or choose a portion of a larger image to use as the texture.

Sampling

In the *Sampling* panel you can control how the information is retrieved from the image.

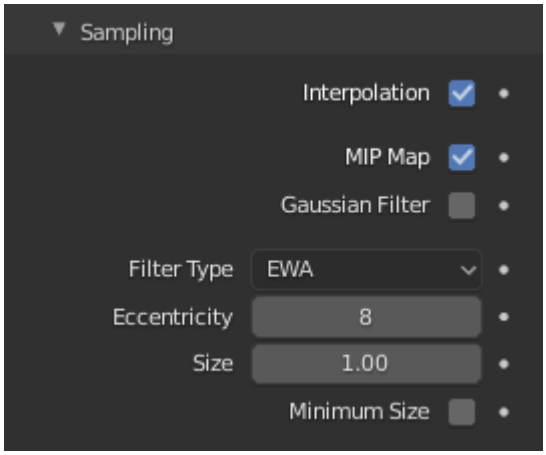


Image Sampling panel.

Interpolation
: This option interpolates the pixels of an image. This becomes visible when you enlarge the picture. By default, this option is on. Turn this option off to keep the individual pixels visible and if they are correctly anti-aliased. This last feature is useful for regular patterns, such as lines and tiles; they remain ‘sharp’ even when enlarged considerably. Turn this image off if you are using digital photos to preserve crispness.



Enlarged Image texture without *Interpolation*.



Enlarged Image texture with *Interpolation*.

MIP Map

Mip-maps are precalculated, smaller, filtered textures for a certain size. A series of pictures is generated, each half the size of the former one. This optimizes the filtering process. By default, this option is enabled and speeds up rendering. When this option is off, you generally get a sharper image, but this can significantly increase calculation time if the filter dimension (see below) becomes large. Without mip-maps you may get varying pictures from slightly different camera angles, when the textures become very small. This would be noticeable in an animation.

Gaussian Filter

Used in conjunction with mip-mapping, it enables the mip-map to be made smaller based on color similarities. In game engines, you want your textures, especially your mip-map textures, to be as small as possible to increase rendering speed and frame rate.

Filter Type

Texture filter to use for image sampling. Just like a *pixel* represents a *pic* ture *el* ement, a *texel* represents a *tex* ture *el* ement. When a texture (2D texture space) is mapped onto a 3D model (3D model space), different algorithms can be used to compute a value for each pixel based on samples from several texels.

Box:: A fast and simple nearest-neighbor interpolation known as Monte Carlo integration.

EWA (Elliptical Weighted Average)::

One of the most efficient direct convolution algorithms developed by Paul Heckbert and Ned Greene in the 1980s. For each texel, EWA samples, weights, and accumulates texels within an elliptical footprint and then divides the result by the sum of the weights.

Eccentricity

Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower.

FELINE (Fast Elliptical Lines)::

Uses several isotropic probes at several points along a line in texture space to produce an anisotropic filter to reduce aliasing artifacts without considerably increasing rendering time.

Light Probes

Number of probes to use. An integer between 1 and 256. Further reading: McCormack, J; Farkas, KI; Perry, R; Jouppi, NP (1999) Simple and Table Feline: Fast Elliptical Lines for Anisotropic Texture Mapping, WRL

Area:: Area filter to use for image sampling.

Eccentricity

Maximum Eccentricity. Higher values give less blur at distant/oblique angles, but is slower.

Size

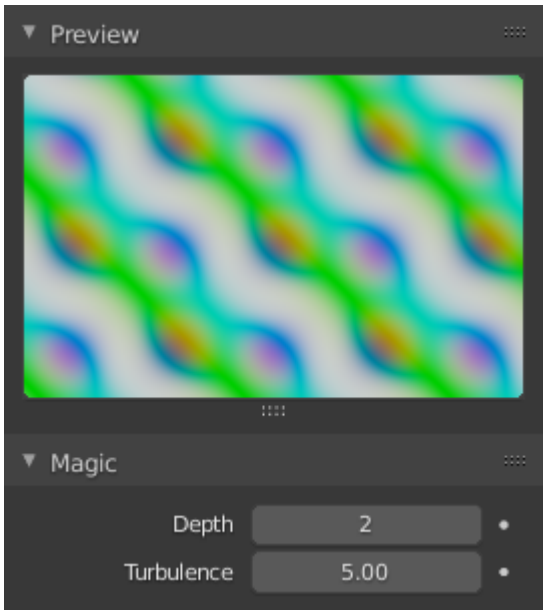
The filter size used in rendering, and also by the options *Mip Map* and *Interpolation*. If you notice gray lines or outlines around the textured object, particularly where the image is transparent, turn this value down from 1.0 to 0.1 or so.

Minimum Size

Use Filter Size as a minimal filter value in pixels.

Magic

The Magic Texture node is used to add a psychedelic color texture. It can be used for “Thin Film Interference” if you set *Mapping* to *Reflection* and use a relatively high *Turbulence*. The RGB components are generated independently with a sine formula.



Magic Texture panels.

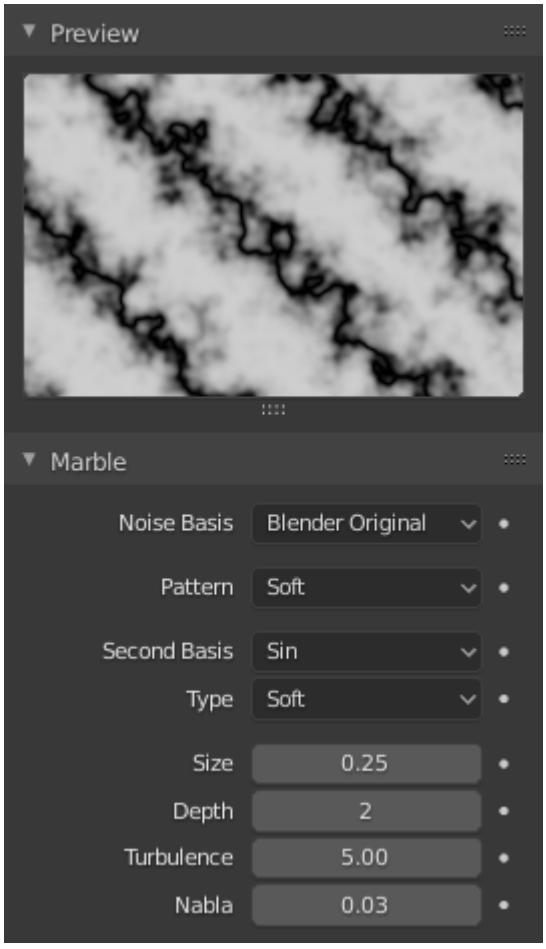
Options

Depth
: The depth of the calculation. A higher number results in a long calculation time, but also in finer details.

Turbulence
: The strength of the pattern.

Marble

The marble texture is used to generate marble, fire, or noise with a structure. Bands are generated based on the sine, saw, or triangular formula and noise turbulence.



Marble Texture panels.

Options

Marble Type
: Three settings for soft to more clearly defined *Marble*.

Soft, Sharp, Sharper

Noise basis

Shape of wave to produce bands.

Sine, Saw, Triangle

Noise Type

The noise function works with two methods.

Soft, Hard

Size

The dimensions of the noise table.

Depth

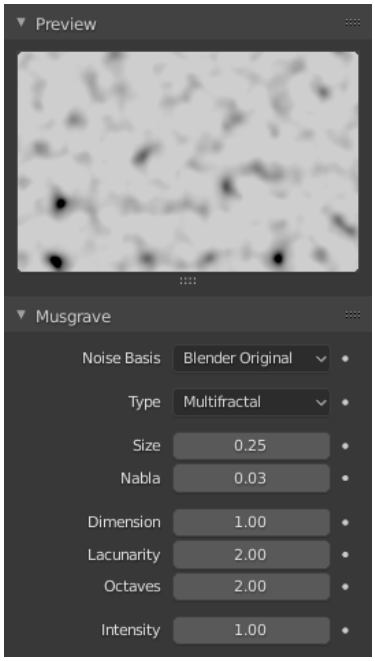
The depth of the *Marble* calculation. A higher value results in greater calculation time, but also in finer details.

Turbulence

The turbulence of the sine bands.

Musgrave

The musgrave texture is used to generate organic materials, but it is very flexible. You can do nearly everything with it.



Musgrave Texture panels.

Options

Type

This procedural texture has five noise types on which the resulting pattern can be based and they are selectable from a select menu at the top of the tab. The five types are:

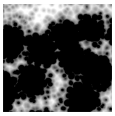
- Hetero Terrain
- Fractal Brownian Motion (fBm)
- Hybrid Multifractal
- Ridged Multifractal
- Multifractal

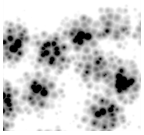
These noise types determine the manner in which Blender layers successive copies of the same pattern on top of each other at varying contrasts and scales.

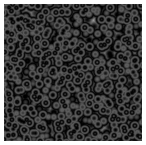
Examples with Basis: Voronoi: F1, Dimension: 0.5, Lacunarity: 0.15, Octave: 2.0.

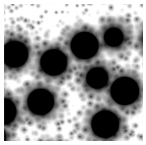


Hetero Terrain.









Fractal Brownian Motion.	Hybrid Multifractal.	Ridged Multifractal.	Multifractal.

The main noise types have four characteristics:

Dimension
: Fractal dimension controls the contrast of a layer relative to the previous layer in the texture. The higher the fractal dimension, the higher the contrast between each layer, and thus the more detail shows in the texture.

Lacunarity
: Lacunarity controls the scaling of each layer of the Musgrave texture, meaning that each additional layer will have a scale that is the inverse of the value which shows on the button. i.e. Lacunarity = 2 –> Scale = 1/2 original.

Octaves
: Octave controls the number of times the original noise pattern is overlaid on itself and scaled/contrasted with the fractal dimension and lacunarity settings.

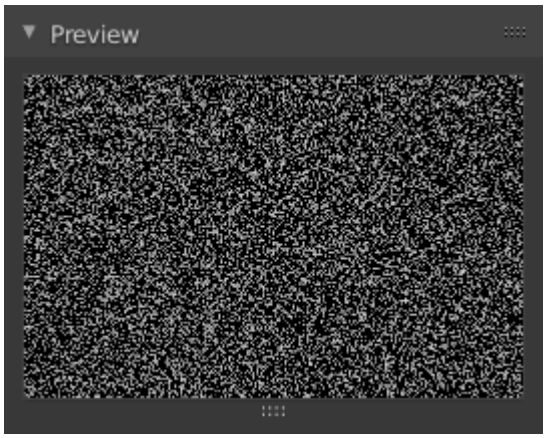
Intensity
: Light intensity. Called *Offset* for *Hetero Terrain*.

The *Hybrid Multifractal* and *Ridged Multifractal* types have these additional settings:

Offset
: Both have a “Fractal Offset” button that serves as a “sea level” adjustment and indicates the base height of the resulting bump map. Bump values below this threshold will be returned as zero.

Gain
: Setting which determines the range of values created by the function. The higher the number, the greater the range. This is a fast way to bring out additional details in a texture where extremes are normally clipped off.

Noise



Noise Texture panel.

Although this looks great, it is not Perlin Noise! This is a true, randomly generated Noise. This gives a different result every time, for every frame, for every pixel.

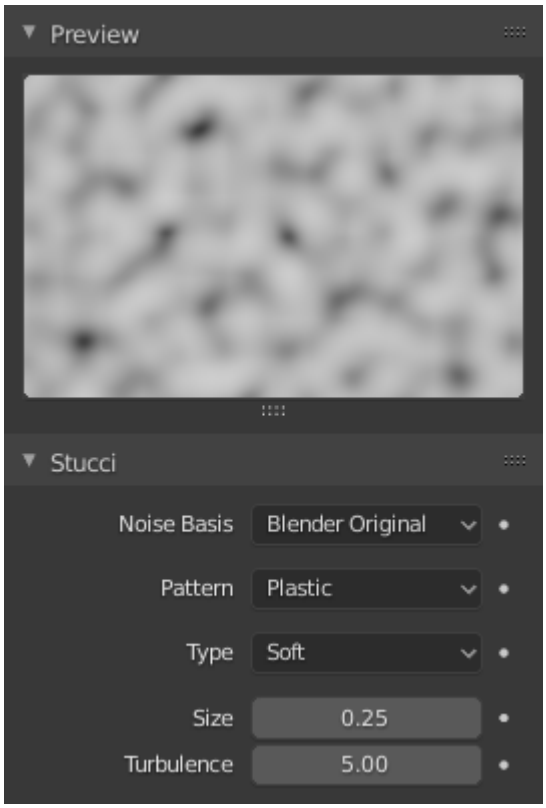
Options

There are no options for this noise.

Often used for
: White noise in an animation. This is not well suited if you do not want an animation. For material displacement or bump, use clouds instead.

Result(s)
: Intensity.

Stucci



Stucci Texture panels.

The *Stucci* texture is based on noise functions. It is often used for stone, asphalt, or oranges, normally for bump mapping to create grainy surfaces.

Options

Plastic / Wall In / Wall out
: Plastic is the standard Stucci, while the “walls” is where Stucci gets it name. This is a typical wall structure with holes or bumps.

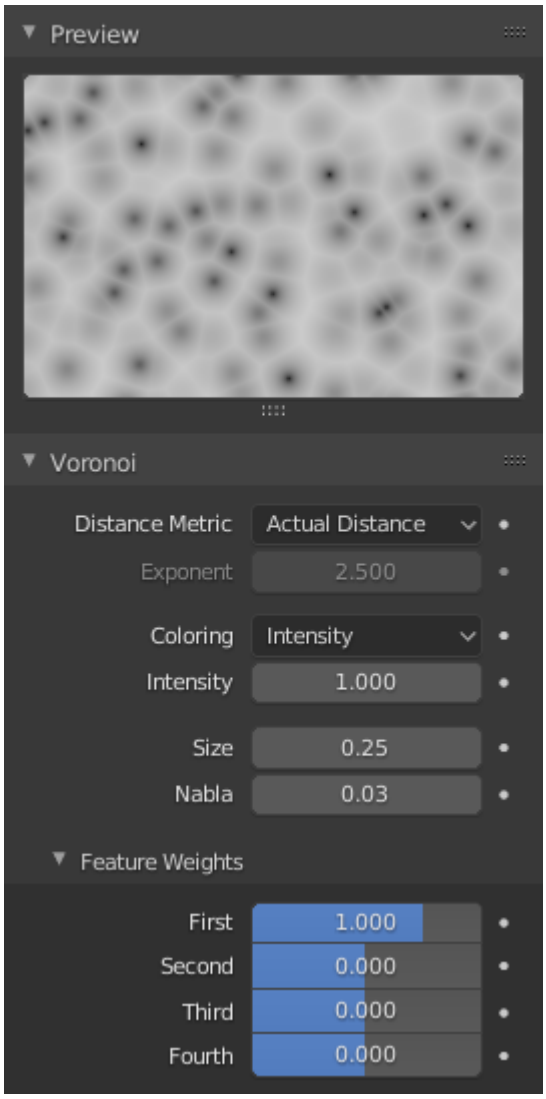
Soft / Hard
: There are two methods available for working with Noise.

Size
: Dimension of the Noise table.

Turbulence
: Depth of the *Stucci* calculations.

Voronoi

The Voronoi texture is used to generate very convincing Metal, especially the “Hammered” effect. Organic shaders (e.g. scales, veins in skin).



Voronoi Texture panels.

Options

Distance Metric

This procedural texture has seven Distance Metric options. These determine the algorithm to find the distance between cells of the texture. These options are:

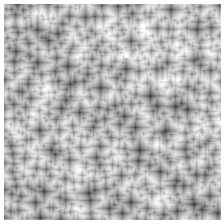
- Minkowski
- Minkowski 4
- Minkowski 1/2
- Chebychev
- Manhattan
- Distance Squared
- Actual Distance

The *Minkowski* setting has a user definable value (the *Exponent* button) which determines the Minkowski exponent *e* of the distance function:

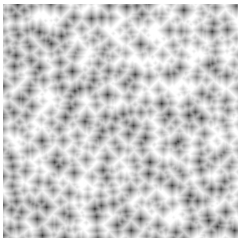
$$(x^e + y^e + z^e)^{1/e}$$

A value of one produces the *Manhattan* distance metric, a value less than one produces stars (at 0.5, it gives a *Minkowski 1/2*), and higher values produce square cells (at 4.0, it gives a *Minkowski 4*, at 10.0, a *Chebychev*). So nearly all Distance Settings are basically the same – a variation of *Minkowski*.

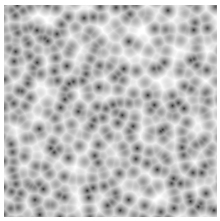
You can get irregularly-shaped rounded cells with the *Actual Distance* / *Distance Squared* options.



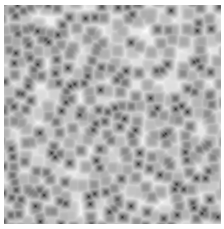
Minkowski Exponent: 0.5 (Minkowski 1/2).



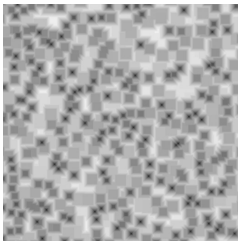
Minkowski Exponent: 1 (Manhattan).



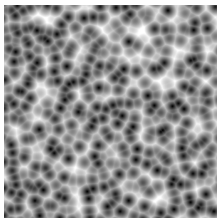
Minkowski Exponent: 2 (Actual Distance).



Minkowski Exponent: 4 (Minkowski 4).



Minkowski Exponent: 10 (Chebychev).



Distance Squared (more contrast than Actual Distance).

Feature Weights

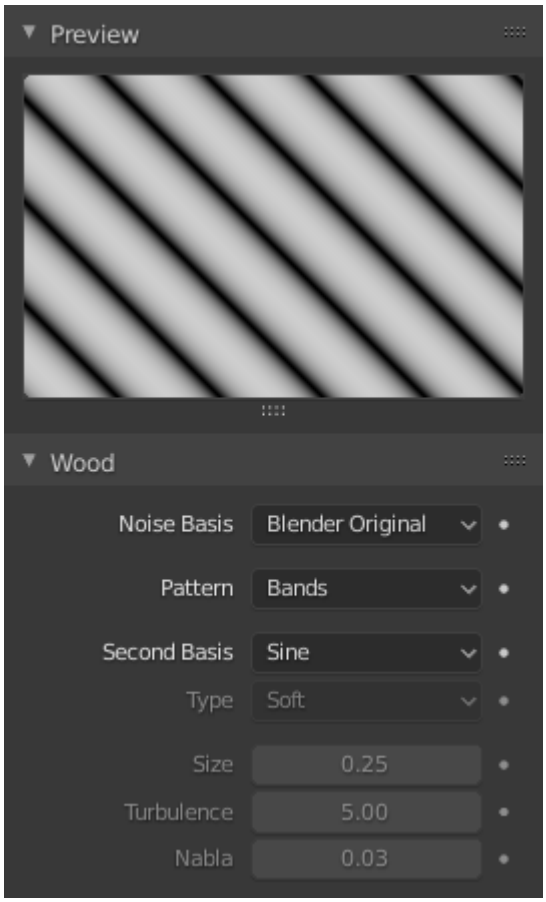
These four sliders at the bottom of the Voronoi panel represent the values of the four Worley constants, which are used to calculate the distances between each cell in the texture based on the distance metric. Adjusting these values can have some interesting effects on the end result…

Coloring

Four settings (*Intensity*, *Position*, *Position and Outline*, and *Position, Outline, and Intensity*) that can use four different noise basis as methods to calculate color and intensity of the texture output. This gives the Voronoi texture you create with the "Worley Sliders" a completely different appearance and is the equivalent of the noise basis setting found on the other textures.

Wood

The wood texture is used to generate wood and ring-shaped patterns.



Wood Texture panels.

Options

Noise Basis
: Shape of wave to produce bands.

  Sine, Saw, Triangle

Wood Type

Set the bands to either straight or ring-shaped, with or without turbulence.

Bands, Rings, Band Noise, Ring Noise

Noise Type

There are two methods available for the Noise function.

Soft, Hard

Size

Dimension of the Noise table.

Turbulence

Turbulence of the Band Noise and Ring Noise types.

Shader Nodes

- Introduction
  - Shaders
  - Textures
  - More
  - Open Shading Language
- Input
  - Ambient Occlusion
  - Attribute Node
  - Bevel Node
  - Camera Data Node
  - Fresnel Node
  - Geometry Node
  - Curves Info Node
  - Layer Weight Node
  - Light Path Node
  - Object Info Node
  - Particle Info Node
  - Point Info
  - RGB Node
  - Tangent Node
  - Texture Coordinate Node
  - UV Map Node
  - Value Node
  - Color Attribute Node
  - Volume Info Node
  - Wireframe Node
- Output
  - AOV Output Node
  - Material Output Node
  - Light Output Node
  - World Output Node
- Shader
  - Add Shader

    - Background
    - Diffuse BSDF
    - Emission
    - Glass BSDF
    - Glossy BSDF
    - Hair BSDF
    - Holdout
    - Mix Shader
    - Metallic BSDF
    - Principled BSDF
    - Principled Hair BSDF
    - Principled Volume
    - Ray Portal BSDF
    - Refraction BSDF
    - Specular BSDF
    - Subsurface Scattering
    - Toon BSDF
    - Translucent BSDF
    - Transparent BSDF
    - Sheen BSDF
    - Volume Absorption
    - Volume Scatter
- Texture
    - Brick Texture Node
    - Checker Texture Node
    - Environment Texture Node
    - Gabor Texture Node
    - Gradient Texture Node
    - IES Texture Node
    - Image Texture Node
    - Magic Texture Node
    - Musgrave Texture Node
    - Noise Texture Node
    - Point Density Node
    - Sky Texture Node
    - Voronoi Texture Node
    - Wave Texture Node

    - White Noise Texture Node
- Color
    - Brightness/Contrast Node
    - Gamma Node
    - Hue/Saturation/Value Node
    - Invert Color Node
    - Light Falloff Node
    - Mix Color Node
    - RGB Curves Node
- Vector
    - Bump Node
    - Displacement Node
    - Mapping Node
    - Normal Node
    - Normal Map Node
    - Vector Curves Node
    - Vector Displacement Node
    - Vector Rotate Node
    - Vector Transform Node
- Converter
    - Blackbody Node
    - Clamp Node
    - Color Ramp Node
    - Combine Color Node
    - Combine XYZ Node
    - Float Curve
    - Map Range Node
    - Math Node
    - Mix Node
    - RGB to BW Node
    - Separate Color Node
    - Separate XYZ Node
    - Shader To RGB
    - Vector Math Node
    - Wavelength Node
- Group
    - Group Input

    - Group Output
    - Node Groups
- Open Shading Language
    - Script Node
    - Writing Shaders
    - Closures
    - Attributes
    - Trace
    - Metadata

Introduction

Materials, lights and backgrounds are all defined using a network of shading nodes. These nodes output values, vectors, colors and shaders.

Shaders

An important concept to understand when building node setups is that of the *shader socket*. The output of all surface and volume shaders is a shader, describing lighting interaction at the surface or of the volume, rather than the color of the surface.

There are a few types of shaders available as nodes:

BSDF shader
: Describe light reflection, refraction and absorption at an object surface.

Emission shader
: Describe light emission at an object surface or in a volume.

Volume shader
: Describe light scattering inside a volume.

Background shader
: Describe light emission from the environment.

Each shader node has a color input, and outputs a shader. These can then be mixed and added together using Mix and Add Shader nodes. No other operations are permitted. The resulting output can then be used by the renderer to compute all light interactions, for direct lighting or global illumination.

See also

Shaders

Textures

Blender has various built in procedural texture nodes, with texture coordinates and various parameters as input, and a color or value as output. No texture data-blocks are needed; instead node groups can be used for reusing texture setups.

For UV mapping and texture painting in the 3D Viewport, the Image Texture node must be used. When setting such a node as active, it will be displayed in the 3D Viewport while using Texture color mode. This method can be used to preview painted textures while texture painting.

The default texture coordinates for all nodes are Generated coordinates, except for Image textures that use UV coordinates by default. Each node includes some options to modify the texture mapping and resulting color, and these can be edited in the texture properties.

See also

Textures

More

Nodes for geometric data, texture coordinates, layering shaders and non-physically-based tricks can be found in:

- Vector Nodes
- Color Nodes
- Converter Nodes

Open Shading Language

In Cycles, custom nodes can be written using the Open Shading Language.

See also

Open Shading Language

Input

- Ambient Occlusion
- Attribute Node
- Bevel Node
- Camera Data Node
- Fresnel Node
- Geometry Node
- Curves Info Node
- Layer Weight Node
- Light Path Node
- Object Info Node
- Particle Info Node
- Point Info
- RGB Node
- Tangent Node
- Texture Coordinate Node
- UV Map Node
- Value Node
- Color Attribute Node
- Volume Info Node
- Wireframe Node

Ambient Occlusion

The *Ambient Occlusion* shader computes how much the hemisphere above the shading point is occluded. This can be used for procedural texturing, for example to add weathering effects to corners only.



For Cycles, this is an expensive shader and can slow down render significantly. If render time is a concern, using Pointiness from the Geometry node or baking Ambient Occlusion will result in faster renders.

Note

Cycles Only

The Ambient Occlusion node will not produce a valid result when:

- The object is either a Caustic caster or Caustic receiver while the scene contains an active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- Open Shading Language is active while using the OptiX rendering backend.

Inputs

Color
: Tint for AO output color.

Distance
: Distance up to which other objects are considered to occlude the shading point.

Normal

Normal used for ambient occlusion; if nothing is connected the default shading normal is used.

Properties

Samples
: Number of samples to use for ray-traced ambient occlusion sampling. Keep as low as possible for optimal performance.

Inside
: Detect convex rather than concave shapes, by computing occlusion inside mesh.

Only Local *Cycles Only*
: Only detect occlusion from the object itself, and not others.

Outputs

Color
: Ambient occlusion with color tint.

AO
: Ambient occlusion factor without color tint.

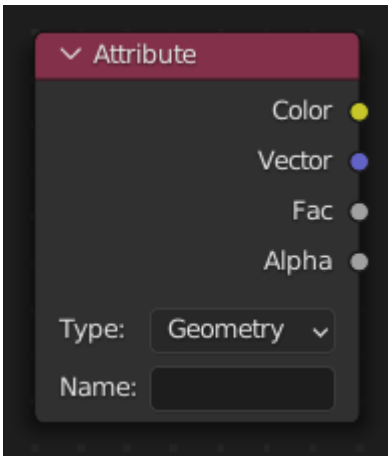
Example



White AO shader.

Attribute Node

The *Attribute* node allows you to retrieve attributes attached to an object or mesh.



Inputs

This node has no inputs.

Properties

Name
: Name of the attribute.

Type
: Specifies the type of the attribute.

Geometry:: The attribute is associated with the geometry of the object, and its value varies from vertex to vertex, or within the volume of the object. Most geometry attributes are directly accessible through the various input nodes, except for these:

Ocean Foam
: Gives a scalar defining where foam might appear when using an Ocean Modifier. This depends on the name you give this property.

See also
For a full list of options see This Thread on the Blender Stack Exchange.

Object:: The attribute name specifies a custom property name, or an RNA path to a built-in property (like the single property driver variables).
The values of attributes of this type are defined once per object. The name or path is looked up first in the object data-block, followed by the mesh data-block if not found. Custom properties have priority over built-in ones.
The property value must be an integer, float, boolean, or a vector of 1 to 4 floats or ints; properties of other types are ignored. If a suitable property is not found, all sockets of the node, including *Alpha*, output 0.

Tip
The `color` attribute will output the value of the Color field in the Viewport Display panel of the object, unless overridden by a custom property.

Instancer:: Similar to *Object*, but the attribute is looked up in the instancer particle system settings, followed by Geometry Node instance attributes (searching from the innermost instancing layer to outer ones), and finally in the instancer object. If the current object is not instanced, or the property is not found, it falls back to the *Object* mode.

Warning
Currently only up to 4 layers of Geometry Node instancing are searched.

View Layer::

The attribute is looked up in the current View Layer, Scene and World, using the same lookup logic as *Object*, and likewise producing all zero outputs including *Alpha* if not found. Attributes of this type have the same uniform value throughout the whole Render Layer.

Tip
This gives access to a number of useful built-in properties, for example:

`color` or `world.color`
: Outputs the value of the Color field in the Viewport Display panel of the World properties.

`render.resolution_x`, `render.resolution_y`
: Outputs the current rendering resolution.

`camera.data.angle_x`, `camera.data.angle_y`,
: Outputs the effective field of view of the active Camera.

See also
An alternative method to access the same set of properties is to use driver Context Properties, possibly with a manually emulated lookup fallback chain.

Outputs

Color
: RGB color interpolated from the attribute.

Vector
: XYZ vector interpolated from the attribute.

Factor
: Scalar value interpolated from the attribute.

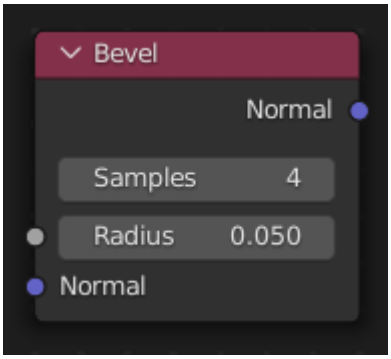
Alpha
: Alpha channel of the attribute, when available. If the attribute has no alpha channel, generally defaults to 1.

Warning

Currently, only *View Layer* attributes are supported in shaders used for the World or Light Objects.

Bevel Node

Cycles Only



The Bevel shader node can be used for rendering rounded corners. Like bump mapping, this does not modify the actual geometry, only the shading is affected. Slight rounding on edges helps to capture specular highlights that you would also see in the real world.

Note that this is a very expensive shader, and may slow down renders by 20% even if there is a lot of other complexity in the scene. For that reason, we suggest to mainly use this for baking or still frame renders where render time is not as much of an issue. The Bevel Modifier is a faster option when it works, but sometimes fails on complex or messy geometry.

Note

The Bevel node will not produce a valid result when:

- **The object is either a Caustic caster**
  or Caustic receiver while the scene contains an active Caustic caster, Caustic receiver, and Shadow Caustic Light.
- Open Shading Language is active while using the OptiX rendering backend.

Inputs

Radius
: Width of the bevel effect on edges.

Normal
: Normal to apply bevel on top of, to be combined with a Bump Node for example. When not connected, uses the surface normal.

Properties

Samples
: Number of samples to take for each shader evaluation. More samples give more accurate results, but are also slower to render. The default value of 4 works well for most cases, with any noise resolved by using more AA samples.

Outputs

Normal
: Standard normal output.

Examples



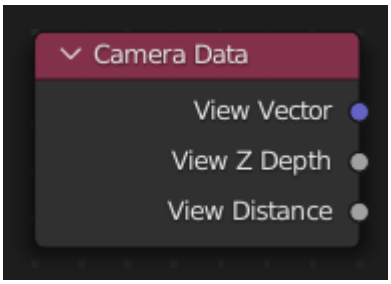
A minimal node setup for working with the Bevel node.



Bevel shader bringing out specular highlights on the edges.

Camera Data Node

The *Camera Data* node returns information about the shading point relative to the camera. This could be used for example to change the shading of objects further away from the camera, or make custom fog effects.



Inputs

This node has no inputs.

Properties

This node has no properties.

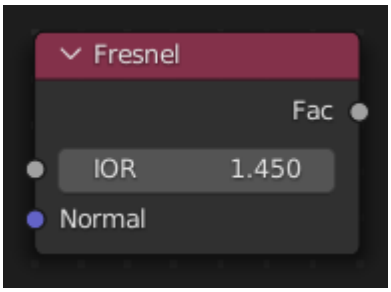
Outputs

View Vector
: A normalized vector, in camera space, from the camera to the shading point.

View Z Depth
: The distance each pixel is away from the camera.

View Distance
: Distance from the camera to the shading point.

Fresnel Node



The *Fresnel* or *Dielectric Fresnel* node computes how much light is reflected off a layer, where the rest will be refracted through the layer. The resulting weight can be used for layering shaders with the *Mix Shader* node. It is dependent on the angle between the surface normal and the viewing direction.

The most common use is to mix between two BSDFs using it as a blending factor in a Mix Shader node. For a simple glass material you would mix between a glossy refraction and glossy reflection. At grazing angles more light will be reflected than refracted as happens in reality.

For a two-layered material with a diffuse base and a glossy coating, you can use the same setup, mixing between a diffuse and glossy BSDF. By using the Fresnel as the blending factor you are specifying that any light which is refracted through the glossy coating layer would hit the diffuse base and be reflected off that.

Inputs

IOR
: Index of refraction (IOR) of the material being entered.

Normal
: Input meant for plugging in bump or normal maps which will affect the output.

Properties

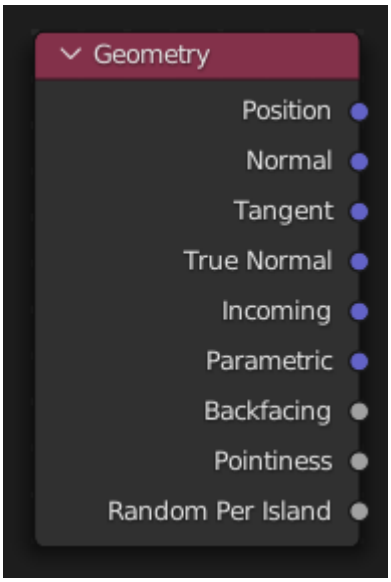
This node has no properties.

Outputs

Factor
: Fresnel weight, indicating the probability with which light will reflect off the layer rather than passing through.

Geometry Node

The *Geometry* node gives geometric information about the current shading point. All vector coordinates are in *World Space*. For volume shaders, only the position and incoming vector are available.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Position
: Position of the shading point.

Normal
: Shading normal at the surface (includes smooth normals and bump mapping).

Tangent
: Tangent at the surface.

True Normal
: Geometry or flat normal of the surface.

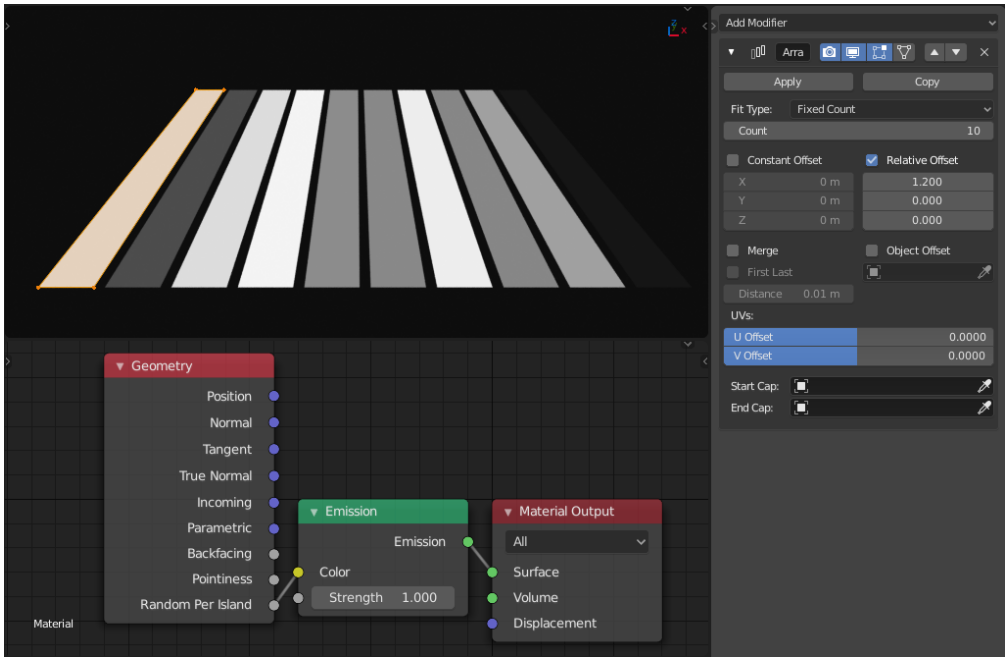
Incoming
: Vector pointing towards the point the shading point is being viewed from.

Parametric
: Parametric coordinates of the shading point on the surface. To area lights it outputs its UV coordinates in planar mapping and in spherical coordinates to point lights.

Backfacing
: 1.0 if the face is being viewed from the back side, 0.0 for the front side.

Pointiness *Cycles Only*
: An approximation of the curvature of the mesh per vertex. Lighter values indicate convex angles, darker values indicate concave angles. It allows you to do effects like dirt maps and wear-off effects.

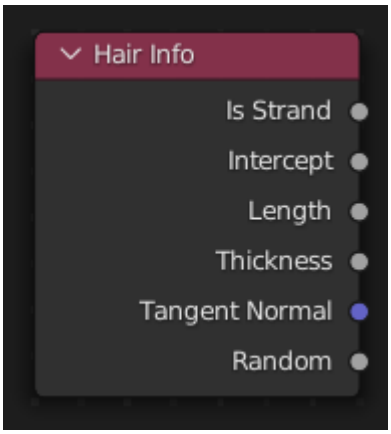
Random per Island *Cycles Only*
: A random value for each connected component (island) of the mesh. It is useful to add variations to meshes composed of separated units like tree leaves, wood planks, or curves of multiple splines.



Get a random value for each instance of the mesh when using an Array modifier.

Curves Info Node

The *Curves Info* node gives access to Hair information.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Strand
: Outputs 1 when the shader is acting on a strand, otherwise 0.

Intercept
: The point along the strand where the ray hits the strand (1 at the tip and 0 at the root).

Length
: The total measurement from the root to the tip of the strand, interpreted as a grayscale value from 0 to infinity.

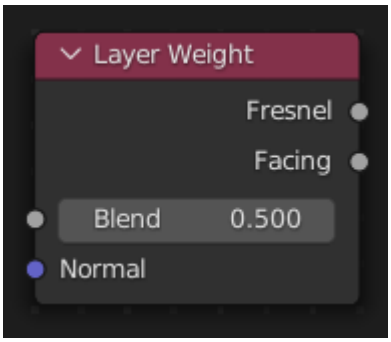
Thickness
: The thickness of the strand at the point where the ray hits the strand.

Tangent Normal
: Tangent normal of the strand.

Random
: A random per-curve value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the curve's color.

Layer Weight Node

The *Layer Weight* node outputs a weight typically used for layering shaders with the *Mix Shader* node.



Inputs

Blend
: Bias the output towards all 0 or all 1. Useful for uneven mixing of shaders.

Normal
: Input meant for plugging in bump or normal maps which will affect the output.

Properties

This node has no properties.

Outputs

Fresnel
: Dielectric Fresnel weight, useful for example for layering diffuse and glossy shaders to create a plastic material. This is like the Fresnel node, except that the input of this node is in the often more convenient 0.0 to 1.0 range.

Facing

Weight that blends from the first to the second shader as the surface goes from facing the viewer to viewing it at a grazing angle.

Light Path Node

The *Light Path* node is used to find out for which kind of incoming ray the shader is being executed; particularly useful for non-physically-based tricks. More information about the meaning of each type is in the Light Paths documentation.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Is Camera Ray
: 1.0 if shading is executed for a camera ray, otherwise 0.0.

Is Shadow Ray
: 1.0 if shading is executed for a shadow ray, otherwise 0.0.

Is Diffuse Ray
: 1.0 if shading is executed for a diffuse ray, otherwise 0.0.

Is Glossy Ray
: 1.0 if shading is executed for a glossy ray, otherwise 0.0.

Is Singular Ray *Cycles Only*
: 1.0 if shading is executed for a singular ray, otherwise 0.0.

Is Reflection Ray *Cycles Only*
: 1.0 if shading is executed for a reflection ray, otherwise 0.0.

Is Transmission Ray *Cycles Only*
: 1.0 if shading is executed for a transmission ray, otherwise 0.0.

Ray Length *Cycles Only*
: Distance traveled by the light ray from the last bounce or camera.

Ray Depth
: Number of times the ray has been reflected or transmitted on interaction with a surface.

  Note

  Passing through a transparent shader does not count as a normal "bounce".

Diffuse Depth *Cycles Only*
: Number of times the ray has gone through diffuse reflection or transmission.

Glossy Depth *Cycles Only*
: Number of times the ray has gone through glossy reflection or transmission.

Transparent Depth *Cycles Only*
: Number of times the ray has gone through a transparent surface.

Transmission Depth *Cycles Only*
: Number of times the ray has gone through a transmissive surface. A typical use case is to avoid black spots in the render (caused by rays hitting the bounce limit) by switching from a transmissive to a diffuse shader past a certain point. See Mix Shader.

EEVEE Support

EEVEE has no real concept of rays, but in order to ease the workflow between Cycles and EEVEE, some of the outputs are supported in particular cases.

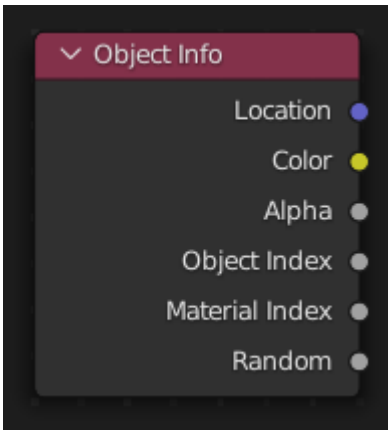
- *Is Camera*: Supported.
- *Is Shadow*: Supported.
- *Is Diffuse*: Supported.
- *Is Glossy*: Supported.
- *Is Singular*: Not supported. Same as *Is Glossy*.
- *Is Reflection*: Not supported. Same as *Is Glossy*.
- *Is Transmission*: Not supported. Same as *Is Glossy*.
- *Ray Length*: Distance from the camera to the shading point.
- *Ray Depth*: Indicates the current bounce when baking the light cache.
- *Diffuse Depth*: Same as Ray Depth but only when baking diffuse light.
- *Glossy Depth*: Same as Ray Depth but only when baking specular light.
- *Transparent Depth*: Not supported. Defaults to 0.
- *Transmission Depth*: Not supported. Same as *Glossy Depth*.

Note

Is Glossy does not work with Screen Space Reflections/Refractions but does work with reflection planes (whether used with SSR or not).

Object Info Node

The *Object Info* node gives information about the object instance. This can be useful to give some variation to a single material assigned to multiple instances, either manually controlled through the object index, based on the object location, or randomized for each instance. For example a Noise texture can give random colors or a Color Ramp can give a range of colors to be randomly picked from.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Location
: Location of the object in world space.

Color
: Object color, same as *Color* in the *Properties* ▸ *Object Properties* ▸ *Viewport Display*.

Alpha
: The Alpha Channel component of the object's viewport display color (see the *Color* output for more details).

Object Index
: Object pass index, same as *Pass Index* in the *Properties* ▸ *Object Properties* ▸ *Relations*.

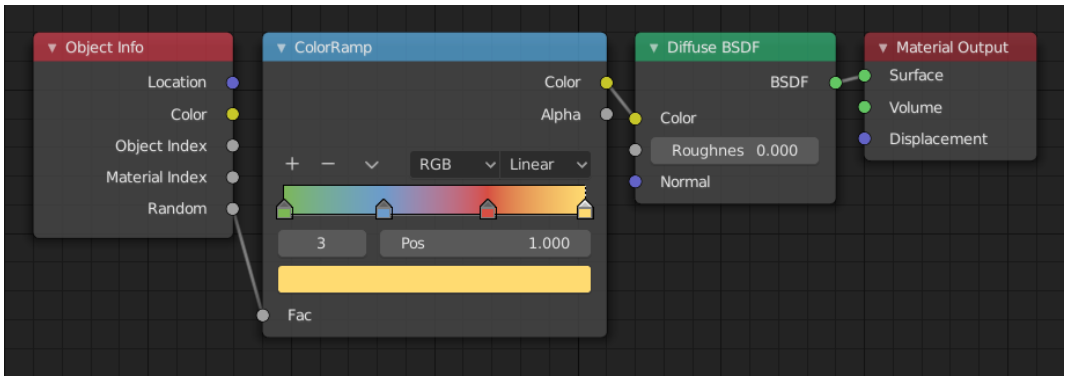
Material Index
: Material pass index, same as *Pass Index* in the *Properties* ▸ *Material* ▸ *Settings*.

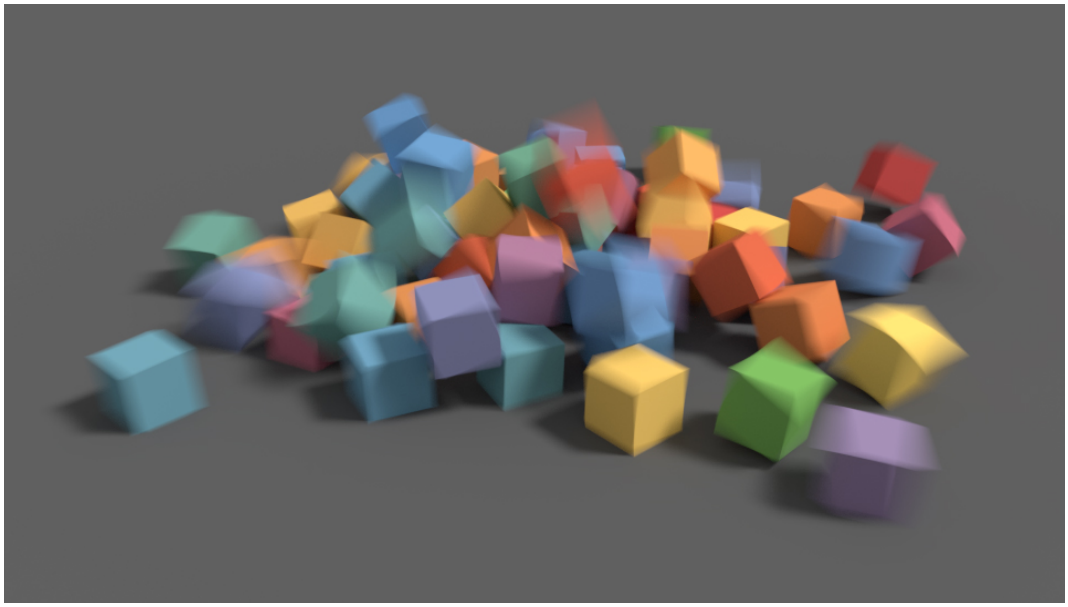
Random
: Random number unique to a single object instance. Output is a Float between 0.0 and 1.0

Note

Note that this node only works for material shading nodes; it does nothing for light and world shading nodes.

Example

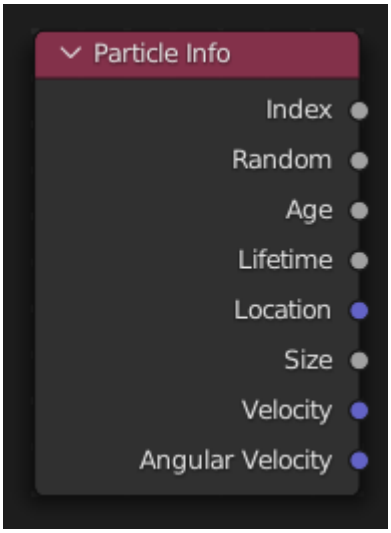




Example blend-file.

Particle Info Node

Cycles Only



The *Particle Info* node can be used in the material node tree for objects that are used as the instancing objects, when you use *Object* or *Collection* Render mode of a particle system.

This node gives access to the data of the particle that spawned the object instance. It can be useful to give some variation to a single material assigned to multiple instances of instancing object.

Note

This node currently only supports parent particles. Info from child particles is not available.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Index
: Index number of the particle (from 0 to number of particles).

Random
: A random per-particle value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the particle color.

Age
: Age of the particle in frames.

Lifetime
: Total lifespan of the particle in frames.

Location
: Location of the particle.

Size
: Size of the particle.

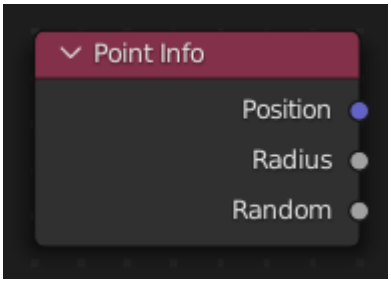
Velocity
: Velocity of the particle.

Angular Velocity
: Angular velocity of the particle.

Point Info

Cycles Only

The *Point Info* node can be used in the material node tree for point cloud objects and gives access to the data of individual points. It can be useful to give some variation to a single material assigned a point cloud object.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Location
: Location of the particle.

Radius
: Size of the particle.

Random
: A random per-point value in the range from 0 to 1. It can for example be used in combination with a color ramp, to randomize the point color.

RGB Node



Inputs

This node has no input sockets.

Properties

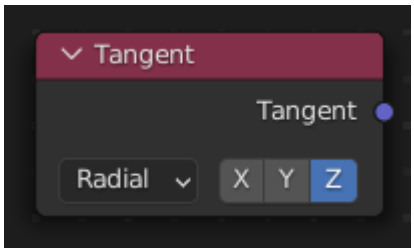
The RGB node uses the color picker widget.

Outputs

Color / RGBA
: A single RGBA color value.

Tangent Node

The *Tangent* node generates a tangent direction for the Anisotropic BSDF.



Inputs

This node has no inputs.

Properties

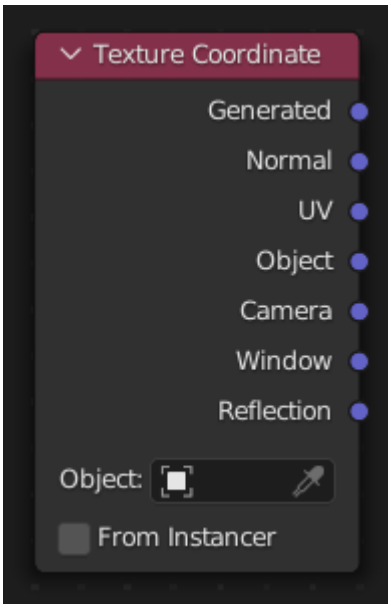
Direction Type
: The tangent direction can be derived from a cylindrical projection around the X, Y, or Z axis (radial), or from a manually created UV Map for full control.

Outputs

Tangent
: The tangent direction vector.

Texture Coordinate Node

The *Texture Coordinate* node is commonly used for the coordinates of textures, typically used as inputs for the *Vector* input for texture nodes.



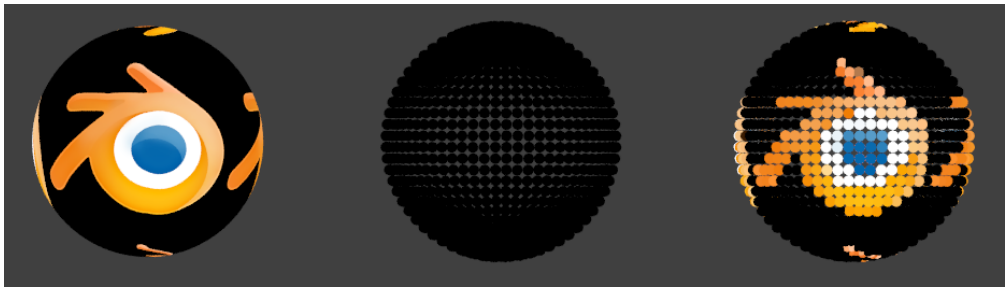
Inputs

This node has no inputs.

Properties

Object
: Specific object to use for object space coordinates. This only affects the *Object* output.

From Instancer *Cycles Only*
: If the object is generated by instancing from vertices or faces, use texture coordinates from instancer. This only affects the *Generated* and *UV* outputs.



From left to right: Sphere with a UV-mapped texture. Small spheres instanced to the faces of the textured sphere using instancing from faces. Small spheres with *From Instancer* enabled, using the UV map of the large sphere.

Note

From Instancer only works with the UV output when the object is instanced, either from particles or from faces.

Outputs

Generated

Automatically-generated texture coordinates from the vertex positions of the mesh without deformation, keeping them sticking to the surface under animation. Range from 0.0 to 1.0 over the bounding box of the undeformed mesh. See Texture Spaces for more information.

Normal

Object space normal, for texturing objects with the texture staying fixed on the object as it transformed. The Normal output can be used on Point and Spot lights. The coordinates will take the rotation of the light into account.

UV

UV texture coordinates from the active render UV map. See UV Mapping for more information.

Note

In order to select UV map other than the active map you must use the UV Map node.

Object
: Uses an object as a source for coordinates. Often used with an empty, this is an easy way to place a small image at a given point on the object. This object can also be animated, to move a texture around or through a surface.

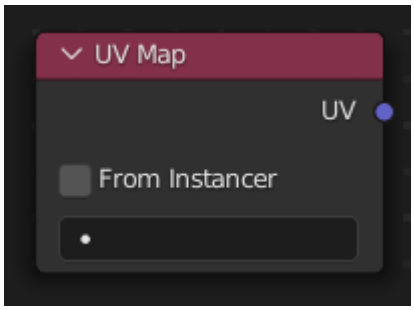
Camera
: Position coordinate in camera space.

Window
: Location of shading point on the screen, ranging from 0.0 to 1.0 from the left to right side and bottom to top of the render. This is well suited for blending two objects.

Reflection
: Uses the direction of the reflection vector as coordinates. This is useful for adding reflection maps. You will need this input when using environment maps.

UV Map Node

The *UV Map* node is used to retrieve specific UV maps. Unlike the Texture Coordinate Node which only provides the active UV map, this node can retrieve any UV map belonging to the object using the material.



Inputs

This node has no inputs.

Properties

From Instancer *Cycles Only*
: See the From Instancer option of the Texture Coordinate Node.

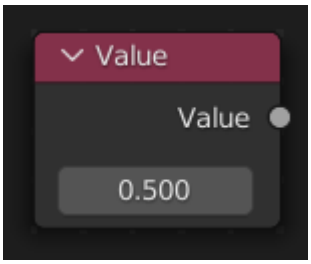
UV Map
: UV map to use.

Outputs

UV
: UV mapping coordinates from the specified UV map.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.



Inputs

This node has no input sockets.

Properties

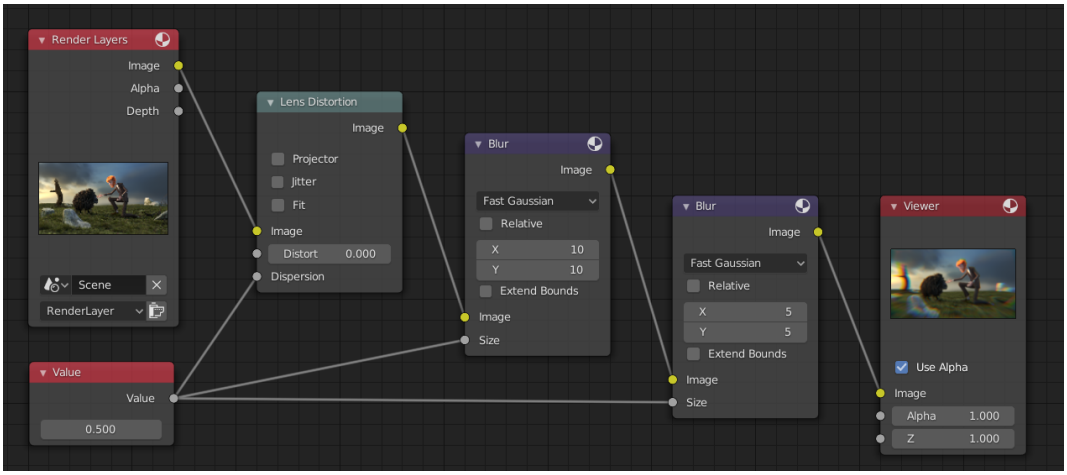
Single numerical value (floating-point).

Outputs

Value
: The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.



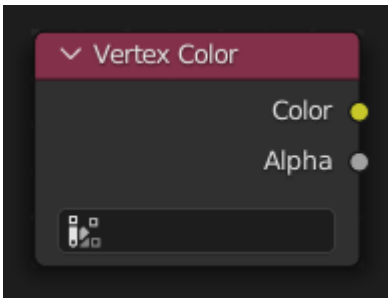
Example of the *Value* node.

Tip

From this you can also make different values proportional to each other by adding a Math Node in between the different links.

Color Attribute Node

The *Color Attribute* node provides access to Color Attributes as well as their alpha value.



Inputs

This node has no inputs.

Properties

Color Attribute
: The target Color Attribute. The listed Color Attributes are those of the mesh of the active object. If the active object has no mesh, a warning will be displayed. If the property is marked in red, it means the Color Attribute is not available in the mesh of the active object, but it may be available in other meshes of objects that share this material!

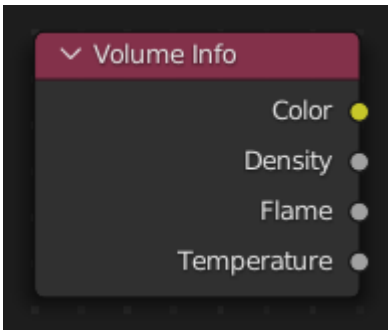
Outputs

Color
: Standard color output.

Alpha
: Alpha value.

Volume Info Node

The *Volume Info* node provides information about *Smoke Domains*.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

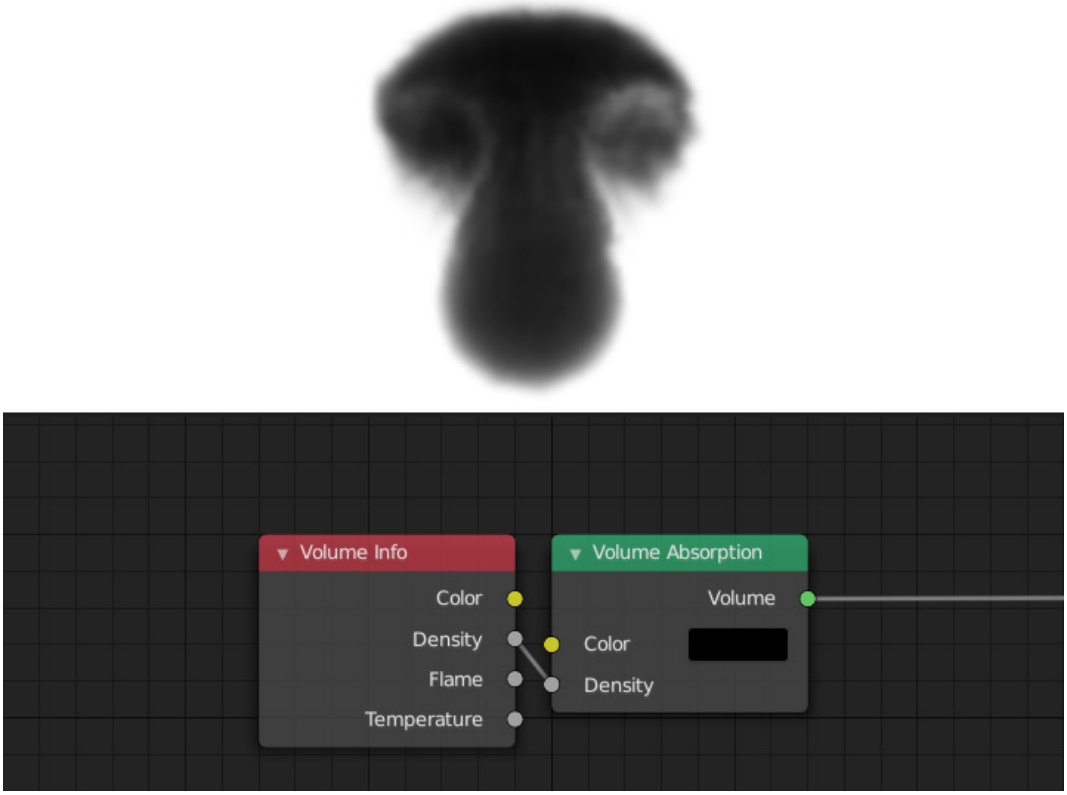
Color
: Gives the color of the smoke inside the Fluid Domain. The color and vector outputs are the same. The Factor output is an average of the channels.

Density
: Gives a scalar defining the density of any smoke inside the Fluid Domain.

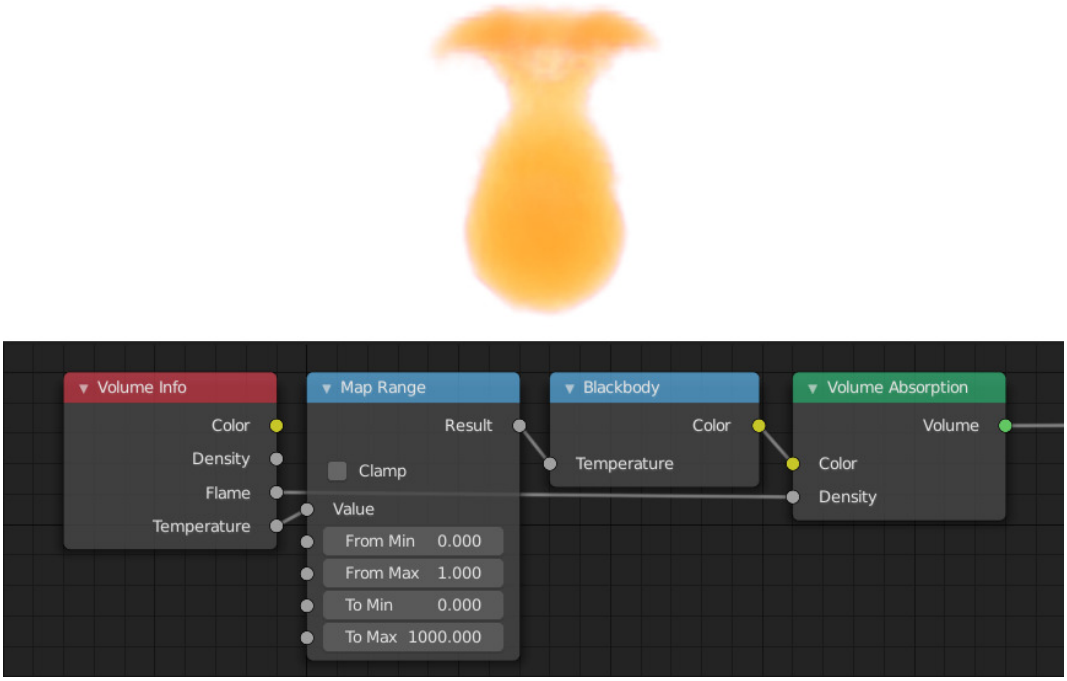
Flame
: Gives a scalar defining the density of any fire inside the Fluid Domain. All three outputs are the same.

Temperature
: Gives a scalar defining the temperature of the volume. Values in the range 0 - 1 map to 0 - 1000 kelvin. This may be used to render physically-based fire with the Blackbody or Principled Volume shaders. All three outputs are the same.

Example



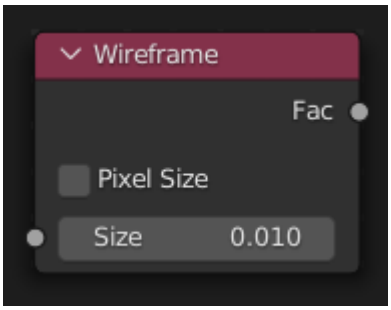
Smoke density.



Computing the color of fire using the *Blackbody* node. Since the *Blackbody* node expects its input in Kelvin, the temperature output has to be remapped first.

Wireframe Node

The Wireframe node is used to retrieve the edges of an object as it appears to Cycles. As meshes are triangulated before being processed by Cycles, topology will always appear triangulated when viewed with the Wireframe node.



Inputs

This node has no inputs.

Properties

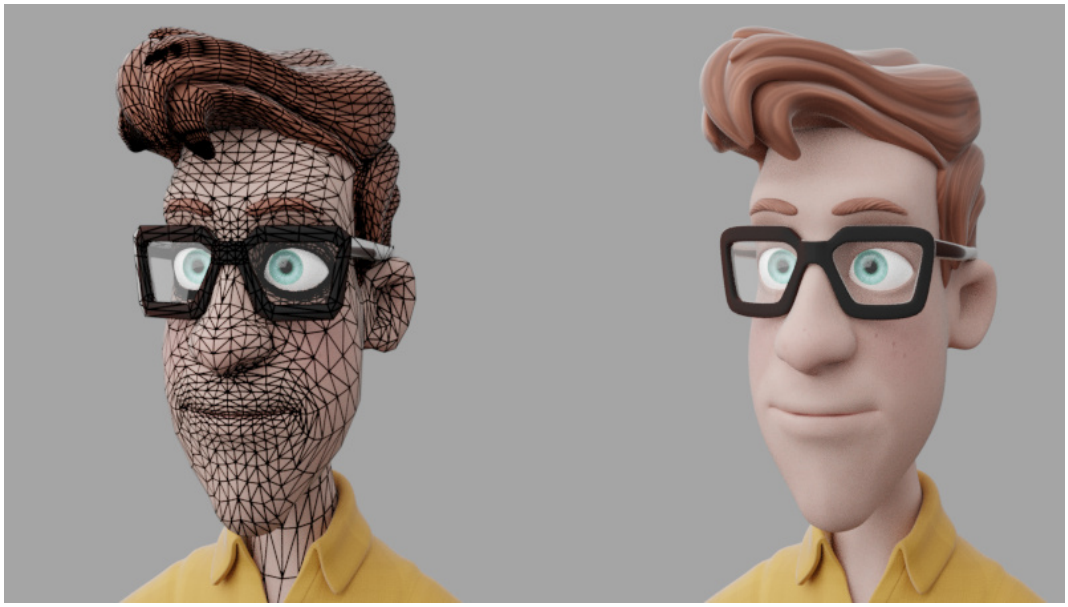
Pixel Size
: When enabled, the size of edge lines is set in screen space.

Size
: Thickness of the edge lines.

Outputs

Factor
: Black-and-white mask showing white lines representing edges according to the object's Topology.

Examples



Using the Wireframe node to showcase the topology of a mesh.

Output

Output nodes are the final node in every node tree. Although you can add more than one, only one will be used (indicated by a colored or darkened header). Output nodes are always preceded by Shaders except in the case of the Displacement of a Material Output.

- AOV Output Node
- Material Output Node
- Light Output Node
- World Output Node

AOV Output Node



Shader AOVs (Arbitrary Output Variables) provide custom render passes for arbitrary shader node components. As an artist this can be a good way to debug or tweak very fine details of a scene in post-processing. To use shader AOVs create the pass in the Shader AOV panel then reference that pass with the *AOV Output* shading node. Shader AOVs can be added or removed in the *Shader AOV* panel.

Tip

The *AOV Output* node can be used in Material and World shader nodes.

Inputs

Color
: Output a color variable; as the name suggest can be used for a color but also a normal value.

Value
: Output a single numerical value.

Properties

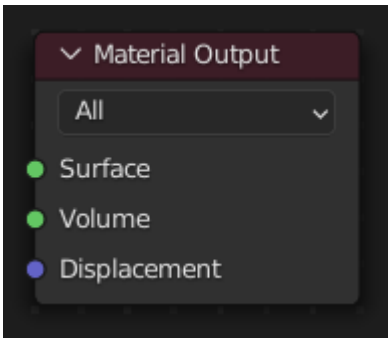
Name
: The name of the render pass to assign the input value to. This property has the same *Name* that is specified in the Shader AOV panel.

Outputs

This node has no outputs.

Material Output Node

The *Material Output* node is used to output surface material information to a surface object.



Inputs

Surface
: Shading for the surface of the object.

Volume
: Shading for the volume inside the object.

Displacement
: Used to create bump mapping or actual subdivided displacement.

Thickness *EEVEE*
: Used to approximate the inner geometry structure of the object without heavy computation. This is currently used for Subsurface Scattering, Translucent BSDF, Refraction BSDF, and the nodes containing these effects.

  If no value is plugged into the output node, a default thickness based on the smallest dimension of the object is used. If a value is connected it will be used as object space thickness (i.e. scaled by object transform). A value of zero will disable the thickness approximation and treat the object as having only one interface.

  This output is only used by the EEVEE render engine.

Note

- The thickness is used to skip the inner part of the object.
- Refraction will not refract objects inside the thickness distance.
- Shadow casting object will not cast shadow within the thickness distance.

Tip

- For large or compound meshes (e.g. vegetation), the thickness should be set to the thickness of individual parts (e.g. leaves, grass blades).
- Thickness can be baked to textures or custom attributes for more accurate result.

See also

Thickness Mode – controls how the thickness value is used.

Properties

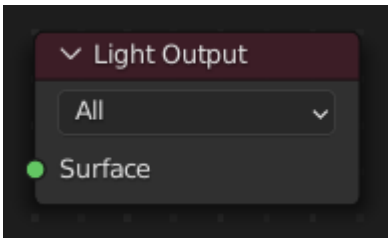
Target
: Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

This node has no outputs.

Light Output Node

The *Light Output* node is used to output light information to a light object.



Inputs

Surface
: Shading for the surface of the light object.

Properties

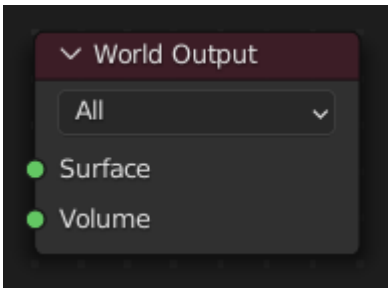
Target
: Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

This node has no outputs.

World Output Node

The *World Output* node is used to output light a color information to the scene's World.



Inputs

Surface
: The appearance of the environment, usually preceded by a Background shader.

Volume
: Used to add volumetric effects to the world. See the shaders Volume Absorption and Volume Scatter for more information.

> **Note**
>
> It is not possible to have an HDR and volumetric due to the fact that HDR's are assumed to be an infinite distance from the camera.

Properties

Target
: Render engine the input shaders are used for. By default shaders are shared between Cycles and EEVEE, with multiple output nodes specialized shader setups can be created for each.

Outputs

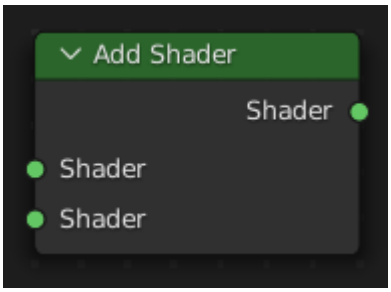
This node has no outputs.

Shader

- Add Shader
- Background
- Diffuse BSDF
- Emission
- Glass BSDF
- Glossy BSDF
- Hair BSDF
- Holdout
- Mix Shader
- Metallic BSDF
- Principled BSDF
- Principled Hair BSDF
- Principled Volume
- Ray Portal BSDF
- Refraction BSDF
- Specular BSDF
- Subsurface Scattering
- Toon BSDF
- Translucent BSDF
- Transparent BSDF
- Sheen BSDF
- Volume Absorption
- Volume Scatter

Add Shader

The *Add* node is used to add two *Shaders* together.



Inputs

Shaders
: Standard shader inputs.

Properties

This node has no properties.

Outputs

Shader
: Standard shader output.

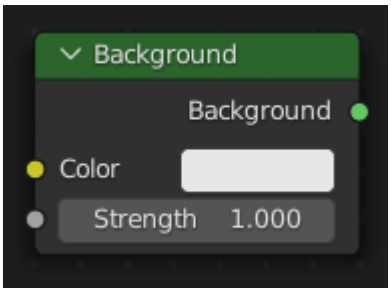
Example



A mix of a glossy and a diffuse shader makes a nice ceramic material.

Background

The *Background* shader node is used to add background light emission. This node should only be used for the world surface output.



Inputs

Color
: Color of the emitted light.

Strength
: Strength of the emitted light.

Properties

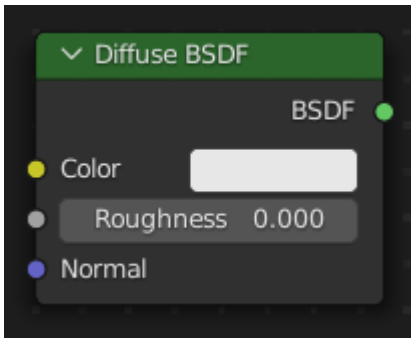
This node has no properties.

Outputs

Background
: Standard shader output.

Diffuse BSDF

The *Diffuse* BSDF node is used to add Lambertian and Oren-Nayar diffuse reflection.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected or transmitted for each wavelength.

Roughness *Cycles Only*
: Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

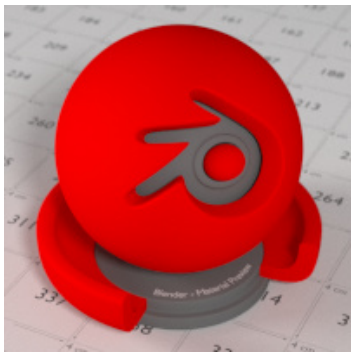
Properties

This node has no properties.

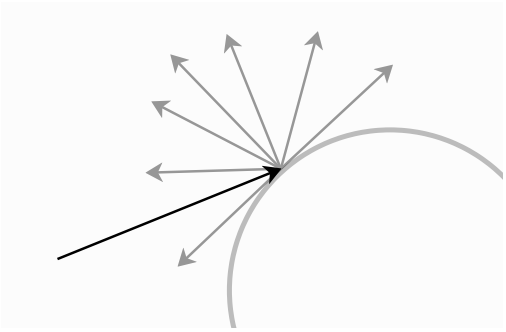
Outputs

BSDF
: Standard shader output.

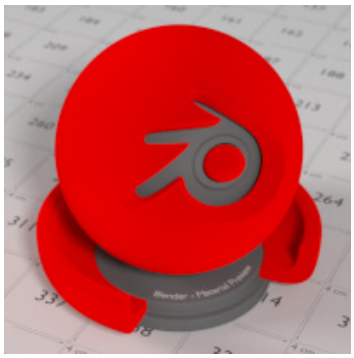
Examples



Lambertian reflection.

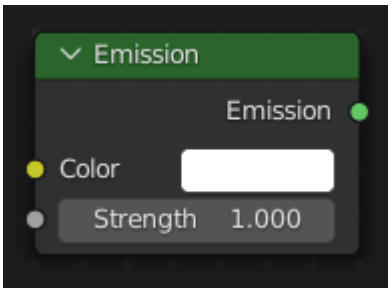


Diffuse shader behavior.



Oren-Nayar reflection.

Emission



The *Emission* node is used to add Lambertian emission shader. This can for example, be used for material and light surface outputs.

Light strength for point, spot and area lights is specified in Watts.

Sun lights are specified in Watts/m^2, which require much smaller values like 1 W/m^2. This can be confusing, but specifying strength in Watts would not have been convenient; the real sun for example has strength 384.6×10^{24}W. Emission shaders on meshes are also in Watts/m^2.

Inputs

Color
: Color of the emitted light.

Strength
: Strength of the emitted light. For point and area lights, the unit is Watts. For materials, a value of 1.0 will ensure that the object in the image has the exact same color as the Color input, i.e. make it ‘shadeless’.

Properties

This node has no properties.

Outputs

Emission

The Emission shader output can both be plugged into the *Surface Input* as well as the *Volume Input* of the Material Output node.

Examples



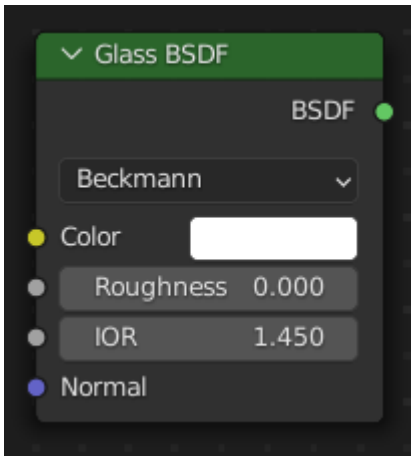
Emission shader, with strength at 1.0.



Emission shader, with strength at 3.0.

Glass BSDF

The *Glass* BSDF is used to add a Glass-like shader mixing refraction and reflection at grazing angles. Like the transparent shader, only pure white will make it transparent. The glass shader tends to cause noise due to caustics. Since the Cycles path tracing integrator is not very good at rendering caustics, it helps to combine this with a transparent shader for shadows; for more details see here.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Roughness
: Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

IOR
: Index of refraction (IOR) defining how much the ray changes direction. At 1. 0 rays pass straight through like transparent; higher values give more refraction.

Normal
: Normal used for shading.

Properties

Distribution

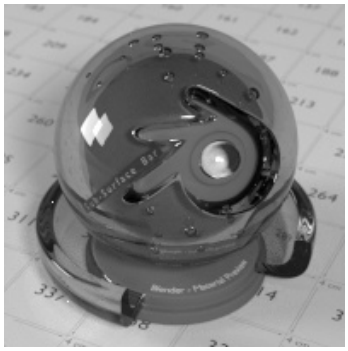
Microfacet distribution to use.

GGX:: GGX microfacet distribution.
Multiscatter GGX:: Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.
Beckmann:: *Cycles Only* Beckmann microfacet distribution.

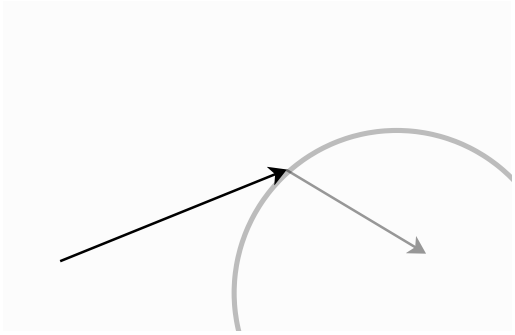
Outputs

BSDF
Standard shader output.

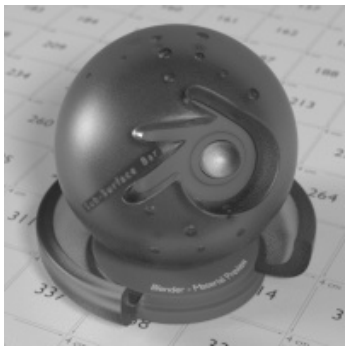
Examples



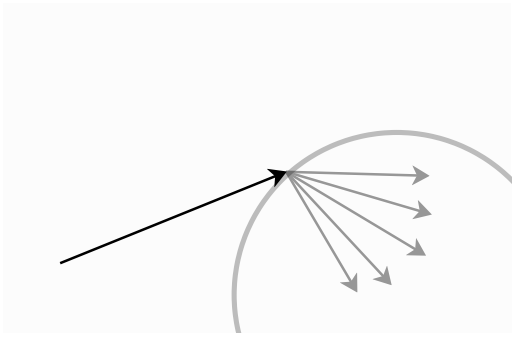
Sharp Glass example.



Sharp Glass behavior.



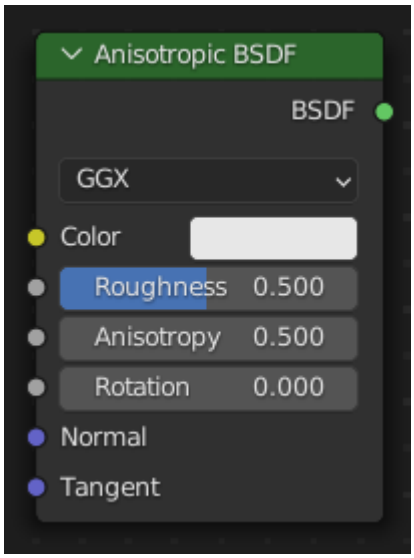
Rough Glass example.



Rough Glass behavior.

Glossy BSDF

The *Glossy* BSDF node is used to add reflection with microfacet distribution, used for materials such as metal or mirrors.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

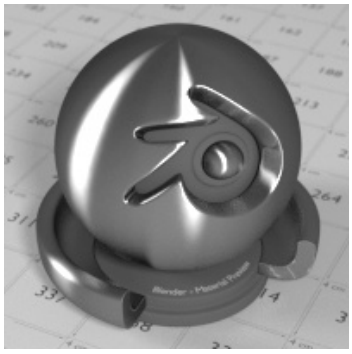
Roughness
: Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

Anisotropy *Cycles Only*
: Controls the amount the reflection stretches the reflection along the surface of the material. A value of 0.0 results in no anisotropic reflections. Higher values give elongated highlights orthogonal to the tangent direction; negative values give highlights shaped along the tangent direction.

  This is a phenomenon know as “Anisotropic Reflections” which is often seen in metallic materials.

Rotation

Rotation of the anisotropic tangent direction. Value 0.0 equals 0° rotation, 0.25 equals 90° and 1.0 equals 360° = 0°. This can be used to texture the tangent direction.



Anisotropic rotation on 0.



Anisotropic rotation on 0.25 (90°).

Normal

Normal used for shading; if nothing is connected the default shading normal is used.

Tangent

Tangent used for shading; if nothing is connected the default shading tangent is used.

Properties

Distribution

Microfacet distribution to use.

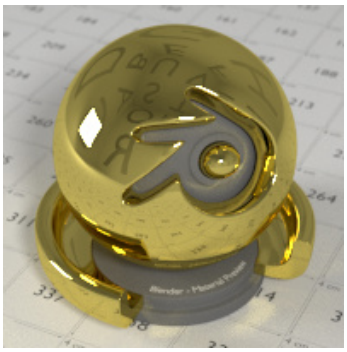
GGX::	GGX microfacet distribution.
Multiscatter GGX::	Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.

Beckmann:: *Cycles Only* Beckmann microfacet distribution.

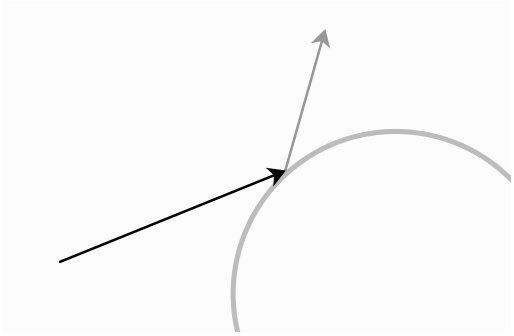
Outputs

BSDF
: Standard shader output.

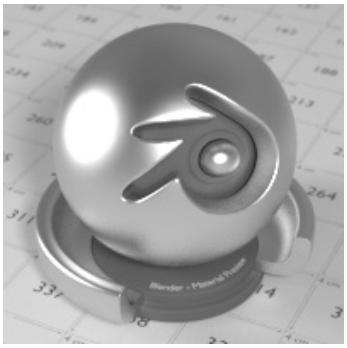
Examples



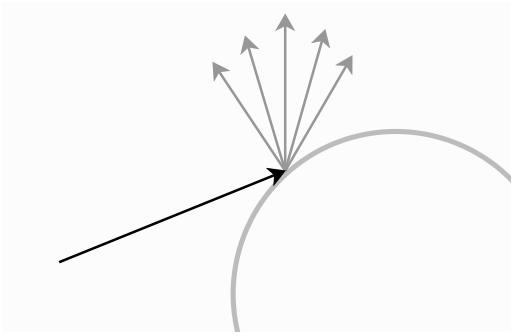
Sharp Glossy example.



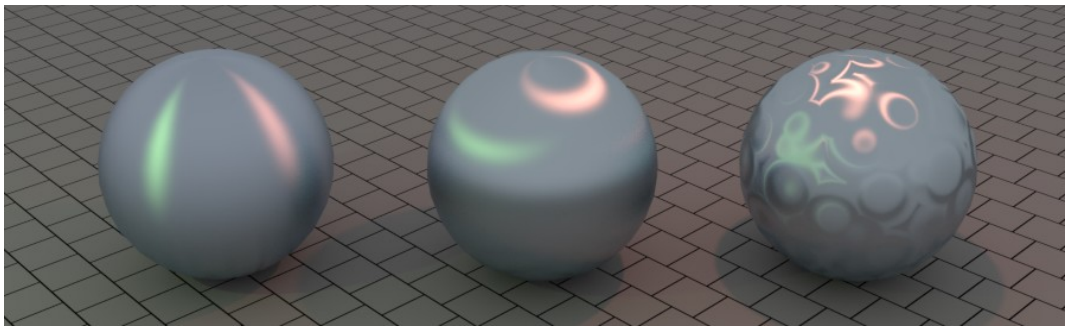
Sharp Glossy behavior.



Rough Glossy example.



Rough Glossy behavior.

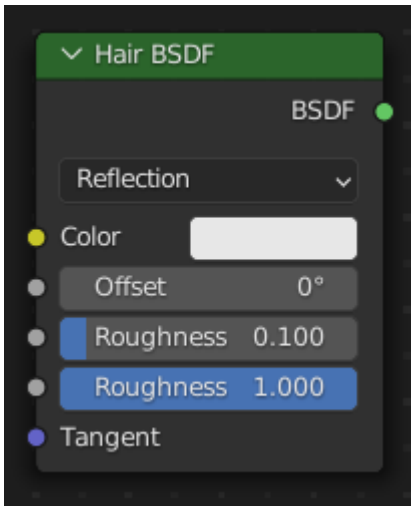


Anisotropic shading with 0° rotation, 90° rotation and textured rotation of the tangent direction. Example blend-file.

Hair BSDF

Cycles Only

The *Hair* BSDF node is used to add shading for Hair.



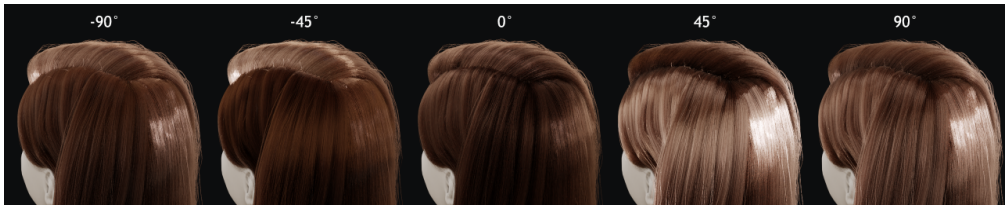
Inputs

Color
: Color of the hair.

Offset
: Controls the way the light is rotated (angular shift) for the reflection/transmission.



Reflection Offset.



Transmission Offset.

Roughness U/V

Controls the roughness in the direction light is skewed, and perpendicular to it.



Roughness when using the Reflection Component.



Roughness when using the Transmission Component.

Tangent

Input tangent.

Properties

Component

There are two components that can be used to control the look of the hair. Usually you are going to want each of these and use a Mix Node.

Reflection:: The light that bounces off the surface of the hair.
Transmission:: The light that passes through the hair and exits on the other side.



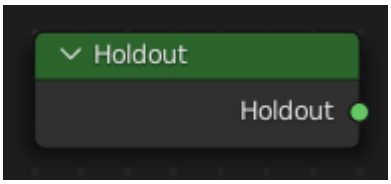
With Mix node: 0 is full Reflection, 1 is full Transmission.

Outputs

BSDF
: Standard shader output.

Holdout

The *Holdout* shader node is used to create a “hole” in the image with zero alpha transparency, which is useful for compositing (see Alpha Channel).



Inputs

This node has no inputs.

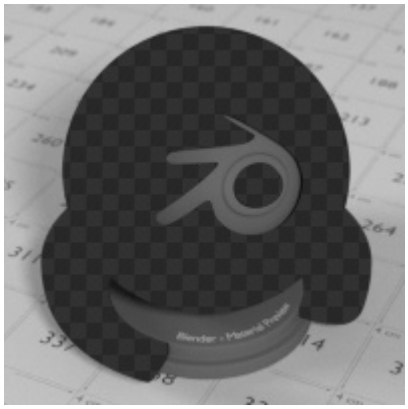
Properties

This node has no properties.

Outputs

Holdout
: Standard shader output.

Examples



The checkered area is a region with zero alpha.

Mix Shader

The *Mix* node is used to mix two shaders together. Mixing can be used for material layering, where the *Factor* input may, for example, be connected to a *Blend Weight* node.



Inputs

Shader
: Shaders to mix, such that incoming rays hit either with the specified probability in the *Factor* socket.

Factor
: Blend weight to use for mixing two shaders; at zero it uses the first shader entirely and at one the second shader.

Properties

This node has no properties.

Outputs

Shader
: Standard shader output.

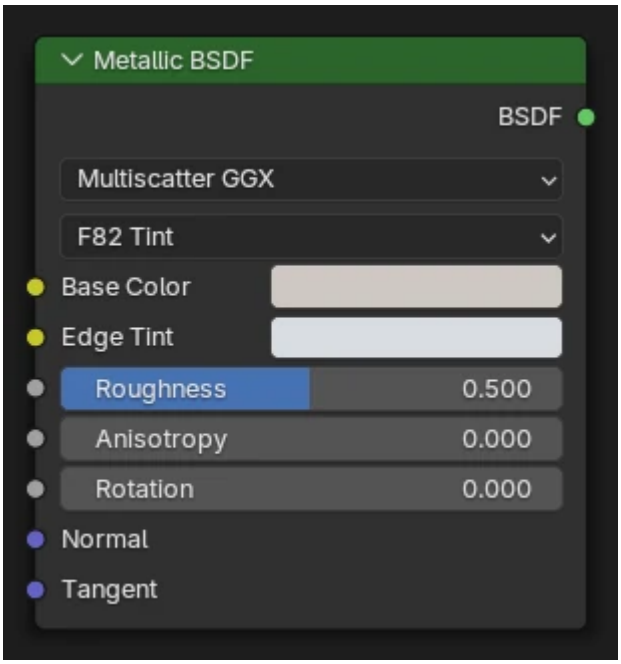
Examples



A mix of a glossy and a diffuse shader makes a nice ceramic material.

Metallic BSDF

The *Metallic* BSDF node is used to recreate the appearance of metals.



Inputs

F82 Tint

Base Color
: Color of the material when viewed straight on.

Edge Tint
: Color of the material when viewed at a 82° angle.

Physical Conductor

IOR
: Refractive index per color channel. This is the real part of a complex refractive index, scientifically denoted as n.

Extinction

Extinction coefficients per color channel. This is the imaginary part of a complex refractive index, scientifically denoted as k.

Common

Roughness
: Sharpness of the reflection; perfectly sharp at 0.0 and smoother with higher values.

Anisotropy *Cycles Only*
: Amount of anisotropy. Higher values give elongated highlights along the tangent direction; negative values give highlights shaped perpendicular to the tangent direction.

Rotation *Cycles Only*
: Rotates the direction of anisotropy, with 1.0 going full circle.

  Compared to the *Glossy BSDF* node, the direction of highlight elongation is rotated by 90°. Add 0.25 to the value to correct.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Tangent
: Tangent used for shading; if nothing is connected the default shading tangent is used.

Properties

Distribution
: Microfacet distribution to use.

  GGX:: GGX microfacet distribution.
  Multiscatter GGX::

Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.

Beckmann:: *Cycles Only* Beckmann microfacet distribution.

Fresnel Type

Models for describing the metal's appearance, by specifying the apparent color or the physical IOR.

F82 Tint:: Uses the Adobe F82-Tint formula for the metallic fresnel. This allows for artist friendly control of the color near the edge of the material to simulate a complex IOR.

Physical Conductor:: Accepts Complex IOR measurements from real world metals to replicate a more accurate rendering of metals than the *F82 Tint* Fresnel type.
Complex IOR values can be found from sources like the Physically Based database for CG artists and Refractive Index nk database.

Outputs

BSDF

Standard shader output.

Examples

F82 Tint






Material	Titanium (Default)	Aluminum	Copper	Gold
Base Color	0.617, 0.576, 0.540	0.911, 0.912, 0.917	0.972, 0.694, 0.486	1.000, 0.735, 0.353
Edge Tint	0.695, 0.726, 0.770	0.848, 0.877, 0.916	0.961, 0.969, 0.942	0.993, 1.000, 1.000

Physical Conductor

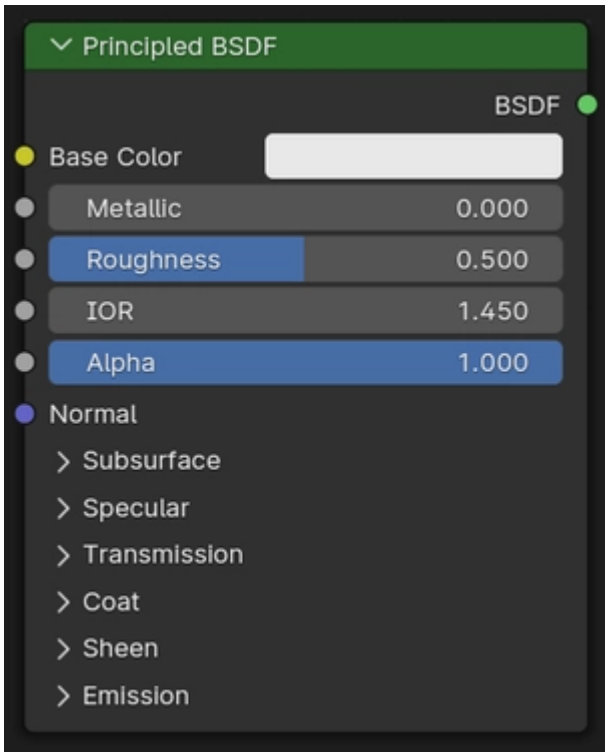





Material	Titanium (Default)	Aluminum	Copper	Gold
IOR	2.757, 2.512, 2.231	1.333, 0.945, 0.582	0.235, 0.729, 1.369	0.000, 0.470, 1.439
Extinction	3.867, 3.404, 3.009	7.434, 6.340, 5.181	5.666, 2.562, 2.227	182.6, 2.189, 1.660

Principled BSDF

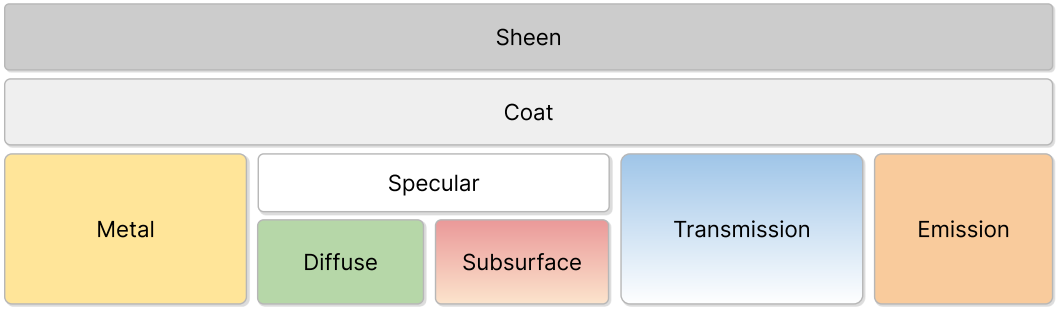
The *Principled* BSDF that combines multiple layers into a single easy to use node. It can model a wide variety of materials.



It is based on the OpenPBR Surface shading model, and provides parameters compatible with similar PBR shaders found in other software, such as the Disney and Standard Surface models. Image textures painted or baked from software like Substance Painter may be directly linked to the corresponding input in this shader.

Layers

The base layer is a mix between metal, diffuse, subsurface, and transmission components. Most materials will use one of these components, though it is possible to smoothly mix between them.



The metal component is opaque and only reflect lights. Diffuse is fully opaque, while subsurface also involves light scattering just below the surface. Both diffuse and subsurface sit below a specular layer. The transmission component includes both specular reflection and refraction.

On top of all base layers there is an optional glossy coat. And finally the sheen layer sits on top of all other layers, to add fuzz or dust.

Light emission can also be added. Light emits from below the coat and sheen layers, to model for example emissive displays with a coat or dust.

Inputs

Base Color

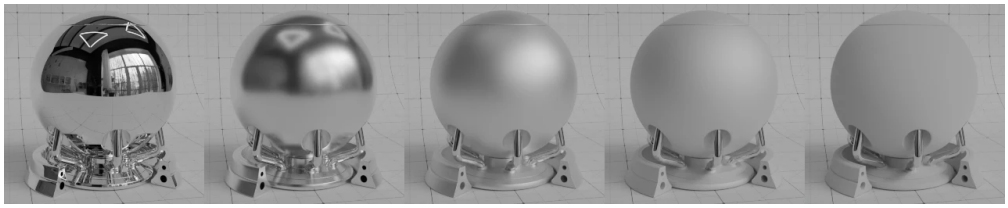
Overall color of the material used for diffuse, subsurface, metal and transmission.



Same base color for multiple materials types

Roughness

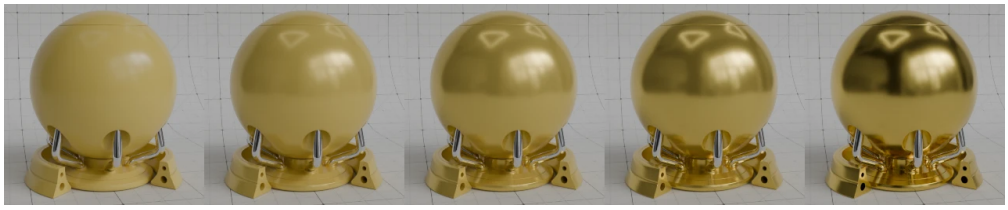
Specifies microfacet roughness of the surface for specular reflection and transmission. A value of 0.0 gives a perfectly sharp reflection, while 1.0 gives a diffuse reflection.



Roughness from 0.0 to 1.0

Metallic

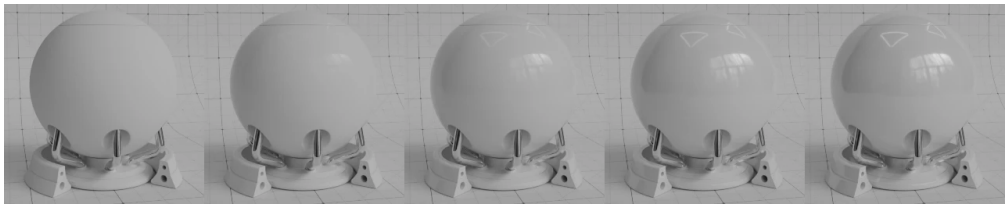
Blends between a dielectric and metallic material model. At 0.0 the material consists of a diffuse or transmissive base layer, with a specular reflection layer on top. A value of 1.0 gives a fully specular reflection tinted with the base color, without diffuse reflection or transmission.



Metallic from 0.0 to 1.0

IOR

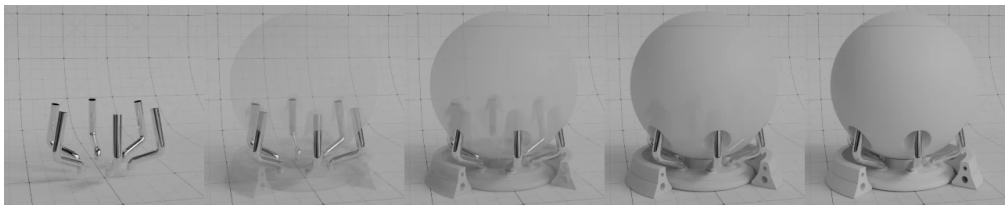
Index of refraction (IOR) for specular reflection and transmission. For most materials, the IOR is between 1.0 (vacuum and air) and 4.0 (germanium). The default value of 1.5 is a good approximation for glass.



IOR from 1.0 to 2.0

Alpha

Controls the transparency of the surface, with 1.0 fully opaque. Usually linked to the Alpha output of an Image Texture node.



Alpha from 0.0 to 1.0

Normal

Controls the normals of the base layers.

Diffuse

Roughness *Cycles Only*
: Surface roughness; 0.0 gives standard Lambertian reflection, higher values activate the Oren-Nayar BSDF.

Subsurface

Subsurface scattering is used to render materials such as skin, milk and wax. Light scatters below the surface to create a soft appearance.

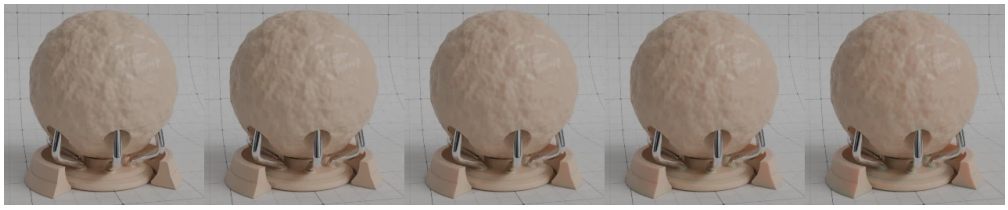
Method
: Rendering method to simulate Subsurface scattering.

Christensen-Burley::
: An approximation to physically-based volume scattering. This method is less accurate than *Random Walk* however, in some situations this method will resolve noise faster.

Random Walk::
: *Cycles Only* Provides accurate results for thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.

Random Walk (Skin)::
: *Cycles Only* Random walk method optimized for skin rendering. The radius is automatically adjusted based on the color texture, and the subsurface entry direction uses a mix of diffuse and specular transmission with custom IOR. This tends to retain greater surface detail and color and matches measured skin more closely.

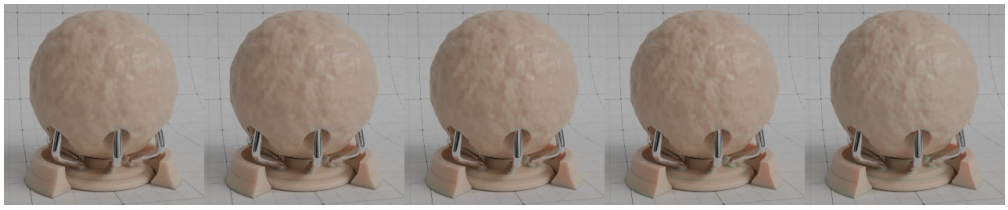
Weight
: Blend between diffuse surface and subsurface scattering. Typically should be zero or one (either fully diffuse or subsurface).



Weight from 0.0 to 1.0

Radius

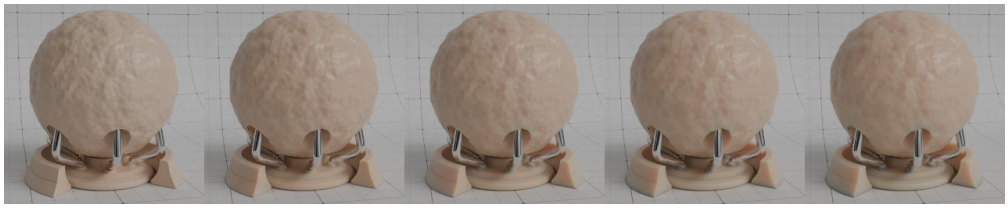
Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows and through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mapped to the R, G and B values, respectively.



Radius from white to red

Scale

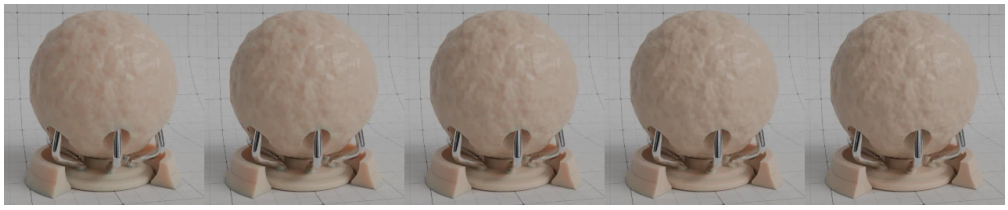
Scale applied to the radius.



Scale from 0 cm to 50 cm

IOR *Cycles Only*

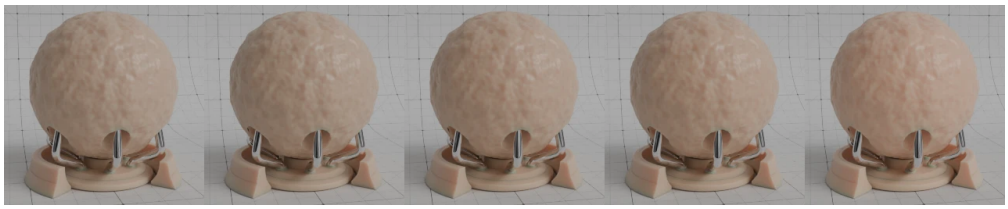
Index of refraction (IOR) used for rays that enter the subsurface component. This may be set to a different value than the global IOR to simulate different layers of skin.



IOR from 1.0 to 2.0

Anisotropy *Cycles Only*

Directionality of volume scattering within the subsurface medium. Zero scatters uniformly in all directions, with higher values scattering more strongly forward. For example, skin has been measured to have an anisotropy of 0.8.



Anisotropy from 0.0 to 1.0

Specular

Controls for both the metallic component and specular layer on top of diffuse and subsurface.

Distribution

Microfacet distribution to use.

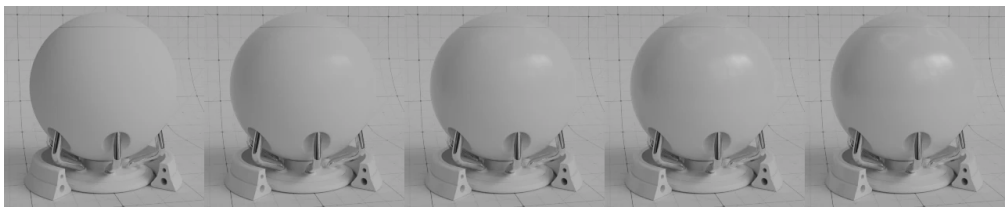
GGX:: A method that is faster than *Multiple-scattering GGX* but is less physically accurate.

Multiscatter GGX:: Takes multiple scattering events between microfacets into account. This gives more energy conserving results, which would otherwise be visible as excessive darkening.

IOR Level

Adjustment to the IOR to increase or decrease intensity of the specular layer. 0.5 means no adjustment, 0 removes all reflections, 1 doubles them at normal incidence.

This input is designed for conveniently texturing the IOR and amount of specular reflection.



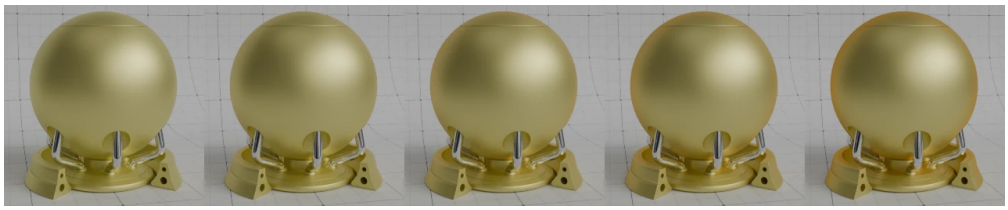
IOR level from 0.0 to 1.0

Tint

Color tint for specular and metallic reflection.

For non-metallic tints provides artistic control over the color specular reflections at normal incidence, while grazing reflections remain white. In reality non-metallic specular reflection is fully white.

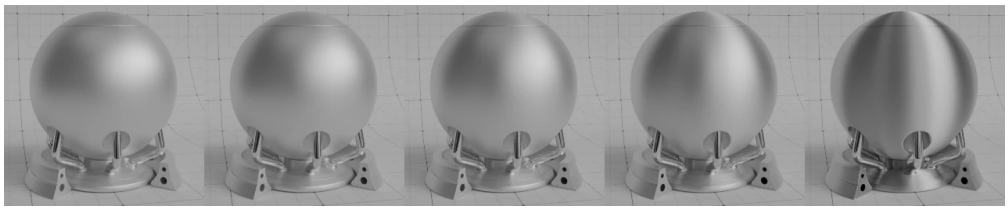
For metallic materials tints the edges to simulate complex IOR as found in materials such as gold or copper.



Tint from white to orange

Anisotropic *Cycles Only*

Amount of anisotropy for specular reflection. Higher values give elongated highlights along the tangent direction; negative values give highlights shaped perpendicular to the tangent direction.

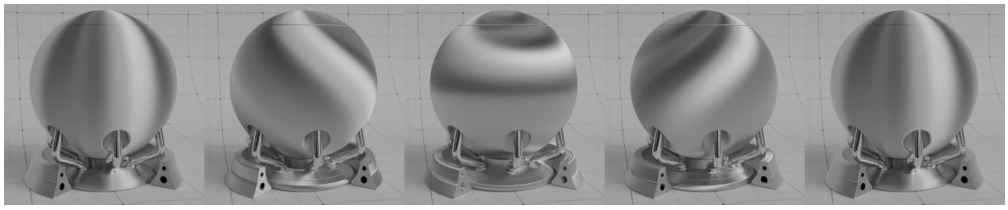


Anisotropic from 0.0 to 1.0

Anisotropic Rotation *Cycles Only*

Rotates the direction of anisotropy, with 1.0 going full circle.

Compared to the *Glossy BSDF* node, the direction of highlight elongation is rotated by 90°. Add 0.25 to the value to correct.



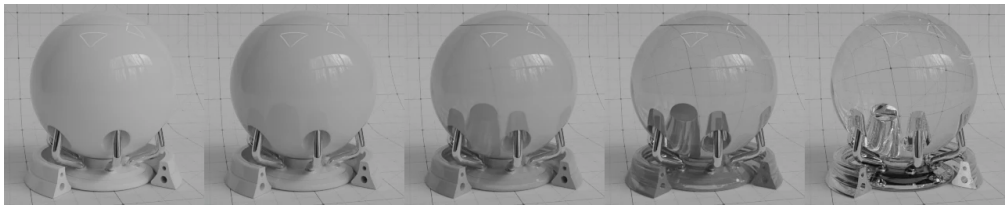
Anisotropic rotation from 0.0 to 1.0

Tangent
: Controls the tangent direction for anisotropy.

Transmission

Transmission is used to render materials like glass and liquids, where the surface both reflects light and transmits it into the interior of the object

Weight
: Mix between fully opaque surface at zero and fully transmissive at one.



Weight from 0.0 to 1.0

Coat

Coat on top of the materials, to simulate for example a clearcoat, lacquer or car paint.

Weight
: Controls the intensity of the coat layer, both the reflection and the tinting. Typically should be zero or one for physically-based materials,

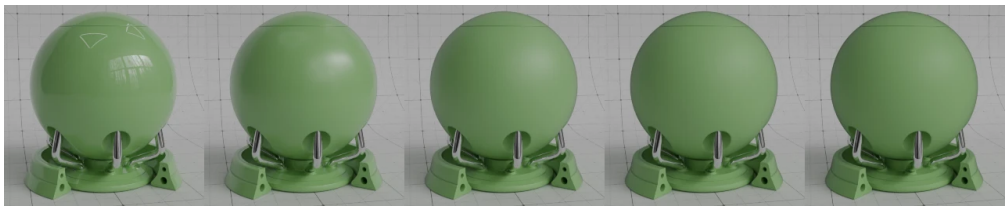
but may be textured to vary the amount of coating across the surface.



Weight from 0.0 to 1.0

Roughness

Roughness of the coat layer.



Roughness from 0.0 to 1.0

IOR

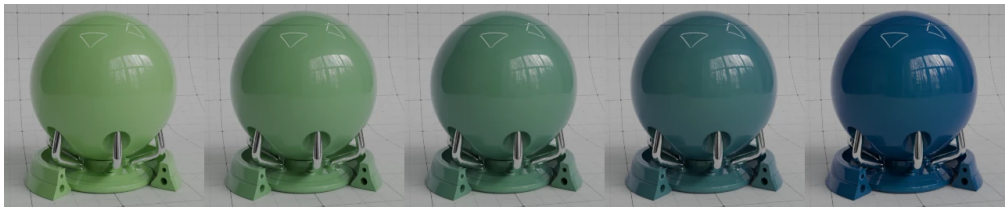
Index of refraction (IOR) of the coat layer. Affects its reflectivity as well as the falloff of coat tinting.



IOR from 1.0 to 2.0

Tint

Adds a colored tint to the coat layer by modeling absorption in the layer. Saturation increases at shallower angles, as the light travels farther through the medium, depending on the IOR.



Tint from white to blue

Normal

Controls the normals of the *Coat* layer, for example to add a smooth coating on a rough surface.

Sheen

Sheen simulates very small fibers on the surface. For cloth this adds a soft velvet like reflection near edges. It can also be used to simulate dust on arbitrary materials.

Weight

Controls in the intensity of the sheen layer.



Weight from 0.0 to 1.0

Roughness

Roughness of the sheen reflection.



Roughness from 0.0 to 1.0

Tint

The color of the sheen reflection.



Tint from white to green.

Emission

Light emission from the surface.

Color

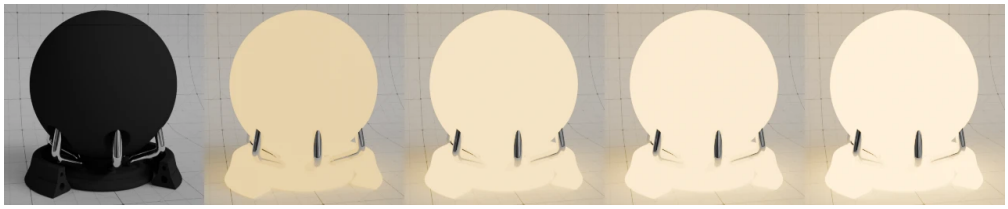
Color of light emission from the surface.



Emission color variations

Strength

Strength of the emitted light. A value of 1.0 ensures that the object in the image has the exact same color as the *Emission Color*, i.e. make it 'shadeless'.



Strength from 0.0 to 10.0

Thin Film *Cycles Only*

Thin Film simulates the effect of interference in a thin film sitting on top of the material. This causes the specular reflection to be colored in a way

which strongly depends on the view angle as well as the film thickness and the index of refraction (IOR) of the film and the material itself.

This effect is commonly seen on e.g. oil films, soap bubbles or glass coatings. While its influence is more obvious in specular highlights, it also affects transmission.

Note

Thin-film interference is currently only applied to dielectric materials. Support for thin films on top of Metallic is planned in the future.

Thickness
: The thickness of the film in nanometers. A value of 0 disables the simulation. The interference effect is strongest between roughly 100 and 1000 nanometers, since this is near the wavelengths of visible light.

IOR
: Index of refraction (IOR) of the thin film. The common range for this value is between 1.0 (vacuum and air) and roughly 2.0, though some materials can reach higher values. The default value of 1.33 is a good approximation for water. Note that when the value is set to 1.0 or to the main IOR of the material, the thin film effect disappears since the film optically blends into the air or the material.

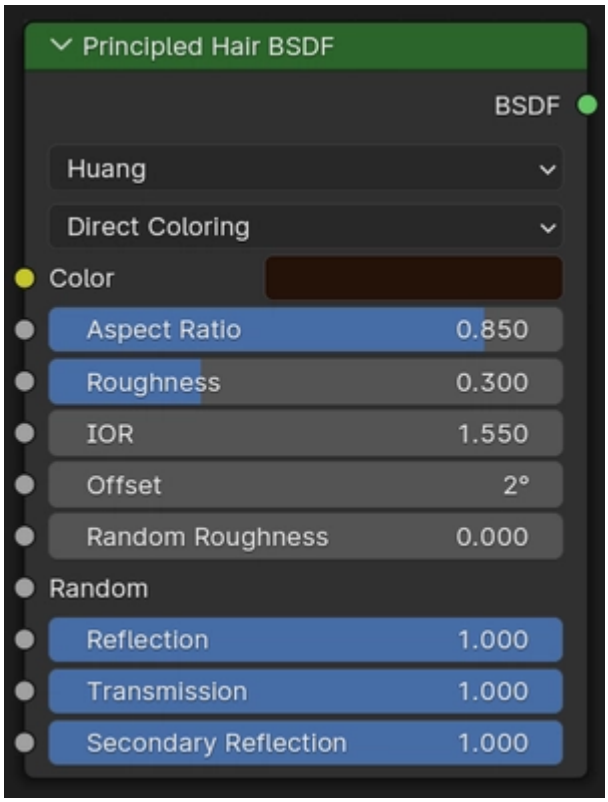
Outputs

BSDF
: Standard shader output.

Principled Hair BSDF

Cycles Only

The *Principled Hair* BSDF is a physically-based, easy-to-use shader for rendering hair and fur.



Tip

Realistic hair should have a minimum of variance between each strand. The shader allows for this by specifying two values, *Random Color* and *Random Roughness*, which remap the specified Melanin/Roughness values to the range \(Color/Roughness \pm Randomization\%\).

Inputs

Common

Color
: The RGB color of the strand. Only used in Direct coloring.

Hint

The chosen color is converted to an absorption coefficient with the following formula (section 4.2 of [CBTB16]):

$$\sigma_{a} = \frac{\ln(Color)} {\left(5.969 - 0.215\beta_{N} + 2.532\beta_{N}^{2} - 10.73\beta_{N}^{3} + 5.574\beta_{N}^{4} + 0.245\beta_{N}^{5}\right)^{2}}$$
where β_{N} is the radial roughness of the hair after applying randomization (if specified).



Coloring hair using the Direct coloring parametrization. (The numbers on top are the RGB values.)

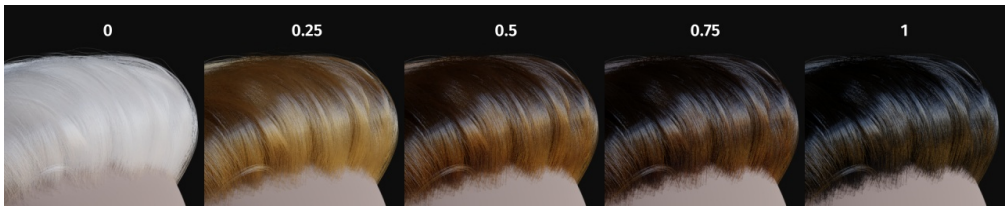
Melanin

Absolute quantity of pigment. Range $[0, 1]$ equivalent to $[0\%, 100\%]$.

Hint

This is a linear mapping to the underlying exponential function:

$$melanin_qty = -\ln(\max(1.0 - Melanin, 0.0001))$$



Melanin.

Melanin Redness

Ratio of pheomelanin to eumelanin. Range $[0, 1]$ equivalent to $[0\%, 100\%]$.

Hint

The ratio formula is: $eumelanin = Melanin \times (1.0 - MelaninRedness)$, $pheomelanin = Melanin \times MelaninRedness$.

The resulting quantities are converted (after randomization, if specified) to absorption concentration via the following formula (section 6.1 of [EFHLA11], adjusted for the range $[0, 1]$):

$$\begin{split}\sigma_{a} = eumelanin \times \begin{bmatrix} 0.506 \\ 0.841 \\ 1.653 \\ \end{bmatrix} + pheomelanin \times \begin{bmatrix} 0.343 \\ 0.733 \\ 1.924 \\ \end{bmatrix}\end{split}$$



Melanin Redness.

Tint

Color used for dyeing the hair after applying the melanin pigment. It is not subject to randomization. It can be disabled by setting the color to white.

Hint

This is converted via the Color mapping above and added to the absorption coefficient of the melanin concentration.



Tint, using Melanin 0.1 and the corresponding RGB values.

Absorption Coefficient

Attenuation coefficient σ.

IOR

Index of refraction (IOR) defining how much the ray changes direction. At 1.0 rays pass straight through like in a transparent material; higher values give more refraction. Default value is \(1.55\).

Offset

Tilts the glint of the hair by increasing the angle of the scales of the hair's cuticle with respect to the hair shaft. Human hair usually has low values.

Random Color

For each strand, vary the melanin concentration by \(RandomFactor\). Range \([0, 1]\) equivalent to \([0\%, 100\%]\) of the initial melanin concentration.

Hint

The melanin concentration is multiplied by \(randomFactor\), where \(randomFactor = 1.0 + 2.0×(Random - 0.5) × RandomColor\).



Random Color.

Random Roughness

For each strand, vary both Roughness values by \(RandomFactor\). Range \([0, 1]\) equivalent to \([0\%, 100\%]\) of the initial roughness values.

Hint

The applied formula is the same one as for *Random Color*.



Random Roughness.

Random
: Random number source. If no node is connected here, it is automatically instanced with the value obtained from *Hair Info* ▸ *Random*.

Chiang Model

The Chiang model is based on a Gaussian distribution with separate roughness along and orthogonal to the hair.

Roughness
: Specify how much the glints are smoothed in the direction of the hair shaft. Too low values will smoothen the hair to the point of looking almost metallic, making glints look like Fireflies; while setting it too high will result in a Lambertian look.

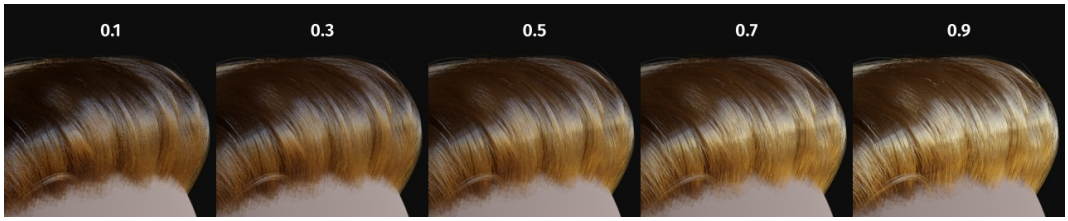


Roughness.

Radial Roughness
: Specify how much the glints are smoothed in the direction of the hair normal. Too low values will concentrate the glint; while setting it too high will spread the light across the width of the strand.

Hint

Mathematically, this parameter is mapped to the logistic distribution's scale factor \(s\) (section 4.1 of [CBTB16]).



Radial Roughness.

Coat
: Simulate a shiny coat of fur, by reducing the Roughness to the given factor only for the first light bounce (diffuse). Range \([0, 1]\) equivalent to a reduction of \([0\%, 100\%]\) of the original Roughness.



Coat.

Huang Model

The Huang model is based on microfacet based reflection and transmission, and supports elliptically shaped hair.

Aspect Ratio
: The ratio of the minor axis to the major axis of an elliptical cross-section. Recommended values are 0.8~1 for Asian hair, 0.65~0.9 for Caucasian hair, 0.5~0.65 for African hair. The major axis is aligned with the curve normal, which can be created with geometry nodes, but is not supported in legacy particle hair.

Roughness
: Microfacet roughness for reflection and transmission.

Reflection
: Optional factor for modulating the first light bounce off the hair surface. The color of this component is always white. Keep this 1.0 for physical

correctness.

Transmission

Optional factor for modulating the transmission component. Picks up the color of the pigment inside the hair. Keep this 1.0 for physical correctness.

Secondary Reflection

Optional factor for modulating the component which is transmitted into the hair, reflected off the backside of the hair and then transmitted out of the hair. This component is oriented approximately around the incoming direction, and picks up the color of the pigment inside the hair. Keep this 1.0 for physical correctness

Properties

Color Parametrization

The shader provides three different ways, or *parametrizations*, to color the hair strands.

Direct Coloring:: Choose the desired RGB color and the shader will approximate the necessary *absorption coefficient* (below).

Melanin Concentration::

This mode defines the color as the quantity and ratio of the pigments which are commonly found in hair and fur, *eumelanin* (prevalent in brown-black hair) and *pheomelanin* (red hair). The quantity is specified in the *Melanin* input, and the ratio between them in *Melanin Redness*. Increasing concentrations darken the hair (the following are with *Melanin Redness* \(1\)):

- White (Melanin \(0\))
- Blonde (Melanin \(0.25\))
- Reddish (Melanin \(0.5\))
- Brown (Melanin \(0.75\))
- Black (Melanin \(1\))

Additionally, the *Tint* inputs allows to dye the hair with the desired color.

Absorption Coefficient::
Specifies the attenuation coefficient \(\sigma_{a}\), as applied by the Beer-Lambert law. This mode is intended mainly for technical users who want to use coefficients from the literature without any sort of conversion.

Outputs

BSDF
Standard shader output.

References

This shader is an implementation of the papers by Chiang et al. [CBTB16] and Huang et al. [HHH22].

[CBTB16] (1,2,3)

Chiang, M. J. , Bitterli, B. , Tappan, C. and Burley, B. (2016), A Practical and Controllable Hair and Fur Model for Production Path Tracing. Computer Graphics Forum, 35: 275-283. doi:10.1111/cgf.12830

[EFHLA11]

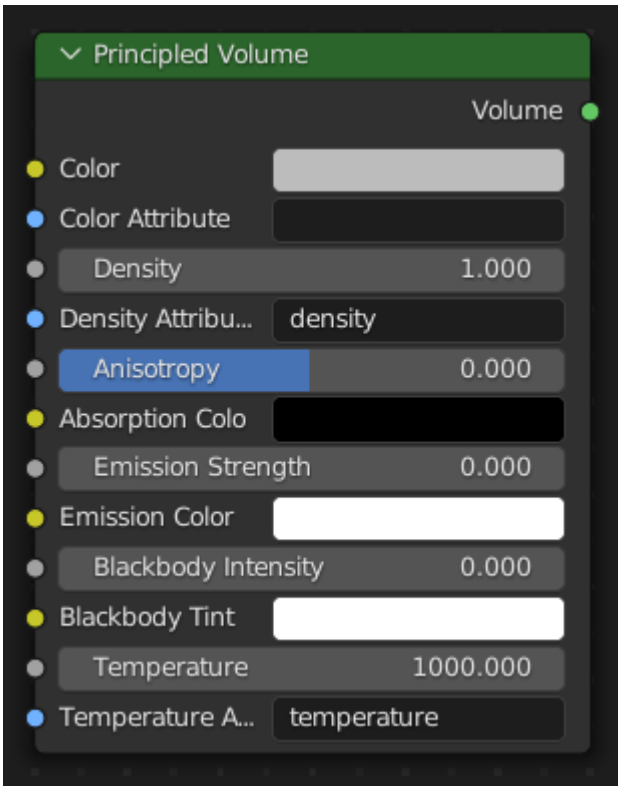
d'Eon, E. , Francois, G. , Hill, M. , Letteri, J. and Aubry, J. (2011), An Energy-Conserving Hair Reflectance Model. Computer Graphics Forum, 30: 1181-1187. doi:10.1111/j.1467-8659.2011.01976.x

[HHH22]

Huang W., Hullin M.B. Hanika J. (2022), A Microfacet-based Hair Scattering Model. Computer Graphics Forum, 41: 79-91. doi:10.1111/cgf.14588

Principled Volume

The *Principled Volume* shader combines all volume shading components into a single easy to use node. Volumes like smoke and fire can be rendered with a single shader node, which includes scattering, absorption and blackbody emission.



Inputs

Color
: Volume scattering color.

Color Attribute
: Volume grid for coloring the volume. Use “color” for smoke simulations.

Density
: Density of the volume.

Density Attribute
: Volume grid to define the density, typically “density”.

Anisotropy
: Backward or forward scattering direction.

Absorption Color
: Volume shadow color tint.

Emission Strength
: Amount of light to emit.

Emission Color
: Emission color tint.

Blackbody Intensity
: Blackbody emission for fire. Set to 1 for physically accurate intensity.

Blackbody Tint
: Color tint for blackbody emission.

Temperature
: Temperature in kelvin for blackbody emission, higher values emit more.

Temperature Attribute
: Volume grid to define the temperature, typically “temperature”.

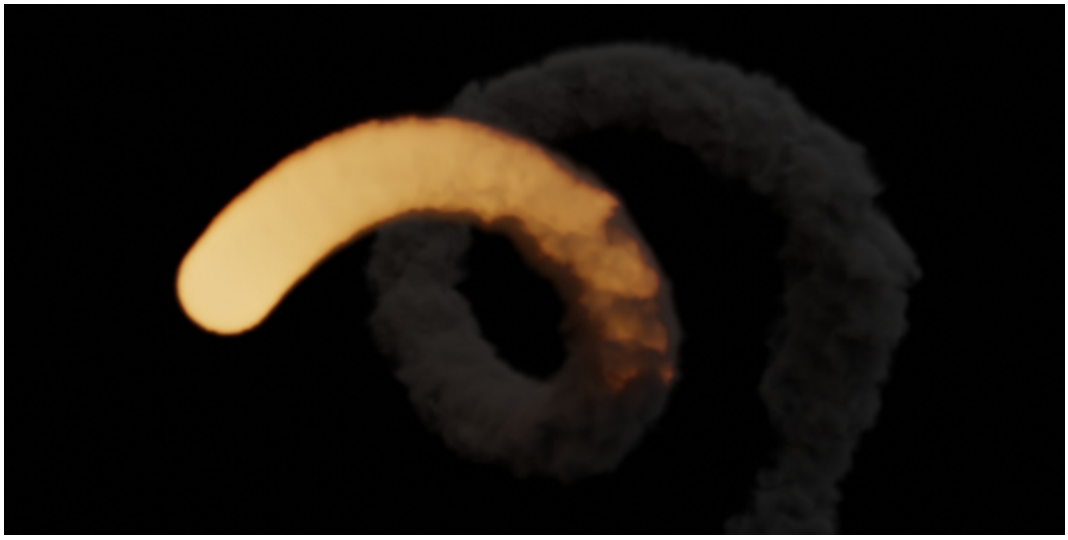
Properties

This node has no properties.

Outputs

Volume
: Standard shader output.

Examples



Ray Portal BSDF

Cycles Only

The *Ray Portal BSDF* node transports rays that enter to another location in the scene. It can be used to render portals for visual effects, and other production rendering tricks.

It acts much like a Transparent BSDF: render passes are passed through, and it is affected by light path max transparent bounces.

Note

- The *Ray Portal BSDF* only allows rays to pass through it in one direction. Add a second portal at the target location to make rays go in the other direction as well.
- Light sampling does not work efficiently through portals. This can lead to increased noise from lights on the other side of portals. Particularly small lights may be very noisy, or not pass through at all.

Inputs

Color
: Tint rays passing through the portal.

Position
: Ray start position at new location. Defaults to the current position, matching the Position output of the Geometry node.

Direction
: Ray direction at the new location. Defaults to the current view direction, which is the same as the negation of the Incoming output of the Geometry node.

Properties

This node has no properties.

Outputs

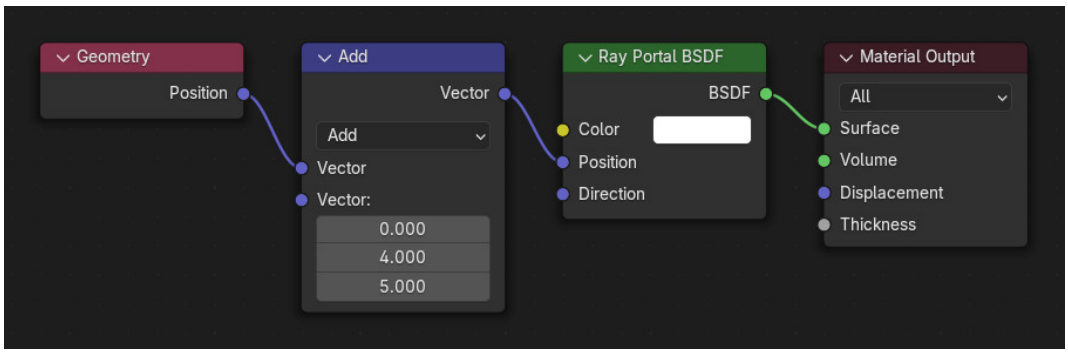
BSDF
: Standard shader output.

Examples

One use case for the *Ray Portal BSDF* is to connect two spaces together to create effects like a portal to an alternative dimension, or "impossible spaces" where something is bigger or smaller on the inside than expected.

To set up a *Ray Portal BSDF* for a technique like this, augment the *Position* and *Incoming* outputs of the Geometry node to set the exit point and direction of the ray through the portal. Here are some examples:

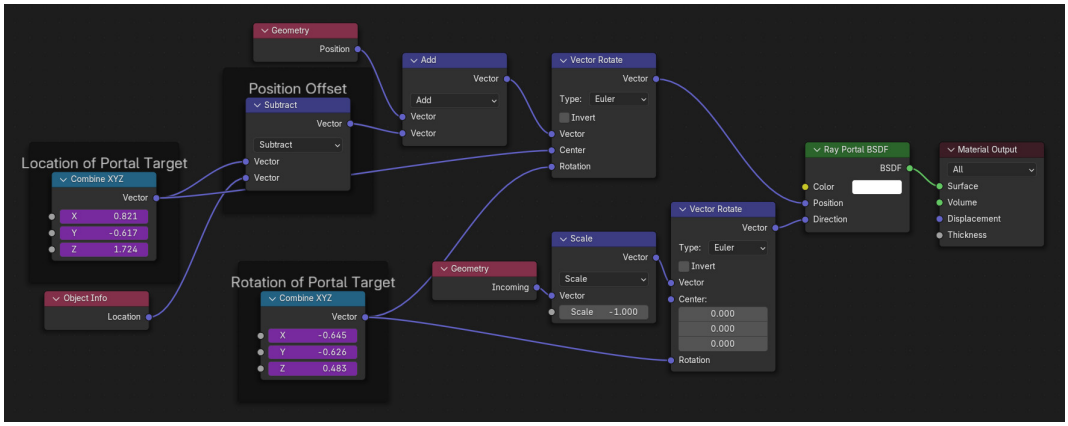
Simple Offset



This simple node setup offsets the ray position. In this example, the ray is offset 0 units along the X axis, 4 units along the Y axis, and 5 units along the Z axis.

Portal





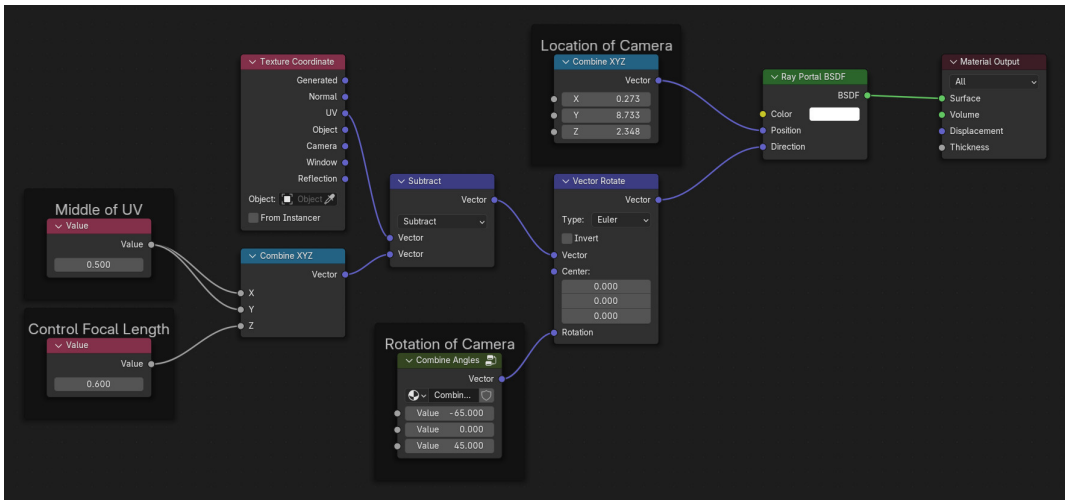
In this example, the *Location of Portal Target* and *Rotation of Portal Target* vectors are obtained from a target portal object using Drivers.

Camera Feed

Along with augmenting rays, the ray position and ray direction can be replaced, for effects like a camera feed on a screen.



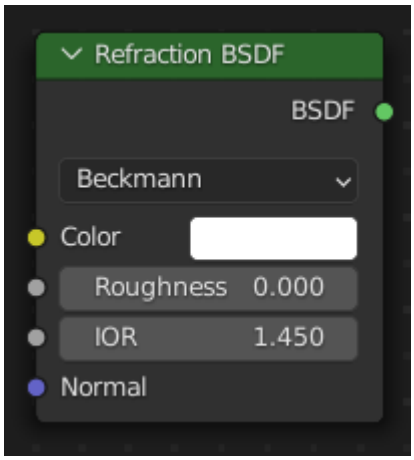
Using the Ray Portal BSDF to replicate the effect of a camera feed on a screen.



Node setup for replicating a camera feed like effect on a screen.

Refraction BSDF

The *Refraction* BSDF is used to add glossy refraction with sharp or microfacet distribution, used for materials that transmit light. For best results this node should be considered as a building block and not be used on its own, but rather mixed with a glossy node using a Fresnel factor. Otherwise it will give quite dark results at the edges for glossy refraction.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is refracted for each wavelength.

Roughness
: Influences sharpness of the refraction; perfectly sharp at 0.0 and smoother with higher values.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Distribution
: Microfacet distribution to use.

GGX::	GGX microfacet distribution.
Beckmann::	*Cycles Only* Beckmann microfacet distribution.

Outputs

BSDF
: Standard shader output.

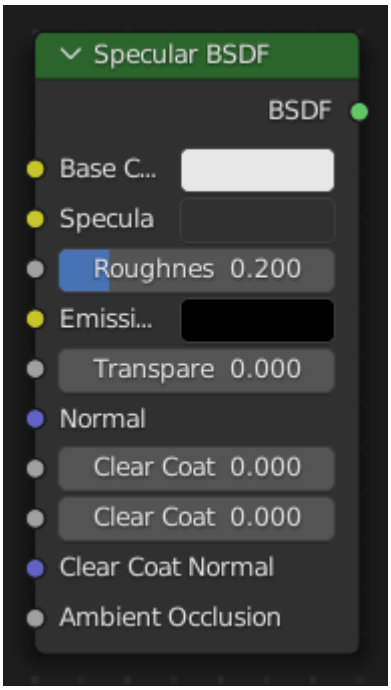
Examples



Refraction Shader.

Specular BSDF

EEVEE Only



The *Specular* BSDF combines multiple layers into a single easy to use node.

It is similar to the Principled BSDF node but uses the *specular* workflow instead of the metallic. It has far fewer parameters and supports less features. Both might be merged into one node in the future.

The specular workflow functions by specifying the facing (along normal) reflection color. The result may not be physically plausible because there is no energy conservation.

Inputs

Base Color
: Diffuse surface color. For conductor materials (metals) it should be black.

Specular
: Amount of specular reflection. Specifies facing (along normal) reflectivity. Conductor materials (metals) can have colored specular reflection.

Hint

To compute this value for a realistic material with a known index of refraction, you may use this special case of the Fresnel formula: \(specular = ((ior - 1)/(ior + 1))^2\)

For example:

- water: ior = 1.33, specular = 0.02
- glass: ior = 1.5, specular = 0.04
- diamond: ior = 2.417, specular = 0.17

Roughness
: Specifies microfacet roughness of the surface for diffuse and specular reflection.

Hint

When converting from the older *Glossy BSDF* node, use the square root of the original value.

Emissive Color
: Color of the emitted light. This light is added to the BSDF result.

Transparency
: Transparency factor. This is the inverse of the alpha channel (1 - alpha) you find in an image. Use an Invert node to convert alpha to transparency. This will only have an effect if the material uses a blend mode other than opaque.

Normal
: Controls the normals of the base layers.

Clear Coat
: Extra white specular layer on top of others. This is useful for materials like car paint and the like.

Clear Coat Roughness:
: Roughness of clear coat specular.

Clear Coat Normal
: Controls the normals of the *Clear Coat* layer.

Ambient Occlusion
: Amount of occlusion to apply to indirect lighting. Usually a bake ambient occlusion map. The final occlusion factor is the minimum of this input and the runtime ambient occlusion effect.

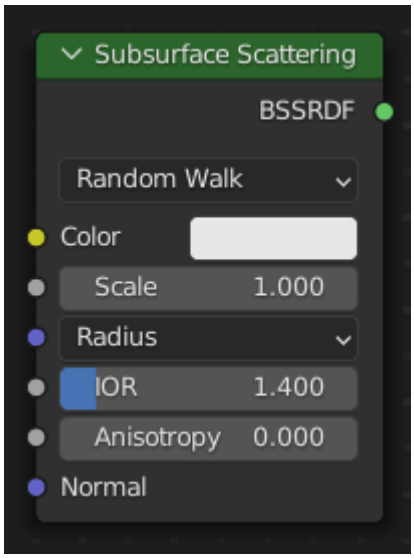
Properties

This node has no properties.

Outputs

BSDF
: Standard shader output.

Subsurface Scattering



The *Subsurface Scattering* node is used to add simple subsurface multiple scattering, for materials such as skin, wax, marble, milk and others. For these materials, rather than light being reflect directly off the surface, it will penetrate the surface and bounce around internally before getting absorbed or leaving the surface at a nearby point.

How far the color scatters on average can be configured per RGB color channel. For example, for skin, red colors scatter further, which gives distinctive red-colored shadows, and a soft appearance.

Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Scale
: Global scale factor for the scattering radius.

Radius
: Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows and through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mapped to the R, G and B values, respectively.

IOR *Cycles Only*

Index of refraction for *Subsurface Scattering*.

Anisotropy *Cycles Only*
: Directionality of subsurface scattering. Higher anisotropy scatters deeper into the object.

Roughness *Cycles Only*
: Roughness of the glossy surface surrounding the subsurface volume.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Subsurface Method
: Rendering method to simulate subsurface scattering.

> **Note**
>
> EEVEE does not support the *Random Walk* methods.

Christensen-Burley::
: An approximation to physically-based volume scattering. This method is less accurate than *Random Walk* however, in some situations this method will resolve noise faster.

Random Walk (Fixed Radius)::
: Provides accurate results for thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.

Random Walk::

Behaves similarly to *Random Walk (Fixed Radius)* but modulates the *Radius* based on the *Color*, *Anisotropy*, and *IOR*. This method thereby attempts to retain greater surface detail and color than *Random Walk (Fixed Radius)*.

Outputs

BSSRDF
: BSSRDF shader output.

Examples

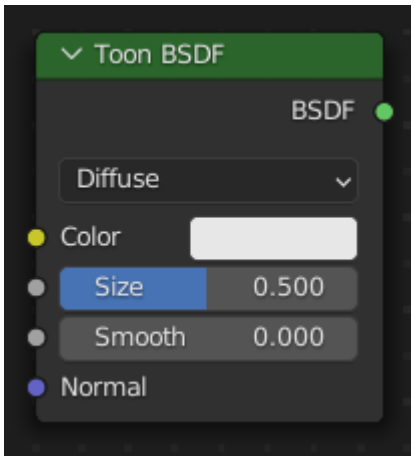


Random walk subsurface scattering.

Toon BSDF

Cycles Only

The *Toon* BSDF is used to create *Diffuse* and *Glossy* materials with cartoon light effects.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Size
: Parameter between 0.0 and 1.0 that gives an angle of reflection between 0° and 90°.

Smooth
: This value specifies an angle over which a smooth transition from full to no reflection happens.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Component

The material component to base the toon effect.

Diffuse:: Use shading based on the Diffuse BSDF.
Glossy:: Use shading based on the Glossy BSDF for specular reflection.

Outputs

BSDF
Standard shader output.

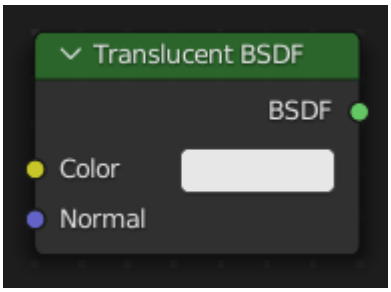
Examples



Example of Toon Shader.

Translucent BSDF

The *Translucent* BSDF is used to add Lambertian diffuse transmission.



Inputs

Color
: Color of the surface, or physically speaking, the probability that light is transmitted for each wavelength.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

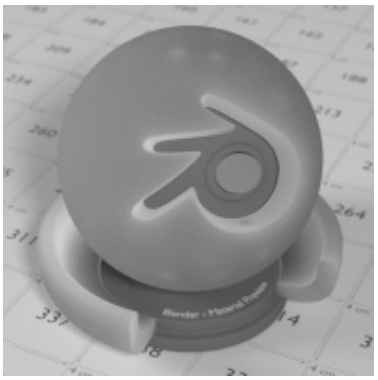
Properties

This node has no properties.

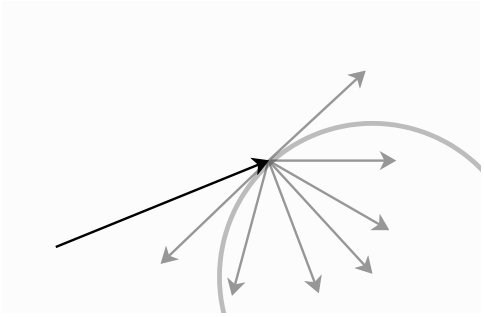
Outputs

BSDF
: Standard shader output.

Examples

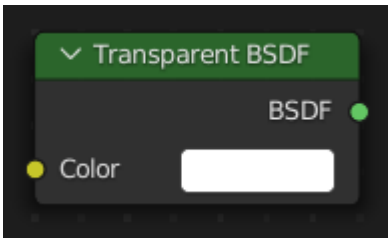


Translucent shader example.



Translucent shader behavior.

Transparent BSDF



The *Transparent* BSDF is used to add transparency without refraction, passing straight through the surface, as if there were no geometry there. Useful with alpha maps, for example. This shader affects light paths somewhat differently than other BSDFs. Note that only pure white transparent shaders are completely transparent.

Inputs

Color
: Color of the surface, or physically speaking, the probability for each wavelength that light is blocked or passes straight through the surface.

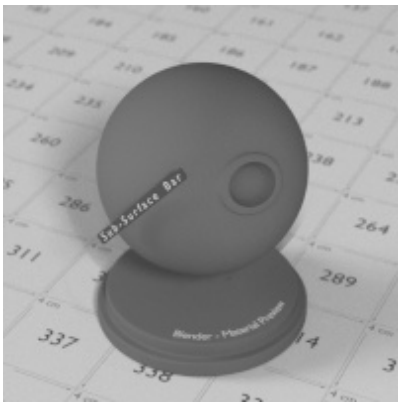
Properties

This node has no properties.

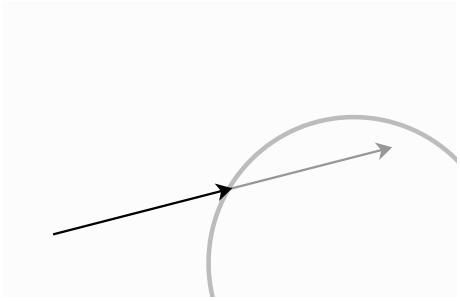
Outputs

BSDF
: Standard shader output.

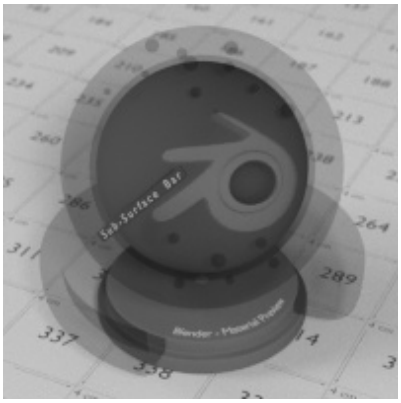
Examples



Transparent shader (pure white).



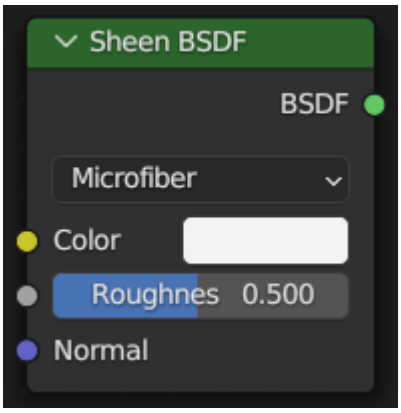
Transparent shader behavior.



Transparent shader (gray).

Sheen BSDF

Cycles Only



The *Sheen* BSDF is used to add reflection to materials that have micro surface details such as cloth or dust. This shader is intended to be layered on top of other shaders such as dielectric or metallic shader setups.

Inputs

Color
: Color of the surface, or physically speaking, the probability that light is reflected for each wavelength.

Roughness
: Controls the amount of color that is reflected back to the camera, higher values reflect more color and can give a dusty appearance, while lower values look fuzzy and darker.

Normal
: Normal used for shading; if nothing is connected the default shading normal is used.

Properties

Distribution
: Sheen shading model.

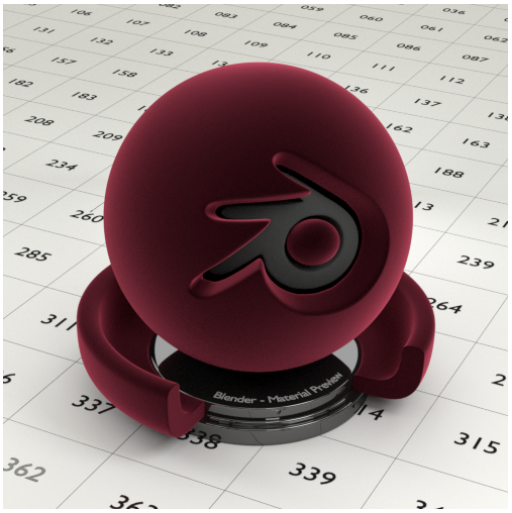
  Ashikhmin:: Classic Ashikhmin velvet, used in Blender versions prior to 4.0

Microfiber:: Microflake-based model of multiple scattering between normal-oriented fibers.

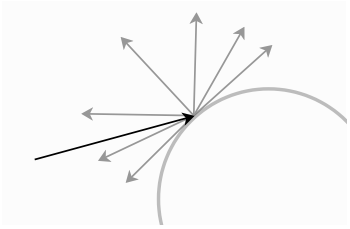
Outputs

BSDF
: Standard shader output.

Examples



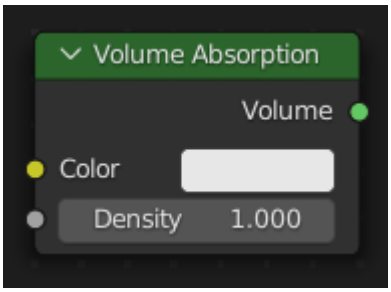
The Sheen shader example.



The Sheen shader behavior.

Volume Absorption

The *Volume Absorption* node allows light to be absorbed as it passes through the volume. Typical usage for this node would be water and colored glass.



Inputs

Color
: Color of the volume.

Density
: The density of the absorption effect.

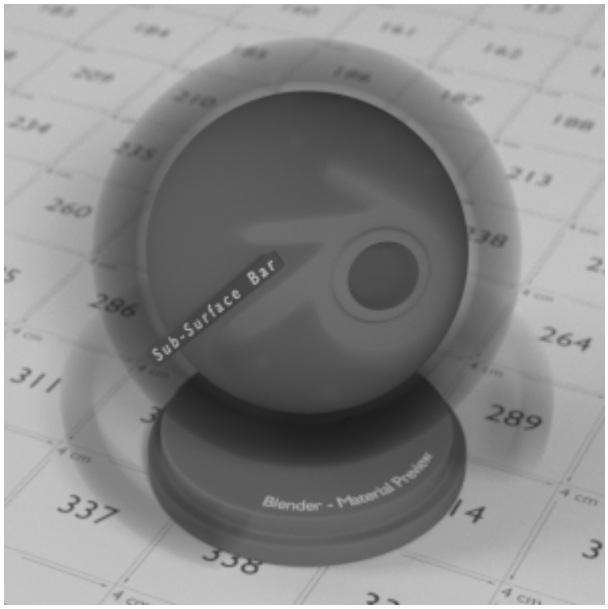
Properties

This node has no properties.

Outputs

Volume
: The Volume Shader output must be plugged into the *Volume Input* of the Material or World Output node.

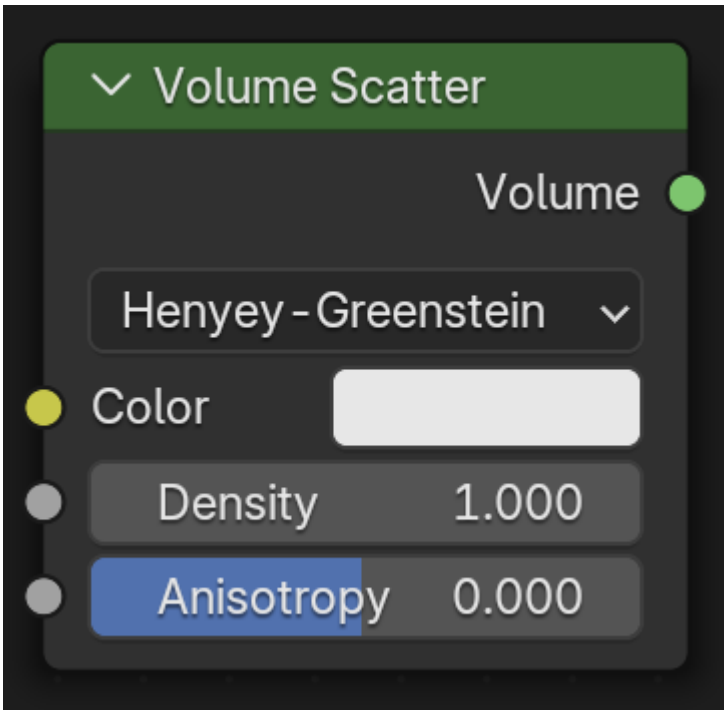
Examples



Example of Volume Absorption.

Volume Scatter

The *Volume Scatter* node allows light to be scattered as it passes through the volume. Typical usage would be to add fog to a scene. It can also be used with the Volume Absorption node to create smoke.



Inputs

Color
: Scattering coefficients per color channel.

Density
: The density of the scatter effect.

Anisotropy *Henyey-Greenstein Draine*
: Controls the relative amount of backward and forward scattering.

IOR *Fournier-Forand*
: Refractive index of the scattering particles relative to water. Common ocean waters range between 1.0 and 1.2, while turbid waters with higher density of particles have higher IORs.

Backscatter *Fournier-Forand*
: Fraction of light that is scattered backwards. Most oceanic particles have backscatter values between 0.001 (e.g., very large phytoplankton) and 0.1 (e.g., very small mineral particles), pure water has a backscatter of 0.5. Values taken from Ocean Optics Web Book.

Alpha *Draine*
: Blending factor between Henyey-Greenstein (\(\alpha = 0\)) and Cornette & Shanks (\(\alpha = 1\)) phase functions.

Diameter *Mie*
: Diameter of the scattering particles in μm.

Properties

Phase
: Volume scattering phase function.

Henyey-Greenstein:: Simple and widely used phase function, useful for approximating scattering in biological tissues.

Fournier-Forand:: *Cycles Only* Suitable for modeling the scattering of light in underwater environments.

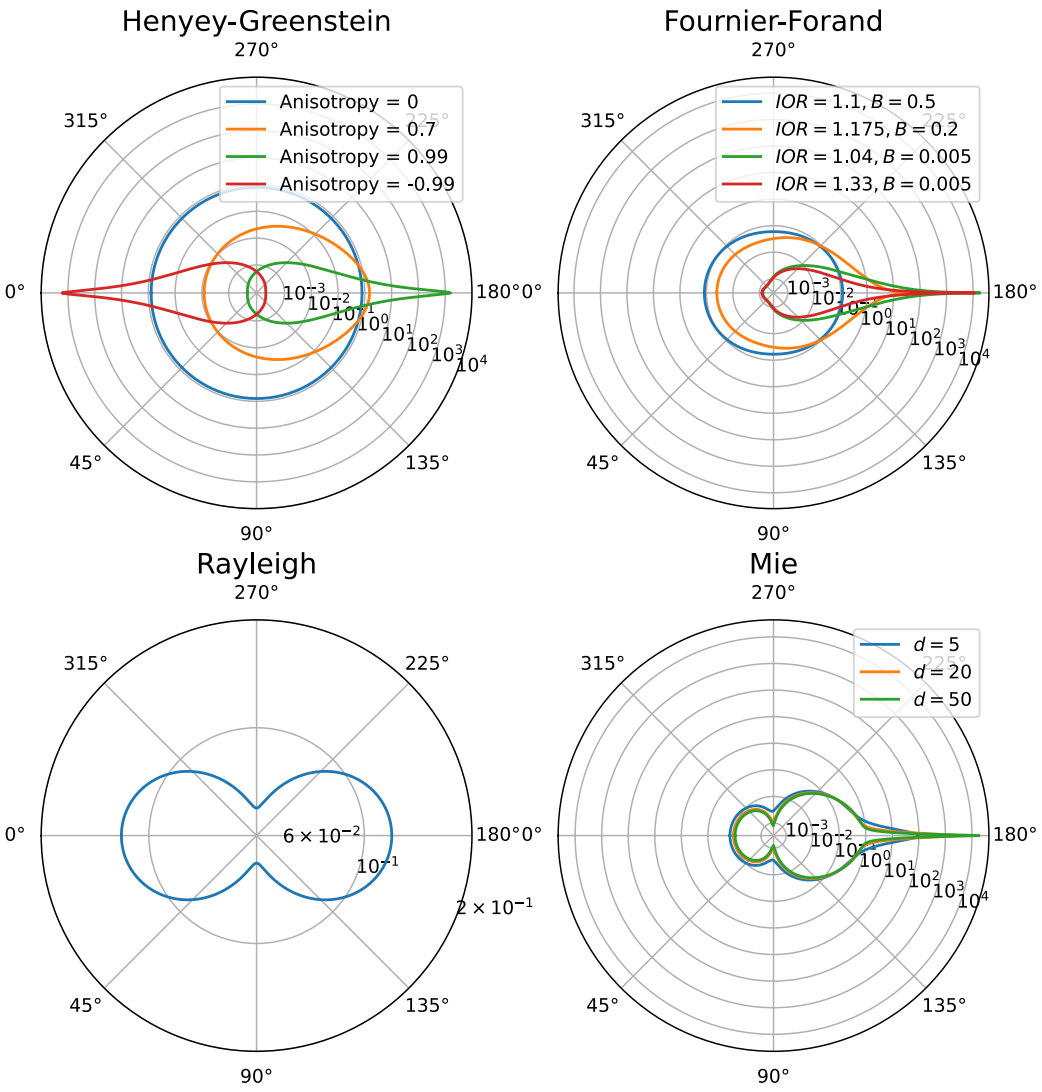
Draine:: *Cycles Only* Suitable for modeling the scattering of interstellar dust.

Rayleigh:: *Cycles Only* Describes the scattering by particles with a size smaller than the wavelength of light, such as the scattering of sunlight in earth's atmosphere.

Mie:: *Cycles Only* Describes the scattering by particles with a size larger than the wavelength of light, such as cloud and fog.

Tip

These phase functions can be combined using a Mix Shader.

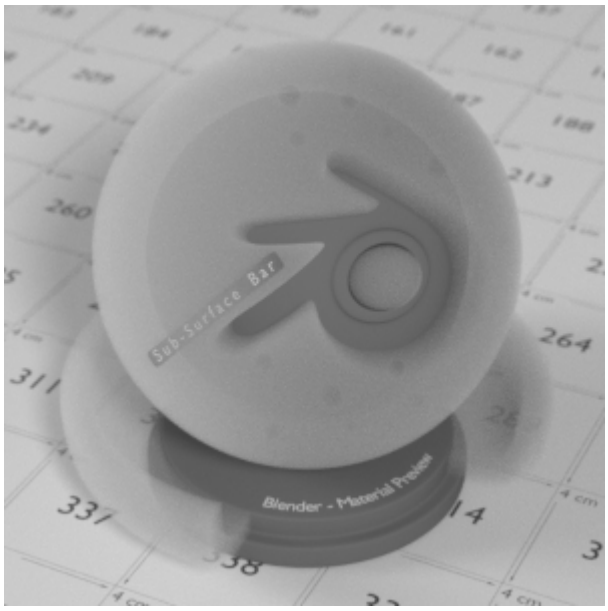


Volume scattering phase as a function of angles between the incoming and the outgoing direction, in logarithmic scale. Light comes from the left side.

Outputs

Volume
: The Volume Shader output must be plugged into the *Volume Input* of the Material or World Output node.

Examples



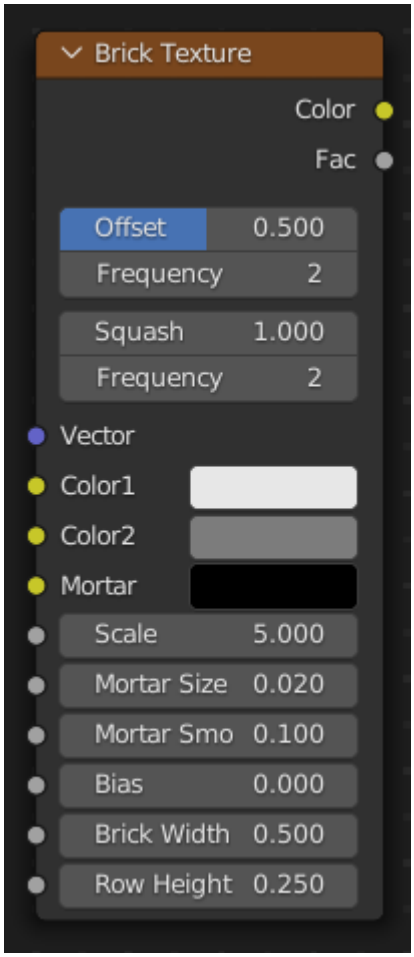
Example of Volume Scatter.

Texture

- Brick Texture Node
- Checker Texture Node
- Environment Texture Node
- Gabor Texture Node
- Gradient Texture Node
- IES Texture Node
- Image Texture Node
- Magic Texture Node
- Musgrave Texture Node
- Noise Texture Node
- Point Density Node
- Sky Texture Node
- Voronoi Texture Node
- Wave Texture Node
- White Noise Texture Node

Brick Texture Node

The *Brick Texture* is used to add a procedural texture producing bricks.



Inputs

Color 1/2
: Color of the bricks.

Mortar
: The color of the area between bricks.

Scale
: Overall texture scale.

Mortar Size
: The size of the filling between the bricks known as “mortar”; 0 means no mortar.

Mortar Smooth
: Blurs/softens the edge between the mortar and the bricks. This can be useful with a texture and displacement textures.

Bias
: The color variation between *Color 1/2*. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width
: The ratio of brick’s width relative to the texture scale.

Row Height
: The ratio of brick’s row height relative to the texture scale.

Properties

Offset
: Determines the brick offset of the various rows.

Frequency
: How often rows are offset; a value of 2 gives an even/uneven pattern of rows.

Squash
: Factor to adjust the brick’s width for particular rows determined by the *Frequency*

Frequency
: How often rows consist of “squished” bricks.

Outputs

Color

Texture color output.

Factor
: Mortar mask (1 = mortar).

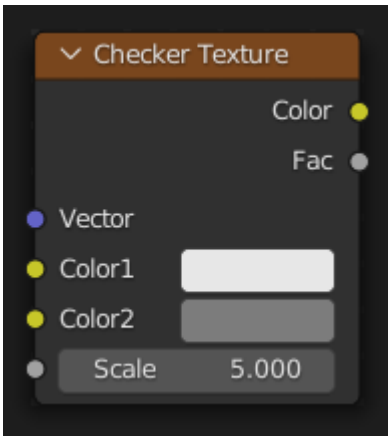
Examples



Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Checker Texture Node

The *Checker Texture* is used to add a checkerboard texture.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

> **Warning**
>
> This node can have precision issues with some vector inputs. See the notes for the White Noise Texture for ways to mitigate this issue.

Color1, Color 2
: Color of the checkers.

Scale
: Overall texture scale. The scale is a factor of the bounding box of the face divided by the scale. For example, a scale of 15 will result in 15 alternate patterns over the overall UV bounding box. Different patterns could be achieved using other nodes to give different input patterns to this socket. For example, using the Math Node.

Properties

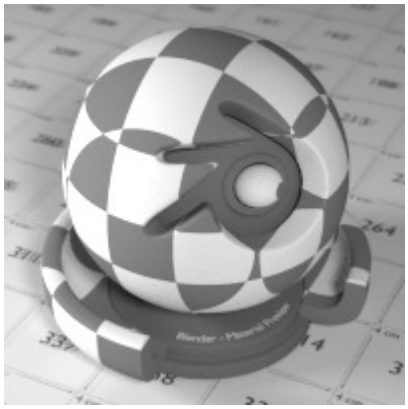
This node has no properties.

Outputs

Color
: Texture color output.

Factor
: Checker 1 mask (1 = Checker 1).

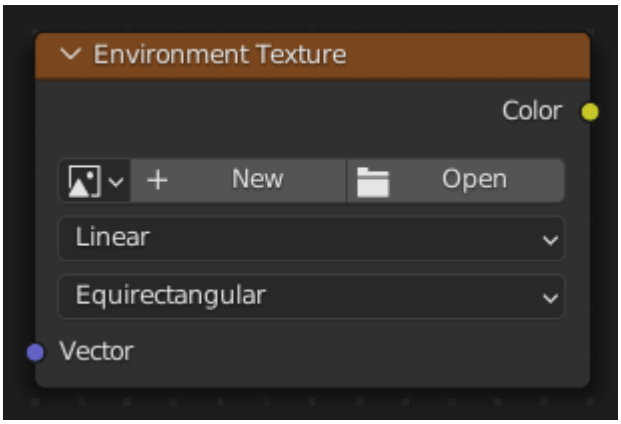
Examples



Default Checker texture.

Environment Texture Node

The Node *Environmental Texture* is used to light your scene using an environment map image file as a texture.



Inputs

Vector
: Texture coordinate for texture look-up. If this socket is left unconnected, the image is mapped as environment with the Z axis as up.

Properties

Image
: Image data-block used as the image source. Additional settings can be found in *Sidebar ▸ Item ▸ Properties*: These include options to control the alpha channel along with addition options for the color space. These addition options are documented with the rest of Common Image Settings.

Color Space
: Type of data that the image contains, either Color or Non-Color Data. For most color textures the default of Color should be used, but in case of e.g. a bump or alpha map, the pixel values should be interpreted as Non-Color Data, to avoid doing any unwanted color space conversions.

The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Texture Interpolation

Interpolation method used for the environment texture. The following interpolations are available:

Linear:: Regular quality interpolation.
Closest:: No interpolation, use closest pixel.
Cubic:: Smoother, better quality interpolation.
Smart:: Bicubic when magnifying, otherwise Bilinear is used. This is only available for OSL.

Projection Method

Allows you to use different types of environmental maps. The following methods are supported:

Equirectangular:: Projection from an Equirectangular photo.
Mirror Ball:: Projection from an orthographic photo or mirror ball.

Outputs

Color

RGB color from the image.

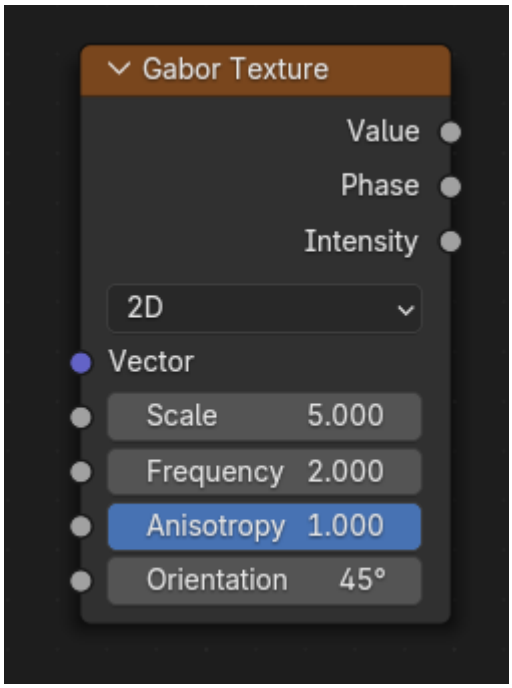
Examples



HDR image from OpenFootage.net.

Gabor Texture Node

The *Gabor Texture* node evaluates a Gabor noise at the input texture coordinates. Gabor noise is visually characterized by random interleaved bands whose direction and width can be controlled. Additionally, it can be used to create omnidirectional noise like the standard Noise Texture node, but since it is more expensive to compute, using the Noise Texture node is probably the better option in those cases. See the examples for more information.



Inputs

Vector
: The coordinates at which Gabor noise will be evaluated. The Z component is ignored in the 2D case. Defaults to *Generated* texture coordinates if the socket is left unconnected.

Scale
: Scale of the Gabor noise.

Frequency
: The rate at which the Gabor noise changes across space. This is different from the Scale input in that it only scales perpendicular to the Gabor noise direction.

Anisotropy

The directionality of Gabor noise. 1 means the noise is completely directional, while 0 means the noise is omnidirectional.

Orientation
: The direction of anisotropic Gabor noise. This is an angle for the 2D case, while it is a unit direction vector for the 3D case.

Properties

Type
: Type of Gabor noise texture.

2D::	Evaluates the noise in 2D space. The Z component of the input vector is ignored.
3D::	Evaluates the noise in 3D space.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Outputs

Value
: The Gabor noise value with both random intensity and phase. This is equal to sine the phase multiplied by the intensity.

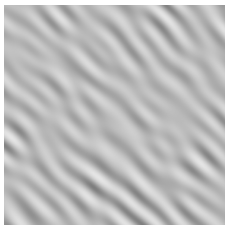
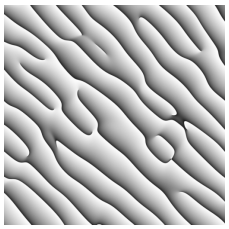
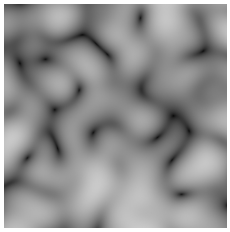
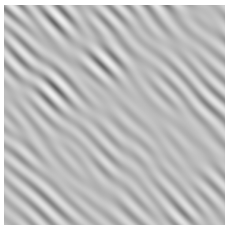
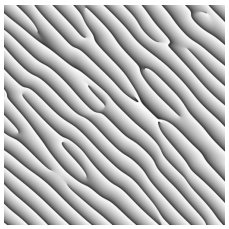
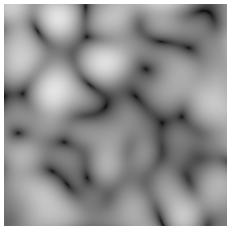
Phase
: The phase of the Gabor noise, which has no random intensity.

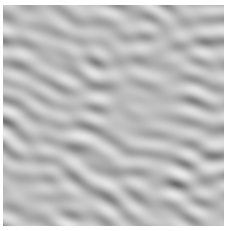
Intensity
: The intensity of the Gabor noise, which has no random phase.

Examples

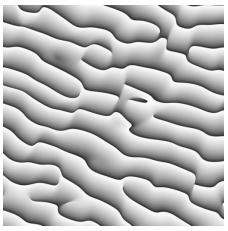
The following table demonstrates different outputs of the node with different parameters. As can be seen, the noise is visually characterized by interleaved bands that are generally oriented in a specific direction. But the *Anisotropy* parameter can be decreased below 1 to make the bands more random in directions. The *Frequency* parameter determines the number of bands perpendicular to the direction of the noise. However, the *Scale* parameter can also be used to globally increase the number of bands, so consider increasing the scale first since high frequency noise can suffer from low contrast and limited interleaving of bands.

Different outputs with different parameters.

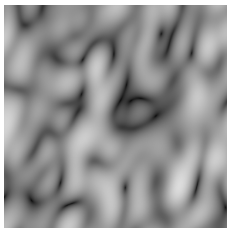
		
Value output. Frequency = 2. Anisotropy = 1.	Phase output. Frequency = 2. Anisotropy = 1.	Intensity output. Frequency = 2. Anisotropy = 1.
		
Value output. Frequency = 3. Anisotropy = 1.	Phase output. Frequency = 3. Anisotropy = 1.	Intensity output. Frequency = 3. Anisotropy = 1.



Value output. Frequency = 2. Anisotropy = 0.7.

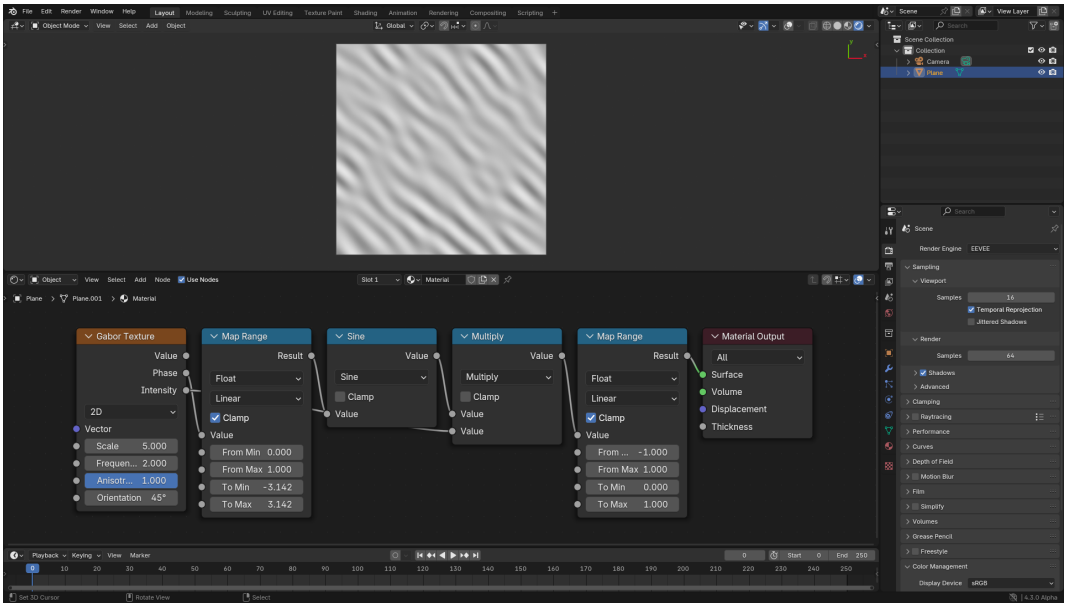


Phase output. Frequency = 2. Anisotropy = 0.7.



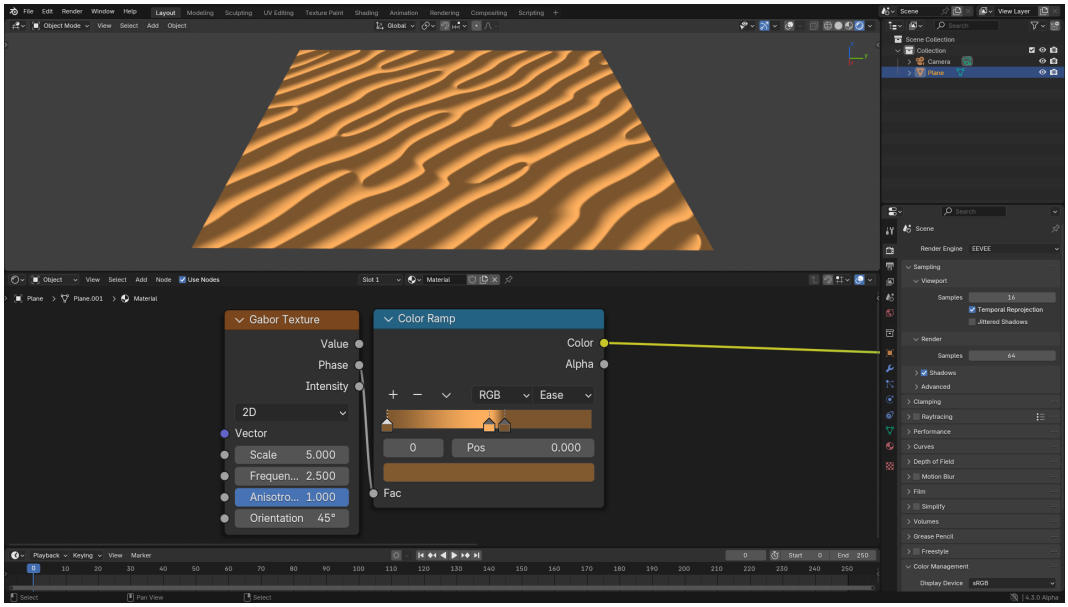
Intensity output. Frequency = 2. Anisotropy = 0.7.

Gabor noise is decomposed into a *Phase* and an *Intensity* components, where the Gabor value is computed as sine the phase multiplied by the intensity, noting that the phase output is normalized to the [0, 1] range.



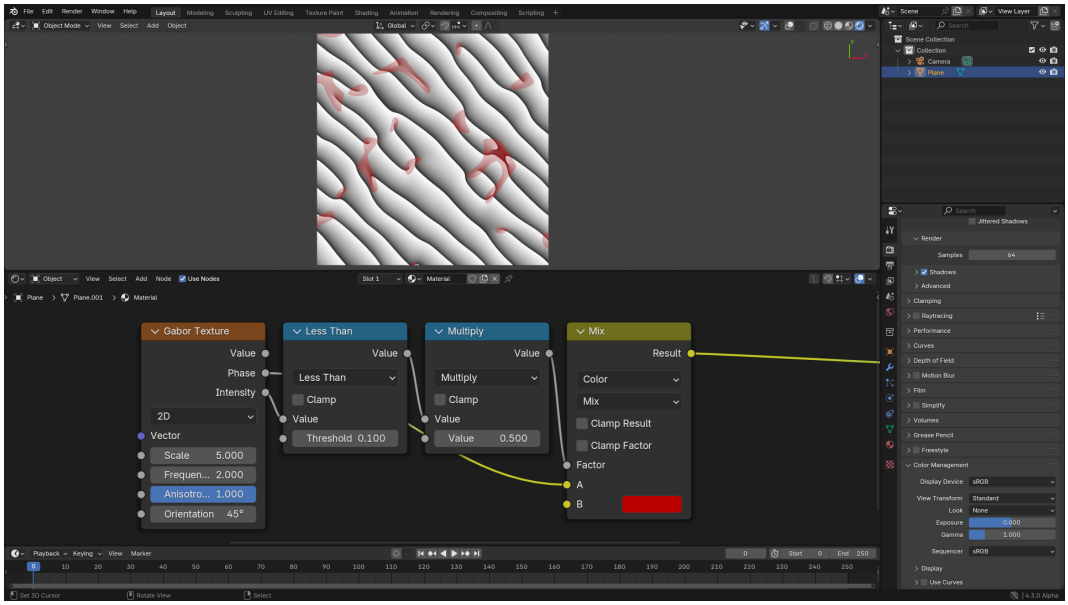
Compute the value output from the phase and intensity outputs.

The advantage of the *Phase* output is that it has no random intensities and no low contrast areas as in the value output, so it can be used as a base for textures that are more structured in nature, like sand dunes.



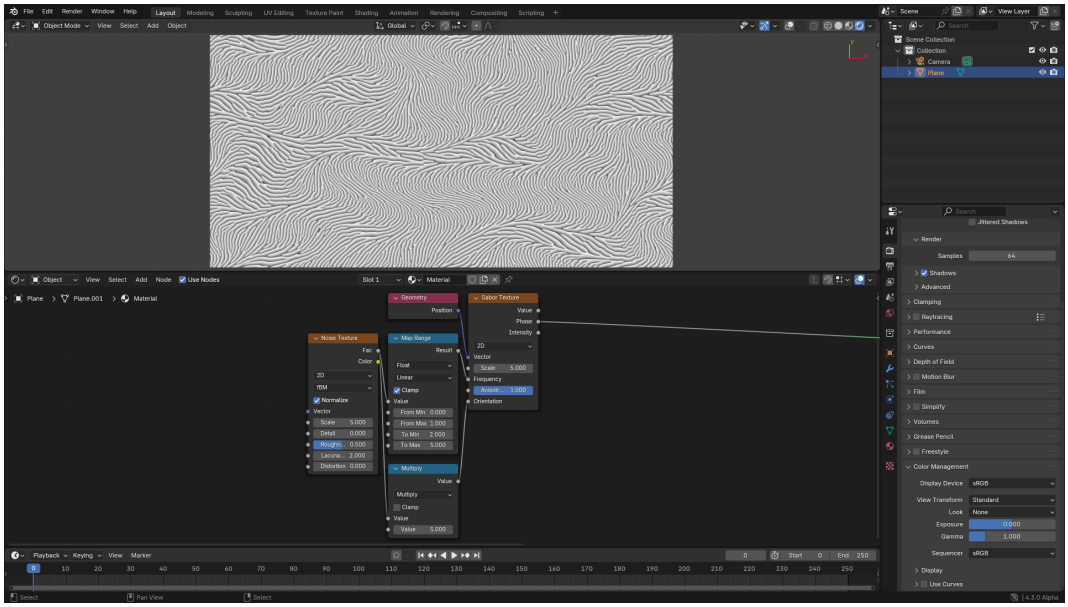
Sand dune-like structures creates using the phase output.

The main advantage and use of the *Intensity* output is that it provides information about the location of *singularities* in the *Phase* output. Singularities are those areas in the phase where the bands meet, which are shown in red in the following figure. Those areas will be close to zero in the *Intensity* output. So if those areas are undesirable, they can be hidden by multiplying by a variant of the *Intensity* output.



Visualization of the areas where singularities happen.

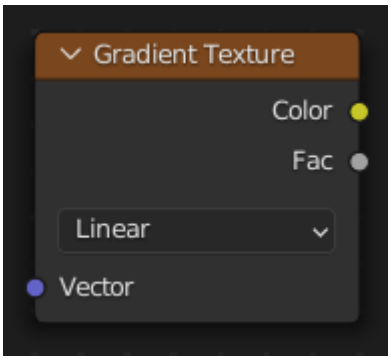
Inputs can be varies across space to get more interesting patterns.



Varying the frequency and orientation across space.

Gradient Texture Node

The *Gradient Texture* node generates interpolated color and intensity values based on the input vector.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Gradient Type
: Controls the type of gradient generated.

Linear:: Directly outputs the input X coordinate.

Quadratic:: Interpolates the input X coordinate quadratically.

Easing:: Uses a combination of quadratic and linear interpolation to generate a smooth gradient from the input X coordinate.

Diagonal:: Averages the input X and Y coordinates.

Spherical:: Creates an inverse gradient using the length of the input vector; the maximum value is at (0, 0, 0).

Quadratic Sphere:: The same as Spherical, except interpolated quadratically.

Radial::

Outputs a value based on the angle of the input around the Z axis.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

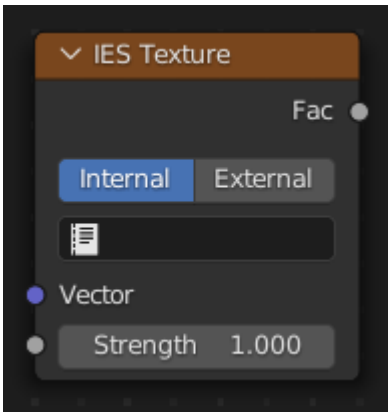
Examples



Gradient texture using object coordinates.

IES Texture Node

The *IES Texture* is used to match real world lights based on IES files (IES). IES files store the directional intensity distribution of light sources.



Inputs

Vector
: Texture coordinate for lookup in the light distribution. Defaults to the normal.

Strength
: Light strength multiplier.

Properties

Mode
: The location to load the IES file from.

Internal::	Use IES profile from a file embedded in a text data-block in the blend-file, for easy distribution.
External::	Load IES profile from a file on the drive.

Outputs

Factor

Light intensity, typically plugged into the Strength input of an Emission node.

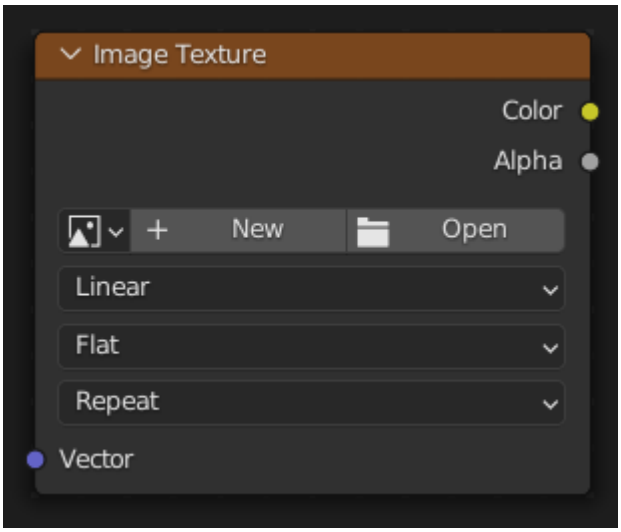
Examples



Lights with different IES profiles.

Image Texture Node

Used for applying an image as a texture.



Inputs

Vector
: 3D coordinate that's projected onto the 2D image using the selected *Projection* method. The node then outputs the color and alpha at this projected point.

  This slot is usually connected to an output of the Texture Coordinate Node. If left unconnected, the coordinate is taken from the object's active UV map (with Z = 0).

Properties

Image
: Image data-block to use.

Interpolation
: Method to scale images up or down for rendering.

Linear:: Regular quality interpolation.

Cubic:: Smoother, better quality interpolation. Bump maps should use this for best results.

Closest:: No interpolation (nearest neighbor). Useful for rendering pixel art.

Smart:: *Cycles Only* Only for Open Shading Language. Use cubic interpolation when scaling up and linear when scaling down, for better performance and sharpness.

Projection

How to project *Vector* onto the image for arriving at a color.

Flat:: Place the image in a unit square (stretching from (0, 0, 0) to (1, 1, 0)) and project the *Vector* vertically onto it. This projection is typically used in combination with UV maps.

Box:: Place the image on each side of a unit cube (stretching from (0, 0, 0) to (1, 1, 1)) and project the *Vector* onto this cube, along the axis that's closest to the mesh normal. This projection is commonly used in architectural models considering these have lots of box-shaped objects.

Blend

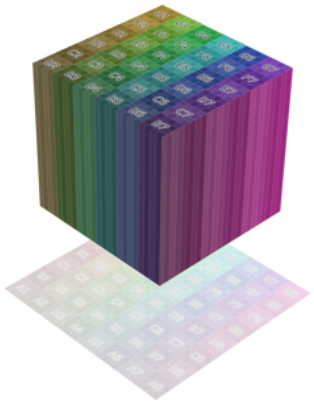
Rather than projecting onto just one side (which creates sharp transitions), project onto multiple sides and blend the results together. The higher the value, the more blending and the smoother the result.

Sphere:: Wrap the image around a sphere with origin (0.5, 0.5, 0.5), and project the *Vector* from this origin onto this sphere. This projection is, of course, perfect for spherical objects such as planets, and is also useful for organic objects.

Tube::

Wrap the image around a cylinder with base (0.5, 0.5, 0) and height 1, and project the *Vector* horizontally from the central axis onto this cylinder. This projection is useful for a label on a bottle, for example. However, it's not suited for the top or bottom side of objects.

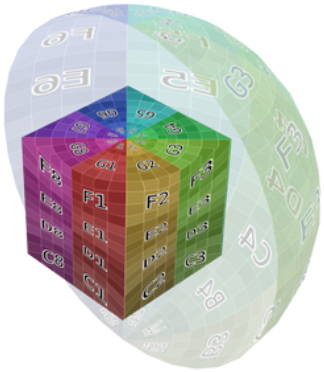
Projections demonstrated using "Object" texture coordinates



Flat projection



Box projection



Sphere projection



Tube projection

Extension

How the image is extrapolated if *Vector* lies outside the regular (0, 0, 0) to (1, 1, 1) bounds:

Repeat:: Repeat the image horizontally and vertically (tiling).

Extend:: Extend the image by repeating the pixels on its edges.

Clip:: Clip to the original image size and set all the exterior pixels values to transparent black.

Mirror:: Repeatedly flip the image horizontally and vertically.

Source

Type of image (Single Image, Movie…). See Image Settings.

Frames

How many frames of the Movie-type image (video) to play. Past this point, the video will be paused (unless *Cyclic* is enabled).

If you want to play the whole video, you can click Match Movie Length in the Image Editor's Sidebar, then copy the *Frames* from there to the node.

Start Frame

Scene frame at which the video should start playing.

Offset

Number of frames to offset the video to an earlier point in time. (Put differently: how many frames at the start of the video to skip.)

Hint

Blender plays video textures at the scene framerate, not their original framerate, meaning they'll be faster or slower than intended if these framerates don't match up. You can put a Driver on the Offset to work around this. Simply type the following into the field, replacing *StartFrame*, *VideoFrameRate* and *SceneFrameRate* by their respective numbers:

#(frame - StartFrame) * (VideoFrameRate - SceneFrameRate) / SceneFrameRate

Cyclic
: Start over after the last frame to create a continuous loop.

Auto Refresh
: Update the video texture in the 3D Viewport when moving through the timeline.

Color Space
: The Color Space the image file was saved in. See Image Settings for details.

Alpha
: How the image uses its Alpha Channel. See Image Settings for details.

Outputs

Color
: RGB color from image. If the image has transparency, the color is premultiplied if the Alpha output is used, and unpremultiplied (straight) otherwise.

Alpha
: Alpha channel from image.

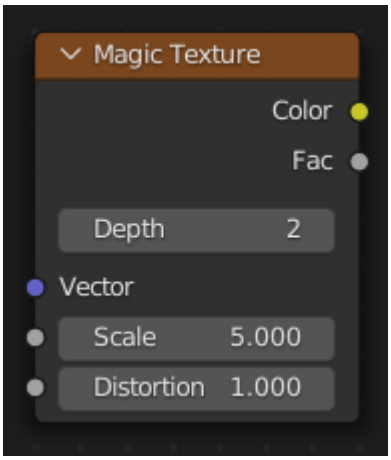
Examples



Image texture from GoodTextures.com.

Magic Texture Node

The Magic Texture node is used to add a psychedelic color texture. It can be used for "Thin Film Interference" if you assign a *Reflection* Texture Coordinate to the Vector input and use a relatively high *Turbulence*. The RGB components are generated independently with a sine formula.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Scale of the texture.

Distortion
: Amount of distortion.

Properties

Depth
: Number of iterations.

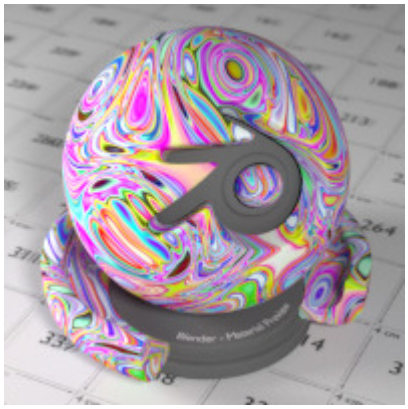
Outputs

Color

Texture color output.

Factor
: Texture intensity output.

Examples



Magic texture: Depth 10, Distortion 2.0.

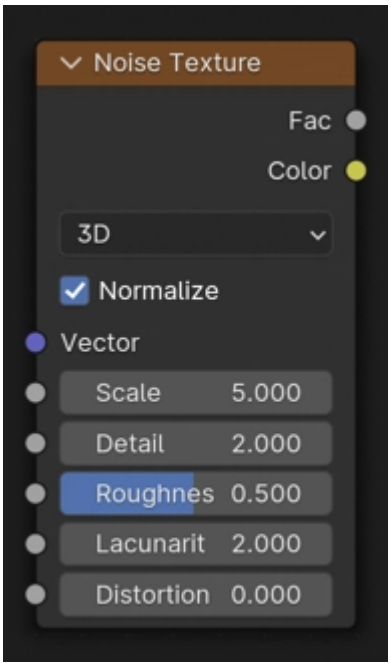
Musgrave Texture Node

The Musgrave texture node was replaced by the Noise Texture node, which includes all the same functionality.

- The Dimension input was replaced by a Roughness input, where \(Roughness = Lacunarity^{-Dimension}\).
- The Detail input value must be subtracted by 1 compared to the old Musgrave Texture node.

Noise Texture Node

The *Noise Texture* node evaluates a fractal Perlin noise at the input texture coordinates. It can be used for a single Perlin noise evaluation, or for combining multiple octaves (layers) with increasingly finer detail.



Inputs

The inputs are dynamic: they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the base noise octave.

Detail

Number of noise octaves. This can have a fractional part, in which case a blend is performed (e.g. a Detail of 2.5 results in a 50% blend between 2 and 3 octaves).

Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

Lacunarity
: The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

Offset
: An added offset to each octave, determines the level where the highest octave will appear.

Gain
: An extra multiplier to tune the magnitude of octaves.

Distortion
: Amount of distortion.

Properties

Dimensions
: The dimensions of the space to evaluate the noise in.

1D::	Evaluate the noise in 1D space at the input *W*.
2D::	Evaluate the noise in 2D space at the input *Vector*. The Z component is ignored.
3D::	Evaluate the noise in 3D space at the input *Vector*.
4D::	Evaluate the noise in 4D space at the input *Vector* and the input *W* as the fourth dimension.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Type
: Type of Noise texture, with different ways to combine octaves.

fBM:: Fractal Brownian motion, produces a homogeneous and isotropic result. Values from octaves are added together.

Multifractal:: More uneven, varying by location similar to real terrain. Values from octaves are multiplied together.

Hybrid Multifractal:: Creates peaks and valleys with different roughness values, like real mountains rise out of flat plains. Combines octaves using both addition and multiplication.

Ridged Multifractal:: Creates sharp peaks. Calculates the absolute value of the noise, creating “canyons”, and then flips the surface upside down.

Hetero Terrain:: Similar to *Hybrid Multifractal* creates a heterogeneous terrain, but with the likeness of river channels.

Normalize *fBM*
: If enabled, ensures that the output values stay in the range 0.0 to 1.0. If disabled, the range is at most -(*Detail* + 1) to *Detail* + 1 (smaller if *Roughness* < 1).

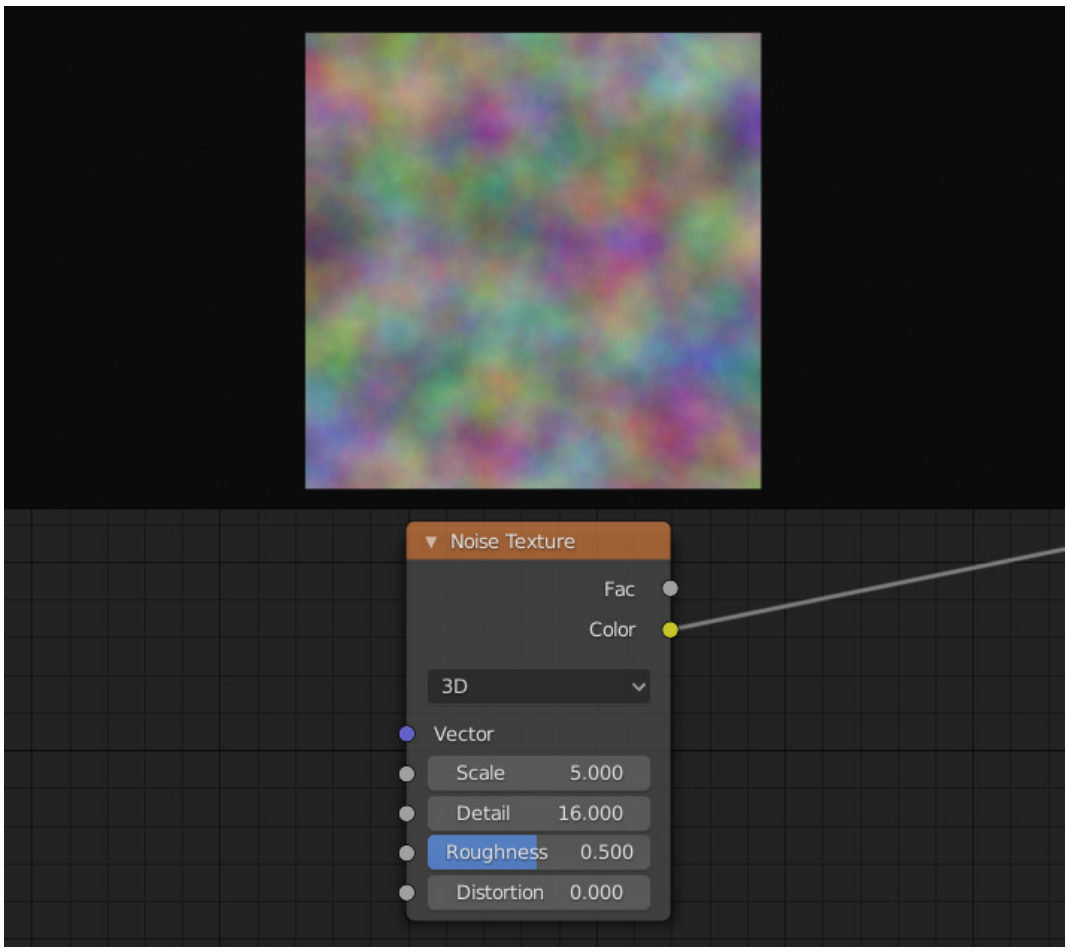
Outputs

Factor
: Value of fractal noise.

Color

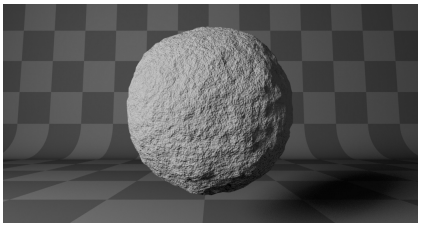
Color with different fractal noise in each component.

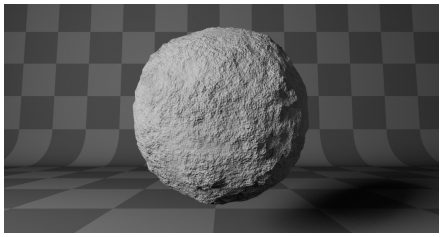
Examples



Noise Texture with high detail.

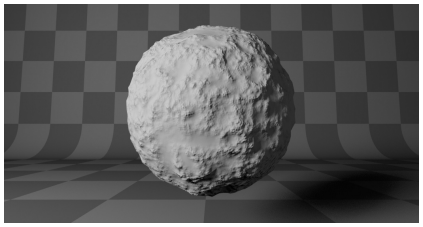
Different Noise types with the same parameters.





Multifractal.

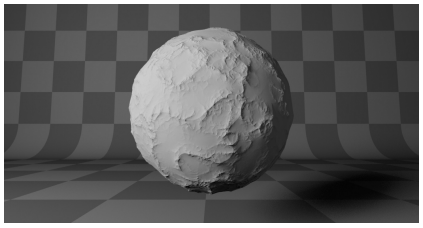
fBM (fractal Brownian Motion).



Hybrid Multifractal.



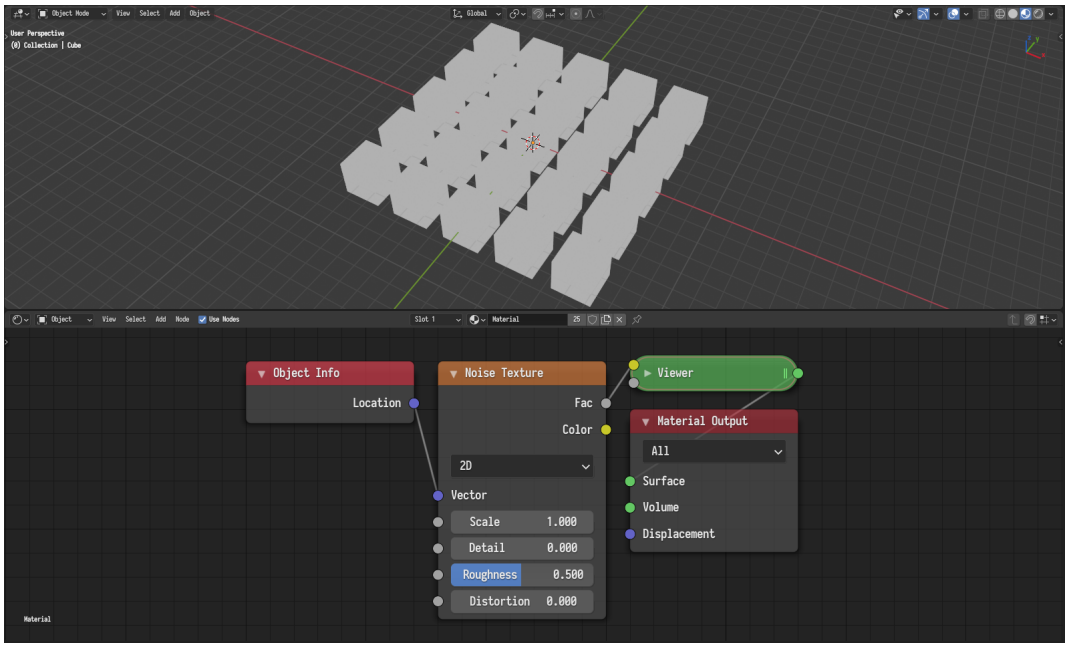
Heterogeneous Terrain.



Ridged Multifractal.

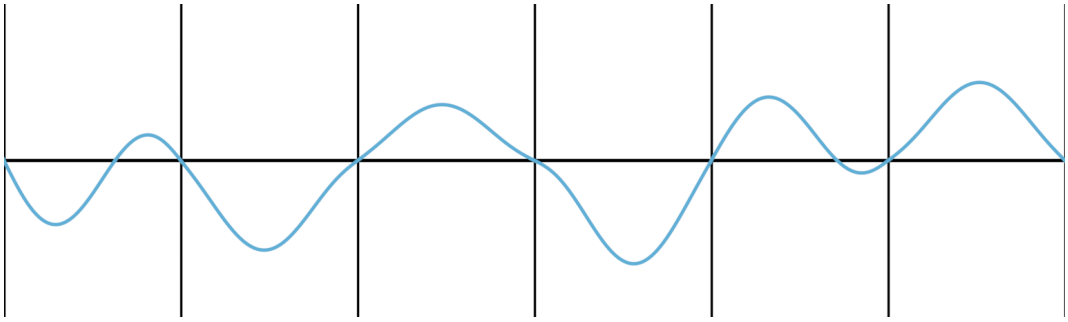
Notes

While the noise is random in nature, it follows a certain pattern that might not evaluate to random values in some configurations. For instance, consider the following configuration where a grid of objects have a material that evaluates a noise texture at their locations. One might expect the objects to have random values since they have different locations, but this is not the case.



An example configuration where the noise evaluates to a constant value.

It seems all objects have a value of 0.5. To understand why this happens, let us look at the following plot of a 1D noise texture.



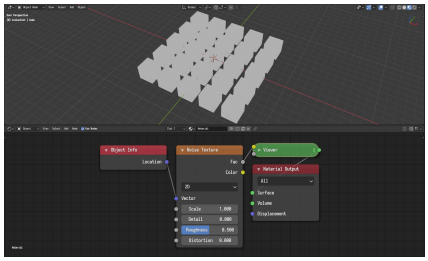
A plot of a 1D noise with zero details and zero distortion.

The horizontal line denotes a value of 0.5 and the vertical lines denotes whole numbers assuming a noise scale of 1. As can be seen, the noise always intersects the 0.5 line at whole numbers. Since the aforementioned objects were distributed on a grid and have whole number locations, they all evaluate to 0.5. Which explains the issue at hand.

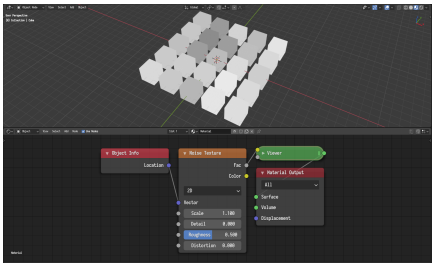
Generally, any discrete evaluation of noise at integer multiples of the reciprocal of the noise scale will always evaluate to 0.5. It also follows that evaluations closer to that will have values close to 0.5. In such cases, it is almost always preferred to use the White Noise Texture.

Regardless, one can mitigate this issue in a number of ways:

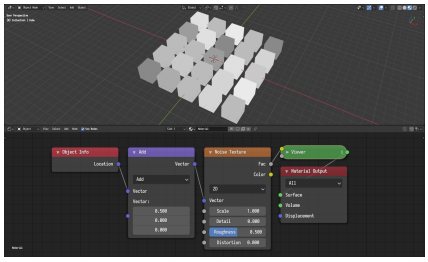
- Adjust the scale of the noise to avoid aligning the noise with the evaluation domain.
- Add an arbitrary offset to the texture coordinates to break the alignment with the evaluation domain.
- Evaluate the noise at a higher dimension and adjust the extra dimension until a satisfactory result is achieved.



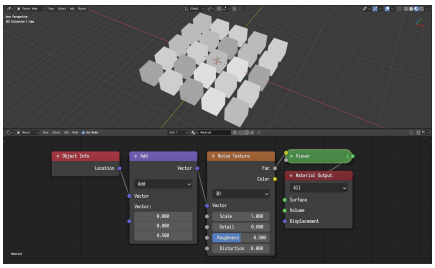
Constant value issue.



Mitigating the issue by adjusting the scale.

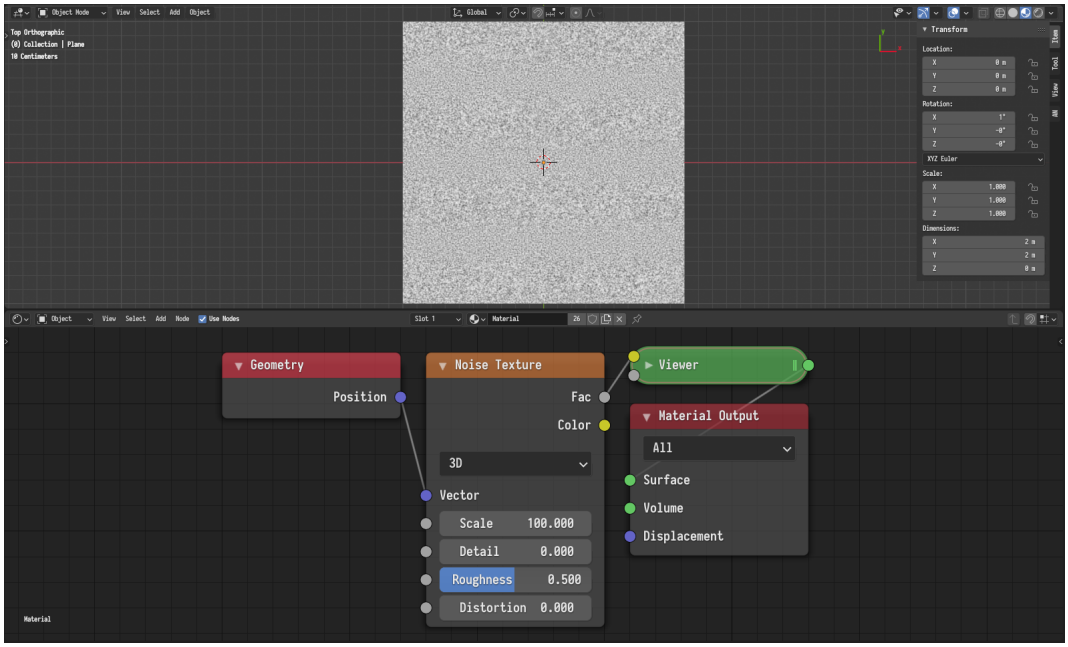


Mitigating the issue by adding an arbitrary offset.



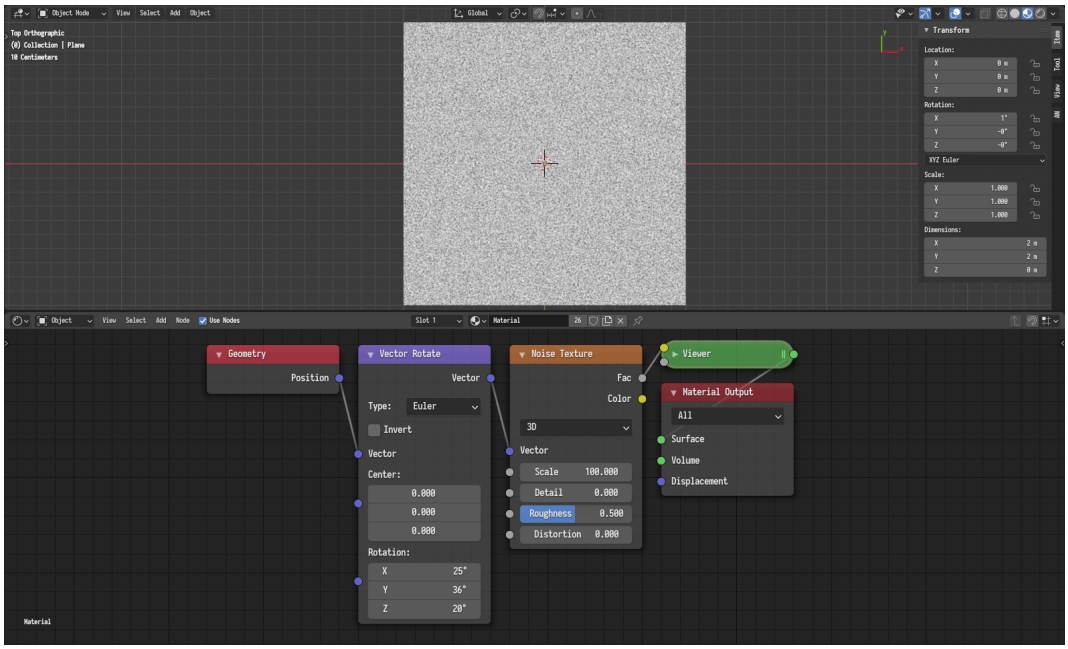
Mitigating the issue by evaluating at a higher dimension.

Similarly, in other configurations, one might experience some banding patterns in the noise, where there are bands of high contrast areas followed by banding of low contrast areas. For instance, planar surfaces that are slightly tilted along one of the axis will have such a banding pattern.



An example configuration where the noise have a banding pattern.

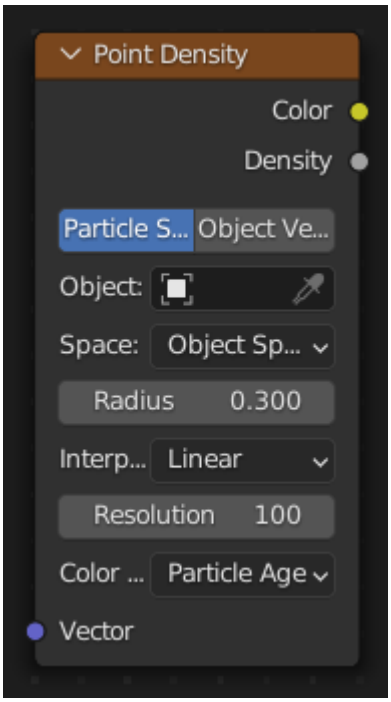
This happens because the slight tilt along one of the axis causes values along the perpendicular axis to change very slowly making the grid structure of the noise more apparent. The easiest way to mitigate this issue to rotate the coordinates by an arbitrary amount.



Mitigating the issue by rotating the coordinates by an arbitrary amount.

Point Density Node

The *Point Density* node is available in volume shaders, to render volumetric points for each particle or vertex of another object.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to global position (*Position* output of *Geometry* node) if the socket is left unconnected.

Properties

Point Data
: Where to get points from.

  Particle System
  : Use each particle position from the specified particle system.

  Object Vertices

Use each vertex position from the specified object.

Object
: Which object's vertices or particle system will be used.

Particle System
: Particle positions from this system will be used.

Space
: The coordinate system for mapping points.

World Space::	Map each point exactly where the source particle or vertex is.
Object Space::	Fit the points from the source particles/vertices inside the bounding box of the object with the point density texture.

Radius
: Size of the points.

Interpolation
: Texel filtering type.

Closest::	No interpolation, use nearest texel. Produces blocky looking points.
Linear::	Interpolate linearly between texels, producing soft, round points.
Cubic::	Use cubic falloff, producing very soft points. Useful when points are very densely packed.

Resolution
: The dimensions of the texture holding the point data.

Color Source
: Which attribute of the particle system or mesh is used to color the output.

  Particle Color Sources

Particle Age:: Lifetime mapped as (0.0 - 1.0) intensity.

Particle Speed:: Particle speed (absolute magnitude of velocity) mapped as (0.0 - 1.0) intensity.

Particle Velocity:: XYZ velocity mapped to RGB colors.

Vertex Color Sources

Vertex Color:: Use a Color Attribute for coloring the point density texture.

> **Note**
> Color Attributes are defined per face corner. A single vertex can have as many different colors as faces it is part of. The actual color of the point density texture is averaged from all vertex corners.

Vertex Weight:: Use weights from a vertex group as intensity values.

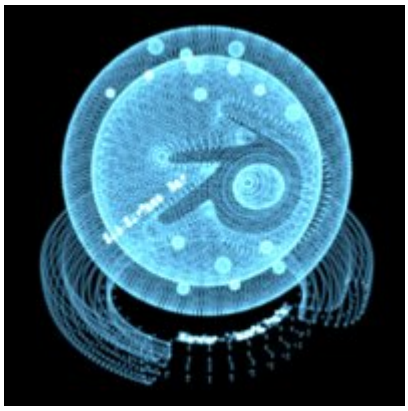
Vertex Normals:: Use object-space vertex normals as RGB values.

Outputs

Color
: Texture color output.

Density
: Density of volume.

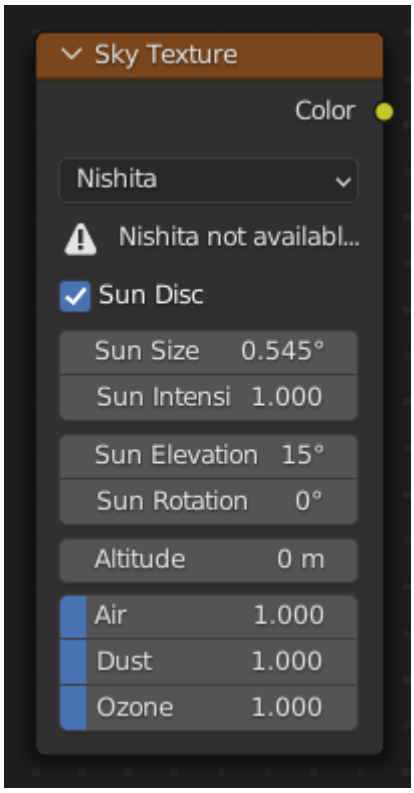
Examples



Domain object with Point Density texture using vertices from ball as points.

Sky Texture Node

The *Sky Texture* node generates a procedural sky. It's typically used in combination with the World Output Node.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Sky Type
: Sky model to use.

  Preetham
  : Based on the 1999 paper by Preetham et al.

Hosek/Wilkie
: Based on the 2012 paper by Hosek and Wilkie.

Nishita
: Improved version of the 1993 model by Nishita et al.

  Note that this sky type is quite bright and makes the image look overexposed with the default scene settings. You can reduce the Exposure setting in *Properties* ▸ *Render* ▸ *Film* to fix this.

Sun Direction
: Sun direction vector.

Turbidity
: Atmospheric turbidity.

  - 2: Arctic like
  - 3: clear sky
  - 6: warm/moist day
  - 10: hazy day

Ground Albedo
: Amount of light reflected from the planet surface back into the atmosphere.

Sun Disc *Cycles Only*
: Enable/Disable sun disc lighting.

Sun Size
: Angular diameter of the sun disc (in degrees).

Sun Intensity
: Multiplier for sun disc lighting.

Sun Elevation
: Rotation of the sun from the horizon (in degrees).

Sun Rotation

Rotation of the sun around the zenith (in degrees).

Altitude
: The distance from sea level to the location of the camera. For example, if the camera is placed on a beach then a value of 0 should be used. However, if the camera is in the cockpit of a flying airplane then a value of 10 km will be more suitable. Note, this is limited to 60 km because the mathematical model only accounts for the first two layers of the earth's atmosphere (which ends around 60 km).

Air
: Density of air molecules.

- 0 no air
- 1 clear day atmosphere
- 2 highly polluted day

Dust
: Density of dust and water droplets.

- 0 no dust
- 1 clear day atmosphere
- 5 city like atmosphere
- 10 hazy day

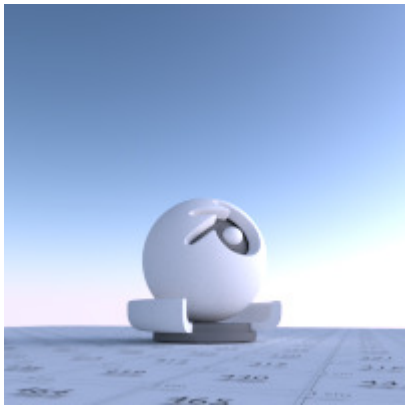
Ozone
: Density of ozone molecules; useful to make the sky appear bluer.

- 0 no ozone
- 1 clear day atmosphere
- 2 city like atmosphere

Outputs

Color
: Texture color output.

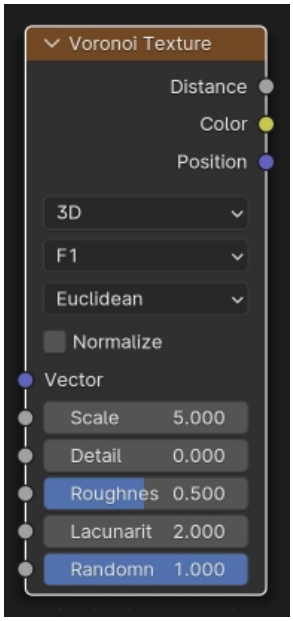
Examples



Example of Sky Texture.

Voronoi Texture Node

The *Voronoi Texture* node evaluates a Worley Noise at the input texture coordinates.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the noise.

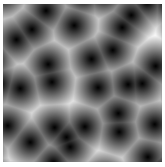
Detail
: Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher evaluation time.

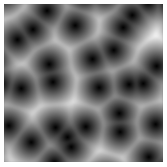
Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

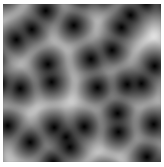
Lacunarity
: The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

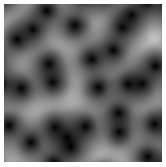
Smoothness

The smoothness of the noise.


Smoothness: 0.0.


Smoothness: 0.25.


Smoothness: 0.5.


Smoothness: 1.0.


Smoothness: 0.0.

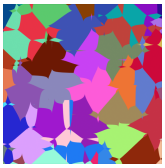

Smoothness: 0.25.


Smoothness: 0.5.


Smoothness: 1.0.

Exponent

Exponent of the Minkowski distance metric.


Exponent: 0.5.


Exponent: 1.0.

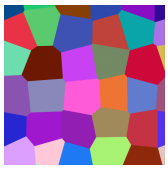

Exponent: 2.0.

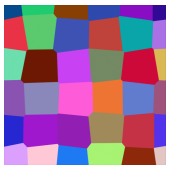

Exponent: 32.0.

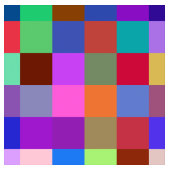
Randomness

The randomness of the noise.


Randomness: 1.0.


Randomness: 0.5.


Randomness: 0.25.


Randomness: 0.0.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

1D:: Evaluate the noise in 1D space at the input W.

2D:: Evaluate the noise in 2D space at the input Vector. The Z component is ignored.

3D:: Evaluate the noise in 3D space at the input Vector.

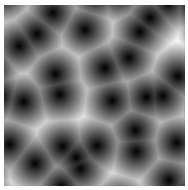
4D:: Evaluate the noise in 4D space at the input Vector and the input W as the fourth dimension.

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Feature

The Voronoi feature that the node will compute.

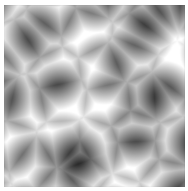
F1:: The distance to the closest feature point as well as its position and color.

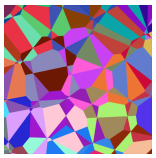

Distance.

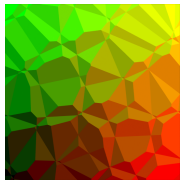

Color.


Position.

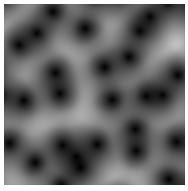
F2:: The distance to the second closest feature point as well as its position and color.


Distance.


Color.


Position.

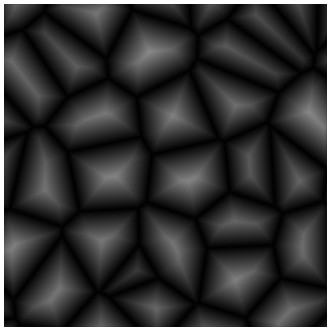
Smooth F1:: A smooth version of F1.


Distance.

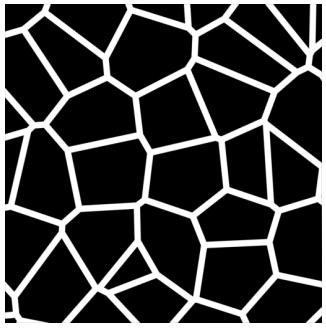

Color.


Position.

Distance to Edge:: The distance to the edges of the Voronoi cells.

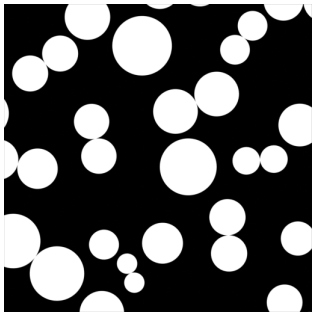


Distance.

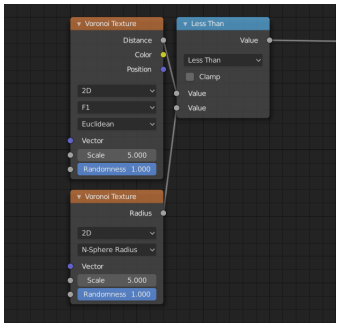


Distance smaller than 0.05.

N-Sphere Radius:: The radius of the n-sphere inscribed in the Voronoi cells. In other words, it is half the distance between the closest feature point and the feature point closest to it.



The n-sphere radius can be used to create tightly packed n-spheres.

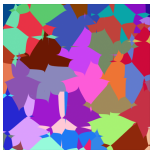


Node tree for the shader to the left.

Distance Metric

The distance metric used to compute the texture.

Euclidean:: Use the Euclidean distance metric.
Manhattan:: Use the Manhattan distance metric.
Chebychev:: Use the Chebychev distance metric.
Minkowski:: Use the Minkowski distance metric. The Minkowski distance is a generalization of the aforementioned metrics with an *Exponent* as a parameter. Minkowski with an exponent of one is equivalent to the *Manhattan* distance metric. Minkowski with an exponent of two is equivalent to the *Euclidean* distance metric. Minkowski with an infinite exponent is equivalent to the *Chebychev* distance metric.









Minkowski Exponent: 0.5 (Minkowski 1/2).	Minkowski Exponent: 1.0 (Manhattan).	Minkowski Exponent: 2.0 (Euclidean).	Minkowski Exponent: 32.0 (approximation of Chebychev).

Normalize
: If enabled, ensures that the output values stay in the range 0.0 to 1.0. In rare cases, the output value may be outside that range when *Feature* is *F2*.

Outputs

Distance
: Distance.

Color
: Cell color. The color is arbitrary.

Position
: Position of feature point.

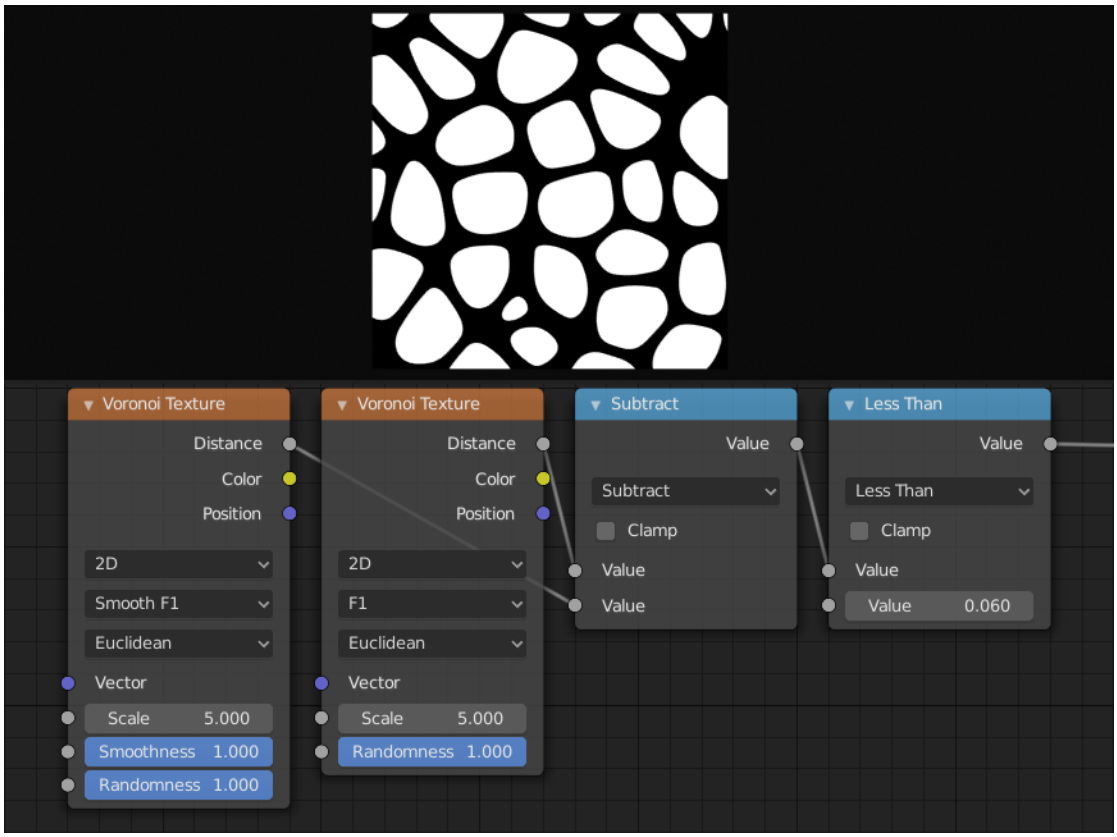
W
: Position of feature point.

Radius
: N-Sphere radius.

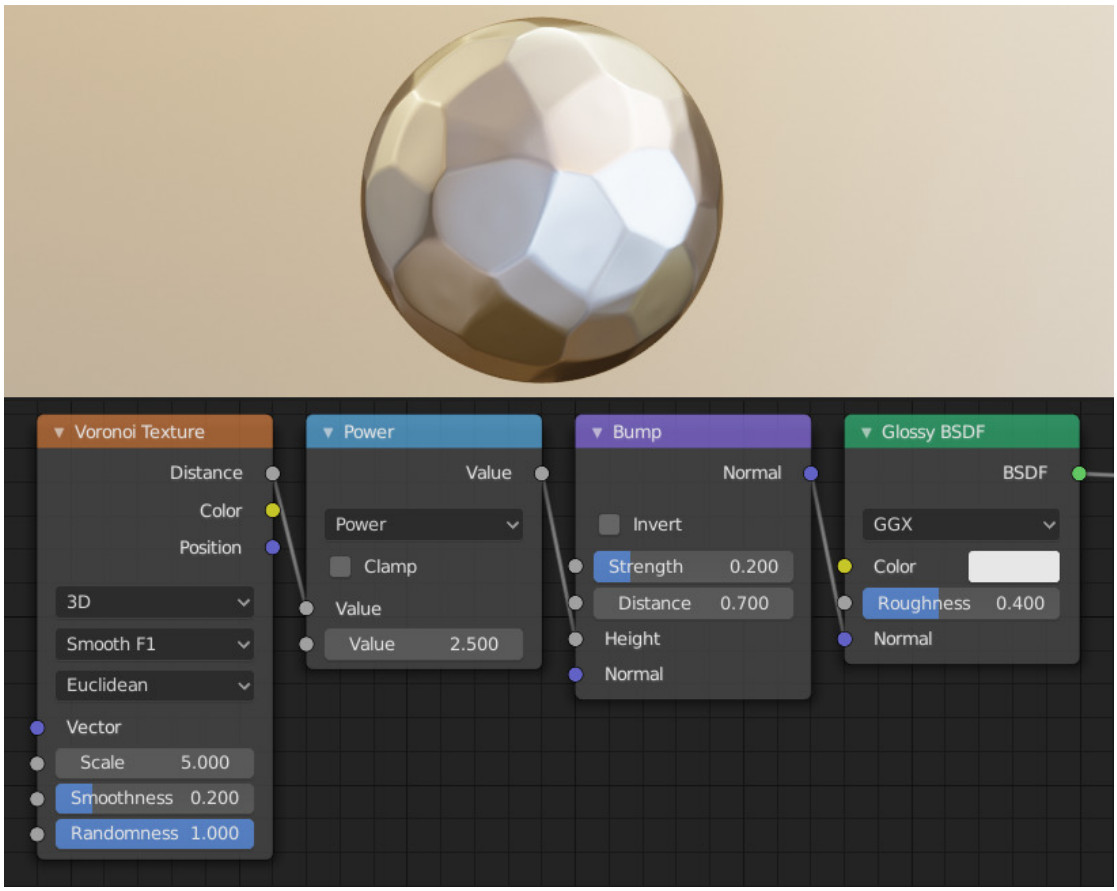
Notes

In some configurations of the node, especially for low values of *Randomness*, rendering artifacts may occur. This happens due to the same reasons described in the Notes section in the White Noise Texture page and can be fixed in a similar manner as described there.

Examples



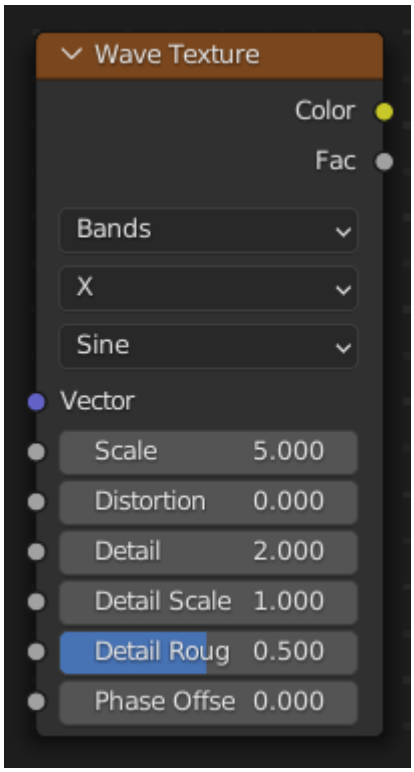
The difference between *F1* and *Smooth F1* can be used to create beveled Voronoi cells.



Creating a hammered metal shader using the *Voronoi Texture* node.

Wave Texture Node

The *Wave Texture* node adds procedural bands or rings with noise distortion.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Overall texture scale.

Distortion
: Amount of distortion of the wave.

Hint

In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.

To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion*/*Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale*, and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Detail
: Amount of distortion noise detail.

Detail Scale
: Scale of distortion noise.

Roughness
: Blend between a smoother noise pattern, and rougher with sharper peaks.

Phase Offset
: Position of the wave along the *Bands Direction*. This can be used as an input for more control over the distortion.

Properties

Type
: *Bands* or *Rings* shaped waves.

Bands/Rings Direction
: The axis the bands or rings propagate from i.e. which axis they are perpendicular to. When using *Bands* a *Diagonal* axis is an option and when using *Rings* the rings can propagate outwards from a single point by using *Spherical* direction.

Wave Profile
: Controls the look of the wave type.

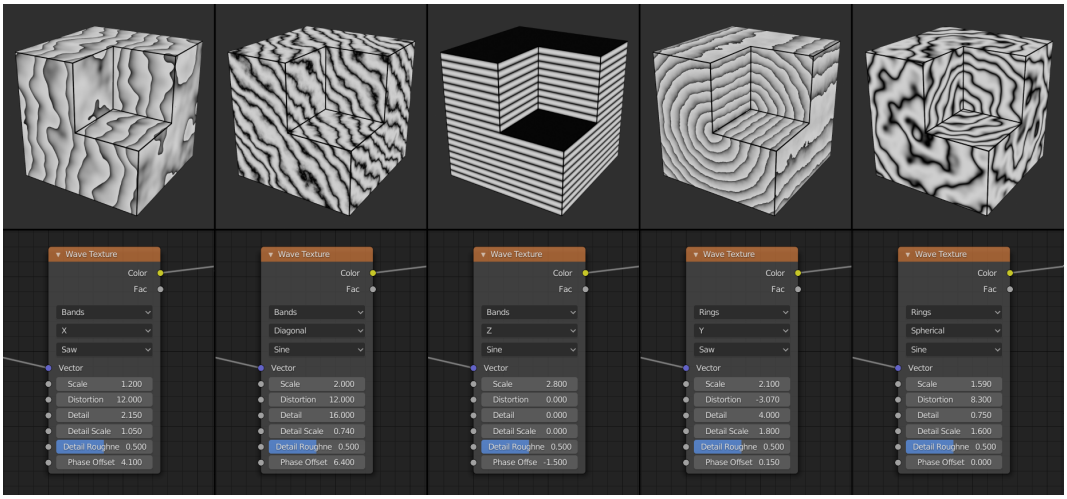
Saw:: Uses a sawtooth profile.
Sine:: Uses the standard sine profile.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

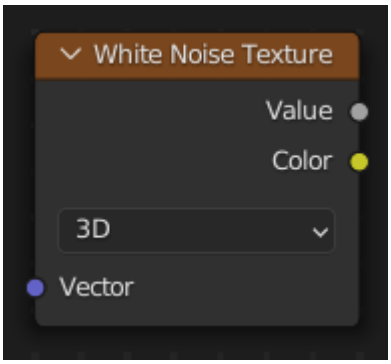
Examples



Wave Texture.

White Noise Texture Node

The *White Noise Texture* node returns a random number based on an input Seed. The seed can be a number, a 2D vector, a 3D vector, or a 4D vector; depending on the *Dimensions* property. The output number ranges between zero and one.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Vector used as seed in 2D, 3D, and 4D dimensions.

W
: Value used as seed in 1D and 4D dimensions.

Properties

Dimensions
: The dimensions of the space to evaluate the noise in.

1D:: The *W* input is used as seed.
2D::

	The X and Y components of the *Vector* input are used as seed.
3D::	The *Vector* input is used as seed.
4D::	Both the *Vector* input and the *W* input are used as seed.

Outputs

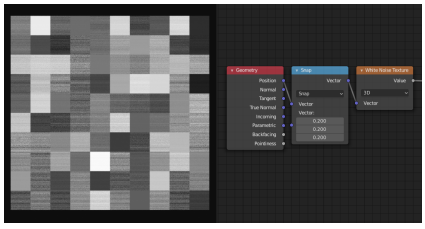
Value
: Output random value.

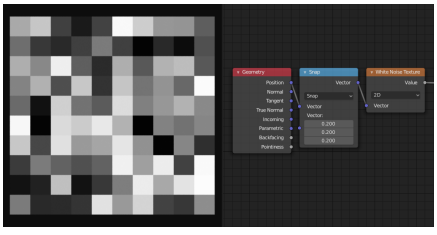
Color
: Output random color.

Notes

The slightest difference in seed values would result in completely different outputs. Consequently, bad precision may have significant impact on the output. Usually, we can mitigate this issue by:

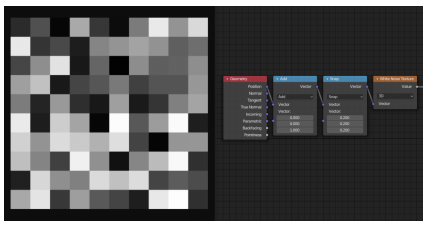
- Eliminating the problematic seed value. If the problematic seed value is constant, it should be eliminated by choosing a lower dimension or multiplying it by zero.
- Adding an arbitrary value to the seed. The issue might only happen at certain boundaries, like unit boundaries, so simply adding an arbitrary value might solve the issue.
- Taking the absolute value of the seed. In computing, zero may be positive or negative, so taking the absolute values unifies the zero into a single value.



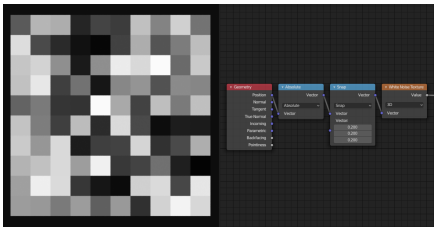


Precision issue due to signed zeros on the Z axis.

Mitigating the issue by eliminating the Z axis.

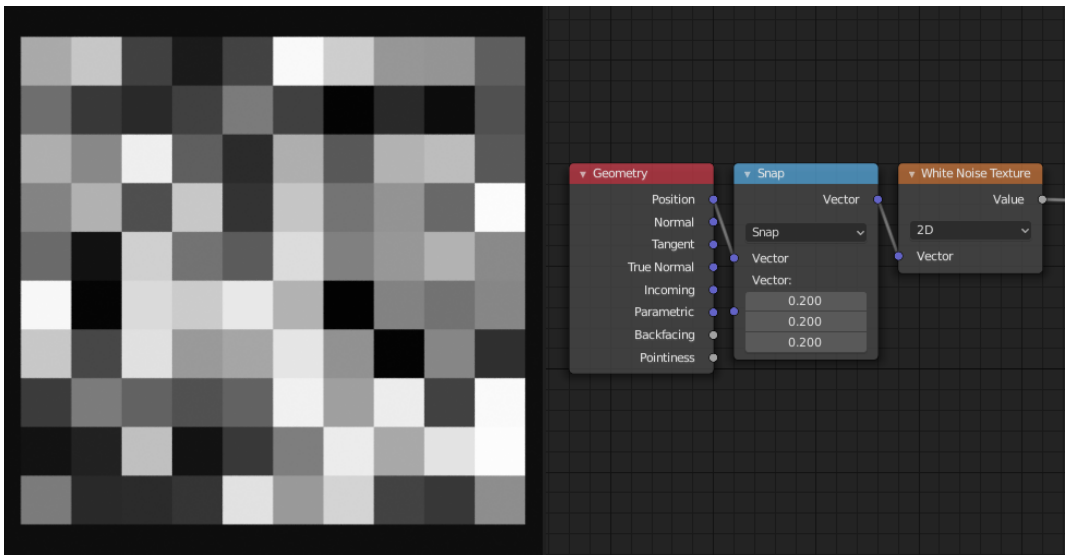


Mitigating the issue by adding an arbitrary value.



Mitigating the issue by taking the absolute value.

Examples

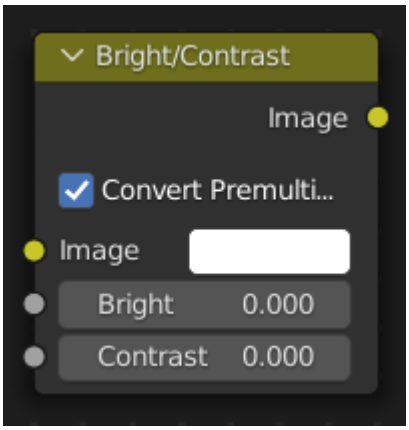


Generating cell noise using the *Snap* vector operation and the *White Noise* node.

Color

- Brightness/Contrast Node
- Gamma Node
- Hue/Saturation/Value Node
- Invert Color Node
- Light Falloff Node
- Mix Color Node
- RGB Curves Node

Brightness/Contrast Node



Inputs

Image
: Standard color input.

Brightness
: An additive-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast
: A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

Convert Premultiplied
: By default, it is supposed to work in *premultiplied* alpha. If the *Convert Premul* checkbox is not enabled, it is supposed to work in *straight* alpha.

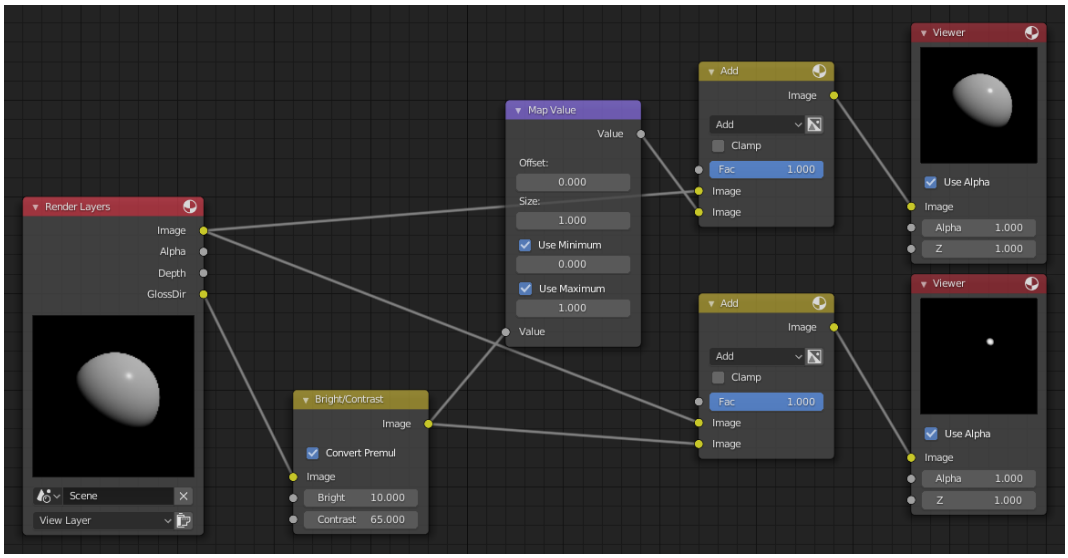
See Alpha Channel.

Outputs

Image
: Standard color output.

Notes

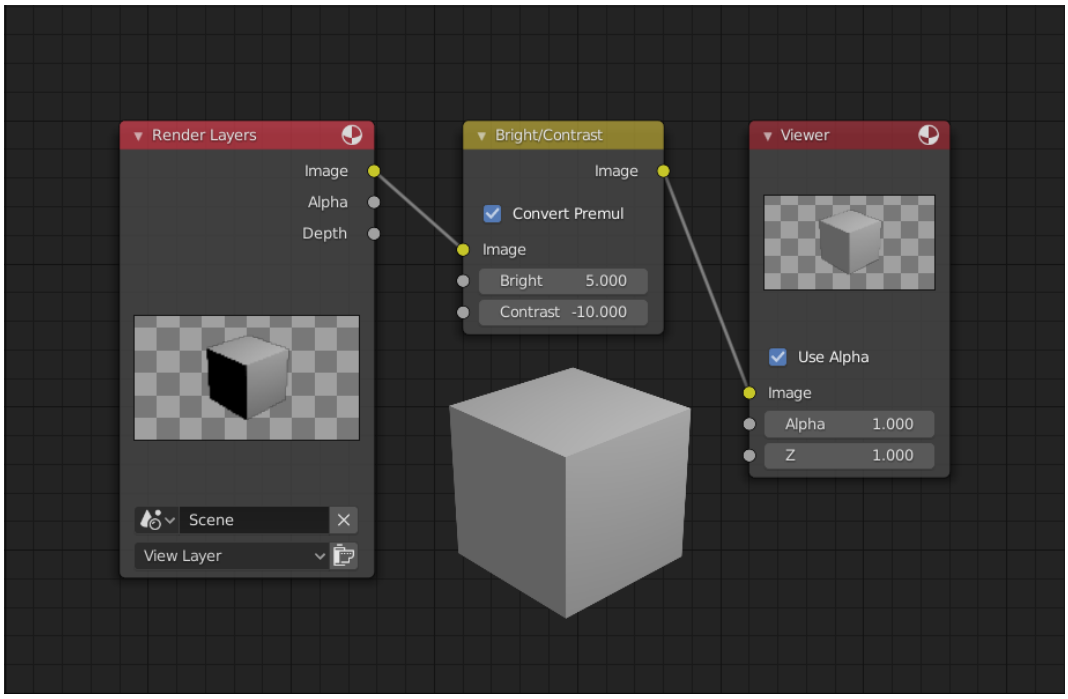
It is possible that this node will put out a value set that has values beyond the normal range, i.e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).



Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to intensify the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has a value much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

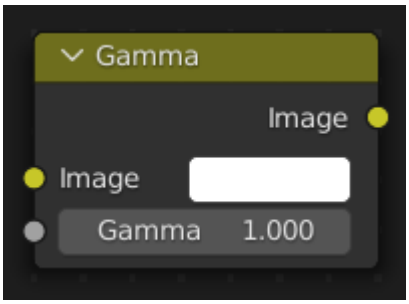
Example



A basic example.

Gamma Node

Use this node to apply a gamma correction.



Inputs

Image
: Standard color input.

Gamma
: An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image
: Standard color output.

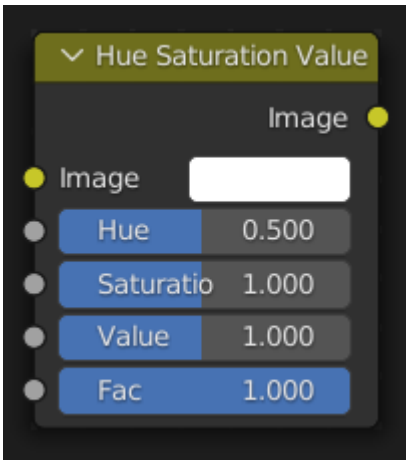
Examples



Example of a Gamma node.

Hue/Saturation/Value Node

The *Hue/Saturation/Value Node* applies a color transformation in the HSV Color Model.



Inputs

Factor
: The amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Hue
: The hue rotation offset, from 0 (-180°) to 1 (+180°). Note that 0 and 1 have the same result.

Saturation
: A value of 0 removes color from the image, making it black-and-white. A value greater than 1.0 increases saturation.

Value
: The value shift. 0 makes the color black, 1 keeps it the same, and higher values make it brighter.

Outputs

Image/Color
: Standard color output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are laid out on a circle
: If you apply a Hue offset of 1 (+180°) to a blue image, you get the diametrically opposite color, which is yellow. If you apply a Hue offset of 1 to that yellow image, you get blue again.

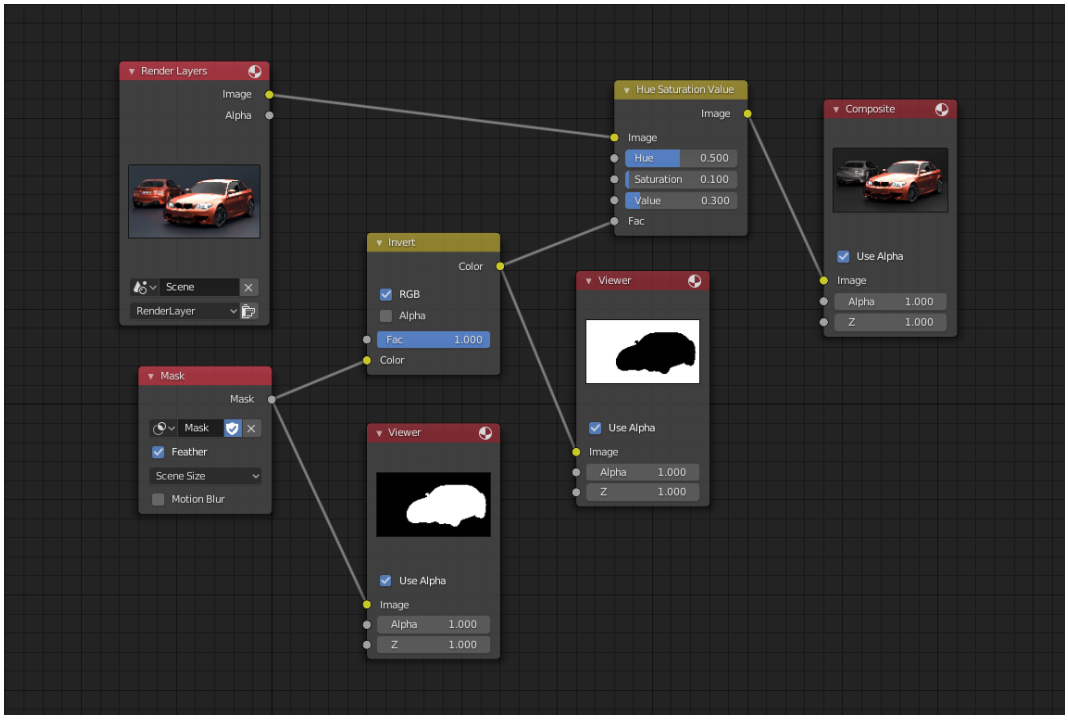
Grayscale images have no hue
: Trying to change the Hue or Saturation of a grayscale image has no effect. You can only brighten or darken it by adjusting the Value. To add color, use the Mix node instead.

Changing the effect over time
: The different values can be animated using a *Time Curve* node or by setting keyframes.

HSV Example



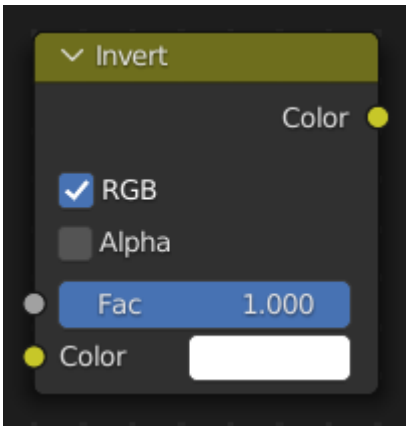
A basic example.



An example of using the Factor input for masking.

Invert Color Node

Inverts the colors in the input image, producing a negative.



Inputs

Factor
: The amount of influence the node exerts on the image.

Color
: Standard color input.

Properties

In the compositing context, this node has the following properties:

RGB
: Invert the color channels.

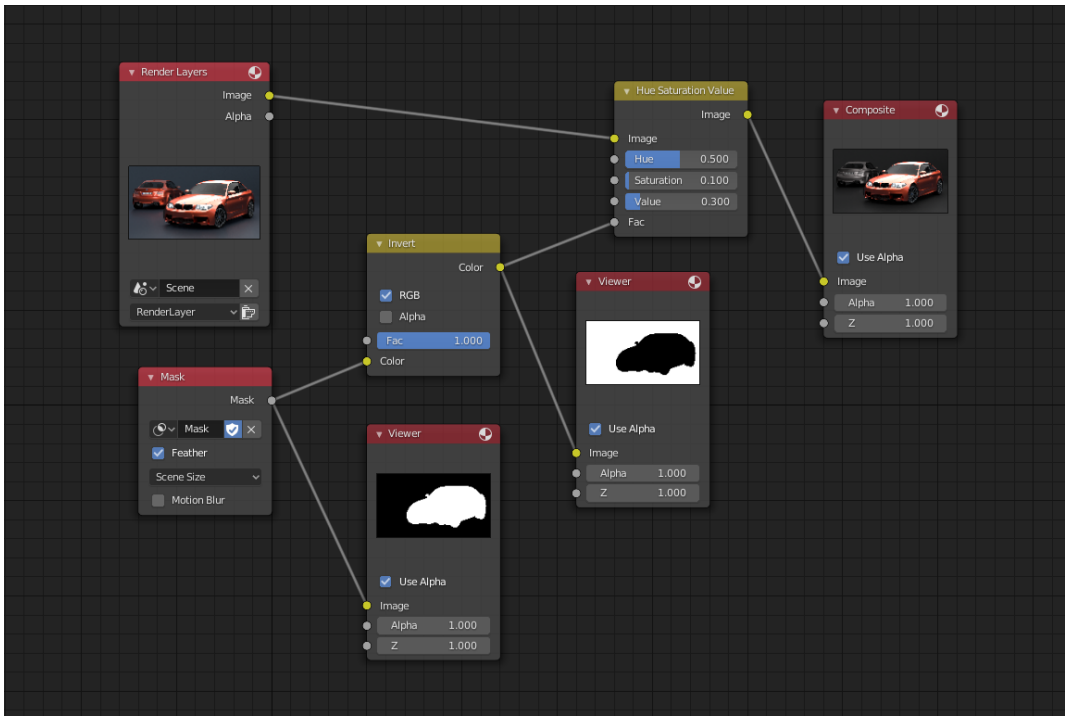
Alpha
: Invert the alpha channel.

Outputs

Color

Standard color output.

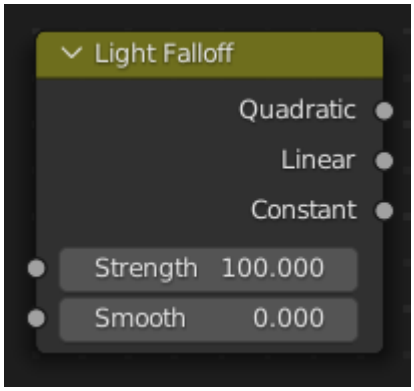
Example



The Invert node is used to invert the mask.

Light Falloff Node

Cycles Only



The *Light Falloff* node allows you to manipulate how light intensity decreases over distance. In reality light will always fall off quadratically; however, it can be useful to manipulate as a non-physically-based lighting trick. Note that using Linear or Constant falloff may cause more light to be introduced with every global illumination bounce, making the resulting image extremely bright if many bounces are used.

Inputs

Strength
: Light strength before applying falloff modification.

Smooth
: Smooth intensity of light near light sources. This can avoid harsh highlights, and reduce global illumination noise. 0.0 corresponds to no smoothing; higher values smooth more. The maximum light strength will be strength/smooth.

Properties

This node has no properties.

Outputs

Quadratic

Quadratic light falloff; this will leave strength unmodified if smooth is 0.0 and corresponds to reality.

Linear
: Linear light falloff, giving a slower decrease in intensity over distance.

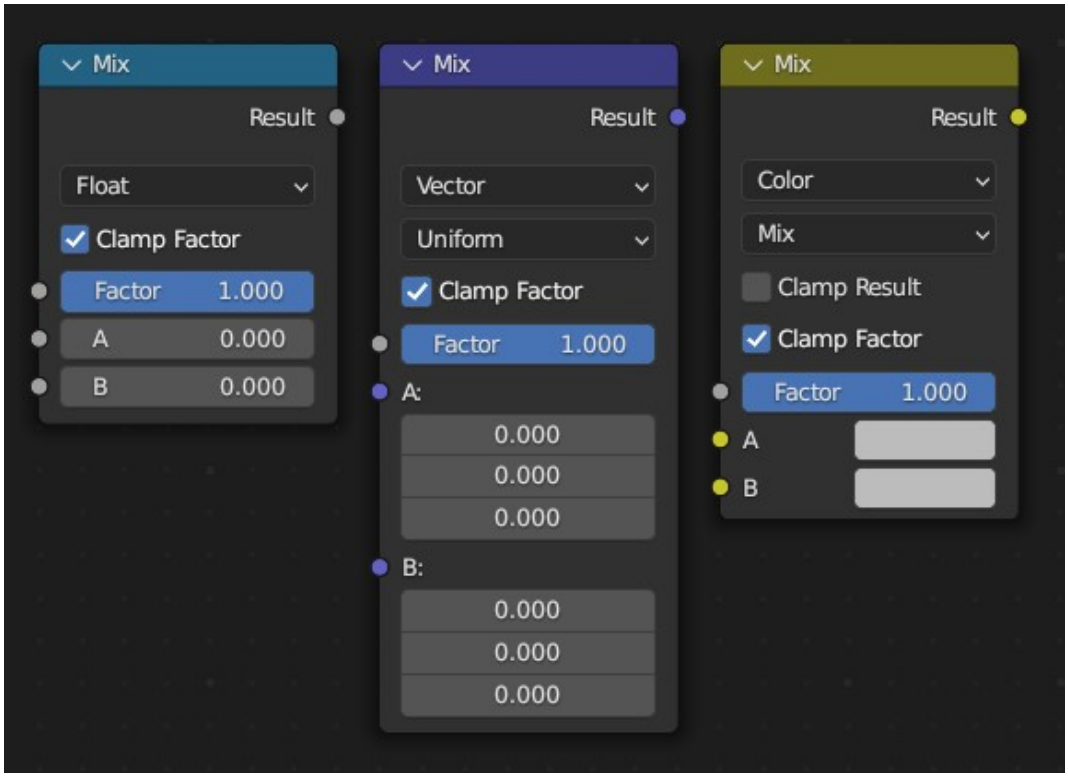
Constant
: Constant light falloff, where the distance to the light has no influence on its intensity.

Examples

Todo <2.8 add.

Mix Color Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.



Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
: The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
: The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
: Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
: Limit the Result to the range between 0.0 and 1.0.

Outputs

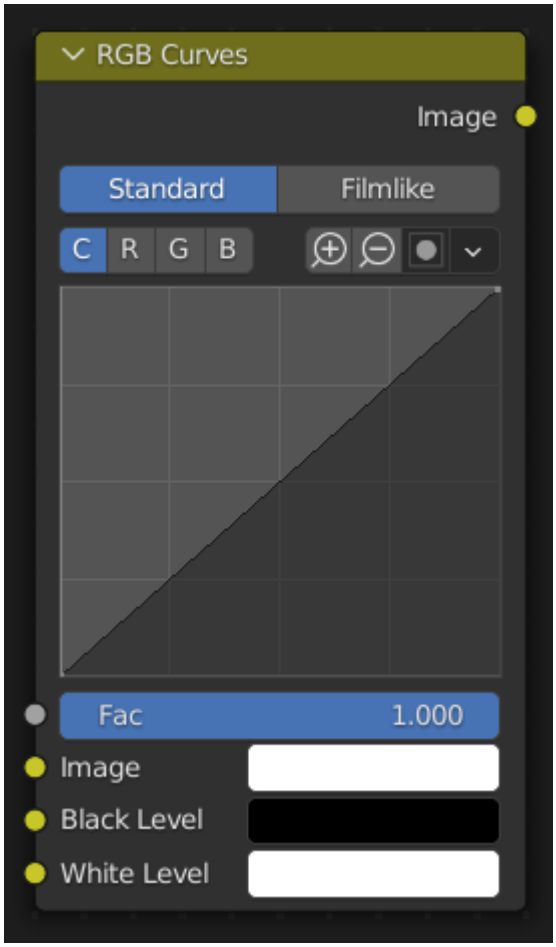
Result
: Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.



Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*
: Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel
: The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve
: A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

Outputs

Image/Color
: Standard color output.

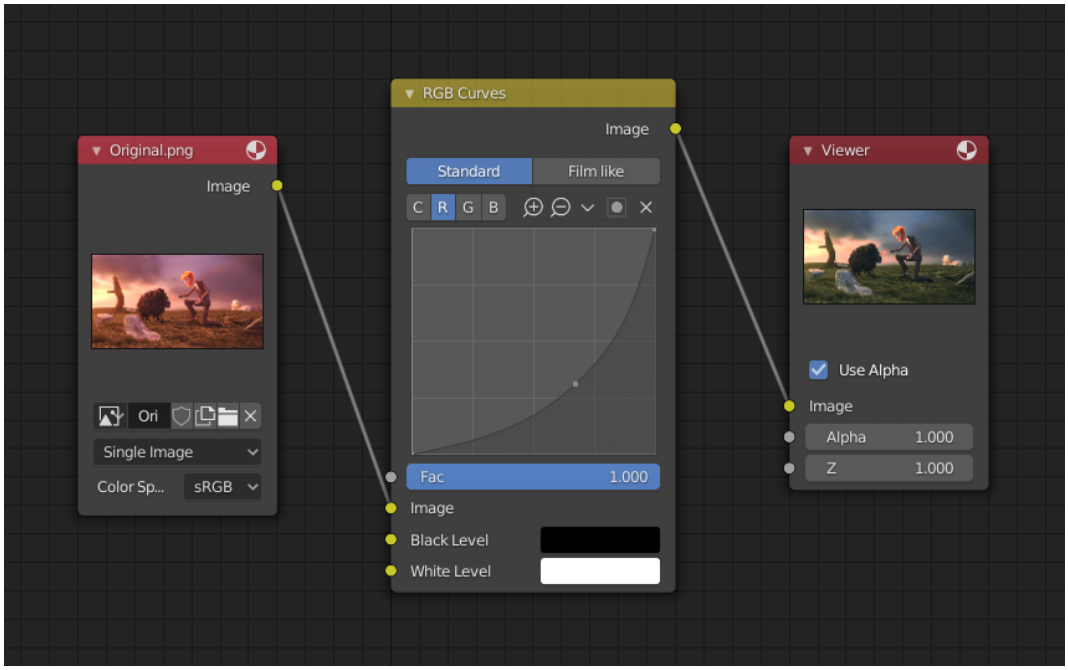
Examples

Below are some common curves you can use to achieve desired effects.



From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

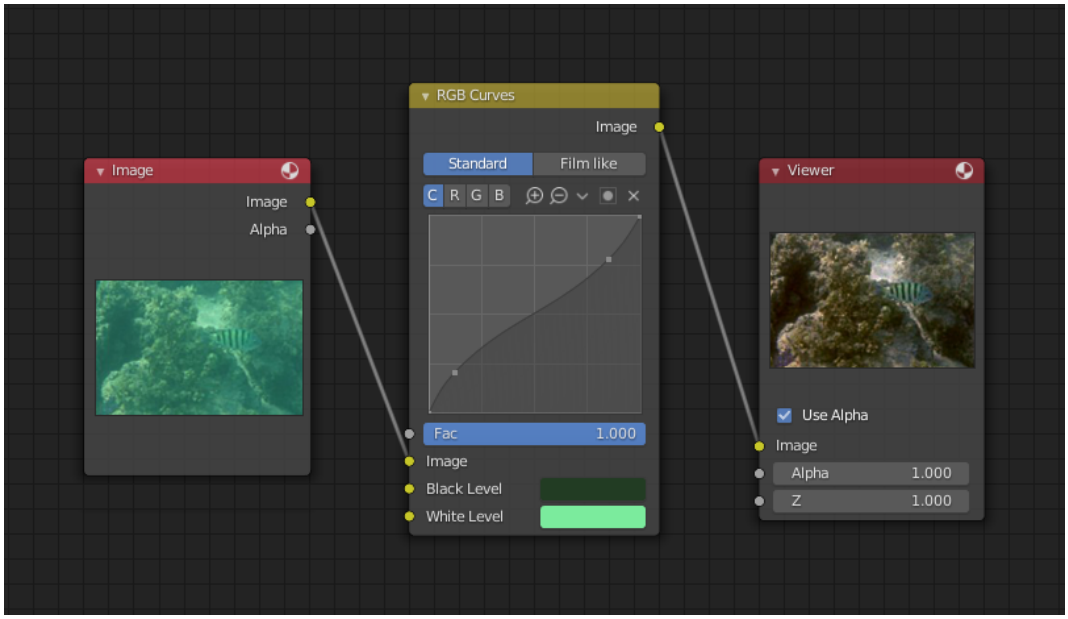


Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



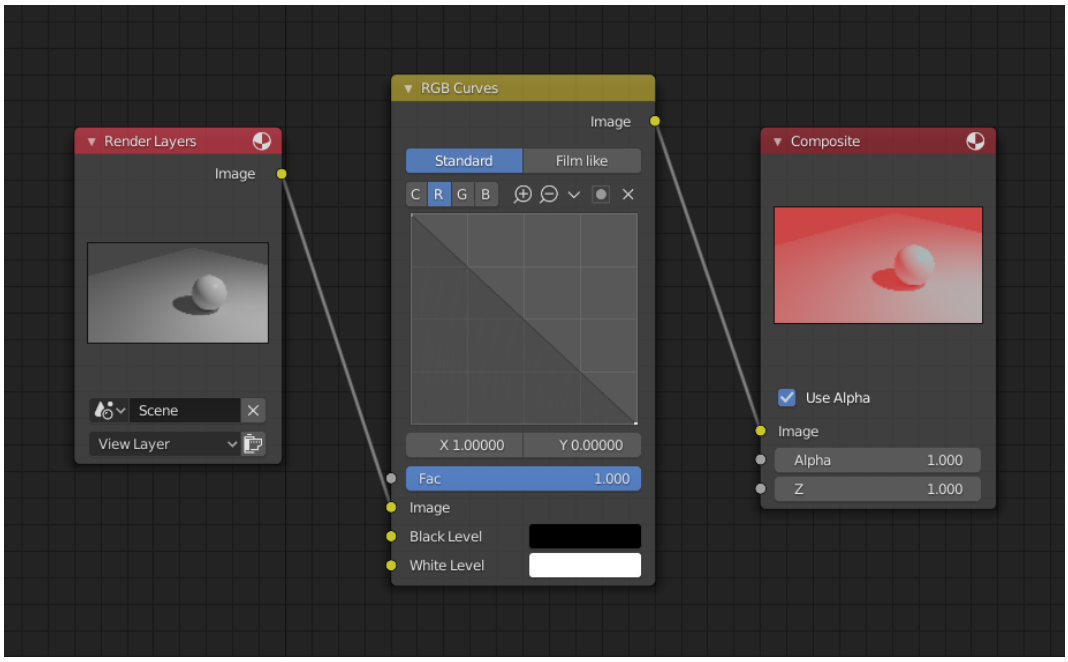
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

Effects



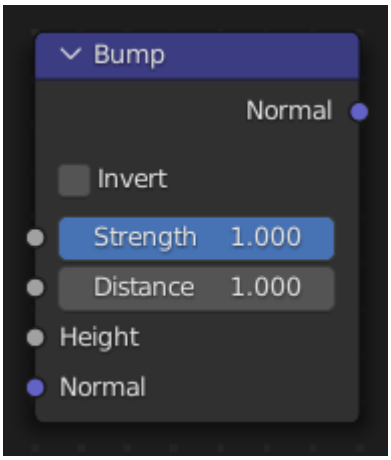
Changing colors by inverting the red channel.

Vector

- Bump Node
- Displacement Node
- Mapping Node
- Normal Node
- Normal Map Node
- Vector Curves Node
- Vector Displacement Node
- Vector Rotate Node
- Vector Transform Node

Bump Node

The *Bump* node generates a perturbed normal from a height texture, for bump mapping. The height value will be sampled at the shading point and two nearby points on the surface to determine the local direction of the normal.



Inputs

Strength
: Strength of the bump mapping effect, interpolating between no bump mapping and full bump mapping.

Distance
: Multiplier for the height value to control the overall distance for bump mapping.

Height
: Scalar value giving the height offset from the surface at the shading point; this is where you plug in textures.

Normal
: Standard normal input.

Properties

Invert
: Invert the bump mapping, to displace into the surface instead of out.

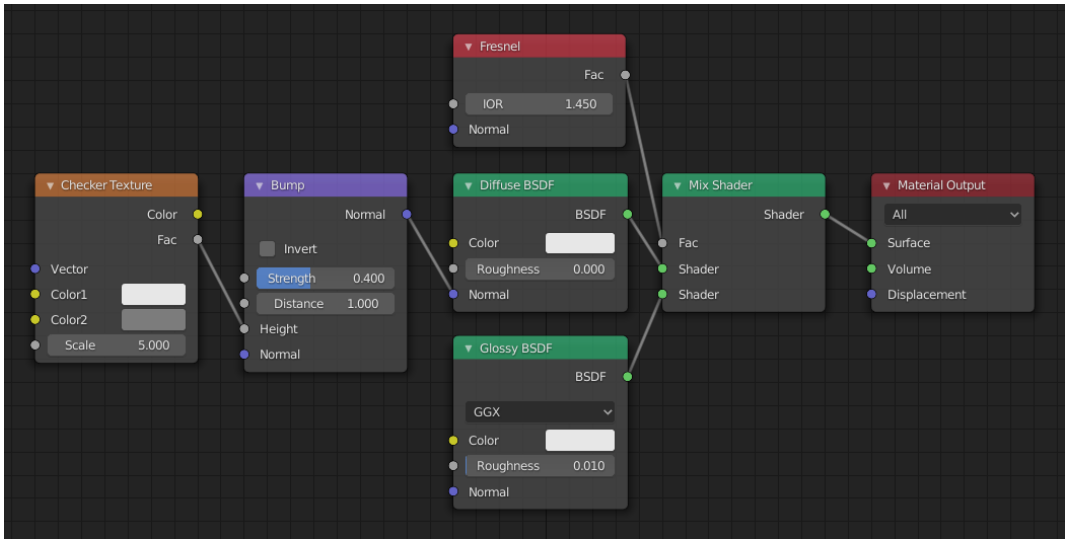
Outputs

Normal
: Standard normal output.

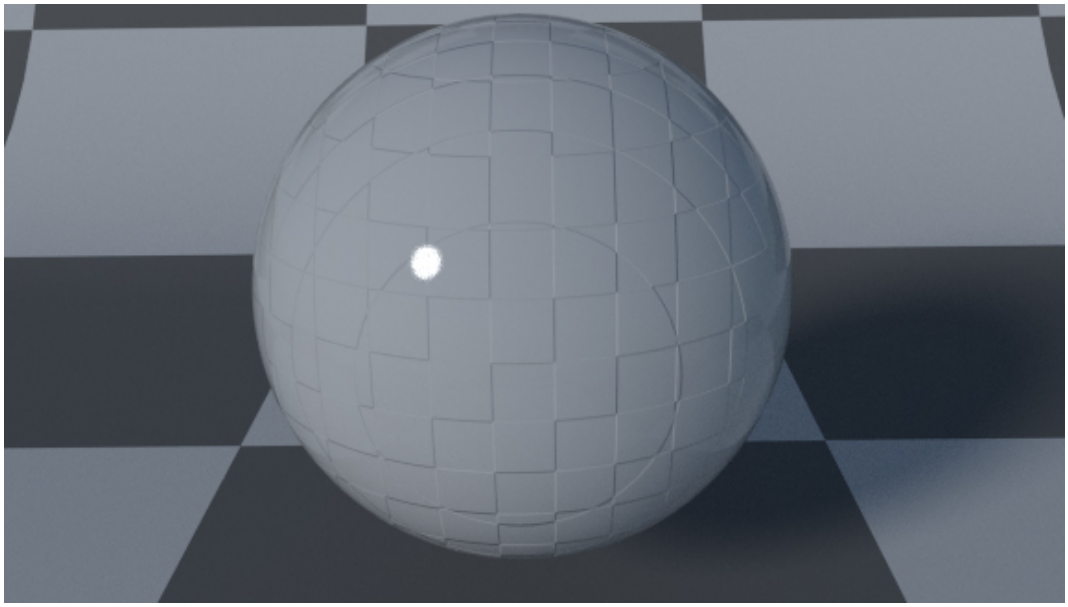
Tip

If the *Height* input is not connected, the node becomes a no-op that outputs its *Normal* input as is, or defaults to the geometry normal if not connected. Routing a node group input via a no-op Bump node before doing math effectively makes it default to normal.

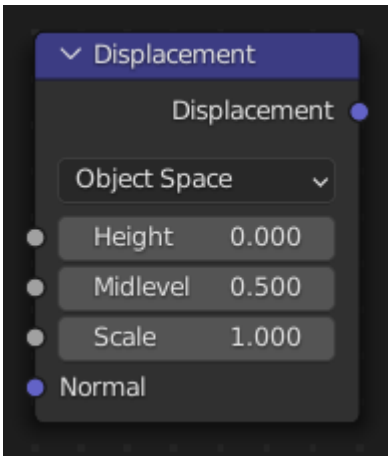
Examples



The above node setup will only bump the diffuse part of the shader, simulating a bumpy diffuse surface coated with a smooth glossy “glaze” layer.



Displacement Node



The *Displacement* node is used to displace the surface along the surface normal, to add more detail to the geometry. Both procedural textures and baked displacement maps can be used.

By default, Blender only uses Bump Mapping to render displacement. However with true displacement, the rendered geometry will be physically displaced. To use true displacement the Displacement method must be set accordingly.

Tip

For best results when using true displacement, the mesh must be subdivided finely to bring out the detail in the displacement texture.

See also

Material Displacement for more details on displacement workflows.

Inputs

Height
: Distance to displace the surface along the normal. This is where a texture node can be connected.

Midlevel
: Neutral displacement value that causes no displacement. With the default 0.5, any lower values will cause the surfaces to be pushed

inwards, and any higher values will push them outwards.

Scale
: Increase or decrease the amount of displacement.

Normal
: Standard normal input.

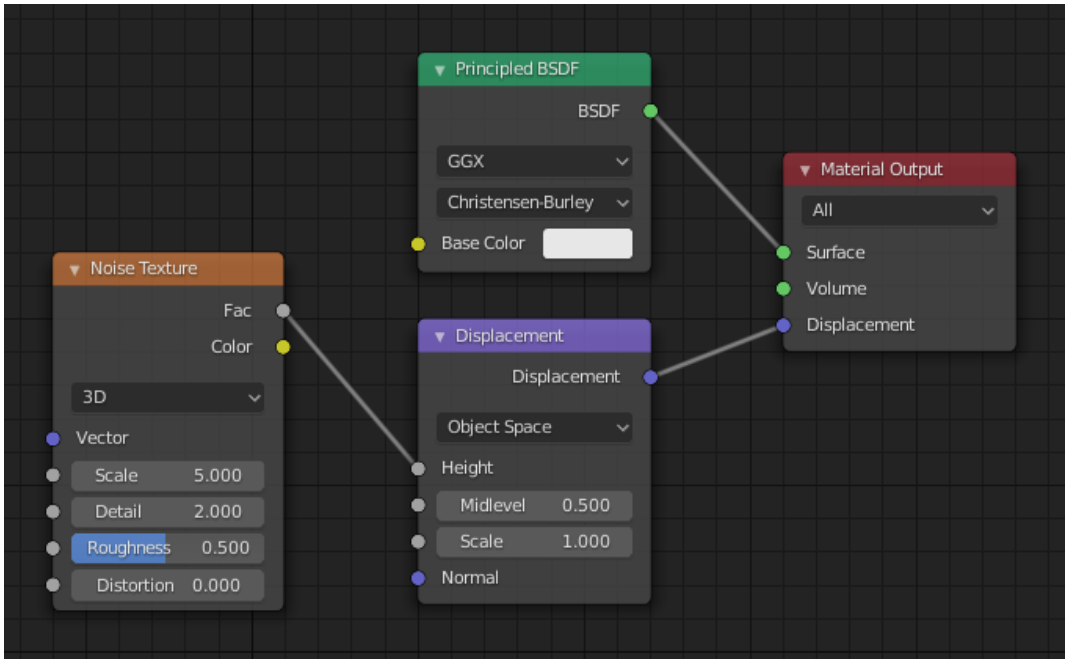
Properties

Space
: Object Space means the displacement scales along with the object.
When using World Space the object scale is ignored.

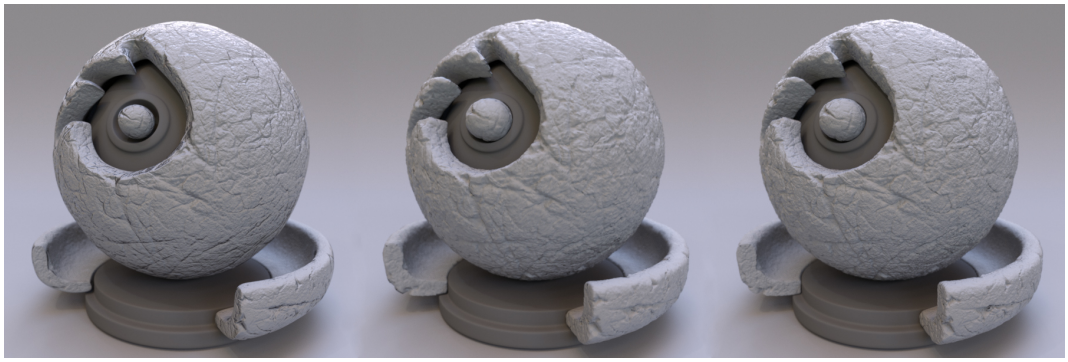
Outputs

Displacement
: Displacement offset to be connected into the Material Output.

Examples



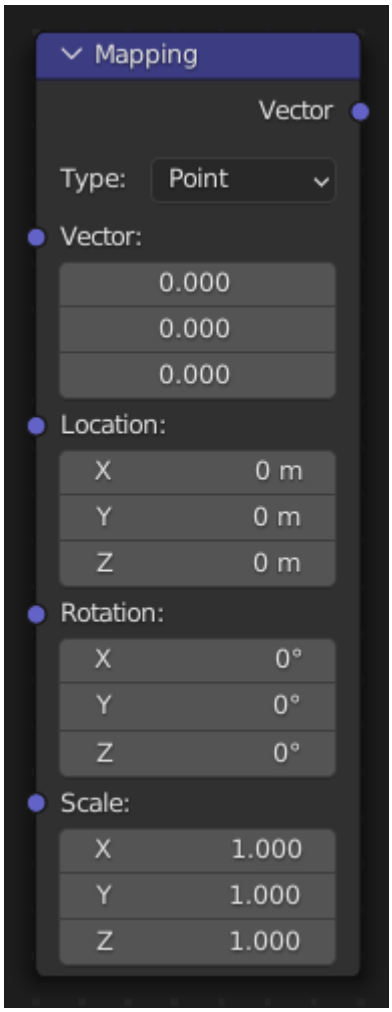
Typical displacement node setup.



Bump only, displacement only, and displacement and bump combined.

Mapping Node

The *Mapping* node transforms the input vector by applying translation, rotation, and scaling.



Inputs

The inputs of the node are dynamic. In particular, the *Location* input is only available in the *Texture* and *Point* vector types.

Vector

The vector to be transformed.

Location
: The amount of translation along each axis.

Rotation
: The amount of rotation along each axis. XYZ order.

Scale
: The amount of scaling along each axis.

Properties

Vector Type
: The node applies the transformation differently depending on the semantic type of the input vector.

Point:: For this vector type, the node performs a straightforward transformation.
Transforming a texture coordinates is analogous to transforming a UV map. For instance, translating the texture coordinates along the **positive** X axis would result in the evaluated texture to move in the **negative** X axis, much like if one translated a UV map. Similarly, scaling the texture coordinates **up** would result in the evaluated texture to scale **down**. So transforming the texture coordinates would appear to have the opposite effect on the evaluated texture.
The order of transformation is: Scale –> Rotate –> Translate, which means:

- Translation moves the input along the local rotation axis.
- Rotation rotates the input around the origin of the space.
- Scaling scales the input along the global axis.

Texture:: For this vector type, the node performs an inverse transformation.
Inverse transforming a texture coordinates would, as opposed to the *Point* type, transform the evaluated texture itself. For instance, translating the texture coordinates along the positive X axis would result in the evaluated texture to move in the positive X axis, as one would expected. Similarly, scaling the texture coordinates up would result in the evaluated texture to scale up, as one would expect.
The order of transformation is: Translate –> Rotate –> Scale, which means:

- Translation moves the input along the global axis.
- Rotation rotates the input around the translation vector.
- Scaling scales the input along the local rotation axis.

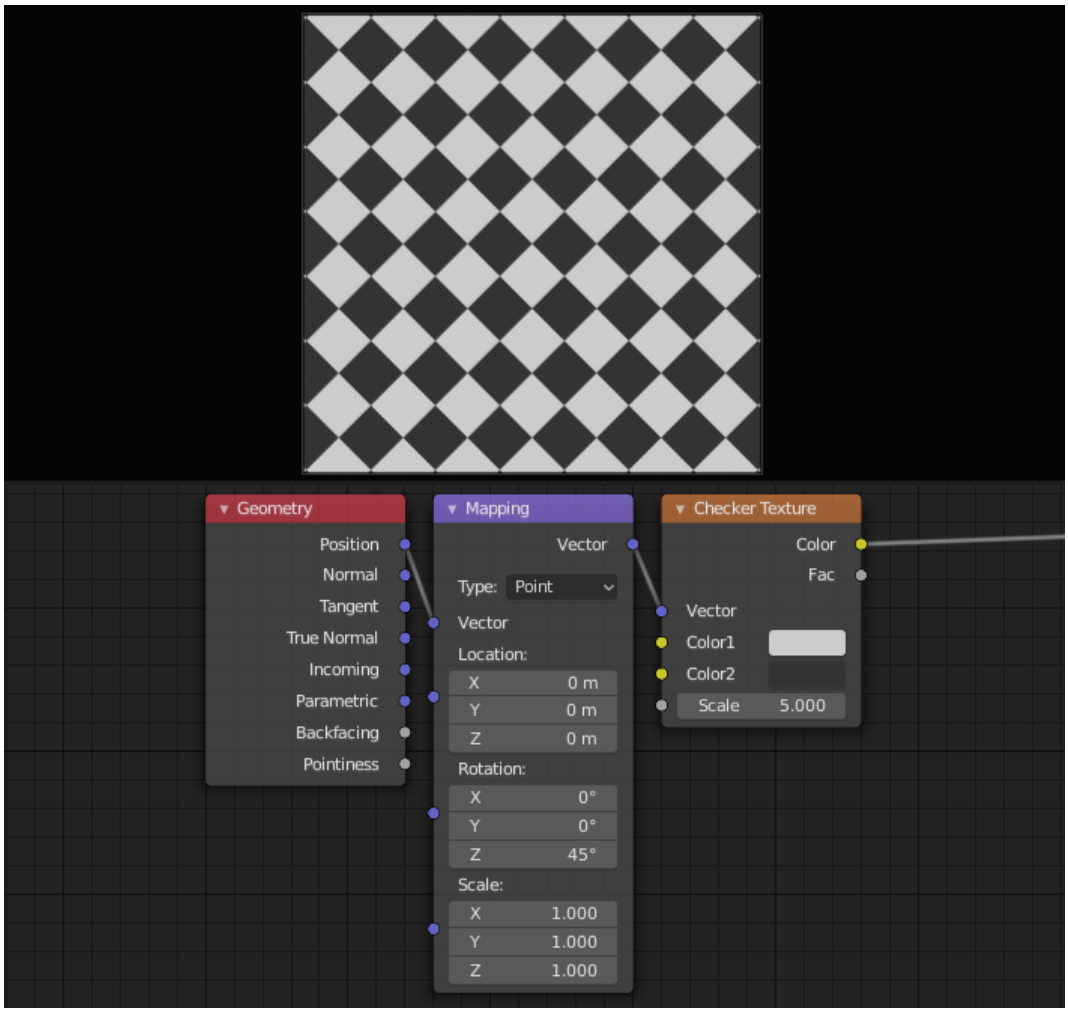
Vector:: For this vector type, a *Point* transformation is performed, but with zero translation.

Normal:: For this vector type, the node performs the inverse transpose of the transformation and normalize the result. Such transformation ensures correct normals after non-uniform scaling. So this type should be used when transforming normals.

Outputs

Vector
The input vector after transformation.

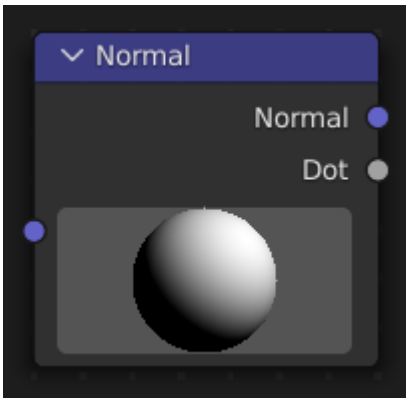
Examples



Mapping node example.

Normal Node

The Normal node generates a normal vector and a dot product.



Inputs

Normal
: Normal vector input.

Properties

Normal Direction
: To manually set a fixed normal direction vector. `LMB` click and drag on the sphere to set the direction of the normal. Holding `Ctrl` while dragging snaps to 45 degree rotation increments.

Outputs

Normal
: Normal vector output.

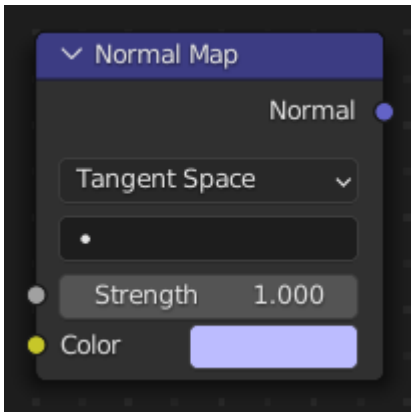
Dot
: Dot product output. The dot product is a scalar value.

  - If two normals are pointing in the same direction the dot product is 1.

- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

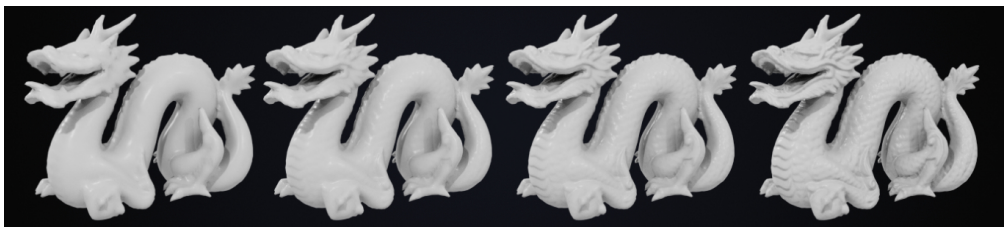
Normal Map Node

The *Normal Map* node generates a perturbed normal from an RGB normal map image. This is usually chained with an *Image Texture* node in the color input, to specify the normal map image. For tangent space normal maps, the UV coordinates for the image must match, and the image texture should be set to *Non-Color* mode to give correct results.



Inputs

Strength
: Strength of the normal mapping effect.



Strength is set to 0, 0.5, 1, 2 (from left to right).

Color
: RGB color that encodes the normal map in the specified space.

Properties

Space
: The input RGB color can be in one of three spaces: Tangent, Object and World space. Tangent space normal maps are the most common, as they support object transformation and mesh deformations. Object space

normal maps keep sticking to the surface under object transformations, while World normal maps do not.

UV Map
: Name of the UV map to derive normal mapping tangents from. When chained with an Image Texture node, this UV map should be the same as the UV map used to map the texture.

Outputs

Normal
: Normal that can be used as an input to BSDF nodes.

Example

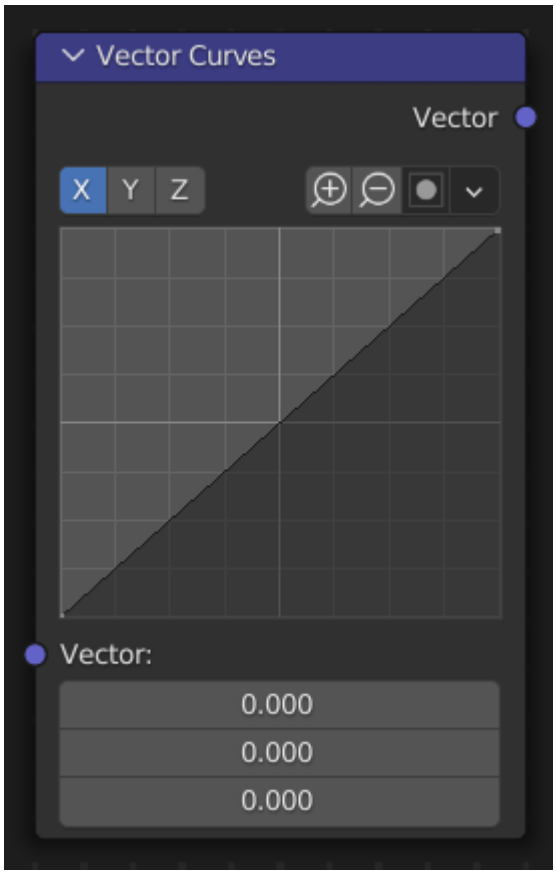


The Normal Map Strength is set to 1.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.



Inputs

In the shader context the node also has an additional Factor property.

Factor
: Controls the amount of influence the node exerts on the output vector.

Vector
: Standard vector input.

Properties

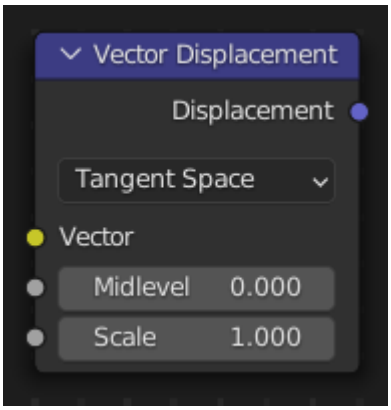
Channel
: X, Y, Z

Curve
: For the curve controls see: Curve widget.

Outputs

Vector
: Standard vector output.

Vector Displacement Node



The *Vector Displacement* node is used to displace the surface along arbitrary directions, unlike the regular Displacement node which only displaces along the surface normal.

It is typically used to apply vector displacement maps created by other sculpting software. Vector displacement maps can fully represent the high resolution detail to be applied on a smooth base mesh, unlike regular displacement maps.

By default, Blender only uses Bump Mapping to render displacement. However with true displacement, the rendered geometry will be physically displaced. To use true displacement the Displacement method must be set accordingly.

Tip

For best results when using true displacement, the mesh must be subdivided finely to bring out the detail in the displacement texture.

See also

Material Displacement for more details on displacement workflows.

Inputs

Vector
: Vector specifying the displacement along three axes. This is where a texture node can be connected.

Typically a baked vector displacement image texture is used. For Object Space, RGB colors in the image are interpreted as an XYZ offset in object space. For Tangent Space, R is an offset along the tangent, G along the normal and B along the bitangent.

Midlevel
: Neutral displacement value that causes no displacement. With the default 0.0, any lower values will cause the surfaces to be pushed inwards, and any higher values will push them outwards.

Scale
: Increase or decrease the amount of displacement.

Properties

Space
: Object Space maps work for static meshes, and will render slightly faster with less memory usage. Tangent Space maps can be used for meshes that will be deformed, like animated characters, so the displacement follows the deformation.

Outputs

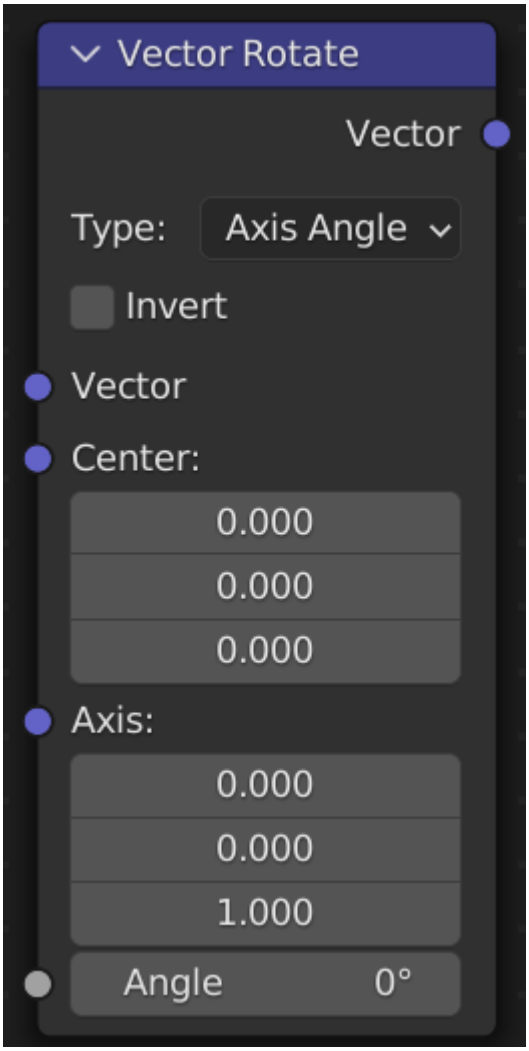
Displacement
: Displacement offset to be connected into the Material Output.

Examples



Regular and exaggerated vector displacement on a smooth base mesh.

Vector Rotate Node



The *Vector Rotate Node* provides the ability to rotate a vector around a pivot point (*Center*).

Inputs

Vector
: Vector to be rotated.

Center

Point to rotate around.

Axis
: Axis to rotate around.

Angle
: Angle to rotate the input vector by.

Rotation
: When *Type* is set to *Euler*, rotate the input vector by these angles around the X, Y, then Z axes in that order.

Properties

Type
: The type of angle input.

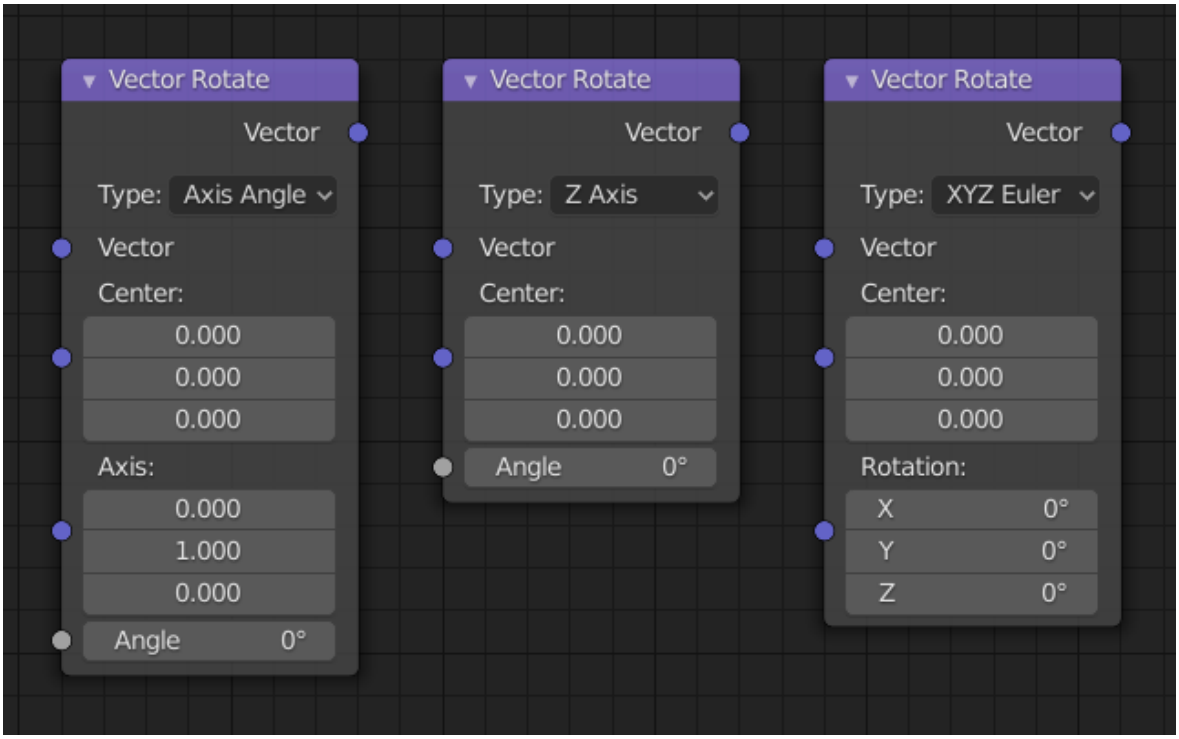
X/Y/Z Axis::	Rotates the vector around the defined axis and the amount of rotation is defined by the *Angle* input.
Axis Angle::	Rotates the vector around an arbitrary axis defined by the *Axis* input vector. The amount of rotation is defined by the *Angle* input.
Euler::	Rotates the vector about a center point defined by the *Center* input vector. The amount of rotation on each axis is defined by the *Rotation* input vector.

Invert
: Inverts the rotation angle.

Outputs

Vector
: The rotated vector.

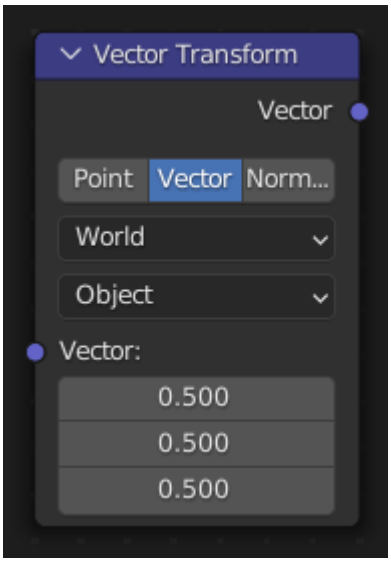
Examples



Vector Rotate node example.

Vector Transform Node

The *Vector Transform* node allows converting a vector, point, or normal between world and camera and object coordinate space.



Inputs

Vector Input
: Standard vector input.

Properties

Type
: Specifies the input/output type.

  Vector, Point, Normal.

Convert From
: Coordinate Space to convert from:

  World, Object, Camera.

Convert To

Coordinate Space to convert to:

World, Object, Camera.

Outputs

Vector Output
: The transformed output vector.

Examples

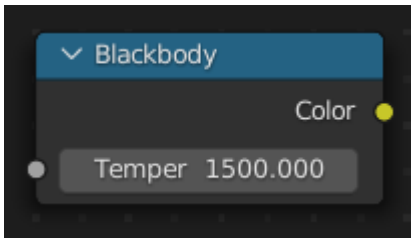
Todo <2.8 add.

Converter

- Blackbody Node
- Clamp Node
- Color Ramp Node
- Combine Color Node
- Combine XYZ Node
- Float Curve
- Map Range Node
- Math Node
- Mix Node
- RGB to BW Node
- Separate Color Node
- Separate XYZ Node
- Shader To RGB
- Vector Math Node
- Wavelength Node

Blackbody Node

The *Blackbody* node converts a blackbody temperature to RGB value. This can be useful for materials that emit light at natural occurring frequencies.



Inputs

Temperature
: The temperature in Kelvin.

Properties

This node has no properties.

Outputs

Color
: RGB color output.

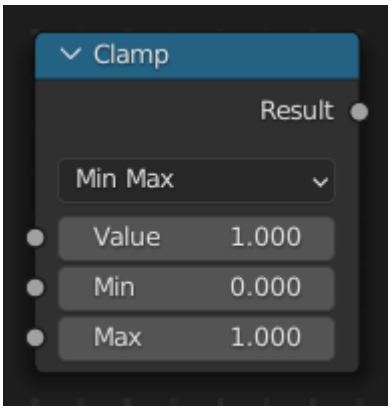
Examples



Example of the color ranges of the Blackbody node.

Clamp Node

The *Clamp* node clamps a value between a minimum and a maximum.



Inputs

Value
: The input value to be clamped.

Min
: The minimum value.

Max
: The maximum value.

Properties

Clamp Type
: Method to clamp.

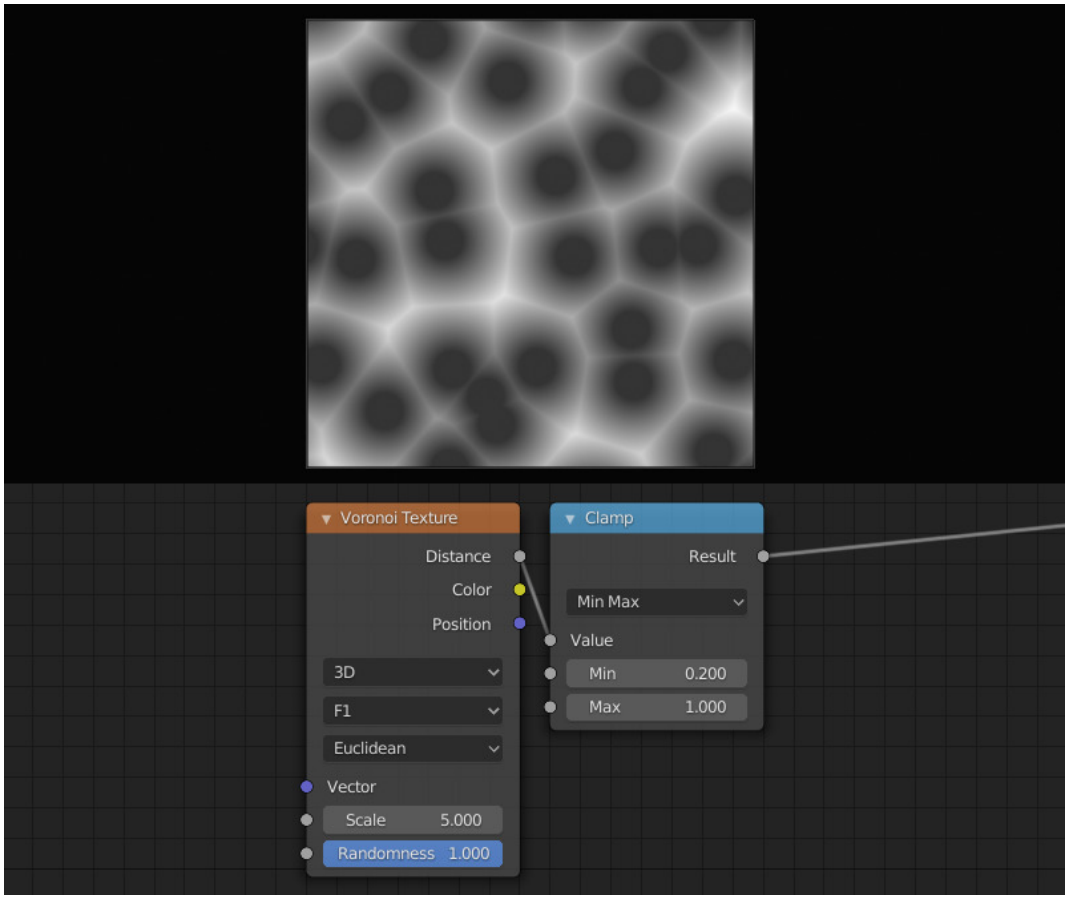
Min Max::	Constrain values between Min and Max.
Range::	Constrain values between Min and Max. When Min is greater than Max, constrain between Max and Min instead.

Outputs

Result
: The input value after clamping.

Examples

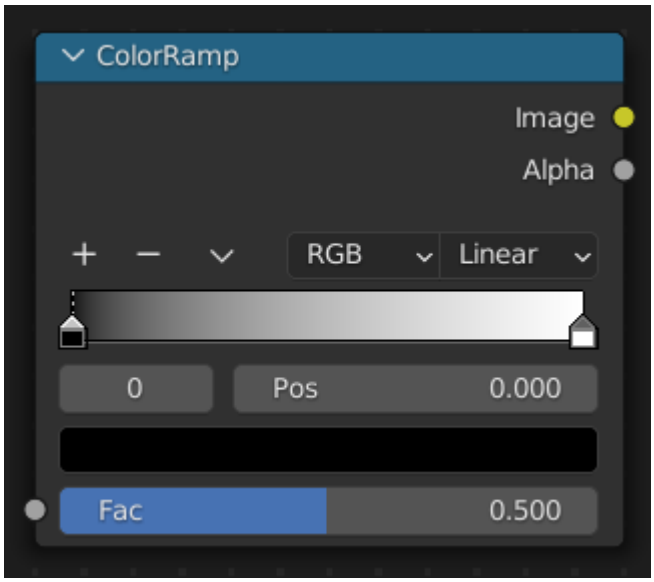
The *Voronoi Texture* node outputs a value whose minimum is zero. We can use the *Clamp* node to clamp this value such that the minimum is 0.2.



Example of Clamp node.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Inputs

Factor
: The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp
: See Color Ramp Widget.

Outputs

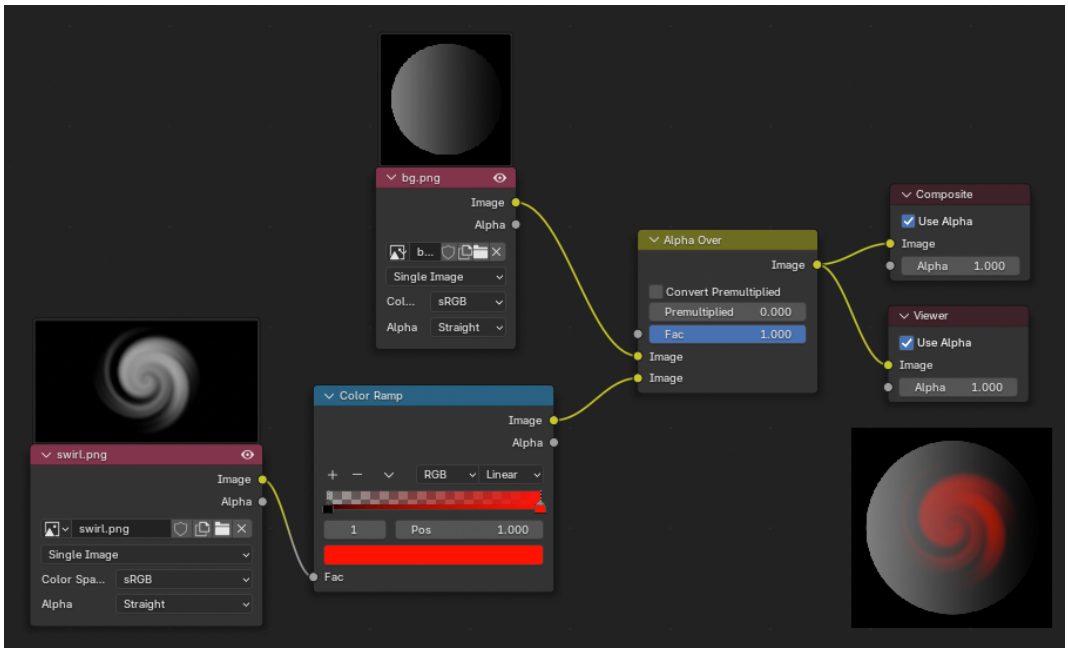
Image/Color
: Standard color output.

Alpha
: Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

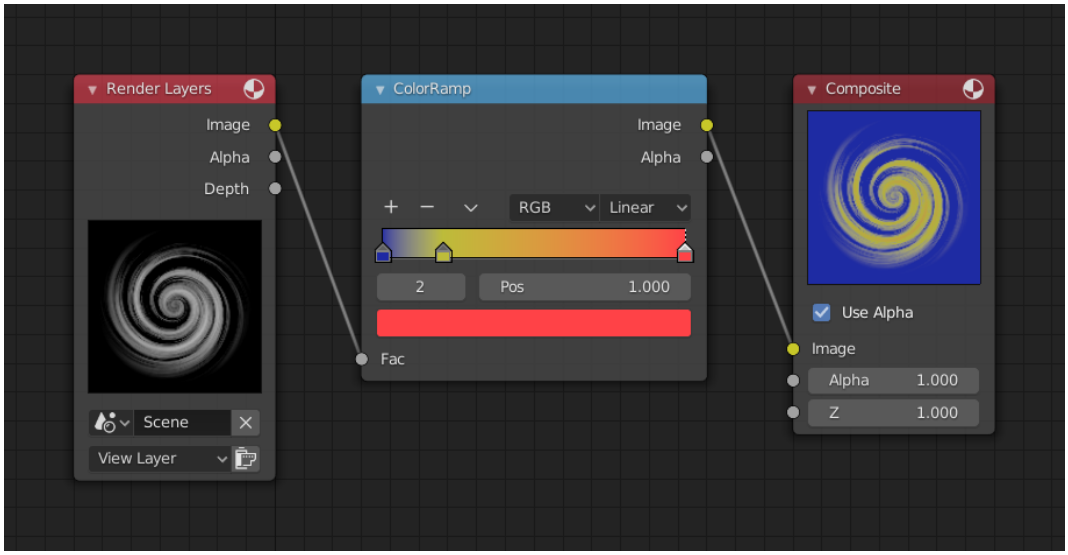


In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

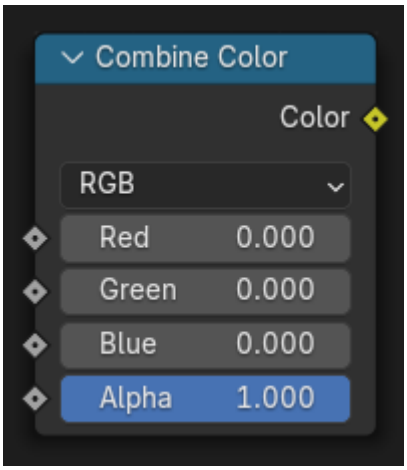
In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.



The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Combine Color Node

Combines four grayscale channels into one color image, based on a particular Color Model.



Inputs

The inputs of this node depend on the Mode property (see below).

Alpha
: The opacity of the output color.

Properties

Mode
: The color model to use.

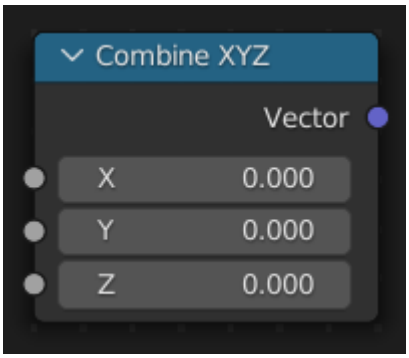
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Output

Color
: Standard color output.

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.



Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

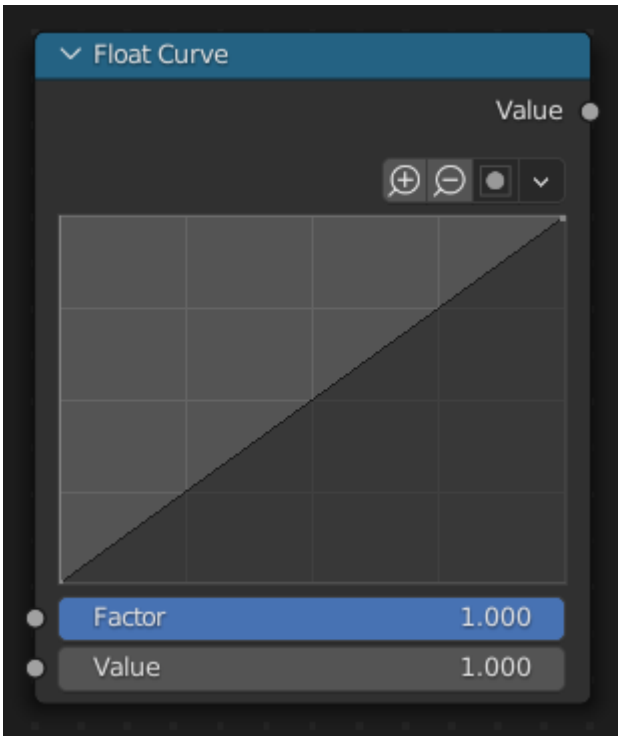
Vector
: Standard vector output.

Note

The vector is not normalized.

Float Curve

The Float Curve node maps an input float to a curve and outputs a float value.



Inputs

Factor
: Controls the amount of influence the node exerts on the output value.

Value
: Standard float input.

Properties

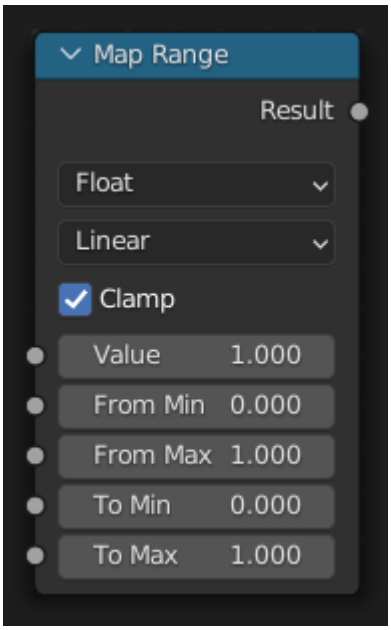
Curve
: For the curve controls see: Curve widget.

Outputs

Float
: Standard float output.

Map Range Node

The *Map Range* node remaps a value from a range to a target range.



Inputs

Value/Vector
: The input value or vector to be remapped.

From Min
: The lower bound of the range to remap from.

From Max
: The higher bound of the range to remap from.

To Min
: The lower bound of the target range.

To Max
: The higher bound of the target range.

Steps

The number of values allowed between *To Min* and *To Max* when using *Stepped Linear* interpolation. A higher value will give a smoother interpolation while lower values will progressively quantize the input.

Properties

Data Type
: Map Range supports both Float and Vector data types. Changing the data type will also update the sockets to reflect the data type chosen.

Interpolation Type
: The mathematical method used to transition between gaps in the numerical inputs.

Linear::	Linear interpolation between From Min and From Max values.
Stepped Linear::	Stepped linear interpolation between From Min and From Max values.
Smooth Step::	Smooth Hermite edge interpolation between From Min and From Max values.
Smoother Step::	Smoother Hermite edge interpolation between From Min and From Max values.

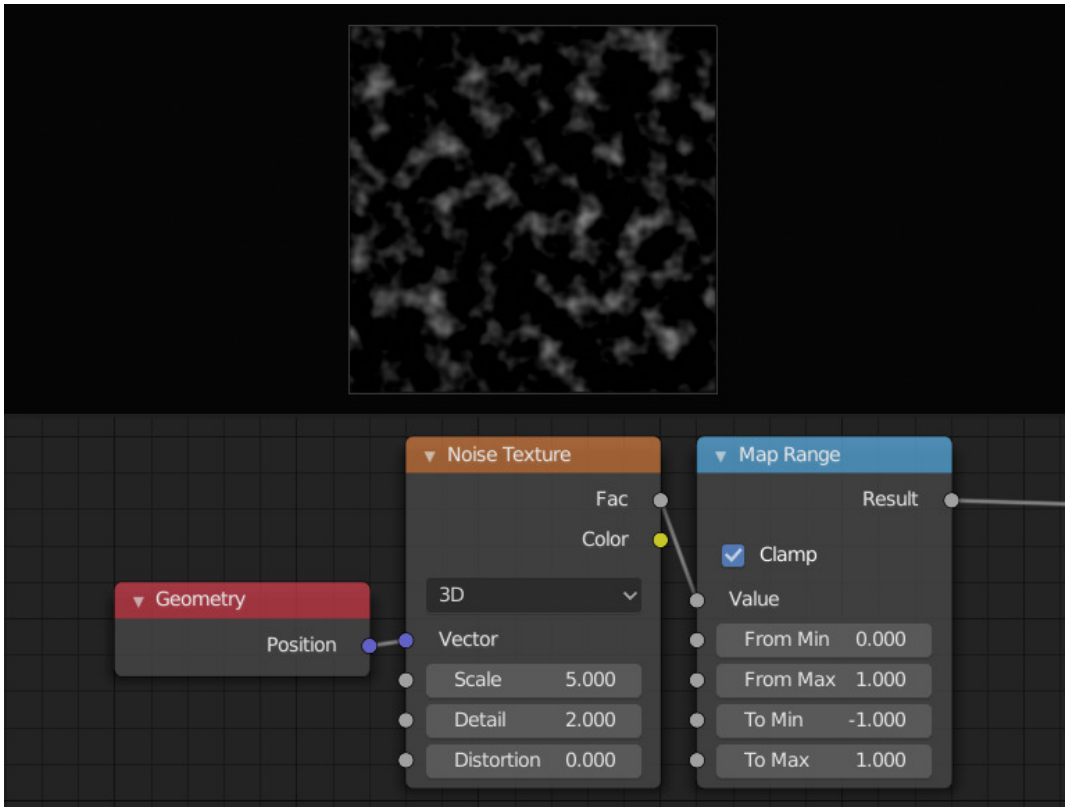
Clamp
: If enabled, the output is clamped to the target range.

Outputs

Result/Vector
: The input value after remapping.

Examples

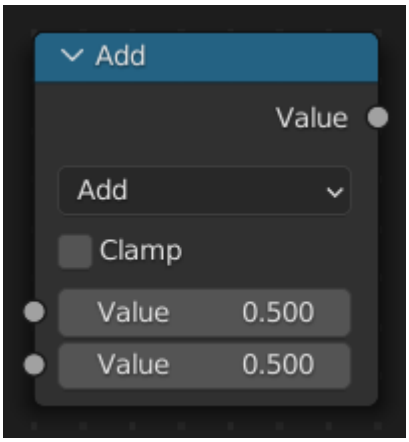
The *Noise Texture* node outputs a value in the range [0, 1]. We can use the *Map Range* node to remap this value into the range [-1, 1].



Example of Map Range node.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value
: Input Value. Trigonometric functions read this value as radians.

Addend
: Input Addend.

Base
: Input Base.

Exponent
: Input Exponent.

Epsilon
: Input Epsilon.

Distance

Input Distance.

Min
Input Minimum.

Max
Input Maximum.

Increment
Input Increment.

Scale
Input Scale.

Degrees
Input Degrees.

Radians
Input Radians.

Properties

Operation
The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

One divided by the square root of the value.

Absolute:: The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises Euler's number to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: Smooth Minimum.

Smooth Maximum:: Smooth Maximum.

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo:: Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.

Snap:: Rounds the input value down to the nearest integer multiple of *Increment*.

Ping-pong:: Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine:: The Sine of the input value.

Cosine:: The Cosine of the input value.

Tangent:: The Tangent of the input value.

Arcsine:: The Arcsine of the input value.

Arccosine:: The Arccosine of the input value.

Arctangent:: The Arctangent of the input value.

Arctan2:: Outputs the Inverse Tangent of the first value divided by the second value measured in radians.

Hyperbolic Sine:: The Hyperbolic Sine of the input value.

Hyperbolic Cosine:: The Hyperbolic Cosine of the input value.

Hyperbolic Tangent:: The Hyperbolic Tangent of the input value.

Conversion

To Radians:: Converts the input from degrees to radians.

To Degrees:: Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

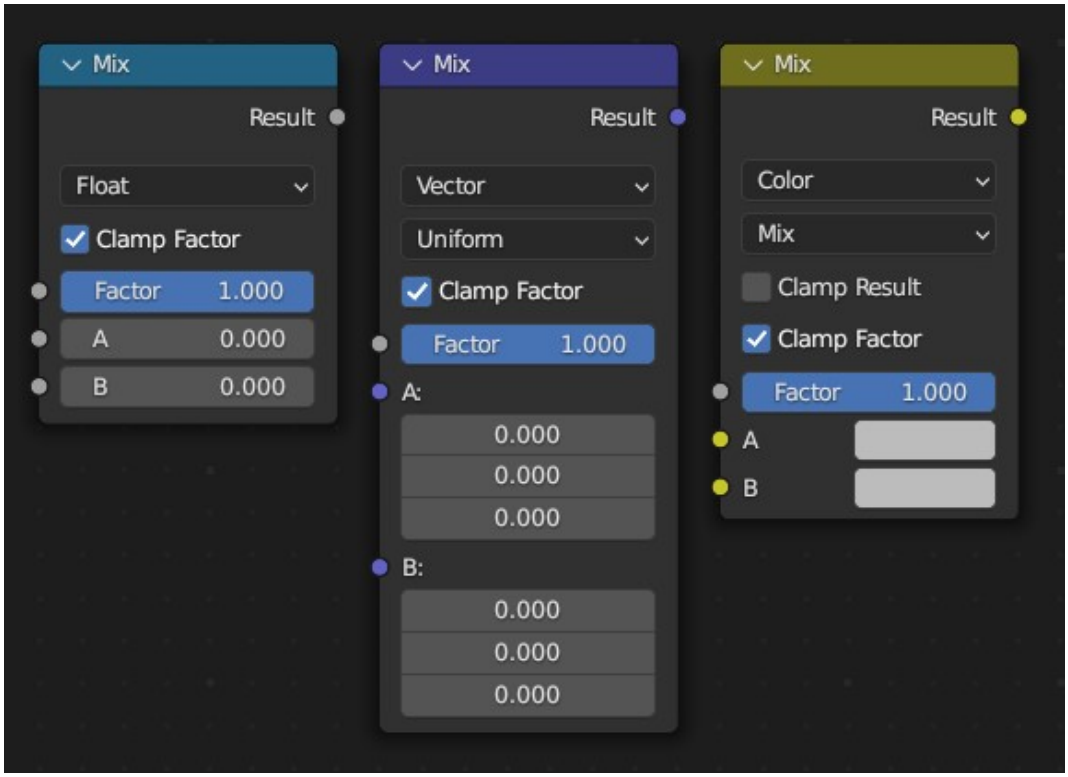
Outputs

Value

Numerical value output.

Mix Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.



Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type
: The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)
: The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor
: Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)
: Limit the Result to the range between 0.0 and 1.0.

Outputs

Result
: Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB to BW Node



The *RGB to BW Node* makes a color image black-and-white by outputting its luminance.

Note

You can directly connect Color sockets to Value sockets in node graphs, which also converts the image to black-and-white. As such, this node is not always necessary.

Inputs

Image
: Color image input.

Properties

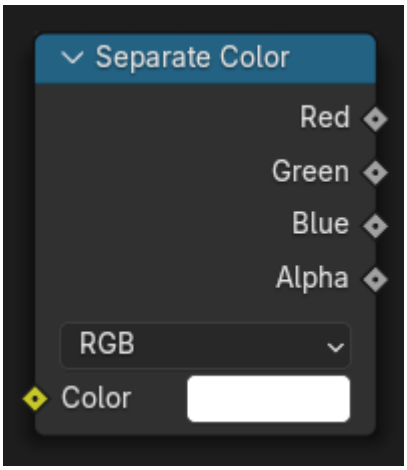
This node has no properties.

Outputs

Value
: Grayscale value output.

Separate Color Node

Splits an image into its channels, based on a particular Color Model.



Inputs

Color
: Standard color input.

Properties

Mode
: The color model to output.

RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

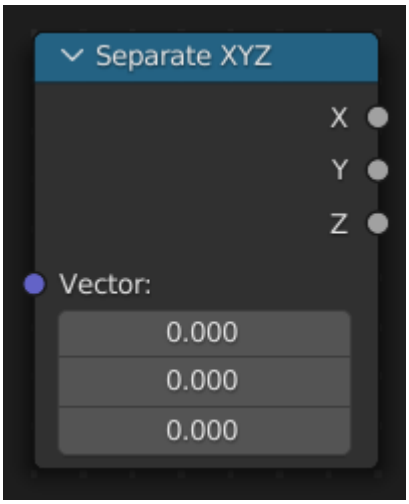
Outputs

The outputs of this node depend on the Mode property (see above).

Alpha
: The opacity value.

Separate XYZ Node

The *Separate XYZ Node* splits a vector into its individual components.



Input

Vector
: Standard vector input.

Properties

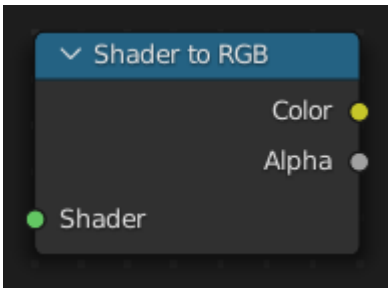
This node has no properties.

Outputs

- X
- Y
- Z

Shader To RGB

EEVEE Only



The Shader to RGB node is typically used for non-photorealistic rendering, to apply additional effects on the output of BSDFs. For example, a color ramp on the output of a diffuse BSDF can be used to create a flexible toon shader.

Using this conversion breaks the PBR pipeline and thus makes the result unpredictable when used in combination with effects such as ambient occlusion, contact shadows, soft shadows and screen space refraction.

Some effects require multiple samples to converge, and applying arbitrary changes to noisy input may not convert to a smooth result.

Warning

If a Shader to RGB node is used, any upstream BSDF will be invisible to the following effects:

- Screen Space Reflection
- Subsurface Scattering
- Alpha Clip and Alpha Hashed blend modes

Shader to RGB node doesn't give expected results in render passes.

Inputs

Shader
: Any shader such as a BSDF or Emission node can be linked here.

Properties

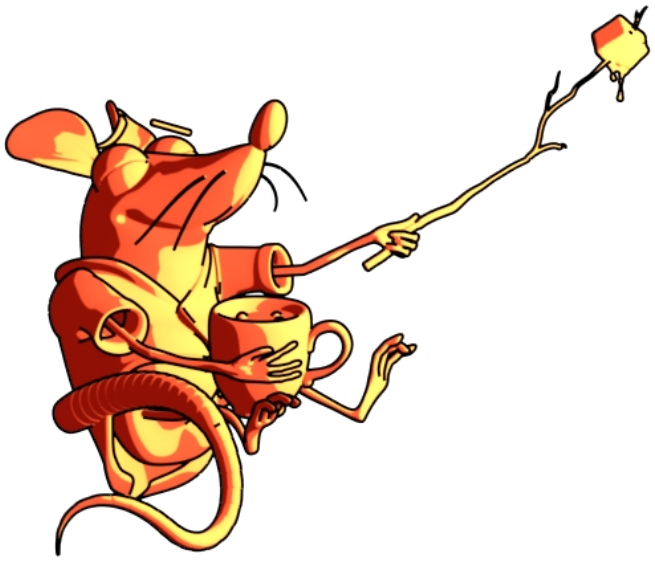
This node has no properties.

Outputs

Color
: Surface color computed from BSDFs and lighting.

Alpha
: Alpha transparency from any Transparent BSDFs in the input.

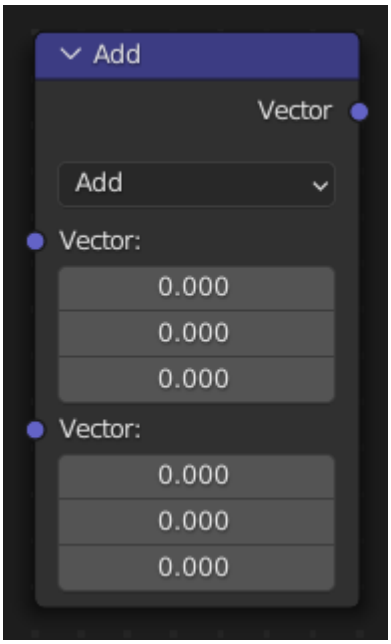
Examples



Simple toon shading with Shader to RGB and Freestyle.

Vector Math Node

The *Vector Math* node performs the selected math operation on the input vectors.



Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Scale* input is only available in the *Scale* operator.

Vector
: Input vector $A = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$.

Vector
: Input vector $B = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$.

Scale
: Input Scale s.

Properties

Operation

The vector math operator to be applied on the input vectors.

Add:: The sum of A and B. $\begin{pmatrix} A_x + B_x \\ A_y + B_y \\ A_z + B_z \end{pmatrix}$

Subtract:: The difference between A and B. $\begin{pmatrix} A_x - B_x \\ A_y - B_y \\ A_z - B_z \end{pmatrix}$

Multiply:: The entrywise product of A and B. $\begin{pmatrix} A_x \cdot B_x \\ A_y \cdot B_y \\ A_z \cdot B_z \end{pmatrix}$

Divide:: The entrywise division of A by B. Division by zero results in zero. $\begin{pmatrix} A_x / B_x \\ A_y / B_y \\ A_z / B_z \end{pmatrix}$

Multiply Add:: The entrywise combination of the multiply and addition operations. $A \times B + C$

Cross Product:: The cross product of A and B. $\begin{pmatrix} A_y \cdot B_z - A_z \cdot B_y \\ A_z \cdot B_x - A_x \cdot B_z \\ A_x \cdot B_y - A_y \cdot B_x \end{pmatrix}$

Project:: The projection of A onto B.

Reflect:: The reflection of A around the normal B. B need not be normalized.

Refract:: For a given incident vector A, surface normal B and ratio of indices of refraction (IOR), refract outputs the refraction vector R.

Faceforward:: Orients a vector A to point away from a surface B as defined by its normal C. Computes $((dot(B, C) < 0) ? A : -A)$.

Dot Product:: The dot product of A and B. $A_x \cdot B_x + A_y \cdot B_y + A_z \cdot B_z$

Distance:: The distance between A and B.

Length:: The length of A. $\sqrt{A_x^2 + A_y^2 + A_z^2}$

Scale:: The result of multiplying A by the scalar input *Scale*. $\begin{pmatrix} s \cdot A_x \\ s \cdot A_y \\ s \cdot A_z \end{pmatrix}$

Normalize::	The result of normalizing A. The result vector points to the same direction as A and has a length of 1. If A is (0, 0, 0), the result is (0, 0, 0) as well.
Wrap::	The entrywise output of a value between Min and Max based on the absolute difference between the input value and the nearest integer multiple of Max less than the value.
Snap::	The result of rounding A to the largest integer multiple of B less than or equal A.
Floor::	Rounds the input value entrywise down to the nearest integer.
Ceil::	Rounds the input value entrywise up to the nearest integer.
Modulo::	The entrywise modulo of A by B.
Fraction::	Returns the fractional part of the *value* entrywise.
Absolute::	The entrywise absolute value of A.
Minimum::	The entrywise minimum value from A and B.
Maximum::	The entrywise maximum value from A and B.
Sine::	The entrywise Sine of A.
Cosine::	The entrywise Cosine of A.
Tangent::	The entrywise Tangent of A.

Outputs

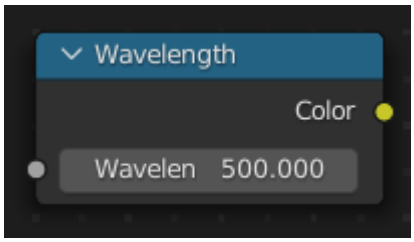
The output of the node is dynamic. It is either a vector or a scalar depending on the operator. For instance, the *Length* operator has a scalar output while the *Add* operator has a vector output.

Vector
: Output vector.

Value
: Output value.

Wavelength Node

The *Wavelength* node converts a wavelength value to an RGB value. This can be used to achieve a specific color on the light spectrum.



Inputs

Wavelength
: The color wavelength from 380 to 780 nanometers.

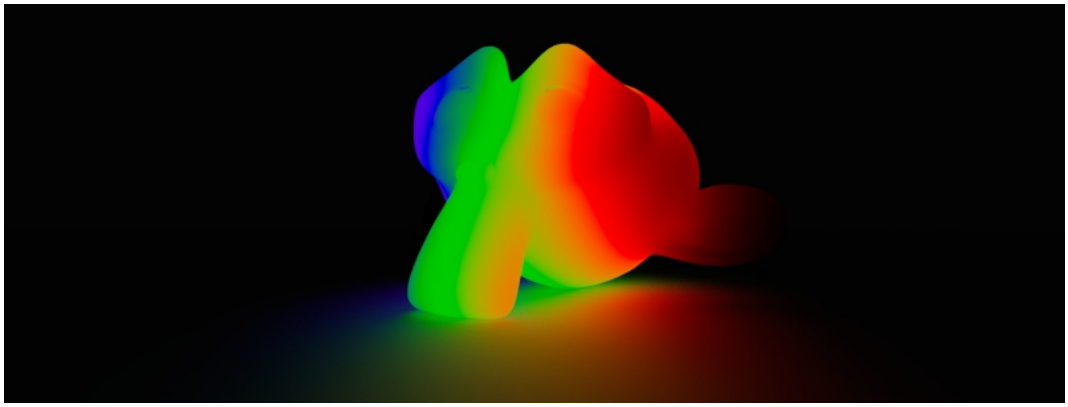
Properties

This node has no properties.

Outputs

Color
: RGB color output.

Examples



Example of Wavelength node.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Open Shading Language

Cycles Only

It is also possible to create your own nodes using Open Shading Language (OSL). These nodes will only work with the CPU and OptiX rendering backend.

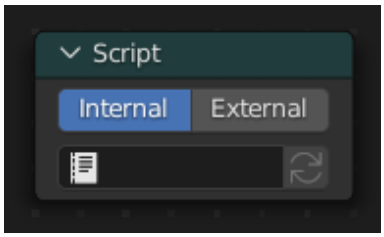
To enable it, select *Open Shading Language* as the shading system in the render settings.

Note

Some OSL features are not available when using the OptiX backend. Examples include:

- Memory usage reductions offered by features like on-demand texture loading and mip-mapping are not available.
- Texture lookups require OSL to be able to determine a constant image file path for each texture call.
- Some noise functions are not available. Examples include *Cell*, *Simplex*, and *Gabor*.
- The trace function is not functional. As a result of this, the Ambient Occlusion and Bevel nodes do not work.

Script Node



OSL was designed for node-based shading, and *each* OSL shader corresponds to *one* node in a node setup. To add an OSL shader, add a script node and link it to a text data-block or an external file. Input and output sockets will be created from the shader parameters on clicking the update button in the Node or the Text editor.

OSL shaders can be linked to the node in a few different ways. With the *Internal* mode, a text data-block is used to store the OSL shader, and the OSO bytecode is stored in the node itself. This is useful for distributing a blend-file with everything packed into it.

The *External* mode can be used to specify a `.osl` file from a drive, and this will then be automatically compiled into a `.oso` file in the same directory. It is also possible to specify a path to a `.oso` file, which will then be used directly, with compilation done manually by the user. The third option is to specify just the module name, which will be looked up in the shader search path.

The shader search path is located in the same place as the scripts or configuration path, under:

Linux

```
$HOME/.config/blender/4.3/shaders/
```

Windows

```
C:\Users\$user\AppData\Roaming\Blender
Foundation\Blender\4.3\shaders\
```

macOS

```
/Users/$USER/Library/Application Support/Blender/4.3/shaders/
```

Tip

For use in production, we suggest to use a node group to wrap shader script nodes, and link that into other blend-files. This makes it easier to make changes to the node afterwards as sockets are added or removed, without having to update the script nodes in all files.

Writing Shaders

For more details on how to write shaders, see the OSL Documentation.

Here is a simple example:

```
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
    color material_color = Diffuse_Color * mix(1.0, noise(P *
10.0), Noise_Factor);
    BSDF = material_color * diffuse(N);
}
```

Closures

OSL is different from, for example, RSL or GLSL, in that it does not have a light loop. There is no access to lights in the scene, and the material must be built from closures that are implemented in the renderer itself. This is more limited, but also makes it possible for the renderer to do optimizations and ensure all shaders can be importance sampled.

The available closures in Cycles correspond to the shader nodes and their sockets; for more details on what they do and the meaning of the parameters, see the shader nodes manual.

See also

Documentation on OSL's built-in closures.

BSDF

- `diffuse(N)`
- `oren_nayar(N, roughness)`
- `diffuse_ramp(N, colors[8])`
- `phong_ramp(N, exponent, colors[8])`
- `diffuse_toon(N, size, smooth)`
- `glossy_toon(N, size, smooth)`
- `translucent(N)`
- `reflection(N)`
- `refraction(N, ior)`

- `transparent()`
- `microfacet_ggx(N, roughness)`
- `microfacet_ggx_aniso(N, T, ax, ay)`
- `microfacet_ggx_refraction(N, roughness, ior)`
- `microfacet_beckmann(N, roughness)`
- `microfacet_beckmann_aniso(N, T, ax, ay)`
- `microfacet_beckmann_refraction(N, roughness, ior)`
- `ashikhmin_shirley(N, T, ax, ay)`
- `ashikhmin_velvet(N, roughness)`

Hair

- `hair_reflection(N, roughnessu, roughnessv, T, offset)`
- `hair_transmission(N, roughnessu, roughnessv, T, offset)`
- `principled_hair(N, absorption, roughness, radial_roughness, coat, offset, IOR)`

BSSRDF

Used to simulate subsurface scattering.

bssrdf(*method*, *N*, *radius*, *albedo*)

Parameters::

- **method** (*string*) –
  Rendering method to simulate subsurface scattering.
  - `burley`: An approximation to physically-based volume scattering. This method is less accurate than `random_walk` however, in some situations this method will resolve noise faster.
  - `random_walk_skin`: Provides accurate results fc thin and curved objects. Random Walk uses true volumetric scattering inside the mesh, which means that it works best for closed meshes. Overlapping faces and holes in the mesh can cause problems.
  - `random_walk`: Behaves similarly to `random_walk_skin` but modulates the *Radius* based on the *Color*, *Anisotropy*, and *IOR*. This method thereby attempts to retain greater surfac detail and color than `random_walk_skin`.
- **N** (*vector*) – Normal vector of the surface point being shaded.
- **radius** (*vector*) – Average distance that light scatters below the surface. Higher radius gives a softer appearance, as light bleeds into shadows an through the object. The scattering distance is specified separately for the RGB channels, to render materials such as skin where red light scatters deeper. The X, Y and Z values are mapped to the R, G and B values, respectively.
- **albedo** (*color*) – Color of the surface, or physicall speaking, the probability that light is reflected for each wavelength.

Volume

- `henyey_greenstein(g)`
- `absorption()`

Other

- `emission()`
- `ambient_occlusion()`
- `holdout()`
- `background()`

Attributes

Geometry attributes can be read through the `getattribute()` function. This includes UV maps, color attributes and any attributes output from geometry nodes.

The following built-in attributes are available through `getattribute()` as well.

`geom:generated`
: Automatically generated texture coordinates, from undeformed mesh.

`geom:uv`
: Default render UV map.

`geom:tangent`
: Default tangent vector along surface, in object space.

`geom:undisplaced`
: Position before displacement, in object space.

`geom:dupli_generated`
: For instances, generated coordinate from instancer object.

`geom:dupli_uv`
: For instances, UV coordinate from instancer object.

`geom:trianglevertices`
: Three vertex coordinates of the triangle.

`geom:numpolyvertices`
: Number of vertices in the polygon (always returns three currently).

`geom:polyvertices`

Vertex coordinates array of the polygon (always three vertices currently).

`geom:name`
Name of the object.

`geom:is_smooth`
Is mesh face smooth or flat shaded.

`geom:is_curve`
Is object a curve or not.

`geom:curve_intercept`
0..1 coordinate for point along the curve, from root to tip.

`geom:curve_thickness`
Thickness of the curve in object space.

`geom:curve_length`
Length of the curve in object space.

`geom:curve_tangent_normal`
Tangent Normal of the strand.

`geom:is_point`
Is point in a point cloud or not.

`geom:point_radius`
Radius of point in point cloud.

`geom:point_position`
Center position of point in point cloud.

`geom:point_random`
Random number, different for every point in point cloud.

`path:ray_length`
Ray distance since last hit.

`object:random`
Random number, different for every object instance.

`object:index`
: Object unique instance index.

`object:location`
: Object location.

`material:index`
: Material unique index number.

`particle:index`
: Particle unique instance number.

`particle:age`
: Particle age in frames.

`particle:lifetime`
: Total lifespan of particle in frames.

`particle:location`
: Location of the particle.

`particle:size`
: Size of the particle.

`particle:velocity`
: Velocity of the particle.

`particle:angular_velocity`
: Angular velocity of the particle.

Trace

CPU Only

We support the `trace(point pos, vector dir, ...)` function, to trace rays from the OSL shader. The "shade" parameter is not supported currently, but attributes can be retrieved from the object that was hit using the `getmessage("trace", ..)` function. See the OSL specification for details on how to use this.

This function cannot be used instead of lighting; the main purpose is to allow shaders to “probe” nearby geometry, for example to apply a projected texture that can be blocked by geometry, apply more “wear” to exposed geometry, or make other ambient occlusion-like effects.

Metadata

Metadata on parameters controls their display in the user interface. The following metadata is supported:

`[[ string label = "My Label" ]]`
: Name of parameter in in the user interface

`[[ string widget = "null" ]]`
: Hide parameter in the user interface.

`[[ string widget = "boolean" ]]` and `[[ string widget = "checkbox" ]]`
: Display integer parameter as a boolean checkbox.

Color Management

Color management is important to create renders and assets that are physically accurate and look great on multiple display devices. It is used both to ensure all parts of the pipeline interpret colors correctly, and to make artistic changes like exposure and color grading.

Blender's color management is based on the OpenColorIO library. By using the same OpenColorIO configuration in multiple applications, the same color spaces and transforms will be available for consistent results.

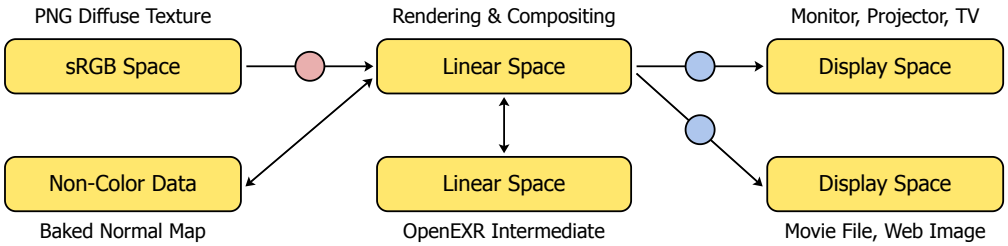


Different views and exposures of the same render.

Workflow

Scene Linear Color Space

For correct results, different Color Spaces are needed for rendering, display and storage of images. Rendering and compositing is best done in *scene linear* color space, which corresponds more closely to nature, and makes computations more physically accurate.



An example of a linear workflow.

If the colors are linear, it means that if in reality, we double the number of photons, the color values are also doubled. Put another way, if we have two photos/renders each with one of two lights on, and add those images together, the result would be the same as a render/photo with both lights on. It follows that such a radiometrically linear space is best for photorealistic rendering and compositing.

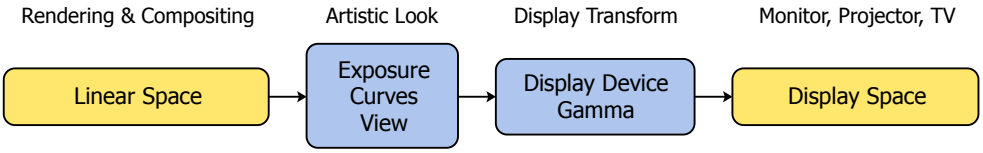
However, these values do not directly correspond to human perception or the way display devices work. and image files are often stored in different color spaces. So we have to take care to do the right conversion into and out of this scene linear color space.

Display Transforms

Transforming scene linear colors to display involves both technical and artistic choices.

Correct display of renders requires a conversion to the display device color space. A computer monitor works differently from a digital cinema projector or HDTV, and so needs a different conversion.

There is also an artistic choice to be made. Partially that is because display devices cannot display the full spectrum of colors and only have limited brightness, so we can squeeze the colors to fit in the gamut of the device. Besides that, it can also be useful to give the renders a particular look, e.g. as if they have been printed on real camera film. The default Filmic transform does this.



Conversion from linear to display device space.

Image Color Spaces

When loading and saving media formats it is important to have color management in mind. File formats such as PNG or JPEG will typically store colors in a color space ready for display, not in a linear space. When they are used as textures in renders, they need to be converted to linear first, and when saving renders for display on the web, they also need to be converted to a display space.

For intermediate files in production, it is recommended to use *OpenEXR* files. These are always stored in scene linear color spaces, without any data loss. That makes them suitable to store renders that can later be composited, color graded and converted to different output formats.

Images can also contain data that is not actually a color. For example normal or displacement maps merely contain vectors and offsets. Such images should be marked as *Non-Color Data* so that no color space conversion happens on them.

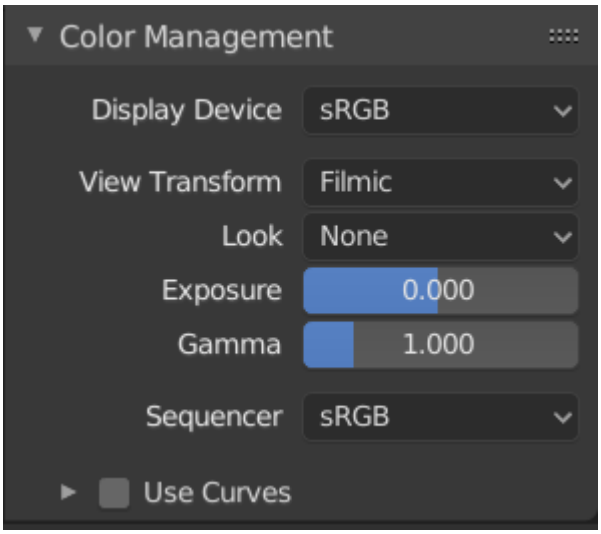
Render Settings

Reference

Editor:: Properties
Panel:: *Render Properties* ▸ *Color Management*

These are color management settings that are used across Blender. These color management settings are Scene specific so settings can be customized

per Scene. Color management can also be overridden when saving images; this behavior can be set in the Output Color Management Properties.



Color Management properties.

Display Device

The color space for the display that Blender is being viewed on.

Most displays are sRGB by default with some newer displays having the option to use Rec. 2020. These displays have a wider color gamut and can display high dynamic range content. If you have an Apple display you probably will want to use Display P3.

It is important to check your OS and display setting to make sure they all match the display in use to view the most accurate image.

sRGB:: Used by most displays.
Display P3:: Used by most Apple devices.
Rec. 1886:: Used by many older TVs.
Rec. 2020:: Used for newer wide gamut HDR displays.

View Transform

These are different ways to view the image on the same display device.

Standard:: Does no extra conversion besides the conversion for the display device. Often used for non-photorealistic results or video editing where a specific look is already baked into the input video.

Khronos PBR Neutral::

A tone mapping transform designed specifically for PBR color accuracy, to get sRGB colors in the output render that match as faithfully as possible the input sRGB base color in materials, under gray-scale lighting. This is aimed toward product photography use cases, where the scene is well-exposed and HDR color values are mostly restricted to small specular highlights.

AgX:: A tone mapping transform that improves on *Filmic*, giving more photorealistic results. AgX offers 16.5 stops of dynamic range and unsaturates highly exposed colors to mimic film's natural response to light.

Filmic:: A tone mapping transform designed to handle high dynamic range colors. Filmic is deprecated and is superseded by AgX which improves handling of saturated colors.

Filmic Log:: Converts to Filmic log color space. This can be used for export to color grading applications, or to inspect the image by flattening out very dark and light areas.

False Color:: Shows a heat map of image intensities, to visualize the dynamic range, and help properly expose an image.
Below is a table that represents how normalized linear color data is represented with False Color.

Luminance Value	**Color**
Low Clip	Black
0.0001% to 0.05%	Blue
0.05% to 0.5%	Blue-Cyan
0.5% to 5%	Cyan
5% to 16%	Green-Cyan
16% to 22%	Grey
22% to 35%	Green-Yellow

Luminance Value	Color
35% to 55%	Yellow
55% to 80%	Orange
80% to 97%	Red
High Clip	White

Raw:: Intended for inspecting the image but not for final export. Raw gives the image without any color space conversion.

Look

Choose an artistic effect from a set of measured film response data which roughly emulates the look of certain film types. Applied before color space conversion.

Exposure

Used to control the image brightness (in stops) applied before color space conversion. It is calculated as follows: \(output_value = render_value × 2^{(exposure)}\)

Gamma

Extra gamma correction applied after color space conversion. Note that the default display transforms already perform the appropriate conversion, so this mainly acts as an additional effect for artistic tweaks.

Sequencer

The color space that the Sequencer operates in. By default, the Sequencer operates in sRGB space, but it can also be set to work in Linear space like the Compositing nodes, or another color space. Different color spaces will give different results for color correction, crossfades, and other operations.

The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Display

High Dynamic Range
: Enable high dynamic range display in rendered viewport, uncapping display brightness. This requires a monitor with HDR support and a view transform designed for HDR (Filmic does not generate HDR colors).

  This feature is currently only supported on macOS.

Use Curves

Adjust RGB Curves to control the image colors before the color space conversion. Read more about using the Curve Widget.

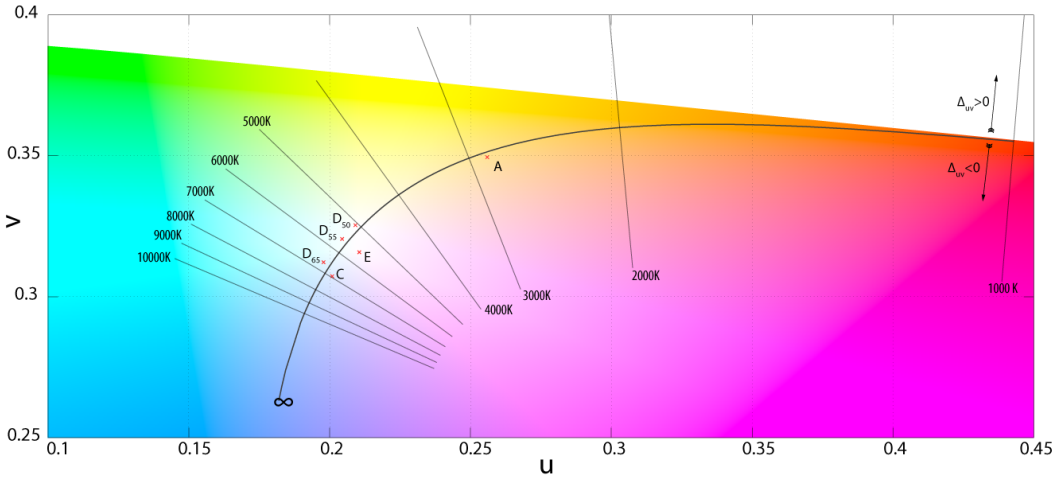
White Balance

Adjusts colors so that a given white point (expressed in color temperature and tint) ends up as white on the display.

As an alternative to manually specifying the values, there's also a color picker. When a color is selected, temperature and tint are set such that this color ends up being balanced to white. This only works if the color is close enough to a blackbody emitter.

Temperature
: The blackbody temperature of the primary illuminant. By default a D65 white point is used.

Tint
: The amount of green/magenta shift of the blackbody curve.



Blackbody temperature curve.

Tip

White balancing can also be accomplished as part of the compositing pipeline by using the Color Balance Node

Image Files

When working with image files, the default color space is usually the right one. If this is not the case, the color space of the image file can be configured in the image settings. A common situation where manual changes are needed is when working with or baking normal maps or displacement maps, for example. Such maps do not actually store colors, just data encoded as colors. Those images should be marked as *Non-Color Data*.

Image data-blocks will always store float buffers in memory in the scene linear color space, while a byte buffer in memory and files in a drive are stored in the specified color space setting.

By default, only renders are displayed and saved with the render *View Transformation* applied. These images are the “Render Result” and “Viewer” image data-blocks, and the files saved directly to a drive with the

Render Animation operator. However, when loading a render saved to an intermediate OpenEXR file, Blender cannot detect automatically that this is a render (it could be e.g. an image texture or displacement map). We need to specify that this is a render and that we want the transformations applied, with these two settings:

View as Render
: Display the image data-block (not only renders) with view transform, exposure, gamma, RGB curves applied. Useful for viewing rendered frames in linear OpenEXR files the same as when rendering them directly.

Save as Render
: Option in the image save operator to apply the view transform, exposure, gamma, RGB curves. This is useful for saving linear OpenEXR to e.g. PNG or JPEG files in display space.

OpenColorIO Configuration

Blender comes with a standard OpenColorIO configuration that contains a number of useful display devices and view transforms. The reference linear Color Space used is the linear color space with Rec. 709 chromaticities and D65 white point.

However, OpenColorIO is also designed to give a consistent user experience across multiple applications, and for this, a single shared configuration file can be used. Blender will use the standard OCIO environment variable to read an OpenColorIO configuration other than the default Blender one. More information about how to set up such a workflow can be found on the OpenColorIO website.

Blender currently use the following color space rules:

`scene_linear`
: Color space used for rendering, compositing, and storing all float precision images in memory.

`data`
: Color space for non-color data.

`aces_interchange`
: ACES2065-1 color space. Used to derive chromaticities of the *scene_linear* color space, for effects such as blackbody emission.

`color_picking`
: Defines the distribution of colors in color pickers. It is expected to be approximately perceptually linear, have the same gamut as the *scene_linear* color space, map 0..1 values to 0..1 values in the scene linear color space for predictable editing of materials' albedo.

`default_sequencer`
: Default color space for the Sequencer, *scene_linear* if not specified.

`default_byte`
: Default color space for byte precision images and files, *texture_paint* if not specified.

`default_float`
: Default color space for float precision images and files, *scene_linear* if not specified.

The standard Blender configuration includes support for saving and loading images in ACES (code and documentation) color spaces. However, the ACES gamut is larger than the Rec. 709 gamut, so for best results, an ACES specific configuration file should be used. OpenColorIO provides an ACES configuration file, though it may need a few more tweaks to be usable in production.

Default OpenColorIO Configuration

Color Spaces
: Blender's OCIO configuration file is equipped by default to read/write files in these color spaces:

sRGB:: Standard RGB display space using Rec. 709 chromaticities and a D65 white point.

Rec.2020:: BT.2020 2.4 Exponent EOTF Display.

Rec.1886:: BT.1886 2.4 Exponent EOTF Display, commonly used for TVs.

Non-Color:: Generic data that is not color, will not apply any color transform (e.g. normal maps).

Linear Rec.709:: Linear BT.709 chromaticities with illuminant D65 white point.

Linear Rec.2020:: Linear BT.2020 with illuminant D65 white point.

Linear FilmLight E-Gamut:: Linear E-Gamut with illuminant D65 white point.

Linear DCI-P3 D65:: Linear DCI-P3 with illuminant D65 white point.

Linear CIE-XYZ E:: 1931 CIE XYZ standard with assumed illuminant E white point.

Linear CIE-XYZ D65:: 1931 CIE XYZ with adapted illuminant D65 white point.

Filmic sRGB:: Similar to *sRGB* but uses the Filmic view transform.

Filmic Log:: Intermediate log color space of Filmic view transform.

Display P3:: Apple's Display P3 with sRGB compound (piece-wise) encoding transfer function, common on Mac devices.

ACEScg::

An ACES color space that is designed to be used for rendering and compositing. It uses the AP1 color primaries, a D60 white point, and a linear transfer function. While similar to ACES2065-1, this color space has a smaller color gamut. The smaller gamut allow it to better represent the colors that fit inside the CIE 1931 chromaticities diagram. Colors that lie outside the CIE 1931 chromaticities are generally not important to rendering and compositing because the human stimulus cannot represent these colors.

ACES2065-1:: An ACES color space using the AP0 color primaries, a D60 white point and a linear transfer function. This color space is meant to store and transfer data with the most amount of possible color information.

Freestyle

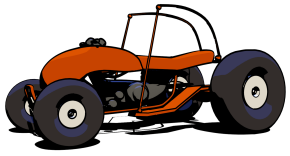
- Introduction
  - The Big Picture
  - Known Limitations
- Render Properties
- View Layer Properties
  - Freestyle
  - Line Set
  - Line Style
- Material Properties
- Python Scripting
  - Writing Style Modules

Introduction

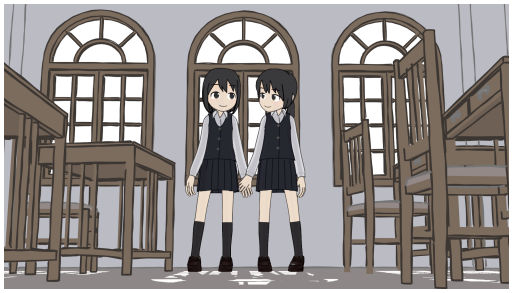
Freestyle is an edge/line-based non-photorealistic (NPR) rendering engine. It relies on mesh data and Z-depth information to draw lines on selected edge types. Various line styles can be added to produce artistic (“hand drawn”, “painted”, etc.) or technical (hard line) looks.

Freestyle can generate a powerful diversity of line styles and results. There are currently, two ways to define the way lines look; the first uses a series of parameter to create a Line Style. This mode allows intuitive editing of features such as dotted lines and easy setup of multiple line types and edge definitions. On top of all of that, with line style modifiers, the sky is the limit!

The second method of generating lines is by using Python Scripting. This method is much more advanced but Blender includes many pre-scripted styles such as Japanese big brush, cartoon, blueprint, and thickness-with-depth.



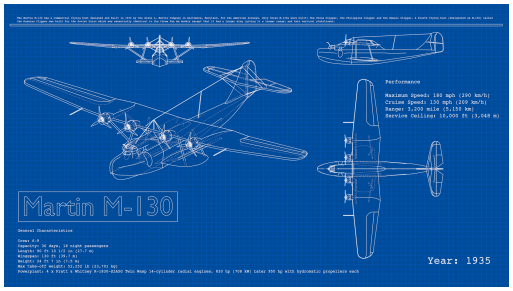
ATV buggy by Rylan Wright (RONIN).
CC BY.
(File:AtvBuggy.zip)



By mato.sus304. CC BY-SA.
(File:Mato_sus304_cut02.zip)



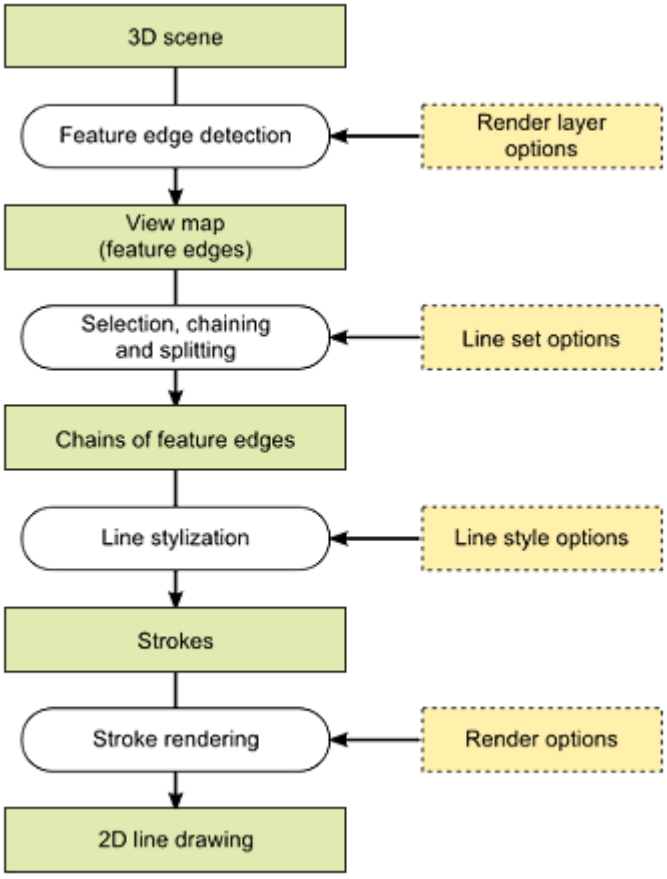
A cartoon scene from OHA Studio © Mechanimotion Entertainment. (blend-file)



Blueprint render of Martin M-130 from 1935 by LightBWK. CC0. Warning: heavy file! designed for stress test Blender to the limits and may crash Blender. (File:M-130Blueprint.zip)

The Big Picture

- Activate Freestyle by the *Properties* ▸ *Render* ▸ *Freestyle* checkbox.
- Freestyle settings are located in the View Layer properties.
- One view layer can only have one view map. A view map holds the edge detection settings (Crease Angle, Culling toggle, Face Smoothness toggle, Material Boundaries toggle, Sphere Radius, and Kr Derivative Epsilon advanced options).
- A view map can have multiple Line Sets.
- A line set controls which line types and selections will be rendered, from lines based on your scene.
- Each line set uses one line style (which can be shared between multiple Line Sets).
- A line style tells Freestyle how to render the linked Line Sets in terms of color, alpha, thickness and other aspects.



Block diagram of Freestyle view map and processes.

Known Limitations

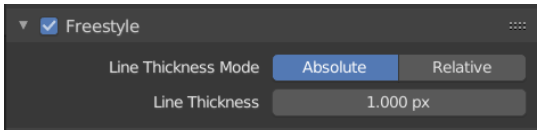
- Highly memory demanding: All mesh objects in a view layer are loaded at once.
- Only faced mesh objects are supported.
- No edges at face intersections are detected yet.
- Freestyle rendering results do not have any Z depth information.
- Panoramic cameras are not supported.

Render Properties

Reference

Panel:: *Properties* ▸ *Render* ▸ *Freestyle*

Freestyle can be activated with the checkbox in the header of the Freestyle panel in the *Render* tab.



Freestyle Render Properties.

Line Thickness Mode
: There are two different modes for defining the base line thickness:

Absolute:: The line thickness is given by a user-specified number of pixels.

Relative:: The unit line thickness is scaled by the proportion of the present vertical image resolution to 480 pixels. For instance, the “unit line thickness” is 1.0 when the image height set to 480px, 1.5 with 720px and 2.0 with 960px.

Line Thickness
: Line thickness to use for rendering (only for *Absolute* line thickness).

View Layer Properties

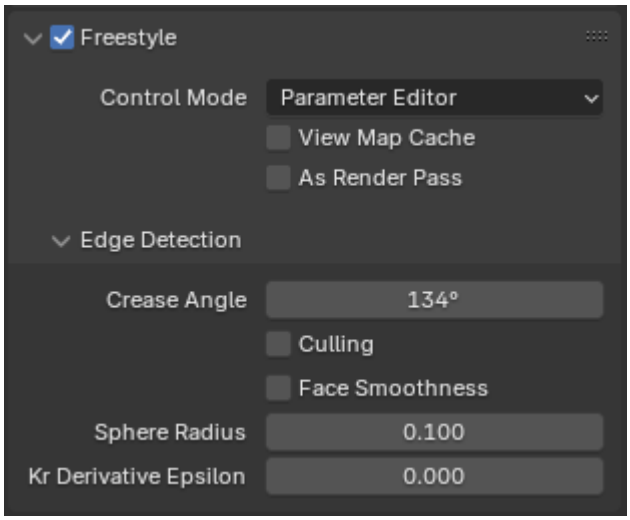
- Freestyle
  - Edge Detection
- Line Set
  - Visibility
  - Edge Types
  - Face Marks
  - Collection
- Line Style
  - Introduction
  - Properties
  - Modifiers

Freestyle

Reference

Panel:: *Properties ▸ View Layer ▸ Freestyle*

There is only one view map per view layer. It controls the edge detection parameters. Freestyle can be enabled/disabled per View Layer by toggling the checkbox in the panel header.



View Layer: Freestyle panel.

Control Mode

Which detected edges are actually rendered, and how, can be controlled either through:

Parameter Editor Mode::

Lines are rendered via parameters defined in a user-friendly interface to define and control Line Sets and line styles.
A view map (hence a view layer) can have multiple Line Sets, and each line set is linked to

one line style.

Python Scripting Mode::
Lines are rendered via Python scripting, powerful but complex.

View Map Cache
An option to reuse a previously computed view map for subsequent rendering. The cache is automatically updated when the mesh geometry of the input 3D scene has been changed.

This functionality offers a major performance boost for Freestyle animation rendering when the camera-space mesh geometry is static, as well as for repeated still renders with updates of line stylization options.

Although the *View Map Cache* checkbox is a view layer option, the cache memory is shared by all view layers and scenes. This means that if Freestyle is used for two or more view layers (possibly in different scenes through the Compositor), then the cached view map for one view layer is replaced by a new view map for another view layer and hence no performance gain is expected.

As Render Pass
Freestyle lines will not immediately be visible on top of the render image. Instead, Freestyle lines are rendered as a Render Pass which can be composited with the rendered image with an Alpha Over node.

Edge Detection

Crease Angle
If two adjacent faces form an angle less than the defined *Crease Angle*, the edge between them will be rendered when using *Crease* edge type selection in a line set. The value also affects *Silhouette* edge type selection.

Culling
Ignore the edges that are out of view. (Saves some processing time and memory, but may reduce the quality of the result in some cases.)

Face Smoothness

Takes *Smooth Shading* into account for edges calculation.

Sphere Radius

Affects the calculation of curvatures for *Ridge*, *Valley* and *Suggestive Contour* edge type selection in a line set. The curvature at each vertex is computed by averaging the shape of the surface within the specified radius. Increasing the value reduces noise and detail.

Kr Derivative Epsilon

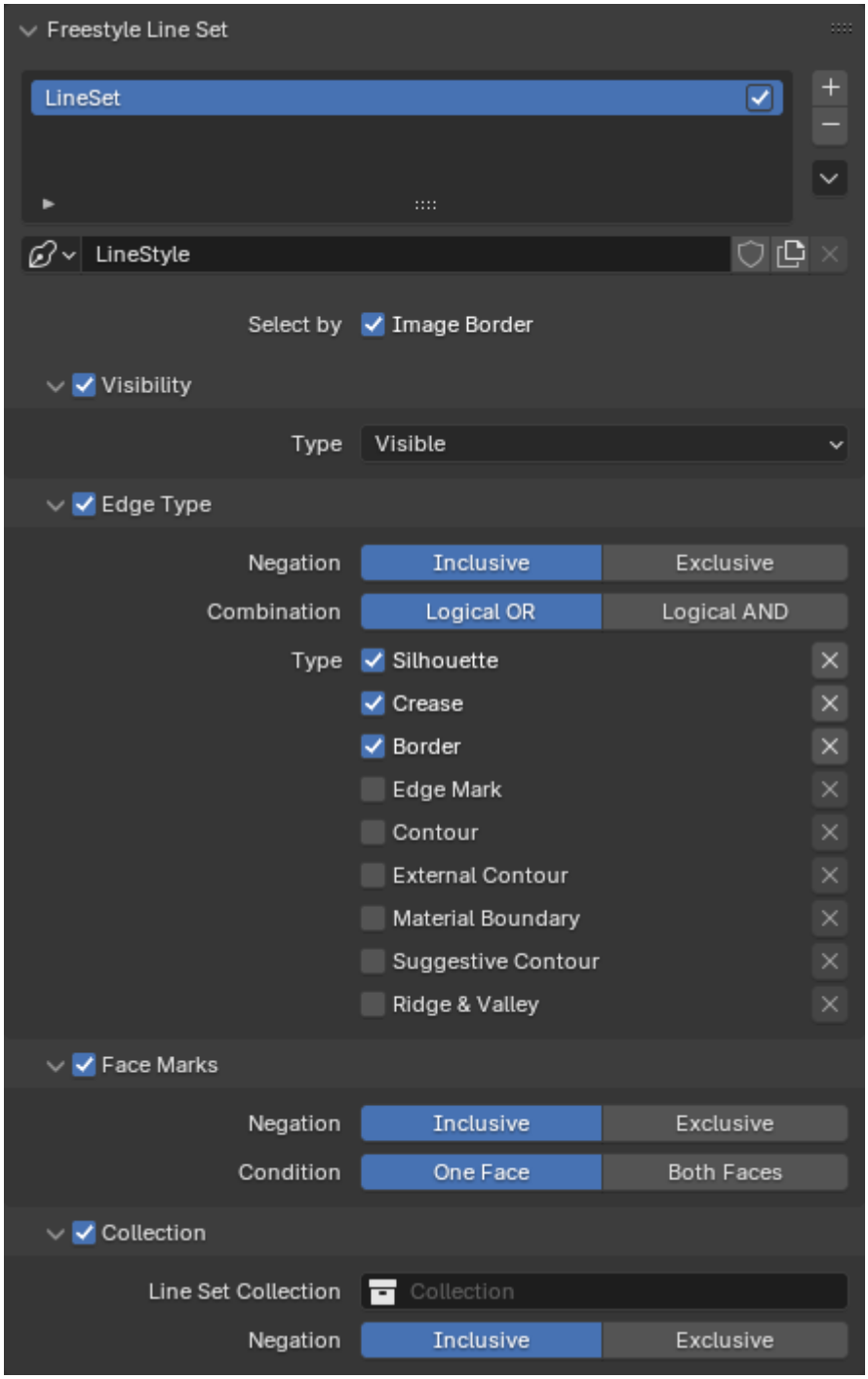
Controls the threshold on the minimum rate of change of curvature used to filter the output of the *Suggestive Contour* edge type selection. Increasing the value reduces the amount of rendered lines, starting from smoother areas of the object (further information in this pdf).

Line Set

Reference

Panel:: *Properties ▸ View Layer ▸ Freestyle Line Set*

A line set selects, among the lines (edges) detected by Freestyle, which ones will be rendered using its attached line style, through various methods.



Freestyle Line Set panel.

Select By

Image Border

Causes Freestyle to only take geometry within the image border into consideration for line calculation. This reduces render times but

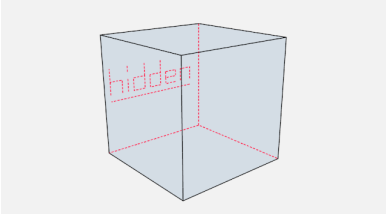
increases continuity problems when geometry is moved out of and into camera view.

Visibility

Type
: Determine how to use visibility for feature edge selection.

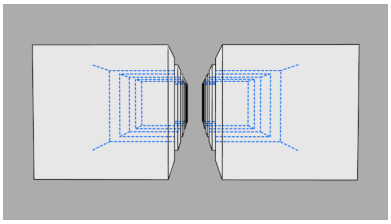
Visible:: Only lines occluded by no surfaces are rendered.

Hidden:: Lines occluded by at least one surface are rendered.



Proof of concept of visible and hidden edges by LightBWK (blend-file).

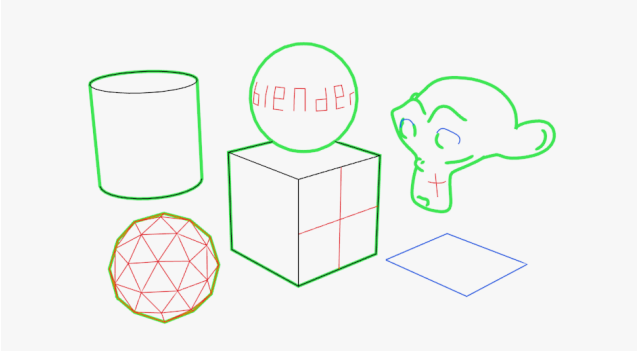
Quantitative Invisibility::
Lines occluded by a number of surfaces in the given range are rendered.
Start, End
: Min/max number of occluding surfaces for a line to be rendered.



QI Range proof of concept demo, Start: 3, End: 7, by LightBWK (blend-file).

Edge Types

Edge types are basic algorithms for the selection of lines from geometry. When using the parameter editor you have to choose at least one edge type in order to get a render output, but several edge types can be combined in one line set. Edge types can also be excluded from calculation by pressing the *X* next to them.



Examples of some basic edge types: Silhouette (green), Crease (black), Border (blue) and Edge Marks (red) (blend-file by LightBWK).

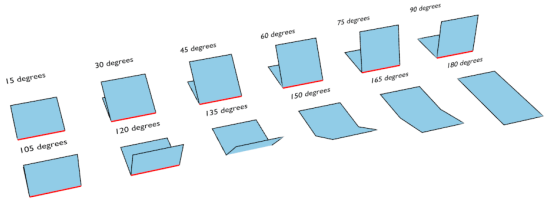
Type

Silhouette

Draws silhouettes around your closed objects by rendering lines where the surface normal transitions between pointing toward and away from the camera. It is often good for organic objects (like Suzanne & Sphere), and bad for sharp edges, like a box. It cannot render open mesh objects like open cylinders and flat planes.

Crease

Shows only edges whose adjacent faces form an angle sharper than the defined view map's *Crease Angle*.



Crease Angle proof of concept for 121° by LightBWK (blend-file).

Border

Border shows open mesh edges, i.e. edges that belong to only one face. An open cylinder has open edges at the top and bottom, and a plane is open all around. Suzanne's eye socket is an open edge.

Edge Mark

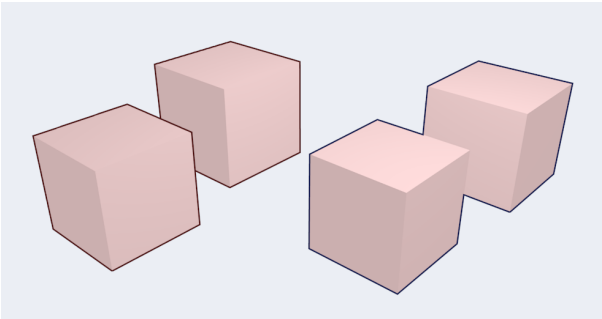
Renders marked edges. See Edge Marks for details.

Contour

Draws lines around each object, separating it from other objects behind it, or the scene background.

External Contour

Draws lines around all objects, separating them from the scene background, but not each other.



Left pair: Contour; Right pair: External Contour.

Material Boundary

Draws lines where two materials meet on the same object.

Suggestive Contour

Draws some lines which would form the *Silhouette* of the mesh if the view point was shifted. Depends on your view map settings for *Kr Derivative Epsilon* and *Sphere Radius* (further information: File:Manual-2.6-Render-Freestyle-PrincetownLinestyle.pdf).

Ridge & Valley

Draws lines marking crests of ridges and valleys, i.e. places where the surface curvature is at its minimum or maximum. Depends on your *Sphere Radius* view map settings.

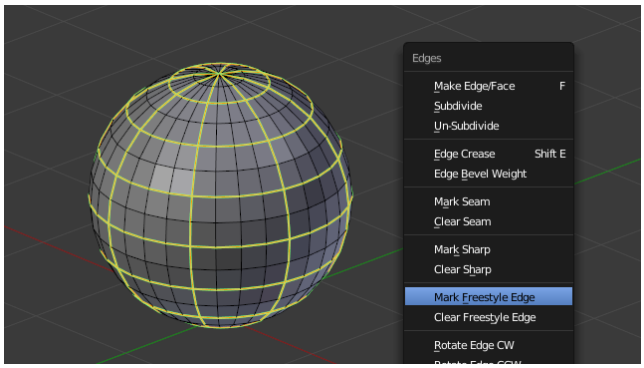
Edge Marks

In Edit Mode you can mark "Freestyle Edges" in the same manner you can mark "Seams" for UV unwrapping or "Sharp" for edge split. These marked edges are available to render when you select *Edge Mark*.

This is done as follows:

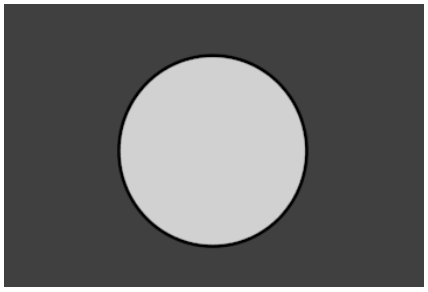
1. Select the mesh object and enter *Edit Mode*.
2. Select the edges you want to be marked.
3. Press `Ctrl-E` and select *Mark Freestyle Edge*.

Edge marks are useful when you want to draw lines along particular mesh edges. The examples below explain the use of edge marks.



Marking Freestyle Edges in Edit Mode; the edge marks are highlighted in green.

With *Edge Marks* enabled, the previously-marked lines are always rendered. You can see the black contour lines and the blue lines that are made with edge marks.



Render without Edge Marks.



Render with Edge Marks enabled.

What are edge marks good for?

- When you need to render marks on an almost-flat plane, when other edge types cannot detect any line.
- When you want full control of edge rendering. Often used for edges of squarish shapes.
- Mark the whole base mesh to be rendered for base mesh preview.

What are edge marks not good for?

- Round outer edges (use instead *Contour*/*External Contour*/*Silhouette*).

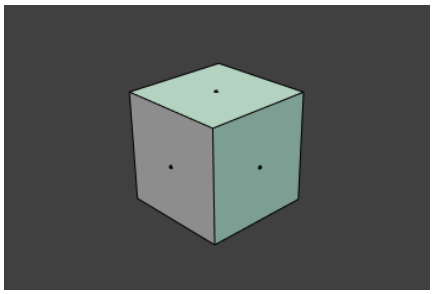
Face Marks

Face marks are useful for removing lines from certain areas of a mesh.

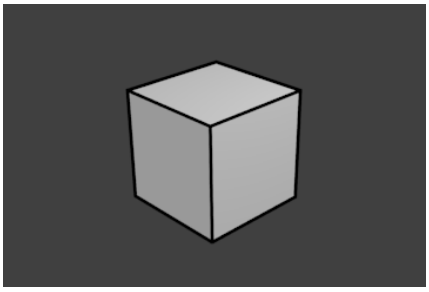
To set a face mark:

1. Select a mesh object and enter *Edit Mode*.
2. Select the faces you want to be marked.
3. Press `Ctrl-F` and select *Face Data* ▸ *Mark Freestyle Face*.

In this example, two faces of the default cube are marked like the image on the left. On the right is a render without face marks activated.



Marked faces (Edit Mode).



Render output.

The line selection can be controlled via inclusion and faces options:

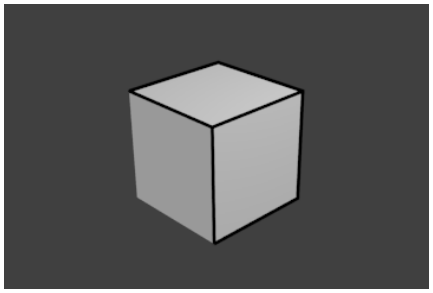
Negation

Whether to include or exclude edges matching defined face mark conditions from the line set.

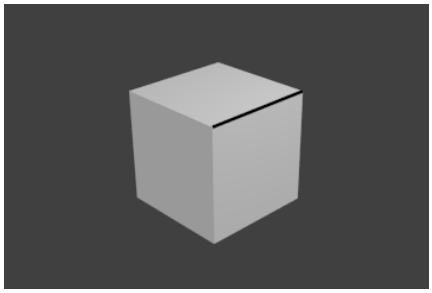
Condition

One Face:: (De)select all edges which have one or both neighbor faces marked.

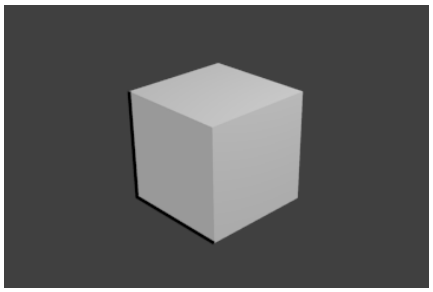
Both Faces:: (De)select all edges which have both of their neighbor faces marked.



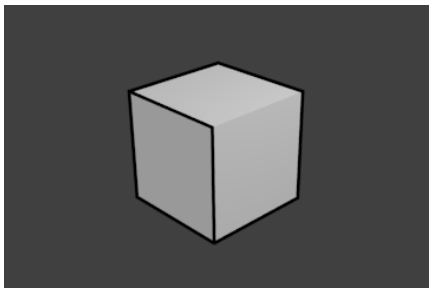
Inclusive, One Face.



Inclusive, Both Faces.



Exclusive, One Face.



Exclusive, Both Faces.

Collection

Include or exclude objects for line calculation, based on their belonging to a Collection.

Line Set Collection
: The name of the object collection to use.

Negation
: Whether to include or exclude lines from those objects in this line set.

Line Style

- Introduction

Properties

- Strokes
- Color
- Alpha
- Thickness
- Geometry
- Texture

Modifiers

- Color
- Alpha
- Thickness
- Geometry

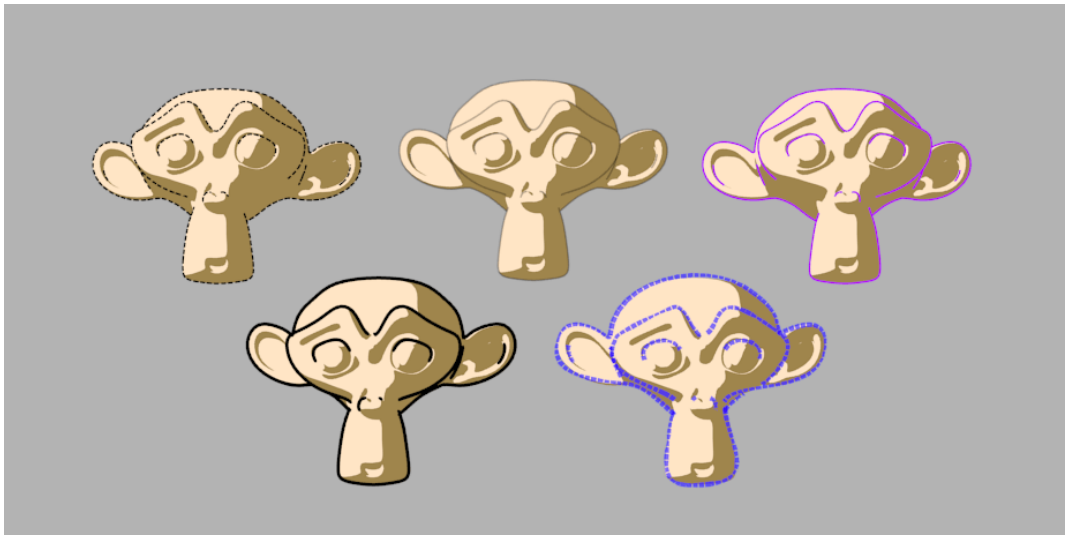
Introduction

In Freestyle, the line style settings define the appearance of a line set using five main aspects: Stroke, Color, Alpha, Thickness, Geometry, and Texture with each on a separate tab. These allow you to get many different styles of renders (technical draw, rough sketch, cartoon, calligraphy, etc.).

You can create as many line styles as you wish, and reuse a given line style for several line sets by selecting it from the select menu next to its name.

Note

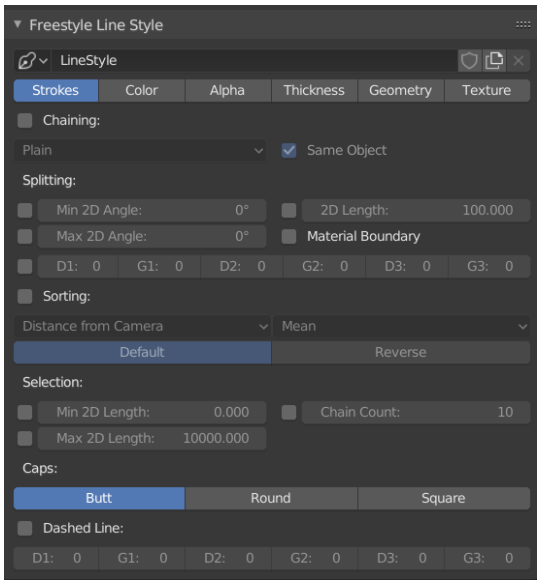
Unless otherwise specified, all lengths in line style settings are in pixels (either relative or absolute, as specified in the core options).



Line Style Example (blend-file).

Strokes

Strokes are the final rendered lines. Yet you can tweak them, for example, by removing the ones longer/shorter than some threshold, chaining lines into a single stroke or breaking a stroke into several ones based on angles, dashed pattern, etc.



Line Style: Strokes.

Caps

You can choose between three types of line caps:

Butt:: Flat cap, exactly at the point the line ends.

Round:: A half circle centered on the end point of the line.

Square:: A square centered on the end point of the line (hence, like the circle, the drawn end of the line is slightly extended compared to its computed value).



Line caps example.

Chaining

By default all retrieved lines from the line set are chained together. There are two basic chaining methods:

Method

Plain:: The default chaining method; it creates simple chains.

Sketchy:: This chaining option allows for generating chains of feature edges with sketchy multiple strokes. Basically, it generates *Round* strokes instead of a single one. It is only really useful if you use some random-driven modifiers in the line style!

Rounds

It specifies the number of rounds in sketchy strokes.

Same Object

If true, only feature edges of the same object are joined.

Chaining can also be turned off to render each line separately, which can be useful for line styles which depend on accurate representation of the line

set.

Splitting

You can split up chains of Freestyle lines by enabling one of the following:

Min/Max 2D Angle
: Splits chains of feature edges when they make a 2D angle above (or below) a minimum (or maximum) threshold.

2D Length
: Splits chains when they are longer than the given value.

Material Boundary
: Splits chains of feature edges if they cross from one material to another.

Split Pattern

Splits the chains using the given dashed pattern (see also Dashed Line).

Dash 1, 2, 3
: Length of the specified dash for splitting.

Gap 1, 2, 3
: Length of the specified gap for splitting.

Sorting

You can sort the order of your strokes, allowing the lines to stack in the order given.

Sort Key
: A sort key is used to determine the stacking order of lines.

  Distance from Camera::
  : Lines closer to the camera lie on top of further lines.

2D Length:: Longer lines lie on top of shorter lines.
Projected X/Y:: Sort by the projected X or Y value in the image coordinate system.

Integration Type
Use in tandem with the Sort Key to determine the range for sorting. Since the distance of a line from the camera may vary over vertices, this option computes the sort key for a line from the values computed at individual vertices. The value computed for the line is:

Mean:: The mean of the values obtained for the vertices.
Min:: The minimum of the values obtained for the vertices.
Max:: The maximum of the values obtained for the vertices.
First:: The value obtained for the first vertex.
Last:: The value obtained for the last vertex.

Sort Order
With the given result you can choose to "Reverse" the sort order.

Selection

You can also choose to only render selected chains.

Min/Max 2D Length
Chains longer and/or shorter than *2D Length*.

Chain Count
Allows the selection of first N chains.

Dashed Line

By enabling the *Dashed Line* checkbox, you can specify three pairs of dash and gap lengths. Dash values define the lengths of dash strokes, while gap values specify intervals between two dashes.

If a zero gap is specified, then the corresponding dash is ignored even if it has a nonzero value.

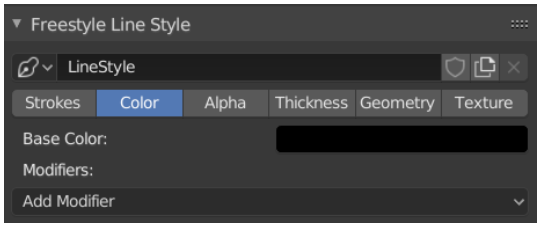
Dashes are treated as separate strokes, meaning that you can apply line caps, as well as color, alpha and thickness modifiers.

Dash 1, 2, 3
: Length of the specified dash for dashed lines.

Gap 1, 2, 3
: Length of the specified gap for dashed lines.

Color

In this tab you control the color of your strokes.



Line Style: Color.

Base Color
: The base color for this line style.

Modifiers

Common Options

Mix
: The modifier output can be mixed with the base property using the usual methods (see for example the Mix compositing node).

Influence
: How much the result of this modifier affects the current property.

Color Ramp
: Each modifier has color ramp that maps the property to a stroke color.

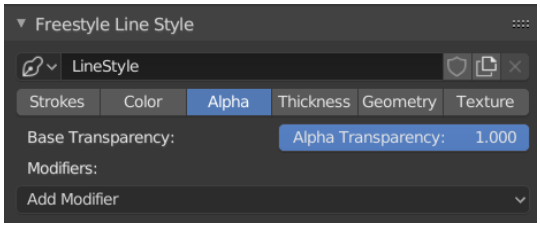
Types

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera

- Distance from Object
- Material
- Noise
- Tangent

Alpha

In this tab you control the alpha (transparency) of your strokes.



Line Style: Alpha.

Base Transparency
: The base alpha for this line style.

Modifiers

Common Options

Mix
: The modifier output can be mixed with the base property using the usual methods (see for example the Mix compositing node).

Influence
: How much the result of this modifier affects the current property.

Mapping
: Either a linear progression (from 0.0 to 1.0), or a custom mapping curve.

> **Note**
>
> Note the linear non-inverted option is equivalent to “do nothing”, as original values from materials are already in the (0.0 to 1.0) range.

That is the case for: Crease Angle, Curvature 3D, Material, Noise, Tangent.

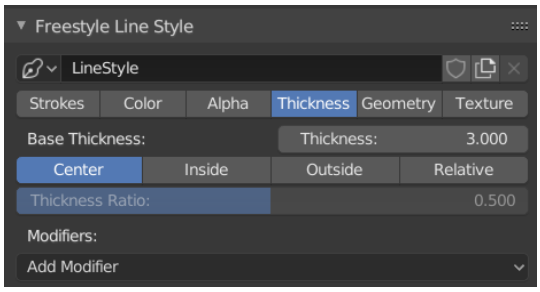
Invert
: Inverts the *Mapping*.

Types

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Material
- Noise
- Tangent

Thickness

Controls the thickness of the Freestyle strokes.



Line Style: Thickness.

Base Thickness

The base thickness for this line style.

Thickness Position

Control the position of stroke thickness from the original (backbone) stroke geometry. There are four choices:

Center:: The thickness is evenly split to the left and right side of the stroke geometry.

Inside:: The strokes are drawn within object boundary.

Outside:: The strokes are drawn outside the object boundary.

Relative:: Specifies the relative position by a number between 0.0 (inside) and 1.0 (outside), in the *Thickness Ratio* number field just below.

Note

The thickness position options are applied only to strokes of edge types *Silhouette* and *Border*, since these are the only edge types

defined in terms of the object boundary. Strokes of other edge types are always drawn using the *Center* option.

Modifiers

Common Options

Mix
: The modifier output can be mixed with the base property using the usual methods (see for example the Mix compositing node).

Influence
: How much the result of this modifier affects the current property.

Types

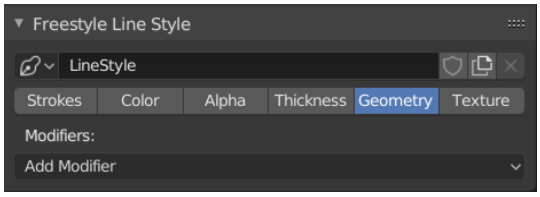
- Along Stroke
- Calligraphy
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Material
- Noise
- Tangent

Geometry

In this tab you control the geometry of your strokes. It contains only the option to add modifiers.

As they always completely apply to the strokes' geometry (like object modifiers do). They take the resulting two-dimensional strokes from the Freestyle line set and displace or deform them in various ways.

As with other modifier stacks in Blender, they are applied from top to bottom.



Line Style: Geometry.

Modifiers

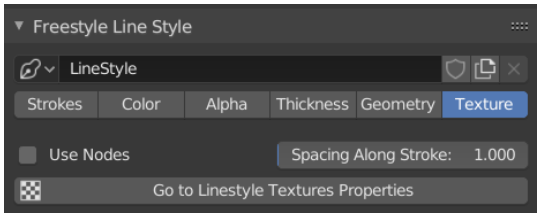
Types

- 2D Offset
- 2D Transform
- Backbone Stretcher
- Bézier Curve
- Blueprint
- Guiding Lines
- Perlin Noise 1D
- Perlin Noise 2D
- Polygonization
- Sampling
- Simplification

- Sinus Displacement
- Spatial Noise
- Tip Remover

Texture

Assigns a texture to the Freestyle stroke.



Line Style: Texture.

Use Nodes
: In Cycles textures are defined by means of shader Nodes.

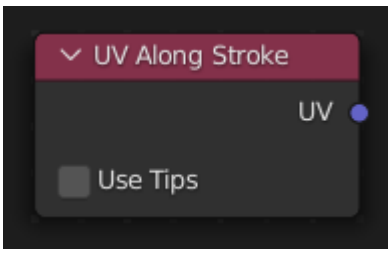
Spacing Along Stroke
: Allows to set the “pace” of textures mapped along the length of strokes.

Go to Linestyle Textures
: A shortcut to the line style texture properties in the other *Textures* tab of the Properties. Make sure to first enable *Use Nodes*.

Nodes

UV Along Stroke Node

The *UV Along Stroke* input node is maps textures along the stroke length, making it possible to mimic pencil, paintbrush, and other art medium marks.



UV Along Stroke Node.

Note

These UV maps become available only during the Freestyle rendering process. Hence, the UV

Along Stroke node cannot be replaced by the conventional UV Map input node which takes an existing UV map already defined as part of mesh data.

Inputs

This node has no inputs.

Properties

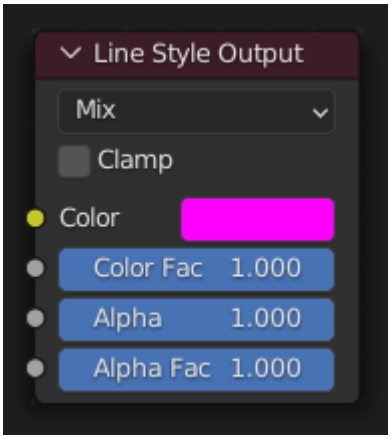
Use Tips
: Allows to use lower quarters of a texture image for the head and tail tips of a stroke, while the upper half for the stroke body.

Outputs

UV
: UV maps defined along strokes.

Line Style Output Node

The *Line Style Output* node specifies how to mix the texture information into the base color of line styles.



Line Style Output Node.

Inputs

Color
: Color input for the texture.

Color Factor
: Standard mix factor of the *Color* value.

Alpha
: Alpha input for the texture.

Alpha Factor
: Standard mix factor of the *Alpha* value.

Properties

Mix
: The Blend mode can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Clamp
: Limit the highest color value to not exceed 1.0.

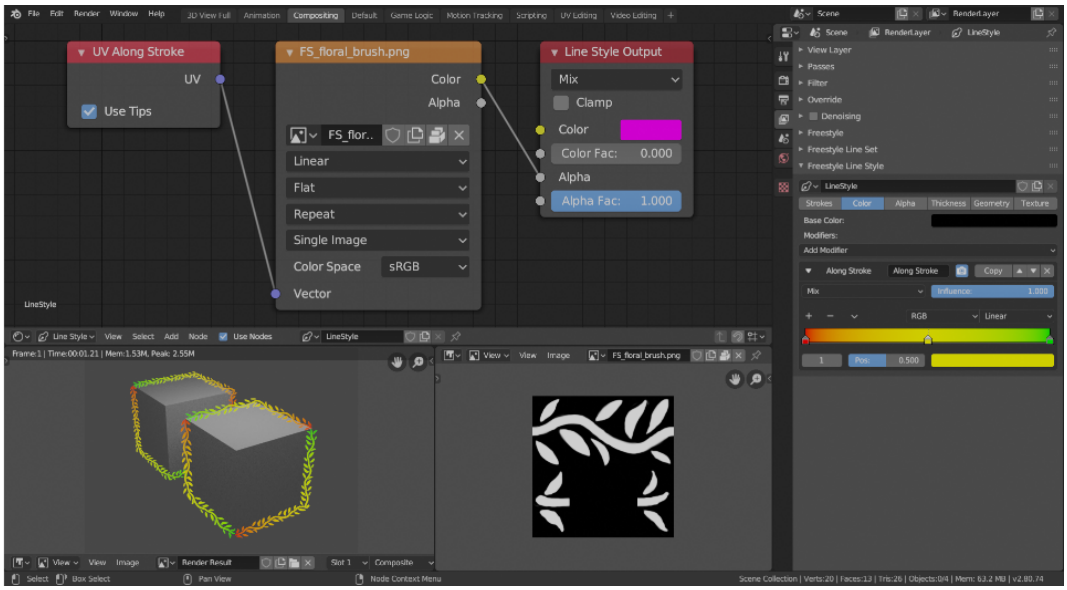
Outputs

This node has no outputs.

Example

The image below shows a typical shader node tree that maps a floral texture image along strokes. The UV Along Stroke input node retrieves UV maps defined by Freestyle along generated strokes, and passes them to the Vector input channel of the Image Texture node. A texture image is selected in the Image Texture node, and its color is inputted to the Alpha channel of the Line Style Output node. Since the Alpha Factor is set to one, the texture image replaces the base alpha transparency of the active line style (shown

in the Freestyle Line Style panel). On the other hand, the Mix blend mode is selected in the Line Style Output node with the Color Factor set to zero, so that the gradient line color specified in the active line style is applied along strokes.



Example of Line Style Nodes (blend-file).

It is noted that the texture image `FS_floral_brush.png` shown in the screen capture is an example of Freestyle brush images with tips. Specifically, the upper half of the image is used as a seamless horizontal tile of the stroke body. Whereas the parts in the lower half are tips (stroke caps) at both ends of the stroke.

Color Modifiers

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Material
- Noise
- Tangent

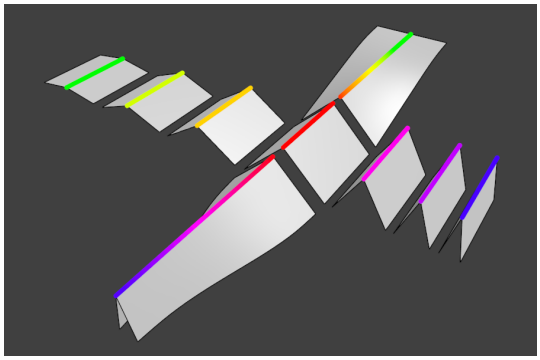
Along Stroke

The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max
: The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



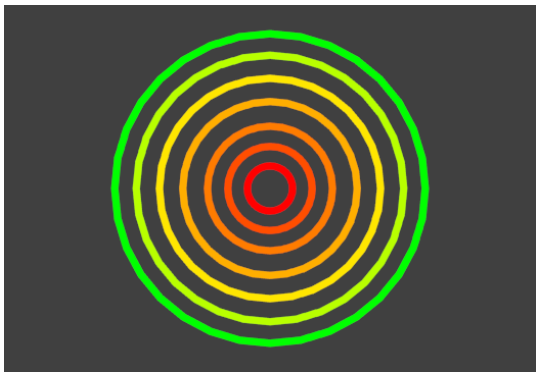
Crease Angle modifier example by T.K. (blend-file).

Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The curvature of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max
: The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. (blend-file).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max
: The limits of the mapping from "distance to camera" to "property in mapping". If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target
: The object to measure distance from.

Range Min, Max
: The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. (blend-file).

Noise

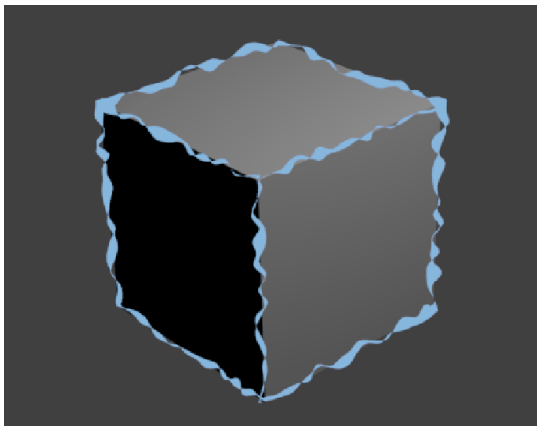
The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude
: The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period
: The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

Seed
: Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*
: Allows the thickness to be distributed unevenly at every point. Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke’s vertices.

Alpha Modifiers

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Material
- Noise
- Tangent

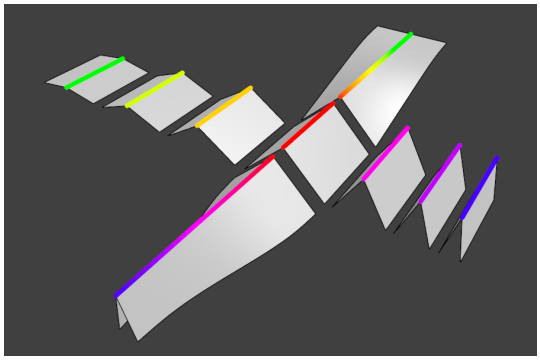
Along Stroke

The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max
: The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



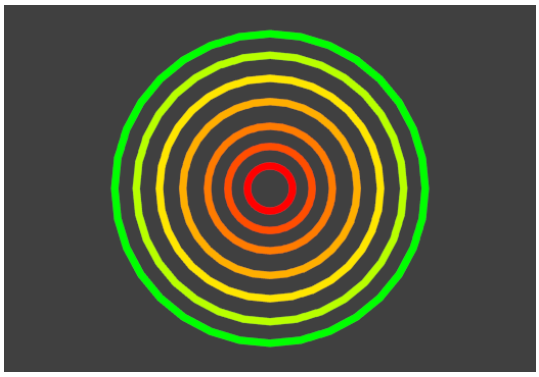
Crease Angle modifier example by T.K. (blend-file).

Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The curvature of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max
: The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. (blend-file).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max
: The limits of the mapping from "distance to camera" to "property in mapping". If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target
: The object to measure distance from.

Range Min, Max
: The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. (blend-file).

Noise

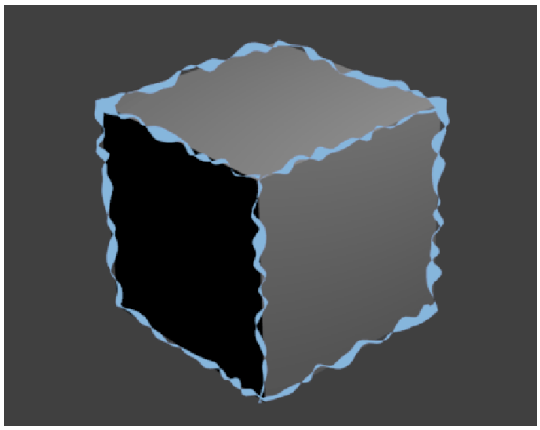
The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude
: The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period
: The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

Seed
: Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*
: Allows the thickness to be distributed unevenly at every point. Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke’s vertices.

Thickness Modifiers

- Along Stroke
- Calligraphy
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Material
- Noise
- Tangent

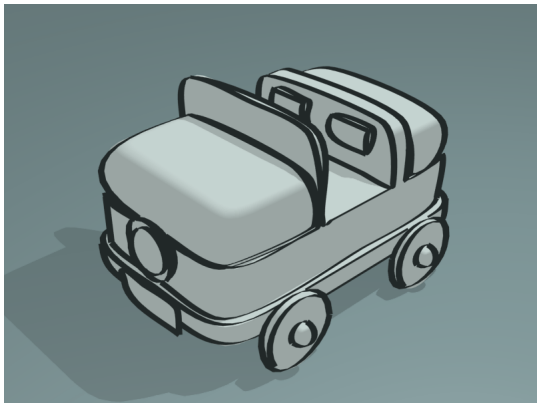
Along Stroke

The *Along Stroke* modifier alters the base property with a new one from a given range mapped along each stroke's length. In other words, it applies a gradient along each stroke.

Calligraphy

The *Calligraphy* modifier mimics some broad and flat pens for calligraphy. It generates different thickness based on the orientation of the stroke.

Orientation
: The angle (orientation) of the virtual drawing tool, from the vertical axis of the picture. For example, an angle of 0.0 mimics a pen aligned with the vertical axis. Hence, the thickest strokes will be the vertical ones i.e. stroke's direction is aligned with the angle, and the thinnest will be the horizontal ones i.e. stroke's direction is perpendicular to the angle.

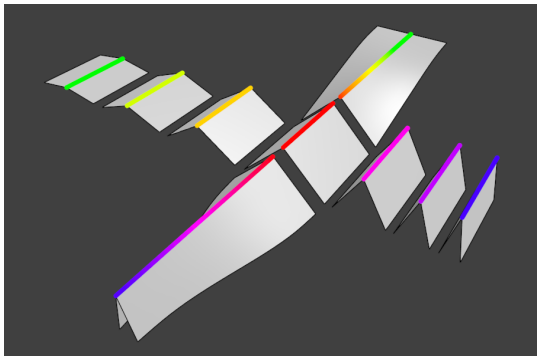


Calligraphy modifier demo by T.K. (blend-file).

Crease Angle

A modifier based on the Crease Angle (angle between two adjacent faces). If a stroke segment does not lie on a crease (i.e. the edge does not have the *Crease Angle nature*), its properties are not touched by the modifier.

Angle Min, Max
: The range of input values to the mapping. Out-of-range crease angle values will be clamped by the Min and Max angles and their corresponding property values.



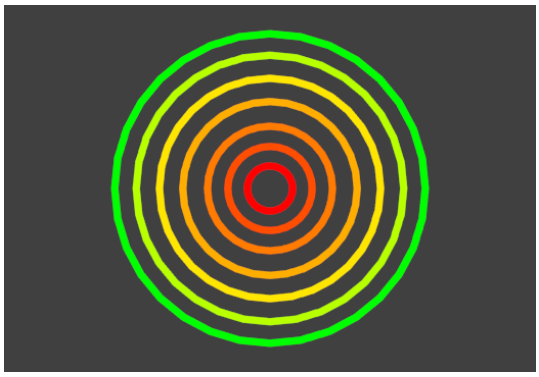
Crease Angle modifier example by T.K. (blend-file).

Curvature 3D

A modifier based on radial curvatures of the underlying 3D surface. The curvature of a 2D curve at a point is a measure of how quickly the curve turns at the point. The quicker the turn is, the larger the curvature is at the point. The curvature is zero if the curve is a straight line. Radial curvatures are those computed for a 2D curve that appears at the cross section between the 3D surface and a plane defined by the view point (camera location) and the normal direction of the surface at the point.

For radial curvatures to be calculated (and therefore for this modifier to have any effect), the *Face Smoothness* option has to be turned on and the object needs to have *Smooth Shading*.

Curvature Min, Max
: The limits of the mapping. If the current point of the stroke is at *Min Curvature* or less from the target, it will take the start point of the mapping. And conversely, if it is at *Max Curvature* or more from the target, it will take the end-point value of the mapping.



Curvature 3D modifier demo by T.K. (blend-file).

Distance from Camera

The *Distance from Camera* modifier alters the base property with a new one from a given range using the distance to the active *camera*.

Range Min, Max
: The limits of the mapping from "distance to camera" to "property in mapping". If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Range Max* or more from the camera/object, it will take the end value. These values are in the current scene's units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Distance from Object

The *Distance from Object* modifier alters the base property with a new one from a given range using the distance to the active *camera* or to a given *object* as the parameter.

Target
: The object to measure distance from.

Range Min, Max
: The limits of the mapping from “distance to camera” to “property in mapping”. If the current point of the stroke is at *Range Min* or less from the active camera or the object, it will take the start value. And conversely, if it is at *Max* or more from the camera/object, it will take the end value. These values are in the current scene’s units, not in pixels!

Fill Range by Selection
: Set the min/max range values from the distances between the current selected mesh vertices and the camera or the target.

Material

The *Material* modifier alters the base property with a new one taken from a given range mapped on the current material under the stroke.

You can use various properties of the materials, among which many are mono-component (i.e. give black-and-white results). In this case for the color modifier, an optional color ramp can be used to map these gray-scale values to colored ones. In the reverse case properties of the materials, which are multi-components (i.e. give RGB results) the mean value will be used for Alpha and Thickness modifiers.

If used with the *Split by Material* option in the *Stroke* tab, the result will not be blurred between materials along the strokes.



Material modifiers demo by T.K. (blend-file).

Noise

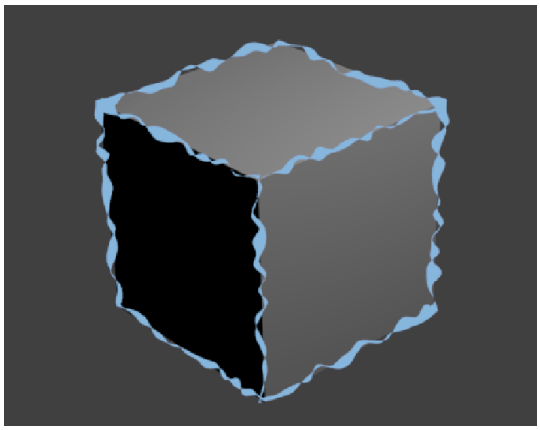
The *Noise* modifier uses a pseudo-random number generator to variably distribute the property along the stroke.

Amplitude
: The maximum value of the noise. A higher amplitude means a less transparent (more solid) stroke.

Period
: The period of the noise. This means how quickly the property value can change. A higher value means a more smoothly changing color along the stroke.

Seed
: Seed used by the pseudo-random number generator.

Asymmetric *Thickness only*
: Allows the thickness to be distributed unevenly at every point. Internally, the stroke is represented as a backbone with a thickness to the right and left side. All other thickness shaders make sure that the left and right thickness values are equal. For the Noise shader however, a meaningful (and good-looking) result can be created by assigning different values to either side of the backbone.



Effect generated with a noise thickness modifier using asymmetric thickness.

Tangent

This modifier bases its effect on the traveling direction of the stroke evaluated at the stroke’s vertices.

Geometry Modifiers

- 2D Offset
- 2D Transform
- Backbone Stretcher
- Bézier Curve
- Blueprint
- Guiding Lines
- Perlin Noise 1D
- Perlin Noise 2D
- Polygonization
- Sampling
- Simplification
- Sinus Displacement
- Spatial Noise
- Tip Remover

2D Offset

The *2D Offset* modifier adds some two-dimensional offsets to the stroke backbone geometry. It has two sets of independent options/effects:

Start, End
: These two options add the given amount of offset to the start (or end) point of the stroke, along the (2D) normal at those points. The effect is blended over the whole stroke, if you for example, set only *Start* to 50, the start of the stroke is offset 50 pixels along its normal, the middle of the stroke, 25 pixels along its own normal, and the end point is not moved.

X, Y
: These two options simply add a constant horizontal and/or vertical offset to the whole stroke.

2D Transform

The *2D Transform* modifier applies two-dimensional scaling and/or rotation to the stroke backbone geometry. Scale is applied before rotation.

Pivot
: The center (pivot point) of these 2D transformations can be:

  Stroke Center:: The median point of the stroke.

  Stroke Start:: The beginning point of the stroke.

  Stroke End:: The end point of the stroke.

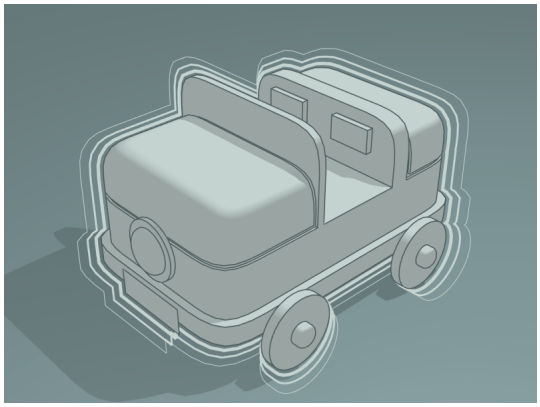
  Stroke Point Parameter:: The *Stroke Point Parameter* factor controls where along the stroke the pivot point is (start point if set to 0.0; end point if set to 1.0).

  Absolute 2D Point:: The *Pivot X* and *Y* values define the position of the pivot point in the final render (from the bottom left corner).

  Important
  Currently, you have to take into account the *real* render size, i.e. resolution **and** resolution percentage.

Scale X, Y
: The scaling factors, in their respective axes.

Rotation Angle
: The rotation angle.



2D Transform modifier (blend-file).

Backbone Stretcher

The *Backbone Stretcher* modifier stretches (adds some length to) the beginning and end of the stroke.

Backbone Length
: Length to add to the strokes' ends.

Bézier Curve

The *Bézier Curve* modifier replaces the stroke by a Bézier approximation of it.

Error
: The maximum distance allowed between the new Bézier curve and the original stroke.



Bézier Curve modifier demo by T.K. (blend-file).

Blueprint

The *Blueprint* modifier produces blueprint-like strokes using either circular, elliptical, or square contours. A blueprint here refers to those lines drawn at the beginning of free-hand drawing to capture the silhouette of objects with a simple shape such as circles, ellipses and squares.

Shape
: Which base shapes to use for this blueprint: *Circles*, *Ellipses* or *Squares*.

Rounds
: How many rounds are generated, as if the pen draws the same stroke several times (i.e. how many times the process is repeated).

Random Radius, Center
: For the *Circles* and *Ellipses* shapes. Adds some randomness to each round in the relevant aspect. Using more than one round with no randomness would be meaningless, as they would draw over each other exactly.

Backbone Length, Backbone
: For the *Squares* shapes. The first adds some extra length to each edge of the generated squares (also affected by the second parameter). The second adds some randomness to the squares.

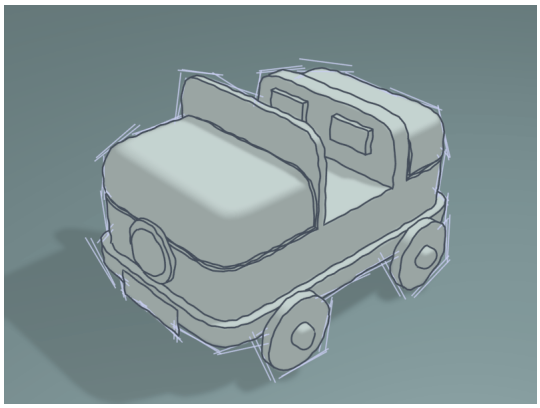
Note that the *Min 2D Length* feature from the *Strokes* settings is quite handy here, to avoid the noise generated by small strokes…

Guiding Lines

The *Guiding Lines* modifier replaces a stroke by a straight line connecting both of its ends.

Offset
: Offset the start and end points along the original stroke, before generating the new straight one.

This modifier will produce reasonable results when strokes are short enough, because shorter strokes are more likely to be well approximated by straight lines. Therefore, it is recommended to use this modifier together with one of the splitting options (by 2D angle or by 2D length) from the *Strokes* panel.



Guiding Lines modifier Demo by T.K. (blend-file).

Perlin Noise 1D

The *Perlin Noise 1D* modifier adds one-dimensional Perlin noise to the stroke. The curvilinear abscissa (value between 0 and 1 determined by a point's position relative to the first and last point of a stroke) is used as the input to the noise function to generate noisy displacements.

This means that this modifier will give an identical result for two strokes with the same length and sampling interval.

Frequency
: How dense the noise is (kind of a scale factor along the stroke).

Amplitude
: How much the noise distorts the stroke in the *Angle* direction.

Seed
: The seed of the random generator (the same seed over a stroke will always give the same result).

Octaves
: The "level of detail" of the noise.

Angle
: In which direction the noise is applied (0.0 is fully horizontal).

Perlin Noise 2D

The *Perlin Noise 2D* modifier adds one-dimensional Perlin noise to the stroke. The modifier generates noisy displacements using 2D coordinates of stroke vertices as the input of the noise generator.

Frequency
: How dense the noise is (kind of a scale factor along the stroke).

Amplitude
: How much the noise distorts the stroke in the *Angle* direction.

Seed
: The seed of the random generator (the same seed over a stroke will always give the same result).

Octaves
: The “level of detail” of the noise.

Angle
: In which direction the noise is applied (0.0 is fully horizontal).

Polygonization

The *Polygonization* modifier simplifies strokes as much as possible (in other words, it transforms smooth strokes into jagged polylines).

Error
: The maximum distance allowed between the new simplified stroke and the original one (the larger this value is, the more jagged/approximated the resulting polylines are).

Sampling

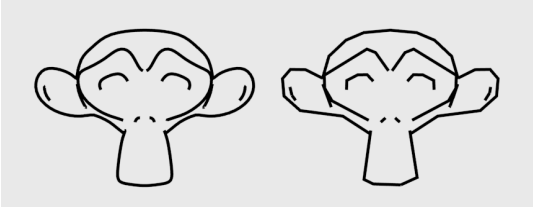
The *Sampling* modifier changes the definition, precision of the stroke, for the following modifiers.

Sampling
: The smaller this value, the more precise are the strokes. Be careful; too small values will require a huge amount of time and memory during render!

Simplification

The *Simplification* modifier merges stroke vertices that lie close to one another, like the *Decimate* modifier for meshes.

Tolerance
: Measure for how close points have to be to each other to be merged. A higher tolerance means more vertices are merged.



Sinus Displacement

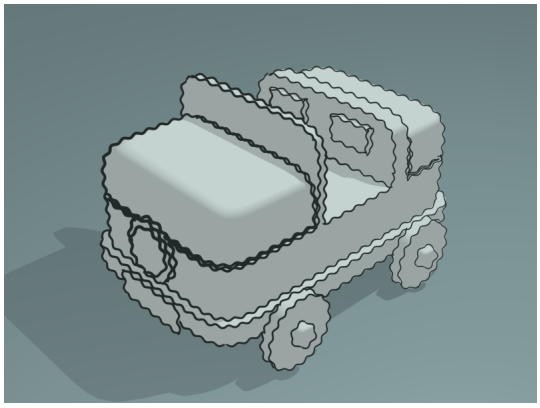
The *Sinus Displacement* modifier adds a sinusoidal displacement to the stroke.

Wavelength
: How wide the undulations are along the stroke.

Amplitude
: How high the undulations are across the stroke.

Phase
: Allows “offsetting” (“moving”) the undulations along the stroke.

> **Tip**
>
> The undulations this modifier produces look exactly the same at a Phase of `0` and any positive or negative multiple of the `Wavelength` set on the modifier. This can be used for rendering short video sequences with wavy lines that can then be seamlessly looped without any visual jumps in the undulations along the line.



Sinus Displacement modifier demo by T.K. (blend-file).

Spatial Noise

The *Spatial Noise* modifier adds some spatial noise to the stroke. Spatial noise displacements are added in the normal direction (i.e. the direction perpendicular to the tangent line) evaluated at each stroke vertex.

Amplitude
: How much the noise distorts the stroke.

Scale
: How wide the noise is along the stroke.

Octaves
: The level of detail of the noise.

Smooth
: When enabled, apply some smoothing over the generated noise.

Pure Random
: When disabled, the next generated random value depends on the previous one; otherwise they are completely independent. Disabling this setting gives a more “consistent” noise along a stroke.

Tip Remover

The *Tip Remover* modifier removes a piece of the stroke at its beginning and end.

Tip Length
: Length of stroke to remove at both of its tips.

Material Properties

Reference

Panel:: *Properties* ▸ *Material* ▸ *Freestyle Line*

Line Color
: Specifies the line colors on a per-material basis.

Priority
: Specify the ordering of competing line colors at material boundaries.

See also

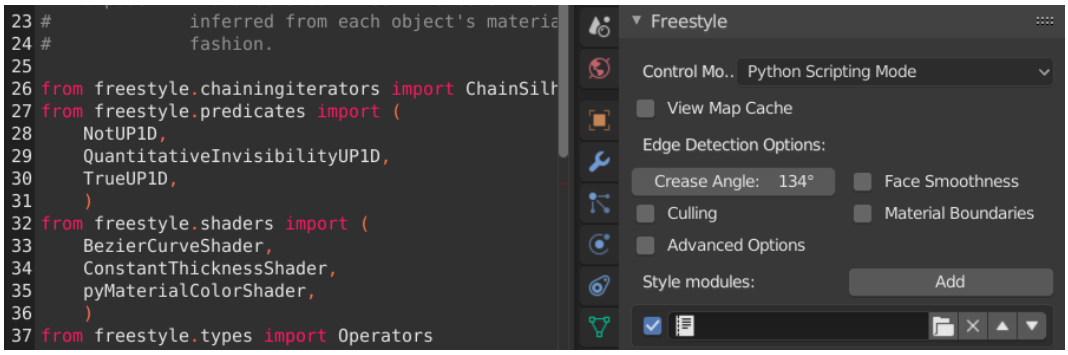
A use case of the line color priority is detailed in a Freestyle development blog article.

Python Scripting

The Python Scripting mode offers full programmable line stylizes. In this control mode, all styling operations are written as Python scripts referred to as style modules in the Freestyle terminology. The input to a style module is a view map (i.e. a set of detected feature edges), and the output is a set of stylized strokes.

A style module is composed of successive calls of five basic operators: selection, chaining, splitting, sorting and stroke creation. The selection operator identifies a subset of input feature edges based on one or more user-defined selection conditions (predicates). The selected edges are processed with the chaining, splitting and sorting operators to build chains of feature edges. These operators are also controlled by user-supplied predicates and functions in order to determine how to transform the feature edges into chains. Finally, the chains are transformed into stylized strokes by the stroke creation operator, which takes a list of user-defined stroke shaders.

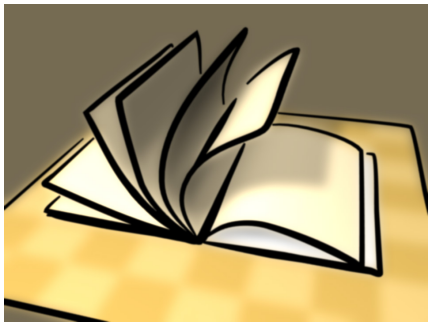
Python style modules are stored within blend-files as text data-blocks. External style module files first need to be loaded in the Text Editor. Then the select menu within an entry of the style module stack allows you to select a module from the list of loaded style modules.



A screen capture of a style module `cartoon.py` loaded in the Text Editor (left), as well as Freestyle options in the Python Scripting

mode in the View Layers buttons (right).

Freestyle for Blender comes with a number of Python style modules that can serve as a starting point of your own style module writing. See also the section of the Freestyle Python API in the Blender Python API reference manual for the full detail of style module constructs.



By T.K. using the Python Scripting mode (blend-file, CC0).



By T.K. using the Python Scripting mode (blend-file, CC0).

Writing Style Modules

A style module is a piece of code responsible for the stylization of Freestyle line drawing. The input of a style module is a set of feature edges called view map (ViewMap). The output is a set of stylized lines also referred to as strokes. A style module is structured as a pipeline of operations that allow for building strokes from the input edges within the view map.

There are five kinds of operations (listed with corresponding operator functions):

- Selection `Operators.select()`
- Chaining `Operators.chain()`, `Operators.bidirectional_chain()`
- Splitting `Operators.sequential_split()`, `Operators.recursive_split()`
- Sorting `Operators.sort()`

- Stroke creation `Operators.create()`

The input view map is populated with a set of ViewEdge objects. The selection operation is used to pick up ViewEdges of interest to artists based on user-defined selection conditions (predicates). Chaining operations take the subset of ViewEdges and build Chains by concatenating ViewEdges according to user-defined predicates and functions. The Chains can be further refined by splitting them into smaller pieces (e.g. at points where edges make an acute turn) and selecting a fraction of them (e.g. to keep only those longer than a length threshold). The sorting operation is used to arrange the stacking order of chains to draw one line on top of another. The chains are finally transformed into stylized strokes by the stroke creation operation applying a series of stroke shaders to individual chains.

ViewEdges, Chains and Strokes are generically referred to as one-dimensional (1D) elements. A 1D element is a polyline that is a series of connected straight lines. Vertices of 1D elements are called 0D elements in general.

All the operators act on a set of active 1D elements. The initial active set is the set of ViewEdges in the input view map. The active set is updated by the operators.

Selection

The selection operator goes through every element of the active set and keeps only the ones satisfying a certain predicate. The `Operators.select()` method takes as the argument a unary predicate that works on any `Interface1D` that represents a 1D element. For example:

```
Operators.select(QuantitativeInvisibilityUP1D(0))
```

This selection operation uses the `QuantitativeInvisibilityUP1D` predicate to select only the visible `ViewEdge` (more precisely, those whose quantitative invisibility is equal to 0). The selection operator is intended to selectively apply the style to a fraction of the active 1D elements.

It is noted that `QuantitativeInvisibilityUP1D` is a class implementing the predicate that tests line visibility, and the `Operators.select()` method takes an instance of the predicate class as argument. The testing of the predicate for a given 1D element is actually done by calling the predicate instance, that is, by invoking the `__call__` method of the predicate class. In other words, the `Operators.select()` method takes as argument a functor which in turn takes an `Interface0D` object as argument. The Freestyle Python API employs functors extensively to implement predicates, as well as functions.

Chaining

The chaining operators act on the set of active `ViewEdge` objects and determine the topology of the future strokes. The idea is to implement an iterator to traverse the ViewMap graph by marching along ViewEdges. The iterator defines a chaining rule that determines the next `ViewEdge` to follow at a given vertex (see `ViewEdgeIterator`). Several such iterators are provided as part of the Freestyle Python API (see `ChainPredicateIterator` and `ChainSilhouetteIterator`). Custom iterators can be defined by inheriting the `ViewEdgeIterator` class. The chaining operator also takes as argument a UnaryPredicate working on `Interface1D` as a stopping criteria. The chaining stops when the iterator has reached a `ViewEdge` satisfying this predicate during the march along the graph.

Chaining can be either unidirectional `Operators.chain()` or bidirectional `Operators.bidirectional_chain()`. In the latter case, the chaining will propagate in the two directions from the starting edge.

The following is a code example of bidirectional chaining:

```
Operators.bidirectional_chain(
        ChainSilhouetteIterator(),
        NotUP1D(QuantitativeInvisibilityUP1D(0)),
        )
```

The chaining operator uses the `ChainSilhouetteIterator` as the chaining rule and stops chaining as soon as the iterator has come to an invisible `ViewEdge`.

The chaining operators process the set of active `ViewEdge` objects in order. The active ViewEdges can be previously sorted using the `Operators.sort()` method (see below). It starts a chain with the first `ViewEdge` of the active set. All ViewEdges that have already been involved in the chaining process are marked (in the case of the example above, the time stamp of each `ViewEdge` is modified by default), in order not to process the same `ViewEdge` twice. Once the chaining reaches a `ViewEdge` that satisfies the stopping predicate, the chain is terminated. Then a new chain is started from the first unmarked `ViewEdge` in the active set. This operation is repeated until the last unmarked `ViewEdge` of the active set was processed. At the end of the chaining operation, the active set is set to the Chains that have just been constructed.

Splitting

The splitting operation is used to refine the topology of each Chain. Splitting is performed either sequentially or recursively. Sequential splitting `Operators.sequentialSplit()` in its basic form, parses the Chain at a given arbitrary resolution and evaluates a unary predicate (working on 0D elements) at each point along the Chain. Every time the predicate is satisfied, the chain is split into two chains. At the end of the sequential split operation, the active set of chains is set to the new chains.

```
Operators.sequentialSplit(TrueUP0D(), 2)
```

In this example, the chain is split every 2 units. A more elaborated version uses two predicates instead of one: One to determine the starting point of the new chain and the other to determine its ending point. This second version can lead to a set of Chains that are disjoint or that overlap if the two predicates are different (see `Operators.sequentialSplit()` for more details).

Recursive splitting `Operators.recursiveSplit()` evaluates a function on the 0D elements along the Chain at a given resolution and find the point that gives the maximum value for the function. The Chain is then split into two at that point. This process is recursively repeated on each of the two new Chains, until the input Chain satisfies a user-specified stopping condition.

```
func = Curvature2DAngleF0D()
Operators.recursive_split(func, NotUP1D(HigherLengthUP1D(5)),
5)
```

In the code example above, the Chains are recursively split at points of the highest 2D curvature. The curvature is evaluated at points along the Chain at a resolution of 5 units. Chains shorter than 5 units will not be split anymore.

Sorting

The sorting operator `Operators.sort()` arranges the stacking order of active 1D elements. It takes as argument a binary predicate used as a "smaller than" operator to order two 1D elements.

```
Operators.sort(Length2DBP1D())
```

In this code example, the sorting uses the `Length2DBP1D` binary predicate to sort the `Interface1D` objects in the ascending order in terms of 2D length.

The sorting is particularly useful when combined with causal density. Indeed, the causal density evaluates the density of the resulting image as it is modified. If we wish to use such a tool to decide to remove strokes whenever the local density is too high, it is important to control the order in which the strokes are drawn. In this case, we would use the sorting operator to ensure that the most "important" lines are drawn first.

Stroke Creation

Finally, the stroke creation operator `Operators.create()` takes the active set of Chains as input and build Strokes. The operator takes two arguments. The first is a unary predicate that works on `Interface1D` that is designed to make a last selection on the set of chains. A Chain that does not satisfy the condition will not lead to a Stroke. The second input is a list of shaders that will be responsible for the shading of each built stroke.

```
shaders_list = [
    SamplingShader(5.0),
```

```
    ConstantThicknessShader(2),
    ConstantColorShader(0.2,0.2,0.2,1),
    ]
Operators.create(DensityUP1D(8,0.1, IntegrationType.MEAN),
shaders_list)
```

In this example, the `DensityUP1D` predicate is used to remove all Chains whose mean density is higher than 0.1. Each chain is transformed into a stroke by resampling it so as to have a point every 5 units and assigning to it a constant thickness of 2 units and a dark gray constant color.

User Control on the Pipeline Definition

Style module writing offers different types of user control, even though individual style modules have a fixed pipeline structure. One is the sequencing of different pipeline control structures, and another is through the definition of functor objects that are passed as argument all along the pipeline.

Different pipeline control structures can be defined by sequencing the selection, chaining, splitting, and sorting operations. The stroke creation is always the last operation that concludes a style module.

Predicates, functions, chaining iterators, and stroke shaders can be defined by inheriting base classes and overriding appropriate methods. See the reference manual entries of the following base classes for more information on the user-scriptable constructs.

See also

Predicates, functions, chaining iterators, and stroke shaders can be defined by inheriting base classes and overriding appropriate methods. See **Freestyle python module** for more information on the user-scriptable constructs.

Layers & Passes

- Introduction
  - View Layers
  - Usage
  - Collections
  - Cycles
- View Layer
- Passes
  - Data
  - Light
  - Cryptomatte
  - Shader AOV
  - Light Groups
  - Combining
  - Known Limitations

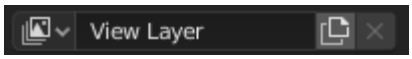
Introduction

Renders can be separated into layers, to composite them back together afterwards.

Some example usages are applying compositing effects to characters separately, blurring the background and foreground layers separately for depth of field, or rendering different lighting variations of the same scene.

Using View Layers can also save you from having to re-render your entire image after each change, allowing you to instead re-render only the layer(s) that you have altered.

View Layers



View Layers.

In the top of the screen there is a list of all the View Layers in the active scene.

Name
: The name of the active view layer, click to edit the name.

Add View Layer
: Will add a new view layer to the active scene.

  New
  : Adds a new view layer.

  Copy Settings
  : Adds a new view layer with all the settings of current view layer.

  Blank

Adds a new view layer with all collections disabled.

Remove View Layer

Will remove the selected view layer from the active scene.

Note

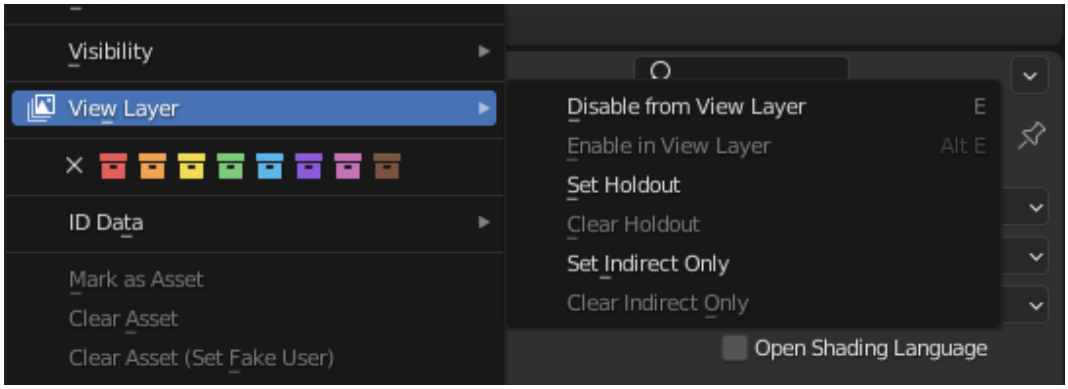
A scene must have at least one view layer.

Usage

Each Scene has an associated set of Collections. The visibility settings of each collection can be changed per View Layer to separate the rendering of different objects and lights into layers.

Collections

Per collection you can adjust the way how the render engine needs to render the objects inside. Based on the render engine different options can be set.



Collection/View layer settings.

Disable from View Layer

Remove this collection from the active view layer. Objects that are only in this collection will not be rendered for the active view layer. This is

useful to sometimes leave out some object influence for a particular view layer.

Enable in View Layer
: Add this collection to the active view layer. Objects inside the collection will be rendered with the active view layer.

Set Holdout
: Objects inside this collection will generate a holdout/mask in the active view layer.

Clear Holdout
: Clear the Set Holdout flag.

Set Indirect Only *Cycles Only*
: Objects inside this collection will only contribute to the final image indirectly through shadows and reflections.

Clear Indirect Only *Cycles Only*
: Clear the *Indirect Only* flag. Objects inside this collection will contribute normally to the final image.

Cycles

Reference

Panel:: *View Layers ▸ Layer*

This section covers only the Render Layer settings appropriate for the Cycles renderer. For the engine-independent settings, see this section.

Filter

Include
: Environment
  : Disables rendering the *Environment* render pass in the final render.

  Surfaces

Disables rendering object materials in the final render.

Curves
: Disables rendering curve strands in the final render.

Volume
: Disables rendering Volumes in the final render.

Use

Motion Blur
: Render motion blur for this Layer, if enabled in the Render Settings.

Override

Material Override
: Overrides all materials in the render layer.

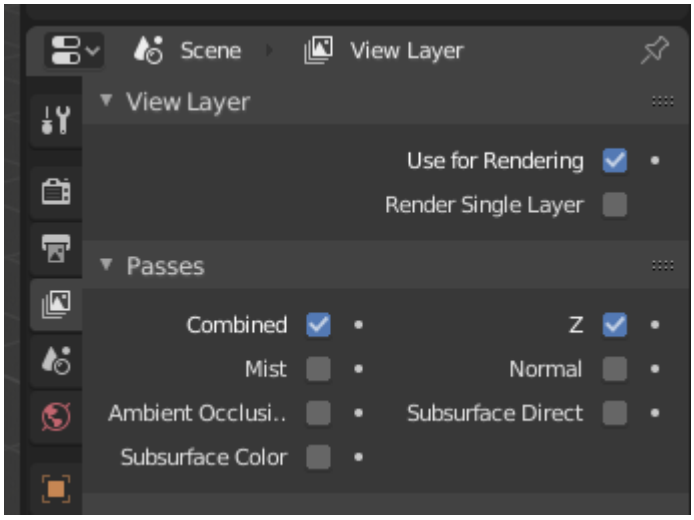
World Override
: Overrides world background in the render layer.

Samples
: View layer samples to override the scene samples. Controlled by the layer samples in the Sampling panel.

View Layer

Reference

Panel:: *Properties ▸ Scene ▸ View Layer*



View Layer panel (shown here for the EEVEE render engine).

The Layer Panel shows the settings of the active View Layer.

Use for Rendering
: The active view layer will be used during rendering.

Render Single Layer
: Only render the active view layer.

Note

This option is ignored when rendering from the command line.

See also

Additional options shown in this panel are different for each render engine. See Render Passes for the options per render engine.

Passes

Reference

Panel:: *Scene ▸ View Layers ▸ Passes*

Passes can be used to split rendered images into colors, direct and indirect light to edit them individually, and also to extract data such as depth or normals.

Data

Cycles

Include

Combined

The final combination of render passes with everything included.

Z

Distance to any visible surfaces.

> **Note**
>
> The Z pass only uses one sample. When depth values need to be blended in case of motion blur or Depth of Field, use the mist pass.

Mist

Distance to visible surfaces, mapped to the 0.0 - 1.0 range. When enabled, settings are in World tab. This pass can be used in compositing to fade out objects that are farther away.

Position

World position of objects in the scene.

Normal
: Surface normal used for shading.

Vector
: Motion vectors for the Vector Blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

UV
: Mapped UV coordinates, used to represent where on a mesh a texture gets mapped too. This is represented through the red and green channels of the image. The blue channel is encoded with a constant value of 1 but does not hold any information.

Denoising Data
: Includes *Denoising Albedo*, *Denoising Normal*, and a render pass of the original combined pass before denoising.

Note

The Z, Position, Object Index, and Material Index passes are not anti-aliased.

Indexes

Object Index
: Creates a mask of the object that can be later read by the ID Mask Node in the Compositor.

Material Index
: Creates a mask of the material that can be later read by the ID Mask Node in the Compositor.

Debug

Sample Count

Number of samples per pixel taken, divided by the maximum number of samples. To analyze adaptive sampling.

Alpha Threshold

Z, Index, normal, UV and vector passes are only affected by surfaces with alpha transparency equal to or higher than this threshold. With value 0.0 the first surface hit will always write to these passes, regardless of transparency. With higher values surfaces that are mostly transparent can be skipped until an opaque surface is encountered.

EEVEE

Include

Combined

The final combination of render passes with everything included.

Z

Distance to any visible surfaces.

Mist

Distance to visible surfaces, mapped to the 0.0 - 1.0 range.

Normal

Surface normal used for shading.

Position

World position of objects in the scene.

Vector

Motion vectors for the Vector Blur node. The four components consist of 2D vectors giving the motion towards the next and previous frame position in pixel space.

Light

Cycles

Diffuse

Direct

Direct lighting from diffuse and subsurface BSDFs. We define direct lighting as coming from lights, emitting surfaces, the background, or ambient occlusion after a single reflection or transmission off a surface. BSDF color is not included in this pass.

Indirect

Indirect lighting from diffuse and subsurface BSDFs. We define indirect lighting as coming from lights, emitting surfaces or the background after more than one reflection or transmission off a surface. BSDF color is not included in this pass.

Color

Color weights of diffuse and subsurface BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Glossy

Direct, Indirect, Color

Same as above, but for glossy BSDFs.

Transmission

Direct, Indirect, Color

Same as above, but for transmission BSDFs.

Volume

Direct, Indirect

Same as above, but for volumetric BSDFs.

Other

Emission

Emission from directly visible surfaces.

Environment

Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

Ambient Occlusion
: Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a ‘normalized’ value between 0 and 1.

Shadow Catcher
: Extra indirect light information collected by objects with the Shadow Catcher option enabled. Multiply this pass with existing footage using the Mix Node in the Compositor to add the indirect lighting information to the footage.

Note

Transparent BSDFs are given special treatment. A fully transparent surface is treated as if there is no surface there at all; a partially transparent surface is treated as if only part of the light rays can pass through. This means it is not included in the Transmission passes; for that a glass BSDF with index of refraction 1.0 can be used.

EEVEE

Diffuse

Light
: Direct lighting from diffuse BSDFs. We define lighting as coming from lights, the background, or ambient occlusion off a surface. BSDF color is not included in this pass.

Color
: Color weights of diffuse BSDFs. These weights are the color input socket for BSDF nodes, modified by any Mix and Add Shader nodes.

Specular

Light, Color
: Same as above, but for specular BSDFs.

Volume

Light

The scattering pass from volume objects or world.

Other

Emission

Emission from directly visible surfaces.

Environment

Emission from the directly visible background. When the film is set to transparent, this can be used to get the environment color and composite it back in.

Shadow

Shadows from light objects. Mostly useful for compositing objects with shadow into existing footage.

Ambient Occlusion

Ambient occlusion from directly visible surfaces. BSDF color or AO factor is not included; i.e. it gives a 'normalized' value between 0 and 1.

Transparent

Contain Blended surfaces, so they can be adjusted in the compositor and later mixed with opaque passes.

This pass only supports monochromatic opacity. Colored opacity will show differently than in combined pass.

Occlusions Distance

Distance of object that contributes to the ambient occlusion effect.

Cryptomatte

Cryptomatte is a standard to efficiently create mattes for compositing. Cycles outputs the required render passes, which can then be used in the

Blender Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing. The mattes will be anti-aliased and take into account effects like motion blur and transparency.

Object
: Render cryptomatte object pass, for isolating objects in compositing.

Material
: Render cryptomatte material pass, for isolating materials in compositing.

Asset
: Render cryptomatte asset pass, for isolating groups of objects with the same parent in compositing.

Levels
: Sets how many unique objects can be distinguished per pixel.

Typical Workflow

1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and link the Render Layer matching Image and Cryptomatte passes to it.
3. Attach a Viewer node to the Pick output of the Cryptomatte node.
4. Use the Cryptomatte Add/Remove button to sample objects in the Pick Viewer node.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

See also

Cryptomatte Node.

Shader AOV

Shader AOVs (Arbitrary Output Variables) provide custom render passes for any shader node components. As an artist this can be a good way to fix or tweak fine details of a scene in post-processing. To use Shader AOVs create the pass in the *Shader AOV* panel then reference this pass with the AOV Output shading node. Shader AOVs can be added or removed in the *Shader AOV* panel. In this panel is a list of all AOV passes; each AOV in the list consists of a *Name* and *Data Type*.

Active AOV Index
: The name of the render pass; this is the *Name* that is referenced in the *AOV Output* node. Any names can be used for these passes, as long as they do not conflict with built-in passes that are enabled.

Data Type
: Shader AOVs can either express a *Color* or a *Value* output. The *Color* type as the name suggest can be used for a color but also for normals. A *Value* type can be used for any single numerical value.

Light Groups

Cycles only

Light Groups provide render passes that only contains information from the lights within that group. Light Groups can be used to easily tweak the lighting color and intensity of specific lights without having to re-render the scene.

Active Light Group Index
: The name of the render pass. This is the name that is used when adding World Shaders, Lights and Objects to *Light Groups*.

Lightgroup Sync

These operators are available in the menu to the right of the Light Group list.

Add Used Lightgroups

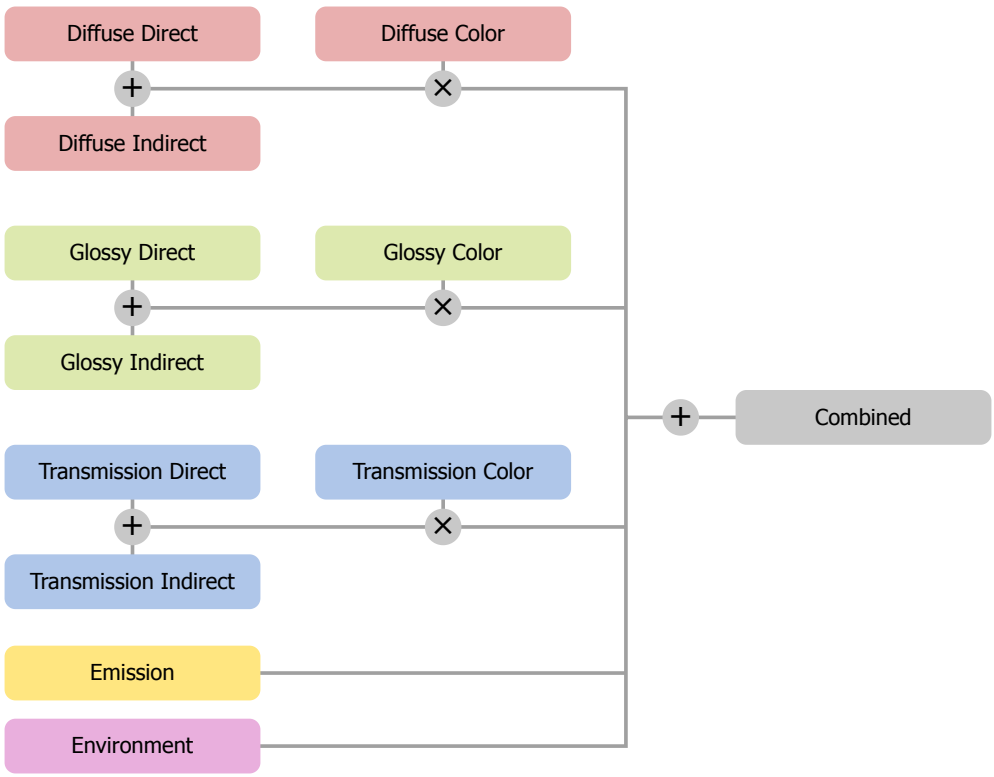
Adds all Light Groups to the View Layer that have been created elsewhere and have lights assigned to them.

Remove Unused Lightgroups
: Deletes all Light Groups that do not have any lights assigned to them.

Combining

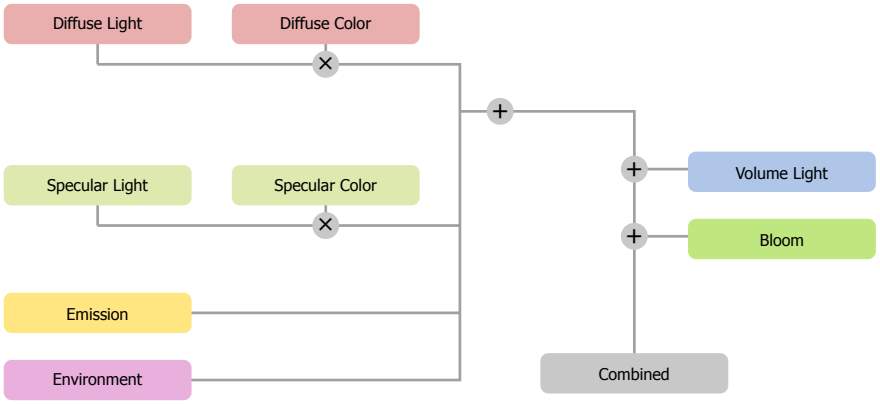
Cycles

All these lighting passes can be combined to produce the final image as follows:



EEVEE

The passes can be combined to produce the final image as follows:



Known Limitations

- Blended materials are not rendered in render passes except the combined pass. Use the *Dithered* as Render Method to render transparent materials in render passes.
- Depth of field is not rendered in render passes except the combined pass. It is possible to add the depth of field back in the Compositor using the Defocus node.
- EEVEE render passes exclude parts of the BSDF equation. Shader to RGB is not supported as it needs the full BSDF equation.
- EEVEE supports upto 16 color and 16 value AOV render passes.

Render Output

- Introduction
- Output Properties
  - Format
  - Frame Range
  - Stereoscopy
  - Output
  - Metadata
  - Post Processing
- Audio Rendering
  - Introduction
  - Speaker
- Rendering Animations
  - Workflow
  - Frame Sequence Workflow
  - Hints
- Animation Player
  - Player Options
  - Hotkeys
  - Frame Cache

Introduction

The first step in the rendering process is to determine and set the output settings. This includes render size, frame rate, pixel aspect ratio, output location, and file type.

When rendering a single frame, the output should be a single image format and not a video. Several image formats are available.

Images can also be used for rendering animations which has a couple advantages. For example, when rendering animations to image file formats the render job can be canceled and resumed at the last rendered frame by changing the frame range. This is useful if the animation takes a long time to render and the computers resources are needed for something else.

Images can then be encoded to a video by adding the rendered image sequence into the Video Sequencer and choosing an appropriate Video Output.

Tip

Rendered image sequences can be played back in the Animation Player.

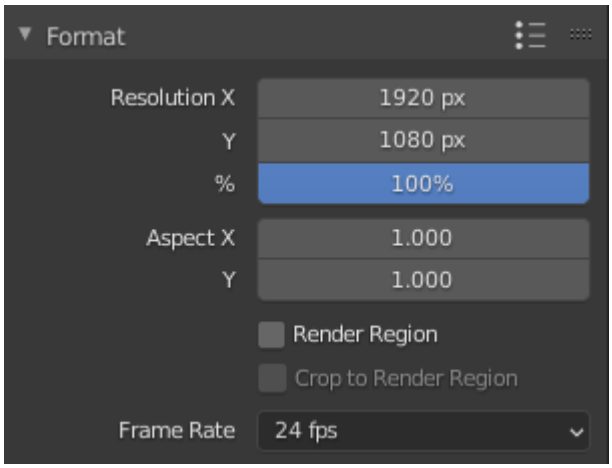
See also

See Render Menu

Output Properties

- Format
- Frame Range
  - Time Stretching
- Stereoscopy
  - Introduction
  - Usage
- Output
  - Color Management
  - Encoding
- Metadata
  - Note
  - Burn into Image
- Post Processing

Format



Format panel.

Several render presets exist with common resolution and frame rates for TVs and screens can be selected in the panel header.

Resolution X, Y
: The number of pixels horizontally and vertically in the image.

Percentage
: Slider to reduce or increase the size of the rendered image relative to the *Resolution* values. This is useful for small test renders that have the same proportions as the final image.

Aspect X, Y
: Older televisions may have non-square pixels, so this can be used to control the shape of the pixels along the respective axis. This will *pre-distort* the images which will look stretched on a computer screen, but which will display correctly on a TV set. It is important that you use the correct pixel aspect ratio when rendering to prevent re-scaling, resulting in lowered image quality.

Render Region

Renders just a portion of the view instead of the entire frame. See the Render Region documentation to see how to define the size of the render region.

Crop to Render Region

Crops the rendered image to the size of the render region, instead of rendering a transparent background around it.

Frame Rate

The number of frames that are displayed per second, relevant for Animation. The menu gives several common frame rates, custom frame rates can be used by selecting *Custom* which gives access to the following properties:

FPS

The frame rate, expressed in frames per second.

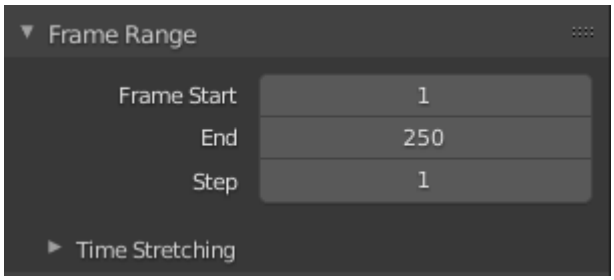
Base

Some standards require a more precise frame rate, for example NTSC. These can be represent as a fraction where the *Base* value is used as the fraction's denominator and the FPS being the numerator: \(\frac{FPS}{Base}\).

See also

Viewport Playback Frame Rate Limited

Frame Range



Frame Range panel.

This panel defines how long an animation will last in terms of frames. Frames can be divided by the scene's Frame Rate to get the animation duration in terms of time. For example, a 250 frame animation at a frame rate of 30 will last 8.3 seconds.

Frame Start, End
: Set the *Start* and *End* frames for Rendering Animations.

Step
: Controls the number of frames to advance by for each frame in the timeline.

Time Stretching

Use to remap the length of an animation; making it run slower or faster. The *Old* and *New* settings may either be used as absolute values or as a ratio: For example, setting *Old* to a value of 2 and *New* to 1 will run the animation twice as fast.

Warning

Using *Time Stretching* will not influence the *Start* or *End* frames set above, so make sure that your animation is not cut off or has extraneous still frames at the end.

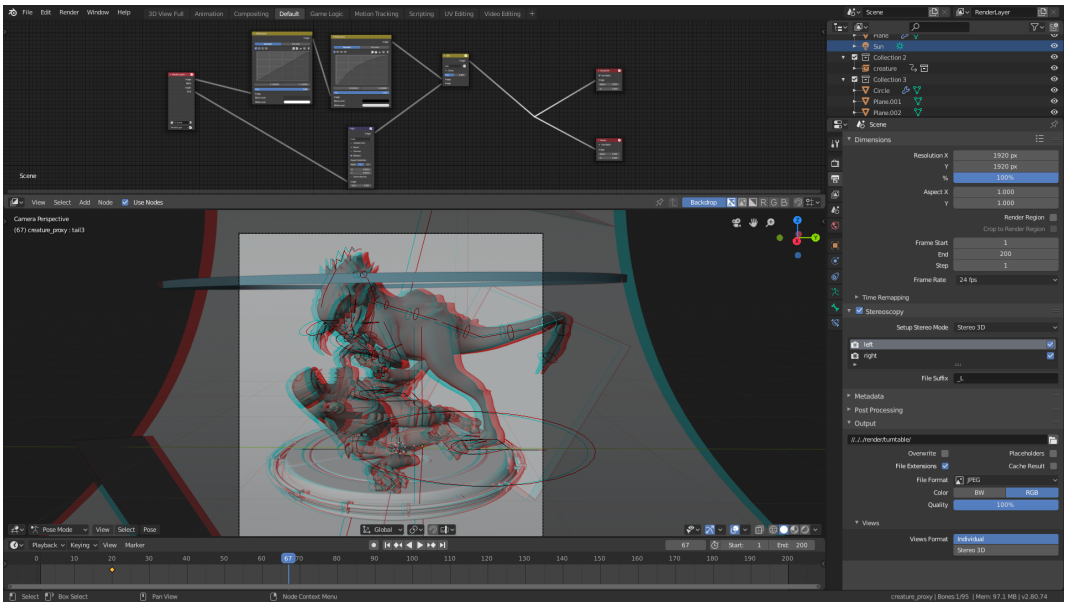
Old
: The length in frames of the original animation.

New
: The length in frames the new animation will last.

Stereoscopy

- Introduction
- Usage
  - Introduction
  - Stereoscopy Setup
  - Camera
  - Viewport
  - Stereo 3D Display
  - Viewport Preview
  - Rendering and Image Editor
  - Image Formats
  - Final Considerations
  - Window Stereo 3D Display
  - Stereo 3D Camera
  - Viewport Stereo 3D
  - Multi-View and Stereo 3D Image I/O
  - Image Editor
  - Compositor

Introduction



Multi-view is a complete toolset for working with stereoscopic rendering in Blender. It works with both the EEVEE and Cycles rendering engines. Cycles additionally supports stereoscopic panoramic cameras. There is support for many different stereo 3D visualization types.

Note

If you have a real 3D display at some point you can change the 3D display mode in the Window menu, by calling the *Stereo 3D* operator. Be aware that some modes require a fullscreen editor to work, and this can be taxing on your CPU.

Usage

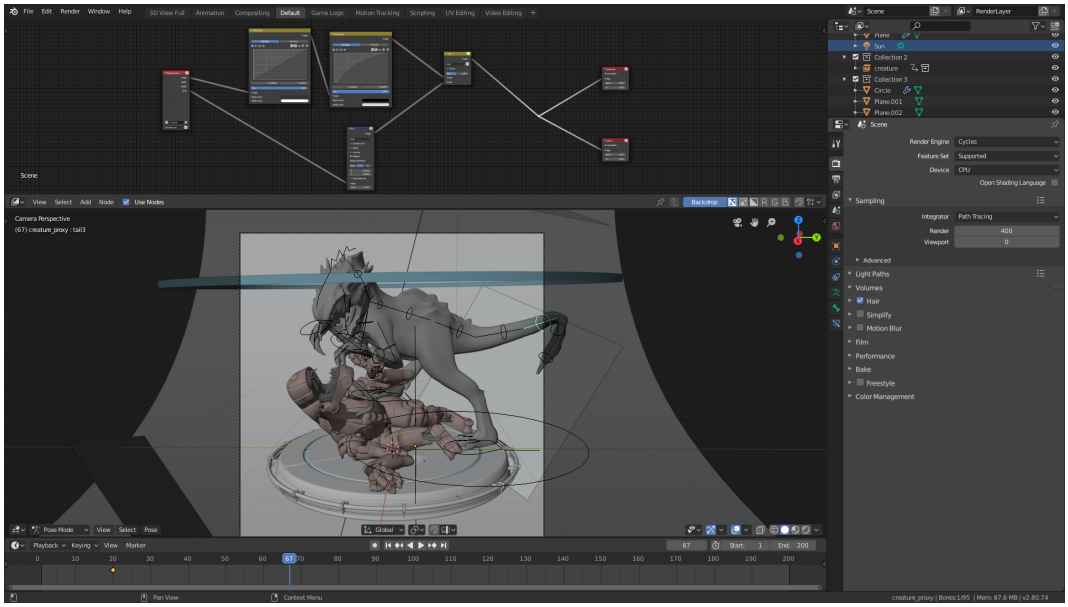
For example, we will take an existing blend-file that was made for monoscopic rendering and transform it to be stereo 3D ready.



Creature Factory 2 by Andy Goralczyk rendered in stereo 3D (anaglyph).

Introduction

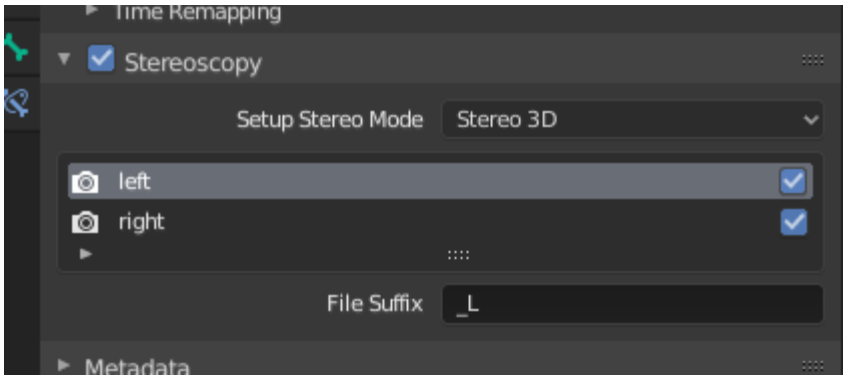
Start opening up your project file, in this case `turntable.blend` from the *Creature Factory 2* Open Movie Workshop series from the Blender Institute by Andy Goralczyk.



Turntable Creature Factory 2.

Stereoscopy Setup

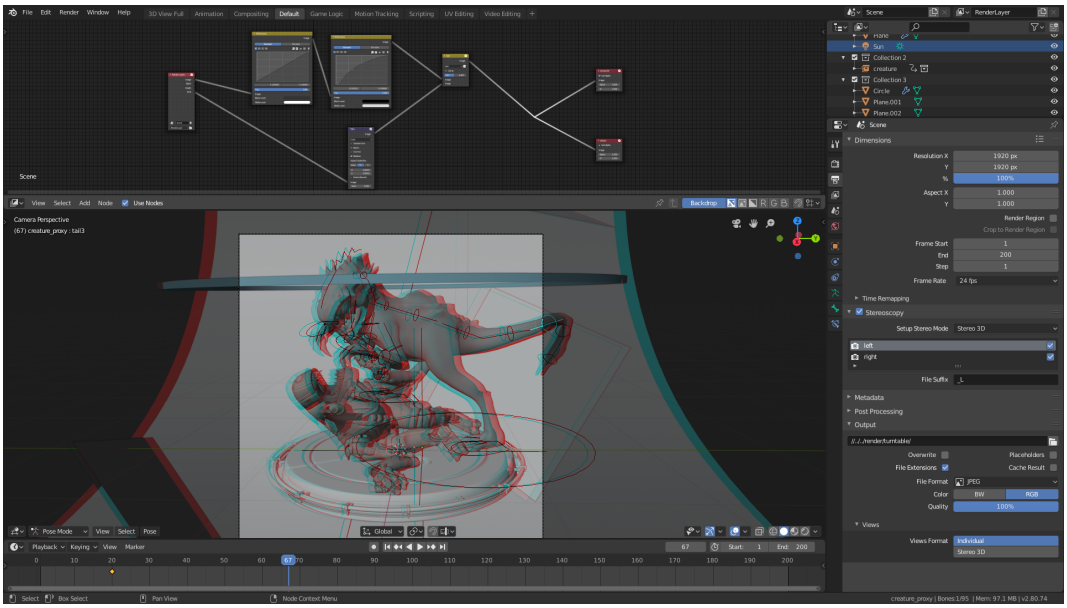
Go to the Output Properties and enable *Stereoscopy* for this scene.



Scene render views.

Note

When you turn on *Stereoscopy* in the scene, you get 3D preview in the viewport, as well as multiple panels that are now accessible all over the user interface.

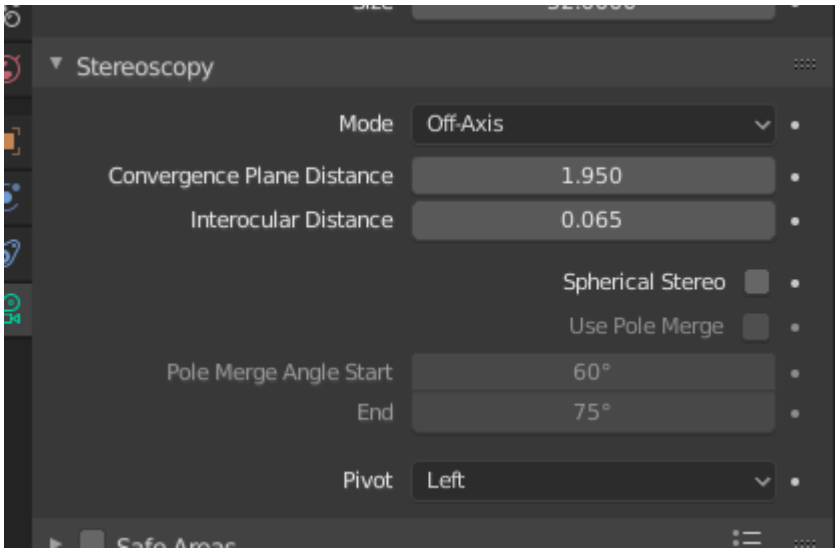


Viewport with 3D visualization.

Camera

To tweak the stereo 3D parameters, select the camera in the Outliner. In the Camera panel go to the Stereoscopy tab and change the *Convergence Distance*.

The viewport will respond in real-time to those changes allowing you to preview the current depth value of the scene.

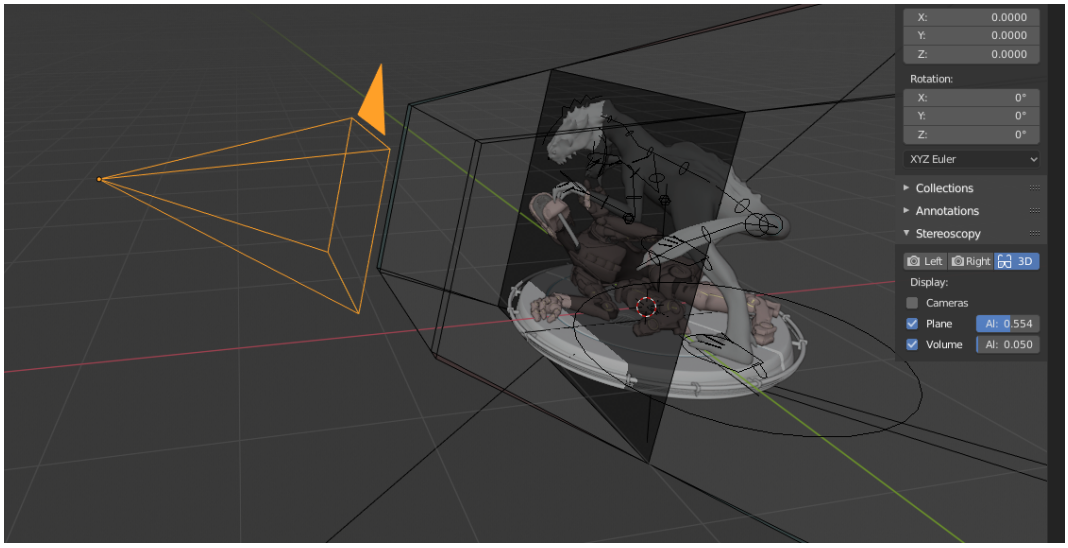


Stereo convergence distance.

Viewport

Before fine-tuning the camera parameters, you can set the convergence plane in the viewport based in your scene depth layout. Go outside the camera view and you will instantly see the convergence plane in front of the camera.

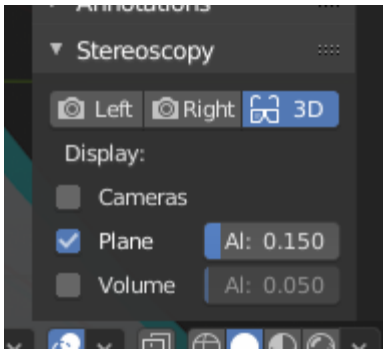
You can toggle this and other display settings in the Stereoscopy panel of the 3D Viewport's Sidebar. In the following image, the camera's frustum volumes are also visible.



Viewport plane and volume stereo preview.

Stereo 3D Display

If you have a real 3D display at some point, you can change the 3D display mode in the Window menu, by calling the Stereo 3D operator. Be aware that some modes require a fullscreen editor to work.

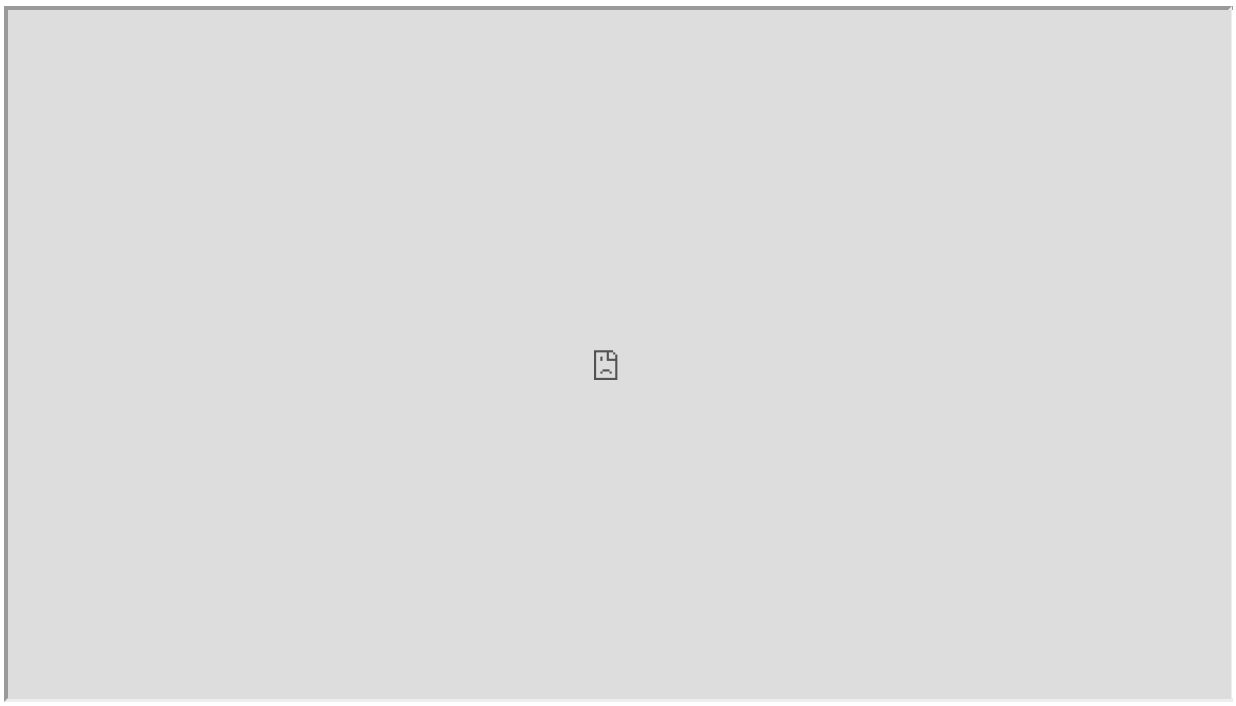


Window menu, stereo 3D operator.

Viewport Preview

Before rendering your scene, you can save a Viewport Preview of the animation for testing in the final display. In the Render Output panel you can choose the output *Views Format*.

The options include individual files per view, top-bottom, anaglyph among others. Pick the one that fits your display requirements.



Rendering and Image Editor

Once you are happy with the results, you can render out the final animation. In the Image Editor you can inspect the individual views and the stereo result.

Image Formats

Your final animation can be saved in more robust formats. In this example we saved as cross-eyed side-by-side stereo 3D.



Side-by-side cross-eye format.

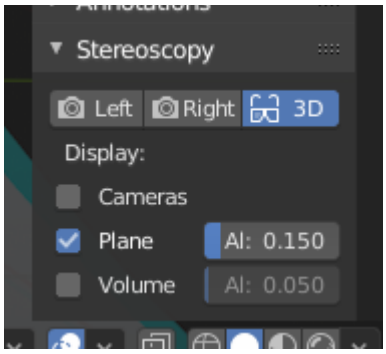
Final Considerations

As this guide showed, there is more to stereo 3D rendering than just generate two images. The earlier the stereo pipeline is considered the smoother it will get. The following sections are a more in-depth view of the individual components we visited in the workflow.

Window Stereo 3D Display

An essential component of the Stereoscopy pipeline is the ability to display the stereo image in a proper display. Blender supports from high-end 3D displays to simple red-cyan glasses. On top of that, you can set a different display mode for each window.

The display mode can be changed via the Window menu or if you create your own shortcuts for the `wm.set_stereo_3d` operator.



Window menu, stereo 3D operator.

Display Mode

Anaglyph
: Render two differently filtered colored images for each eye. Anaglyph glasses are required. We support red-cyan, green-magenta and yellow-blue glasses.

Interlace
: Render two images for each eye into one interlaced image. A 3D-ready monitor is required. We support Row, Column and Checkerboard Interleaved. An option to Swap Left/Right helps to adjust the image for the screen. This method works better in fullscreen.

Time Sequential
: Render alternate eyes. This method is also known as Page Flip. This requires the graphic card to support Quad Buffer and it only works in fullscreen.

Side-by-Side
: Render images for left and right eye side-by-side. There is an option to support Cross-Eye glasses. It works only in fullscreen, and it should be used with the Full Editor operator.

Top-Bottom
: Render images for left and right eye one above another. It works only in fullscreen, and it should be used with the Full Editor operator.

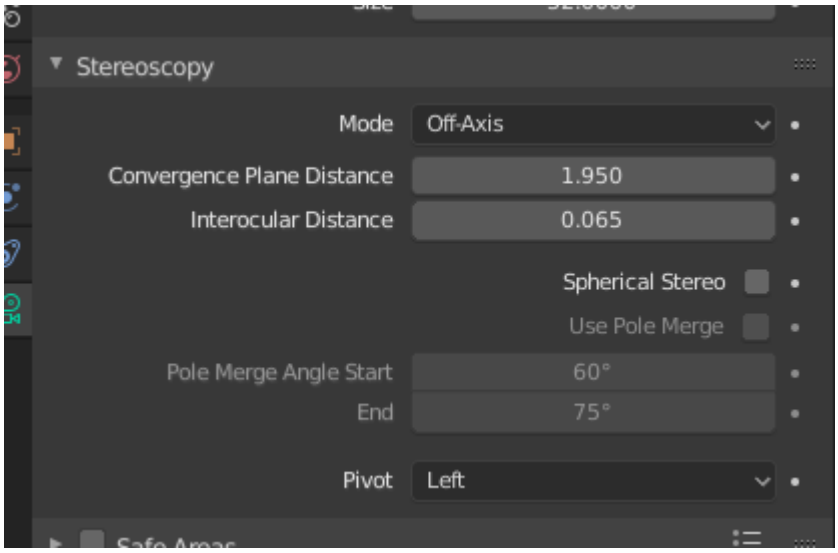
Note

Full Screen Stereo 3D Modes

If you have a 3D display most of the time, you will use it to see in stereo 3D, you will have to go to the fullscreen mode. In fact some modes will only work in the full window mode that hides most of the user interface from the work area. In this case it is recommended to work with two monitors, using the 3D screen for visualizing the stereo result while the other screen can be used for the regular Blender work.

Stereo 3D Camera

When using the Stereo 3D scene view setup, a stereo pair is created on-the-fly and used for rendering and previsualization. For all the purposes this works as two cameras that share most parameters (focal length, clipping, …). The stereo pair, however, is offset, and can have unique rotation and shift between itself.



Stereo 3D camera settings.

Interocular Distance

Set the distance between the camera pair. Although the convergence of a stereo pair can be changed in post-production, different interocular distances will produce different results due to the parts of the scene being occluded from each point of view.

Convergence Plane Distance
: The converge point for the stereo cameras. This is often the distance between a projector and the projection screen. You can visualize this in the 3D Viewport.

Spherical Stereo
: Render every pixel rotating the camera around the middle of the interocular distance.

Use Pole Merge
: Fade interocular distance to 0 after the given cutoff angle.

  Pole Merge Start Angle
  : Angle at which interocular distance starts to fade to 0.

  Pole Merge End Angle
  : Angle at which interocular distance is 0.

Convergence Mode

Off-Axis
: The stereo camera pair is separated by the interocular distance, and shifted inwards so it converges in the convergence plane. This is the ideal format since it is the one closest to how the human vision works.

Parallel
: This method produces two parallel cameras that do not converge. Since this method needs to be manually converged it cannot be used for viewing. This method is common when combining real footage with rendered elements.

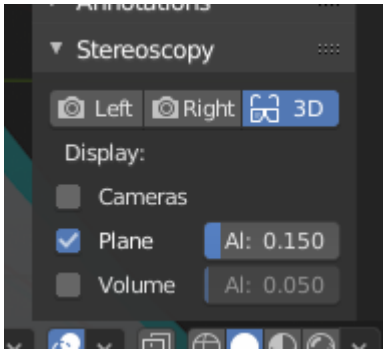
Toe-in

A less common approach is to rotate the cameras instead of shifting their frustum. The Toe-in method is rarely used in modern 3D productions.

Pivot
: The stereo pair can be constructed around the active camera with a new camera built for each eye (Center Pivot) or using the existing camera and creating (Left or Right). The latter is what is used when only one eye needs to be rendered for an existing mono 2D project.

Viewport Stereo 3D

When you enable *Views* in the Render Layer panel, a new area is available in the 3D Viewport Sidebar region. In this panel you can pick whether to see the stereo 3D in the viewport, or which camera to see. It also allow you to see the *Cameras*, the *Plane* and the *Volume* of the stereo cameras.



Viewport stereo 3D settings.

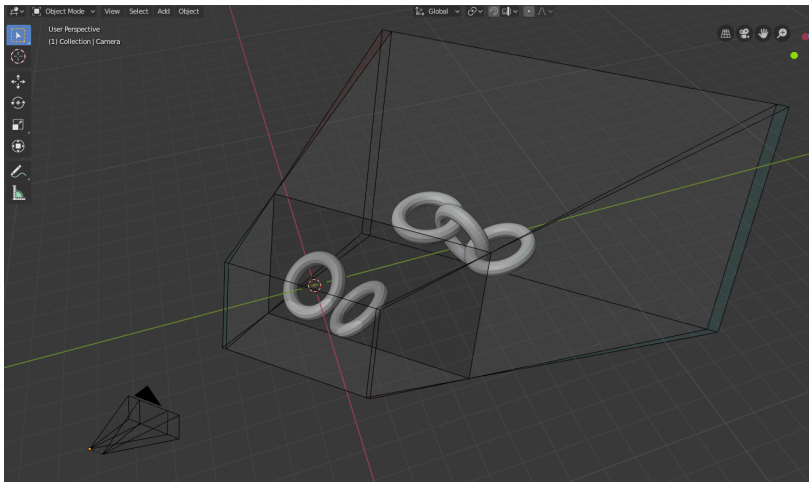
Cameras
: When working with the Stereo 3D Viewports setup, you can inspect what each individual generated camera is looking or the combined result of them. In the Multi-View mode you can see the combined result of the left and right cameras (when available) or the current selected camera.

Plane

The convergence plane represents the screen as it is perceived by the audience. Visualizing it in the 3D Viewport allows you to layout your scene based on your depth script outside the camera view.

Volume

The intersection of the stereo cameras frustums helps planning the show by avoiding elements being visible by only one camera. The volume is defined by the camera's start and end clipping distances. The areas that are in the frustum of one camera only are known as *retinal rivalry areas*. They are tolerated in the negative space (the region from the convergence plane into the image) but are to be avoided at all costs in the positive space (the area from the convergence plane to the camera).



Viewport 3D: convergence plane and volume display.

Multi-View and Stereo 3D Image I/O

Multi-View and Stereo 3D

Multi-view images can be saved in special formats according to the production requirements. By default the system saves each view as an individual file, thus generating as many files as views to be rendered. In stereo 3D productions, for the final deployment or even intermediary previews it is convenient to save stereo 3D images, that are ready to use with 3D displays or simple anaglyph glasses. The formats supported match the display modes available for the window.

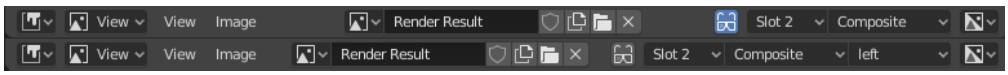
Lossy-Formats
: Some stereo 3D formats represent a considerable loss of data. For example, the Anaglyph format will cap out entire color channels from the original image. The Top-Bottom compressed will discard half of your vertical resolution data. The Interlace will mash your data considerably. Once you export in those formats, you can still import the image back in Blender, for it to be treated as Stereo 3D. You will need to match the window stereo 3D display mode to the image stereo 3D format though.

Lossless Formats
: Some formats will preserve the original data, leading to no problems on exporting and importing the files back in Blender. The Individual option will produce separate images that (if saved in a lossless encoding such as `PNG` or `OpenEXR`) can be loaded back in production with no loss of data. For the Stereo 3D formats the only lossless options are *Top-Bottom* and *Side-by-Side* without the Squeezed Frame option.

Multi-View OpenEXR
: Another option is to use multi-view OpenEXR files. This format can save multiple views in a single file and is backward compatible with old OpenEXR viewers (you see only one view though). Multi-view native support is only available to OpenEXR.

Image Editor

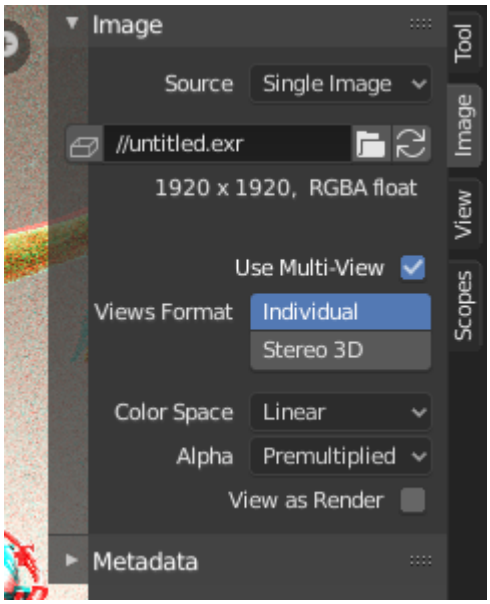
View Menu
: After you render your scene with Stereo 3D you will be able to see the rendered result in the combined stereo 3D or to inspect the individual views. This works for Viewer nodes, render results or opened images.



Stereo 3D and view menu.

Views Format

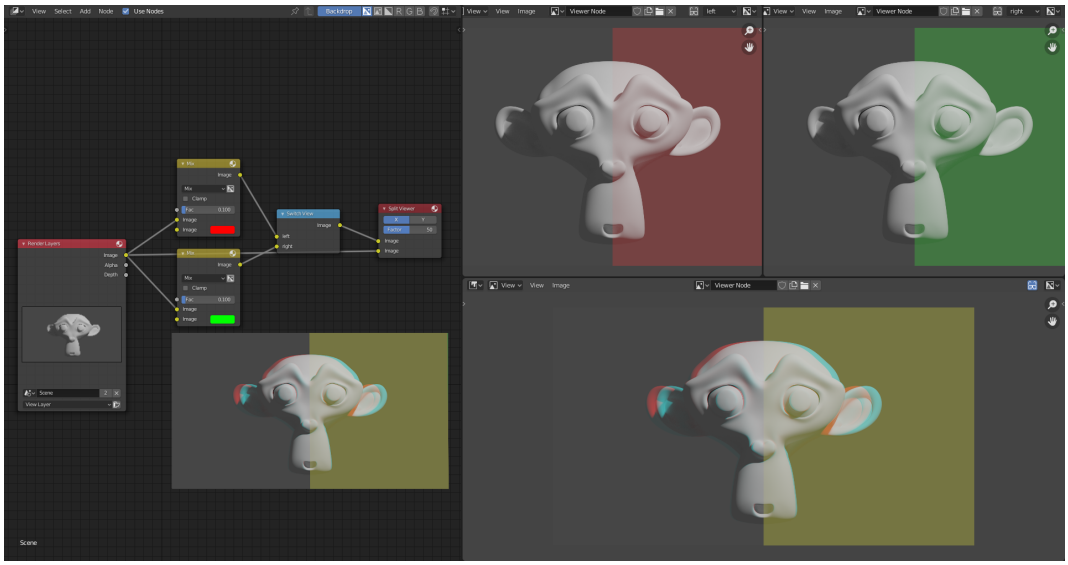
When you drag and drop an image into the Image Editor, Blender will open it as an individual images at first. If your image was saved with one of the Stereo 3D formats, you can change how Blender should interpret the image by switching the mode to Stereo 3D, turning on *Use Multi-View* and picking the corresponding stereo method.



Views formats and stereo 3D.

Compositor

The Compositor works smoothly with multi-view images. The compositing of a view is completed before the remaining views start to be composited. The pipeline is the same as the single-view workflow, with the difference that you can use images, movies or image sequences in any of the supported multi-view formats.



Compositor, backdrop and Split Viewer node.

The views to render are defined in the current scene views, in a similar way as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Note

Single-View Images

If the image from an Image node does not have the view you are trying to render, the image will be treated as a single-view image.

Switch View Node

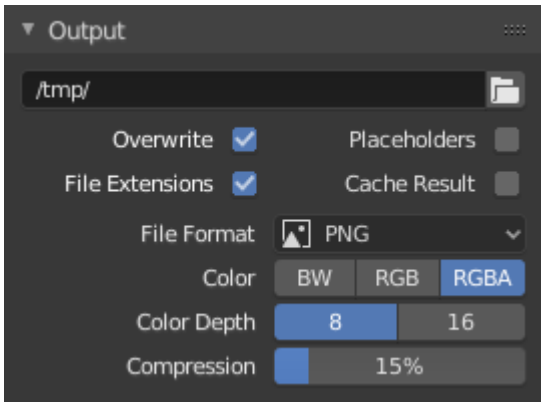
If you need to treat the views separately, you can use the Switch View node to combine the views before an Output node.

Tip

Performance

By default, when compositing and rendering from the user interface all views are rendered and then composited. During test iterations you can disable all but one view from the Scene Views panel, and re-enable it after you get the final look.

Output



Output panel.

This panel provides options for setting the location of rendered frames for animations, and the quality of the saved images.

File Path

Choose the location to save rendered frames.

When rendering an animation, the frame number is appended at the end of the file name with four padded zeros (e.g. `image0001.png`). You can set a custom padding size by adding the appropriate number of `#` anywhere in the file name (e.g. `image_##_test.png` translates to `image_01_test.png`).

This setting expands Relative Paths where a `//` prefix represents the directory of the current blend-file.

Saving

File Extensions

Adds the correct file extensions per file type to the output files.

Cache Result

Saves the rendered view layers and their passes to a multi-layer OpenEXR image. The Compositor can then use this file to improve performance, especially for heavy compositing.

The image is stored in the *Render Cache* folder as specified in the File Paths Preferences. You can also load it back into the Image Editor's Render Result, even after closing and reopening Blender; see Open Cached Render.

File Format

Choose the file format to save to. Based on which format is used, other options such as channels, bit depth and compression level are available.

For rendering out to images see: saving images, for rendering to videos see the Encoding panel.

Color

Choose the color format to save the image to. Note that *RGBA* will not be available for all image formats.

BW, RGB, RGBA

Image Sequence

Overwrite

Overwrite existing files when rendering.

Placeholders

Create empty placeholder frames while rendering.

Hint

Primitive Render Farm

An easy way to get multiple machines to share the rendering workload is to:

- Set up a shared directory over a network file system.
- Disable *Overwrite*, enable *Placeholders* in the Render *Output* panel.

- Start as many machines as you wish rendering to that directory.

Color Management

This panel controls how Color Management is applied when saving images.

Follow Scene:: Uses the same color management settings defined by the active Scene. These properties are defined in the Render Settings

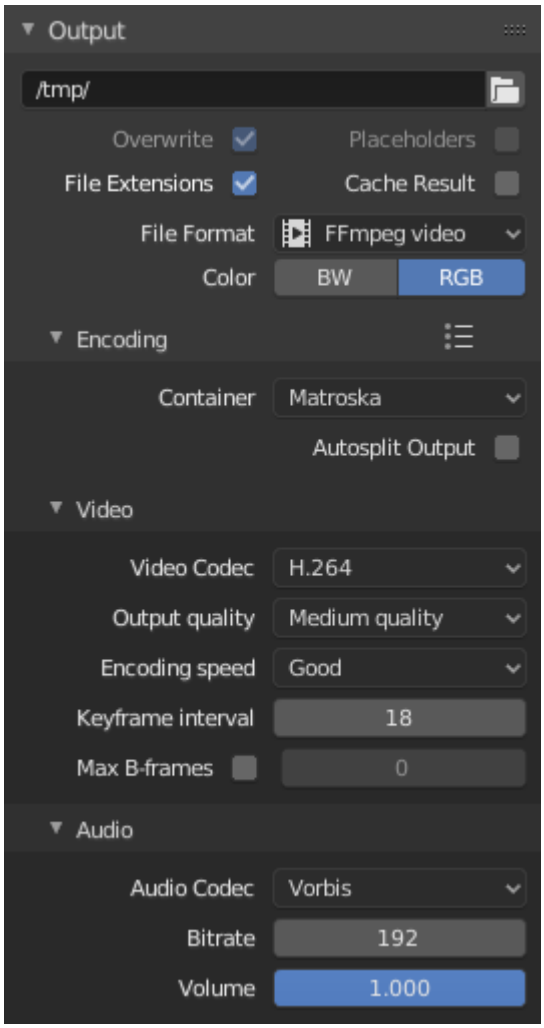
Override:: Uses custom color management settings defined by the properties below in the panel; disregarding any color management settings set at the Scene level.

For a detailed description of color management properties, see the Color Management page.

Encoding

Reference

Panel:: *Properties* ▸ *Output* ▸ *Encoding*



Encoding panel.

Here you choose which video container, codec, and compression settings you want to use. With all of these compression choices, there is a trade-off between file size, compatibility across platforms, and playback quality. In the header, you can use the presets, which choose optimum settings for you for that type of output.

Tip

When you view the System Console, you can see some of the output of the encoding process. You will see even more output if you execute Blender as `blender -d`.

Container
: Video container or file type. For a list of all available options, see video formats.

Autosplit Output
: If your video is huge and exceeds 2GiB, enable Autosplit Output. This will automatically split the output into multiple files after the first file is 2GiB in size.

Video

Video Codec
: Chooses the method of compression and encoding. For a list of all available options see video formats.

Note

Standards

Some containers and codecs are not compatible with each other, so if you are getting errors check that your container and codec are compatible. Like containers and codecs are sometimes not compatible with each other, some codecs do not work with arbitrary dimensions. So, try to stick with common dimensions or research the limitations of the codec you are trying to use.

Output Quality
: These are preset Rate.

Encoding Speed
: Presets to change between a fast encode (bigger file size) and more compression (smaller file size).

Keyframe Interval
: The number of pictures per Group of Pictures. Set to 0 for “intra_only”, which disables inter-frame video. A higher number generally leads to a

smaller file but needs a higher-powered device to replay it.

Max B-frames
: Enables the use of B-frames.

  Interval
  : The maximum number of B-frames between non-B-frames.

Rate

Bitrate
: Sets the average bit rate (quality), which is the count of binary digits per frame. See also: FFmpeg -b:v.

Minimum / Maximum
: Video files can use what is called variable bit rate (VBR). This is used to give some segments of the video less compressing to frames that need more data and less to frames with less data. This can be controlled by the *Minimum* and *Maximum* values.

Buffer
: The decoder bitstream buffer size.

Mux Rate
: Maximum bit rate of the multiplexed stream. Multiplexing is the process of combining separate video and audio streams into a single file, similar to packing a video file and MP3 audio file in a zip-file.

Mux Packet Size
: Reduces data fragmentation or muxer overhead depending on the source.

Audio

These settings change how sound is exported while rendering. To control how sound is played back from within Blender, see the audio settings in the Preferences.

Audio Codec
: Audio format to use. For a list of all available options, see video formats.

Audio Channels
: The number of audio source "locations" to encode.

Mono::	Output a single audio channel.
Stereo::	Output two audio channels; typically a left and right channel.
4 Channels::	Output a four audio channels.
5.1 Surround::	Output a five audio channels with one LFE channel.
7.1 Surround::	Output a seven audio channels with one LFE channel.

Sample Rate
: Sets the audio sampling rate.

Bitrate
: For each codec, you can control the bit rate (quality) of the sound in the movie. Higher bit rates are bigger files that stream worse but sound better. Use powers of 2 for compatibility.

Volume
: Sets the output volume of the audio.

Tips

Tip

The choice of video format depends on what you are planning to do.

It's not recommended to render directly to a video format in the first instance. If a problem occurs while rendering, the file might become unplayable and you will have to re-render all frames from the beginning. If you first render out a set of static images such as the default PNG format or the higher-quality OpenEXR (which can retain HDR pixel data),

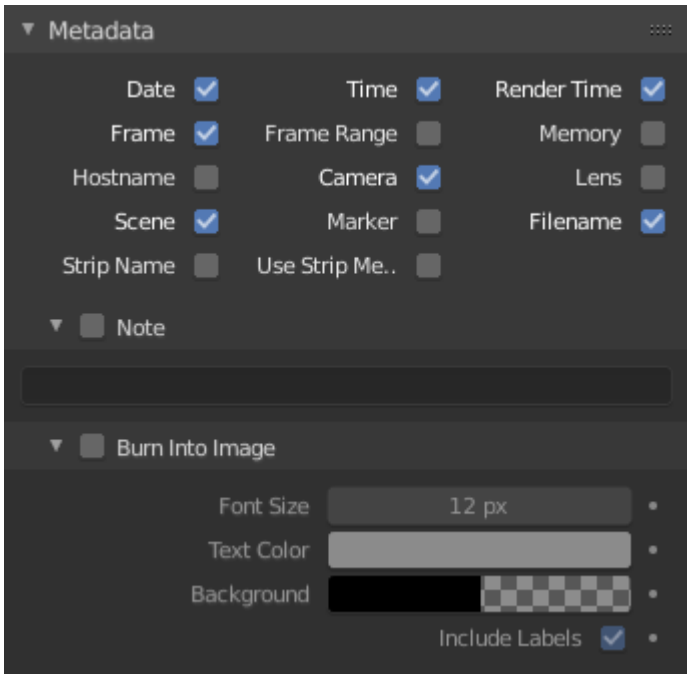
you can combine them as an Image Strip in the Video Sequencer. This way, you can easily:

- Restart the rendering from the place (the frame) where any problem occurred.
- Try out different video encoding options in seconds, rather than minutes or hours as encoding is usually much faster than rendering the 3D scene.
- Enjoy the rest of the features of the Video Sequencer, such as adding Image Strips from previous renders, audio, video clips, etc.

Tip

You shouldn't post-process a lossy-compressed file as the compression artifacts may become visible. Lossy compression should be reserved as a final 'delivery format'.

Metadata



Metadata panel.

The *Metadata* panel includes options for writing metadata into render output.

Note

Only some image formats support metadata: See image formats.

Metadata Input
: Where to grab metadata from.

Scene:: Use metadata from the current scene.
Sequencer Strips:: Use metadata from the strips in the Sequencer.

Include
: Date

Includes the current date and time.

Time
: Includes the current scene time and render frame at `HH:MM:SS.FF`.

Render Time
: Includes the render time.

Frame
: Includes the frame number.

Frame Range
: Includes the start and end frame numbers.

Memory
: Includes the peak memory usage.

Hostname
: Includes the rendering machine's hostname.

Camera
: Includes the name of the active camera.

Lens
: Includes the name of the active camera's lens value.

Scene
: Includes the name of the active scene.

Marker
: Includes the name of the last marker.

Filename
: Includes the filename of the blend-file.

Strip Name
: Includes the name of the foreground sequence strip.

Note

Includes a custom note.

Hint

It can be useful to use the *Note* field if you are setting up a render farm. Since you can script any information you like into it, such as an identifier for the render node or the job number. For details on stamping arbitrary values, see: this page.

Burn into Image

Add metadata as text to the render.

Font Size
: Set the size of the text.

Text Color
: Set the color and alpha of the stamp text.

Background
: Set the color and alpha of the color behind the text.

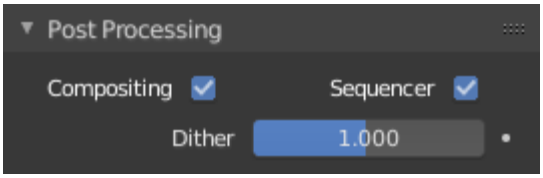
Include Labels
: Displays the labels before the metadata text. For example, “Camera” in front of the camera name, etc.

Post Processing

Reference

Panel:: *Properties ▸ Output ▸ Post Processing*

The Post Processing panel is used to control different options used to process your image after rendering.



Post Processing panel.

Pipeline

Todo.

Compositing

Renders the output from the compositing node setup, and then applies the Composite node tree on all images, displaying the image inputted in the Composite Output node.

Sequencer

Renders the output of the Video Sequence editor, instead of the view from the 3D scene's active camera. If the sequence contains Scene strips, these will also be rendered as part of the pipeline. If *Compositing* is also enabled, the Scene strip will be the output of the Compositor.

Dither

Dithering is a technique for blurring pixels to prevent banding that is seen in areas of gradients, where stair-stepping appears between colors. Banding artifacts are more noticeable when gradients are longer, or less steep. Dithering was developed for graphics with low bit depths, meaning they had a limited range of possible colors.

Dithering works by taking pixel values and comparing them with a threshold and neighboring pixels then does calculations to generate the

appropriate color. Dithering creates the perceived effect of a larger color palette by creating a sort of visual color mixing. For example, if you take a grid and distribute red and yellow pixels evenly across it, the image would appear to be orange.

Audio Rendering

- Introduction
  - Options
- Speaker
  - Options

Introduction

Audio can be rendered from the Render Menu.

Options

Relative Path
: Select the file path relative to the blend-file.

Accuracy
: Sample accuracy, important for animation data (the lower the value, the more accurate).

Container
: See here.

Channels
: The number of audio source “locations” to encode. Each channel can be mixed to a separate file by enabling *Split Channels* (see below).

Mono::	Output a single audio channel.
Stereo::	Output two audio channels; typically a left and right channel.
Stereo LFE::	Output a two audio channels with a “low-frequency effects” channel for frequencies below 120.
4 Channels::	Output a four audio channels.
5.1 Surround::	Output a five audio channels with one LFE channel.
6.1 Surround::	Output a six audio channels with one LFE channel.
7.1 Surround::	Output a seven audio channels with one LFE channel.

Format

Some *Audio Containers* also have option to choose a codec. For more information see here.

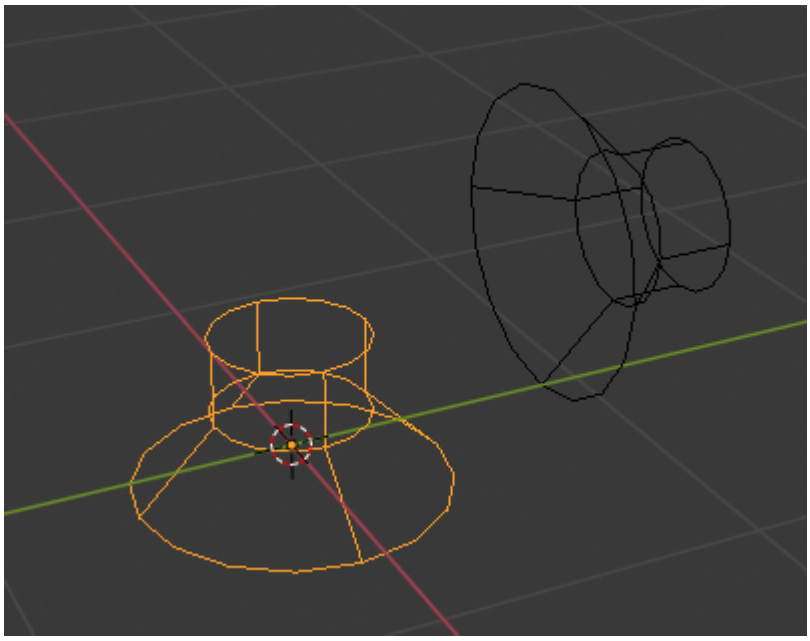
Sample Rate
: Sets the audio sampling rate.

Split Channels
: Each audio channel will be rendered into a separate file.

See also

- See Scene Audio settings.
- See Audio Output settings.
- See Audio Preferences.

Speaker



Speaker objects.

The speaker object is used to give sound in the 3D Viewport. After adding the object, the various settings can be changed in the Properties.

Options

Sound

Open
: The Data-Block Menu for loading audio files. There are two properties you can check when loading a sound:

  Cache
  : This means the whole sound will be decoded and the raw audio data will be buffered in memory, which results in faster playback, but uses quite a lot of memory. So this should be used for short sound

effects that get played more often, but not for longer audio files such as music.

Mono
: For any 3D audio or panning effects the sound source has to be single channel, otherwise it's assumed that the 3D audio and panning information is already present in the multichannel file. Enable this if you want to use those effects for a file with multiple channels.

Mute
: Toggles whether or not the sound can be heard.

Volume
: Adjust the loudness of the sound.

Pitch
: Can be used to bend the pitch of the sound to be either deeper or higher. This basically changes the playback speed of the sound which also results in a pitch change.

Playback Time

There is no setting to choose the start time when the speaker should start playing, because you might want a single speaker to play multiple times. Therefore, you have to open the *NLA Editor* where you can add Sound strips that define when the sound should start (nothing else, so any other properties of the strips, like length don't matter). When you add a speaker object such a strip will be added at the current frame.

Distance

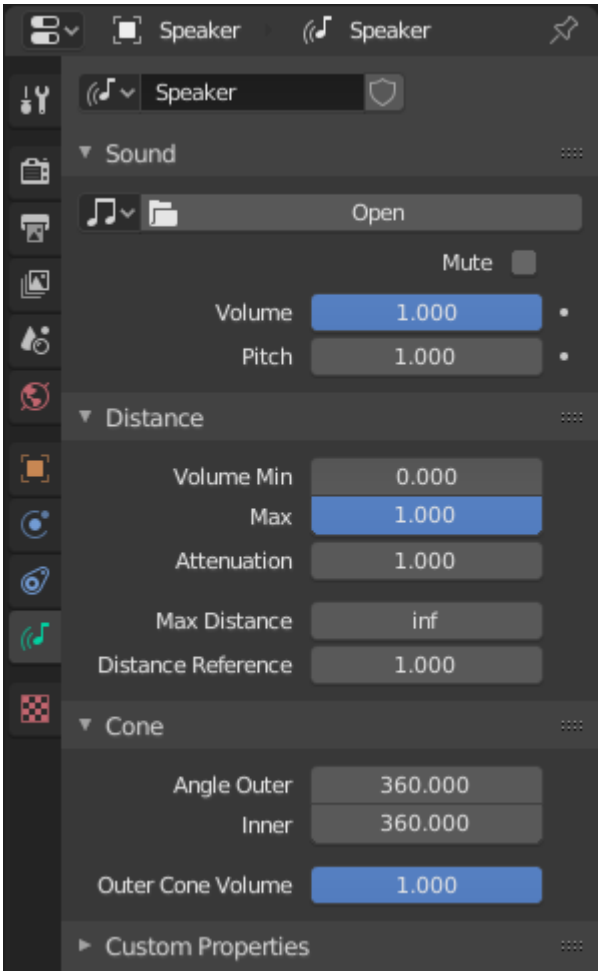
Distance attenuation relevant settings.

Volume
: Minimum/Maximum
  : No matter how far away/close the object is, the distance-based volume won't be lower/higher than this value.

Attenuation
: How strong the distance affects the volume. This factor sets the strength of the distance-based volume change, depending on the chosen distance model (see scene settings).

Distance
: Maximum
: If the object is farther away than this distance, this distance is used to calculate the distance-based volume. Influence of this value also depends on the distance model.

: Reference
: The distance at which the volume is full (1.0). Set this value to the distance used for recording the sound. Usually sound effects recordings should be made exactly 1 m away from sound to get an accurate volume.



Speaker properties.

Cone

Directionality relevant settings.

Imagine a cone with the top at the original of the speaker object and the main axis of it facing in the same direction as the speaker. There are two cones an inner and an outer cone. The angles represent their opening angles, so 360° mean the cone is fully open and there is no directionality anymore. Inside the inner cone the volume is full (1.0), outside the outer cone the

volume is, whatever one sets for the outer cone volume and the volume between those two cones, linearly interpolated between this two volumes.

Angle

Outer

Angle of the outer cone in degrees. Outside this cone, the volume is equal to the *Outer* volume.

Inner

Angle of the inner cone in degrees. Inside the cone, the volume is full.

Volume

Outer

Volume outside the outer cone.

Rendering Animations

While rendering stills will allow you to view and save the image from the render buffer when it is complete, animations are a series of images, or frames, and are automatically saved directly out to a drive after being rendered.

After rendering the frames, you may need to edit the clips, or first use the Compositor to do green-screen masking, matting, color correction, DOF, and so on to the images. That result is then fed to the Sequencer where the strips are cut and mixed and a final overlay is done.

Finally you can render out from the Sequencer and compress the frames into a playable movie clip.

Workflow

Generally, you do a lot of intermediate renders of different frames in your animation to check for timing, lighting, placement, materials, and so on. At some point, you are ready to make a final render of the complete animation for publication.

There are two approaches you can use when making a movie, or animation, with or without sound. The approach you should use depends on the amount of CPU time you will need to render the movie. You can render a “typical” frame at the desired resolution, and then multiply by the number of frames that will ultimately go into the movie, to arrive at a total render time.

If the total render time is an hour or more, you want to use the “Frame Sequence” approach. For example, if you are rendering a one-minute video clip for film, there will be (60 seconds per minute) X (24 frames per second) or 1440 frames per minute. If each frame takes 30 seconds to render, then you will be able to render two frames per minute, or need 720 minutes (12 hours) of render time.

Rendering takes all available CPU time; you should render overnight, when the computer is not needed, or set Blender to a low priority while rendering, and work on other things (be careful with the RAM space!).

Direct Approach

The Direct Approach, which is highly **not** recommended and not a standard practice, is where you set your output format to an AVI or MOV format, and click *Animation* to render your scene directly out to a movie file. Blender creates one file that holds all the frames of your animation. You can then use Blender's Video Sequencer to add an audio track to the animation and render out to an MPEG format to complete your movie.

Frame Sequence

The Frame Sequence is a much more stable approach, where you set your output format to a still format (such as JPG, PNG or a multi-layer format). Click *Animation* to render your scene out to a set of images, where each image is a frame in the sequence.

Blender creates a file for each frame of the animation. You can then use Blender's Compositor to perform any frame manipulation (post-processing). You can then use Blender's Video Sequencer to load that final image sequence, add an audio track to the animation, and render out to an MPEG format to complete your movie. The Frame Sequence approach is a little more complicated and takes more drive space, but gives you more flexibility.

Here are some guidelines to help you choose an approach.

Direct Approach

- Short segments with total render time under one hour.
- Stable power supply.
- Computer not needed for other uses.

Frame Sequence Approach

- Total render time over one hour.
- Post-production work needed:
  - Color/lighting adjustment
  - Green screen/matte replacement
  - Layering/compositing
  - Multiple formats and resolutions of the final product
- Intermediate frames/adjustments needed for compression/codec.
- Precise timing (e.g. lip-sync to audio track) needed in parts.
- May need to interrupt rendering to use the computer, and want to be able to resume rendering where you left off.

Frame Sequence Workflow

1. First prepare your animation.

2. In the *Format* panel, choose the render size, Pixel Aspect Ratio, and the Range of Frames to use, as well as the frame rate, which should already be set.

3. In the Output panel set up your animation to be rendered out as images, generally using a format that does not compromise any quality.

4. Choose the output path and file type in the Output panel as well, for example `//render/my-anim-`.

5. Confirm the range of your animation (frame Start and End).

6. Save your blend-file.

7. Press the *Animation* button and once the animation is finished, use your file manager to navigate to the output folder (`render` in this example). You will see lots of images that have a sequence number attached to. These are the single frames.

8. In Blender, open the Video Sequencer.

Note

The Video Sequencer does not support multi-layer EXR files. To render to a video format you will have to skip the next three steps and instead use an Image Input node in the Compositor.

9. Choose *Add Image* from the add menu. Select all the frames from your output folder that you want to include in your animation. They will be added as a strip in the Sequence editor.
10. Now you can edit the strip and add effects or leave it like it is. You can add other strips, like an audio strip.
11. Scrub through the animation to check if you have included all the frames.
12. In the Output panel, choose the container and codec you want (e.g. `MPEG H.264`) and configure them. The video codecs are described in Output Options.
13. Click the *Animation* render button and Blender will render out the Sequence editor output into a movie.

Hints

Your computer accidentally turns off in the middle of rendering your movie!

Unless your animation renders in a few minutes, it is best to render the animation as separate image files. Instead of rendering directly to a compressed movie file, use a lossless format (e.g. `PNG`).

This allows you an easy recovery if there is a problem and you have to re-start the rendering, since the frames you have already rendered will still be in the output directory.

Just disable the *Overwrite* option to start rendering where you left off.

You can then make a movie out of the separate frames with Blender's Sequence editor or use 3rd party encoding software.

Animation Preview

It can be useful to render a subset of the animated sequence, since only part of an animation may have an error.

Using an image format for output, you can use the *Frame Step* option to render every *N'th* frame. Then disable *Overwrite* and re-render with *Frame Step* set to 1.

Animation Player

Reference

Menu::	*Topbar ▸ Render ▸ View Animation*
Shortcut::	`Ctrl-F11`

The animation player is a utility typically used for previewing rendered animations, supporting all image and video formats also supported by Blender. This is a convenient way to play back image sequences at the correct frame rate.

Launching the animation player opens a new window, playing back images or a video located at the render output of the current scene. You can also drop images or movie files in a running animation player. It will then restart the player with the new data.

Tip

An external player can also be used instead of the one included in Blender. To do this, select it in the Preferences.

Player Options

Ping Pong
: When enabled, playback loops forwards than backwards.

X/Y Flip
: Flip the image horizontally or vertically.

  Viewing the animation from a different perspective can help you see the animation with "fresh eyes".

Hotkeys

The following table shows the available hotkeys for the animation player.

Playback

Action	Hotkey
Start/Pause:	`Spacebar`
Start playback (when paused):	`Return`
Quit:	`Esc`

Timeline

Action	Hotkey
Scrub in time:	`LMB`
Step back one frame:	`Left`
Step forward one frame:	`Right`
Step back 10 frames:	`Down`

Action	Hotkey
Step forward 10 frames:	`Up`
Manual frame stepping:	`NumpadPeriod`

Playback Options

Action	Hotkey
Backward playback:	`Shift-Down`
Forward playback	`Shift-Up`
Slow down playback:	`NumpadMinus`
Speed up playback:	`NumpadPlus`
Toggle looping:	`Numpad0`
Toggle frame skipping:	`A`
Toggle ping-pong:	`P`

Display

Action	Hotkey
Toggle Playhead (Indicator):	`I`
Flip image on the X axis:	`F`
Flip image on the Y axis:	`Shift-F`
Hold to show frame numbers:	`Shift`
Zoom in:	`Ctrl-NumpadPlus`
Zoom out:	`Ctrl-NumpadMinus`

Frame Rate

Action	Hotkey
60 fps	`Numpad1`
50 fps	`Numpad2`
30 fps	`Numpad3`

Action	Hotkey
25 fps	`Numpad4`
24 fps	`Shift-Numpad4`
20 fps	`Numpad5`
15 fps	`Numpad6`
12 fps	`Numpad7`
10 fps	`Numpad8`
6 fps	`Numpad9`
5 fps	`NumpadSlash`

Frame Cache

Image files are cached during playback for faster access.

While loading images is rarely a bottleneck, there are situations where high resolution images may slow down playback causing frame skipping.

See also

Memory Cache Limit preference to control this limit, which may be increased to cache more images during playback. Animation Playback Options to specify this value when launching from the command line.

Compositing

- Introduction
  - Getting Started
  - Examples
  - Image Size
  - Saving your Composite Image
- Sidebar
  - View
  - Options
- GPU Compositor
  - Data
  - Compositing Space
  - Output

Node Types

- Input Nodes
- Output Nodes

- Color Nodes
- Filter Nodes

- Keying Nodes
- Mask Nodes

- Tracking Nodes

- Transform Nodes
- Utilities Nodes

- Vector Nodes

- Group
- Layout Nodes

Introduction

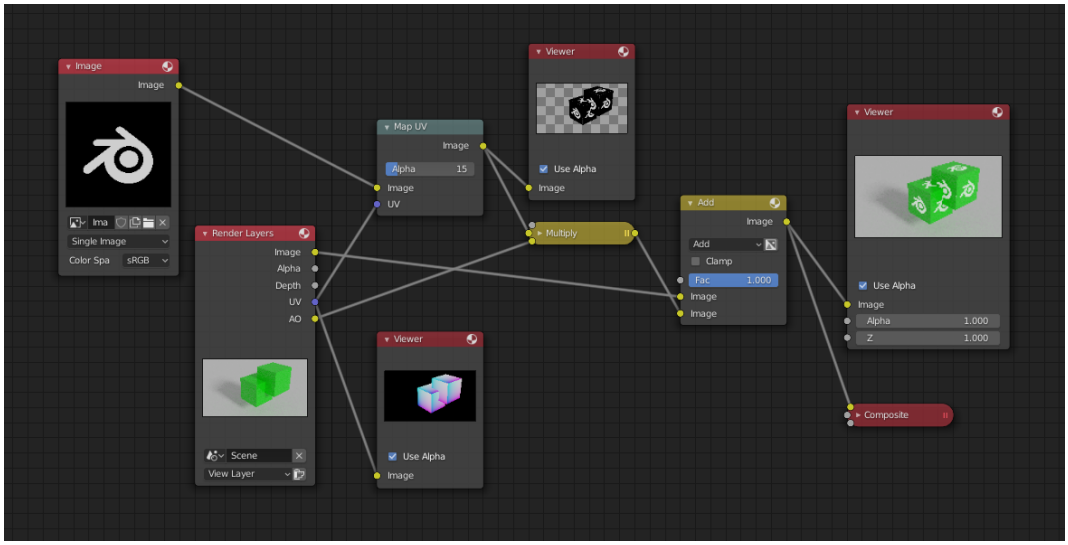
Compositing Nodes allow you to assemble and enhance an image (or movie). Using composition nodes, you can glue two pieces of footage together and colorize the whole sequence all at once. You can enhance the colors of a single image or an entire movie clip in a static manner or in a dynamic way that changes over time (as the clip progresses). In this way, you use composition nodes to both assemble video clips together and enhance them.

Note

Term: Image

The term *Image* may refer to a single picture, a picture in a numbered sequence of images, or a frame of a movie clip. The Compositor processes one image at a time, no matter what kind of input you provide.

To process your image, you use nodes to import the image into Blender, change it, optionally merge it with other images, and finally, save it.



An example of a composition.



An example of color correction.

Getting Started

Access the *Compositor* and activate nodes for compositing by clicking the *Use Nodes* checkbox in the header (see Introduction).

Note

After clicking *Use Nodes* the Compositor is enabled, however, it can also be disabled in the Post Processing.

You now have your first node setup, from here you can add and connect many types of Compositing Nodes, in a sort of map layout, to your heart's content (or physical memory constraints, whichever comes first).

Note

Nodes and node concepts are explained in more detail in the Nodes reference.

Examples

You can do just about anything with images using nodes.

Raw footage from a foreground actor in front of a blue screen, or a rendered object doing something, can be layered on top of a background. Composite both together, and you have composited footage.

You can change the mood of an image:

- To make an image 'feel' colder, a blue tinge is added.
- To convey a flashback or memory, the image may be softened.
- To convey hatred and frustration, add a red tinge or enhance the red.
- A startling event may be sharpened and contrast-enhanced.
- To convey a happy feeling add yellow (equal parts red and green, no blue).
- Dust and airborne dirt are often added as a cloud texture over the image to give a little more realism.

Image Size

It is recommended to pay attention to image resolution and color depth when mixing and matching images. Aliasing, color *flatness*, or distorted images can all be traced to mixing inappropriate resolutions and color depths.

The Compositor can mix images with any size, and will only perform operations on pixels where images have an overlap. When nodes receive inputs with differently sized Images, these rules apply:

- The first/top Image input socket defines the output size.

- The composite is centered by default, unless a translation has been assigned to a buffer using a *Translate* node.

So each node in a composite can operate on different sized images as defined by its inputs. Only the *Composite* output node has a fixed size, as defined by the Format Output Properties The *Viewer* node always shows the size from its input, but when not linked (or linked to a value) it shows a small 320×256 pixel image.

Saving your Composite Image

The *Render* button renders a single frame or image. Save your image using Save Image. The image will be saved using the image format settings on the Render panel.

To save a sequence of images, for example, if you input a movie clip or used a Time node with each frame in its own file, use the *Animation* button and its settings. If you might want to later overlay them, be sure to use an image format that supports an Alpha channel (such as `PNG`). If you might want to later arrange them front to back or create a depth of field effect, use a format that supports a Z-depth channel (such as `EXR`).

To save a composition as a movie clip (all frames in a single file), use an `AVI` or `QuickTime` format, and use the *Animation* button and its settings.

Sidebar

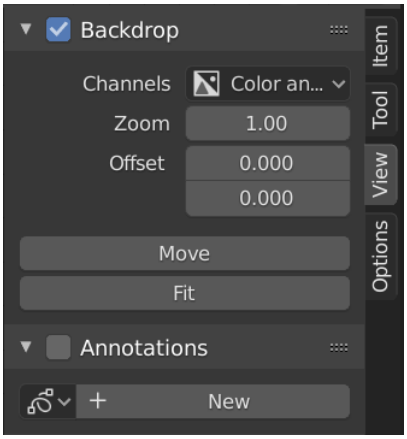
View

Reference

Panel:: *Sidebar region ▸ View*

Backdrop

The backdrop is the output of a Viewer node in the background of the Compositor. For example, when `Shift-Ctrl-LMB` on an Image node, it displays a preview of the image, or on a Mix node, will show the result of the mixing. You can toggle the backdrop by clicking the checkbox in the *Backdrop* panel title or by clicking on the *Backdrop* button in the header.



Backdrop panel.

Channels
: The color channels to display of the backdrop image.

Zoom `Alt-V V`
: Sets the size of the backdrop image.

Offset
: Change the screen space position of the backdrop.

Move `Alt-MMB`
: Changes the position of the backdrop.

Fit
: Scales the backdrop to fit the size of the Compositor.

Reset Backdrop
: Sets back to the default values of Zoom to 1 and Offset to 0.

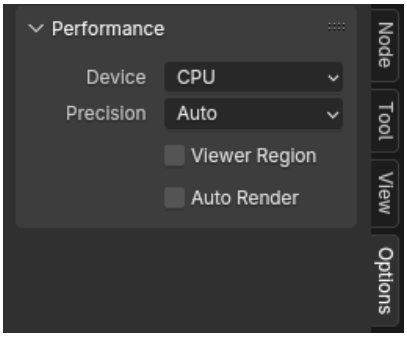
Options

Reference

Panel:: *Sidebar region ▸ Options*

Performance

This panel helps you tweak the performance of the Compositor.



Performance panel.

Device
: The device used for compositing.

CPU:: Use the CPU for compositing.
GPU:: Use the GPU for compositing.

Precision
: The precision of compositor intermediate result.

Auto:: Use full precision for final renders, half precision otherwise.
Full:: Use full precision for final renders and viewport.

Viewer Region
: This allows to set an area of interest for the backdrop. Press `Ctrl-B` and select a rectangular area in the preview which will become the next preview in the backdrop. `Ctrl-Alt-B` discards the region back to a full

preview. This is only a preview option, final compositing during a render ignores this region.

GPU Compositor

The new GPU accelerated compositor introduced in Blender 3.5 and is currently used for viewport compositing.

Data

Dimensionality

Compositing nodes operate on data that is either an image or a dimensionless single value. For instance, the Levels node outputs a single value, while the Render Layers node outputs an image. Node inputs that expect a single value assume a default value if an image is given and ignore the image completely, for instance, the Transform node expects single values for its inputs and will assume default values if images were given to those inputs. The default values are those that are considered identity and thus have no effect on the output, so for the Transform node, the *X*, *Y*, and *Angle* inputs will have a default value of zero, while the *Scale* input will have a default value of one. On the other hand, if node inputs that expect an image are given a single value, the single value will be assumed to cover the whole compositing space. For instance, the Filter node expect its *Factor* input to be an image, but if a single value is given, it will be assumed to be the same for all pixels.

Type

Three types of data exist, all of which are stored in half precision formats:

Float
: A signed floating-point number. Integer data is also stored as floats because no integer type exist.

Vector

A 4D vector. While it is 4D, it can have different interpretations depending on the node that uses it. It can be treated as a 2D vector with the last two components ignored, for instance, the *Vector* input of the Displace node is treated as a 2D vector. It can be treated as a 3D vector with the last component ignored, for instance, the *Vector* input of the Seperate XYZ node is treated as a 3D vector. It can be treated as two consecutive 2D vectors. For instance the *Velocity Pass* as expected by the Vector Blur node is assumed to have the *2D Previous Velocity* in the X and Y components of the vector and the *2D Next Velocity* in the Z and W components of the vector.

Color
: A 4D vector storing the Red, Green, Blue, and Alpha of the color. The color is free form and does not conform to a specific color space or alpha storage model, instead, appropriate nodes will have settings to control the representation of their output and nodes exist to convert between the different representations.

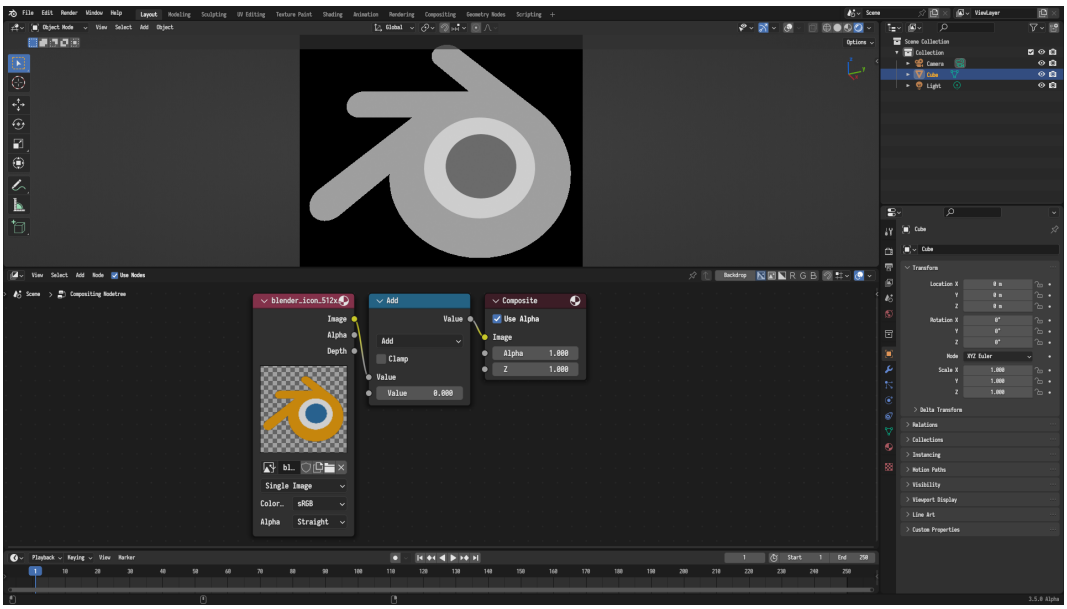
Implicit Conversion

In case a node input is given data of type other than its own type, the following implicit conversions are performed:

Source	Target	Conversion
Float	Vector	f => Vector(f, f, f, 0)
Float	Color	f => Color(f, f, f, 1)
Vector	Float	(x, y, z, w) => Average(x, y, z)
Vector	Color	(x, y, z, w) => Color(x, y, z, 1)

Source	Target	Conversion
Color	Float	(r, g, b, a) => Average(r, g, b)
Color	Vector	(r, g, b, a) => Vector(r, g, b, 0)

The following example demonstrates implicit conversion between a color type and a float type, since the Math node expect float inputs.



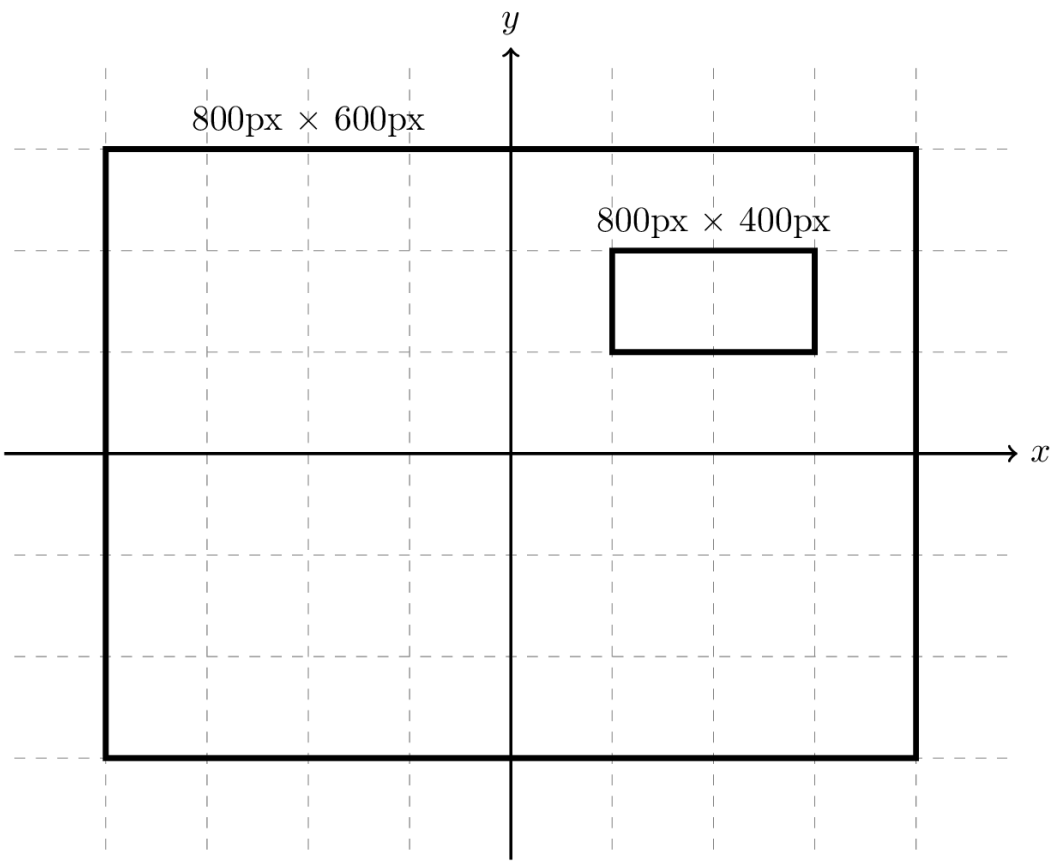
An example that demonstrates implicit conversion between a color type and a float type, since the *Math* node expects float inputs.

Compositing Space

Image Domain

The compositor is designed in such a way as to allow compositing in an infinite compositing space. Consequently, images are not only represented

by their size, but also by their transformation in that space, much like 3D objects have transformations. An identity transformation represents an image that is centered in space. The rectangular area occupied by an image in that space as defined by its transformation and size is called the *Domain* of the image. The figure below demonstrates the domains of two example images.



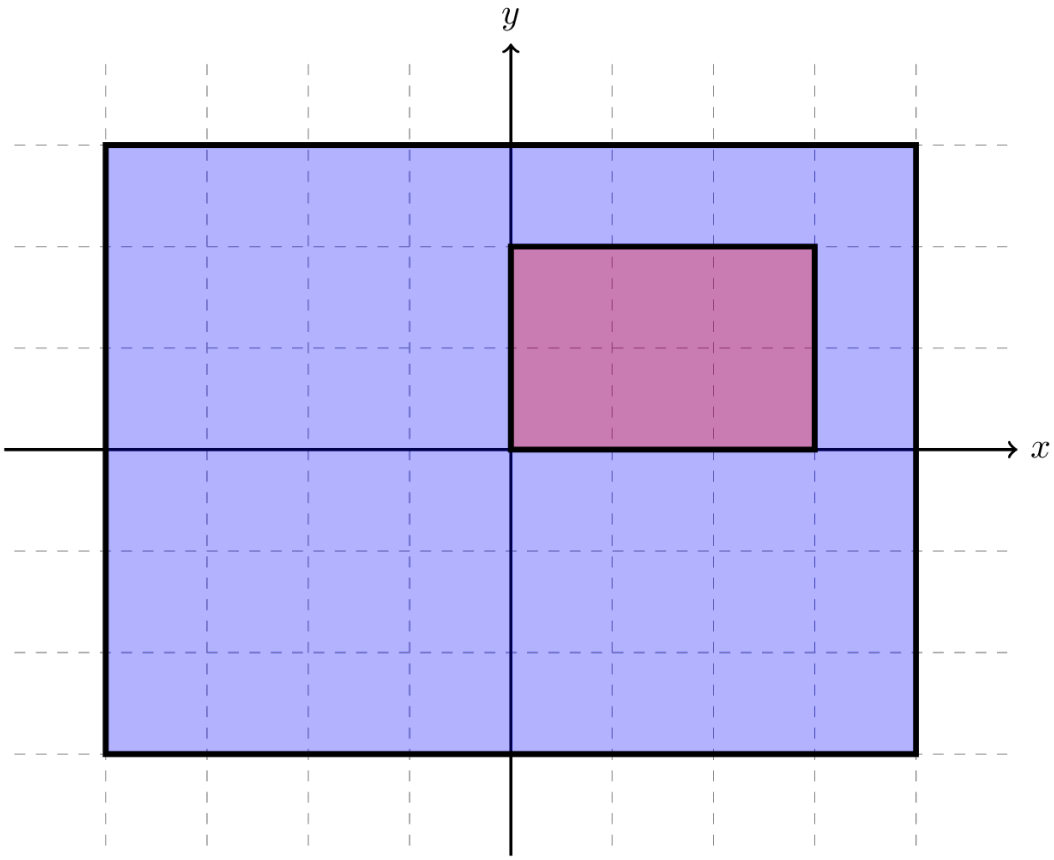
The domains of two example images are illustrated on the compositing space. One of the images is centered in space and the other is scaled down and translated such that it lies in the upper right quadrant of the space. Notice that both images have similar sizes in pixels, yet their apparent sizes are different.

Images can be transformed using nodes like the Transform, Translate, and Rotate nodes.

Operation Domain

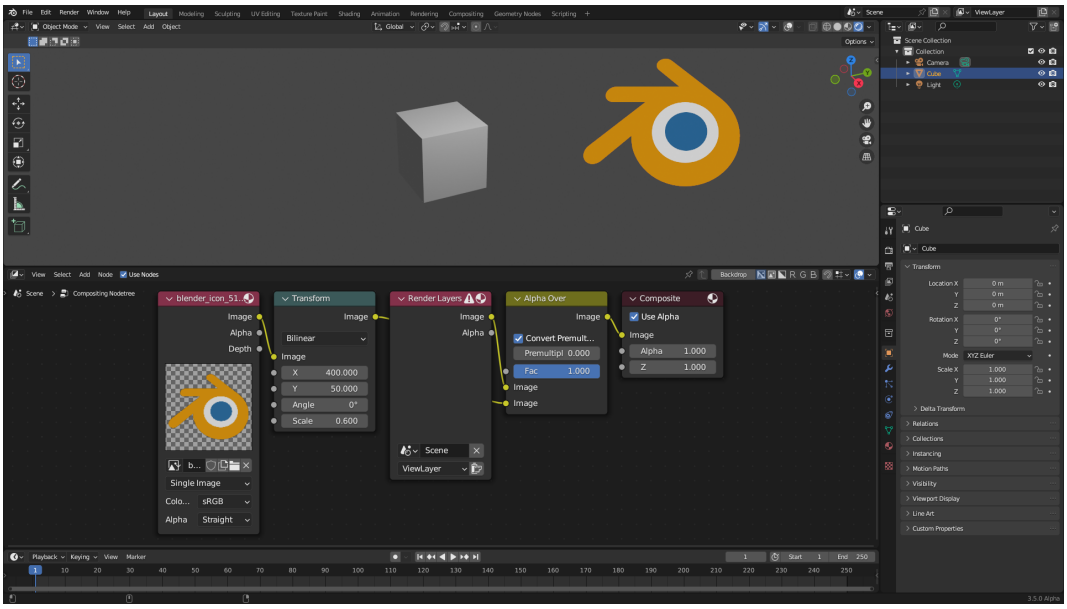
Compositor Nodes operate on a specific rectangular area of the compositing space called the *Operation Domain*. The nodes only consider the area of the input images that overlap the operation domain and ignores the rest of the images. If an input image doesn't completely overlap the operation domain, the rest of the operation domain for that input will be assumed to be a zero value, a zero vector, or a transparent zero color depending on the type.

For instance, the figure below illustrates a case where the operation domain of a node is the large blue area and the domain of an input image is the small red area. In that case, the input image doesn't completely overlap the operation domain, so the rest of the blue area for that input image is assumed to be zero.



An example case where the operation domain of a node is shown in blue and the domain of an input image is shown in red. Since the input image doesn't completely cover the operation domain of the node, the rest of the blue area for that input image is assumed to be zero.

The previous illustration is a representation of a real world example where one uses the Alpha Over node to overlay a small logo on an image, as shown in the figure below. In that case, the operation domain covers the entirety of the viewport — as will later be demonstrated, but the logo covers only a small area of it, so the rest of the area is assumed to be a zero transparent color, which is convenient for the use case.



A real world example where the Alpha Over node is used to over a small logo on an image. The logo only covers a small area of the operation domain, which is the whole viewport in this case, so the rest of the area is assumed to be a zero transparent color.

Interpolation

If an input image to a node is not perfectly aligned with the operation domain of the node or have a different size in pixels, the node would typically need to do a process called Interpolation, where the input image is read at the exact positions of the pixels of the operation domain. This can be done using different interpolation methods, including Nearest-Neighbor, Bilinear, and Bicubic interpolations. Those interpolation methods are demonstrated in the following Wikipedia gallery. Transformation nodes like the Transform and Rotate nodes include an interpolation option to set how they prefer their output image to be read and interpolated.

Determining Operation Domain

The question remains on how nodes determine their operation domain. Different node types can have different mechanisms for determining their operation domain. But generally, three classes of nodes exist when it comes to the mechanism of determining the operation domain, each of which is presented in one of the following sections.

Input Nodes

The operation domain of input nodes like the Image node is a domain with an identity transformation and the same size as their outputs, so for the *Image* node, the operation domain will be the domain whose size is the size of the image and whose transformation is an identity one.

Output Nodes

The operation domain of output nodes like the Viewer node is a domain with an identity transformation and the same size as the final compositor output. For viewport compositing, that size would be the viewport size, and for final render compositing, that size would be the scene render size.

Other Nodes

Unless stated otherwise in their respective documentation pages, all other nodes use the following mechanism. One of the inputs of the nodes is designated as the *Domain Input* of the node, and the operation domain of the node is identical to the domain of that designated input. For many nodes, the domain input can be intuitively identified as the main input of the node, for instance, the domain input for the Filter node is the *Image* input. But there are some caveats to note, which requires a deeper understanding of the mechanism.

Each input in the node has the so called *Domain Priority* property, the operation domain of the node is the domain of the non single value input with the highest domain priority. So for instance, the Filter node has two

inputs, the domain priority of the *Image* input is higher than that of the *Factor* input, and there are four possible configurations:

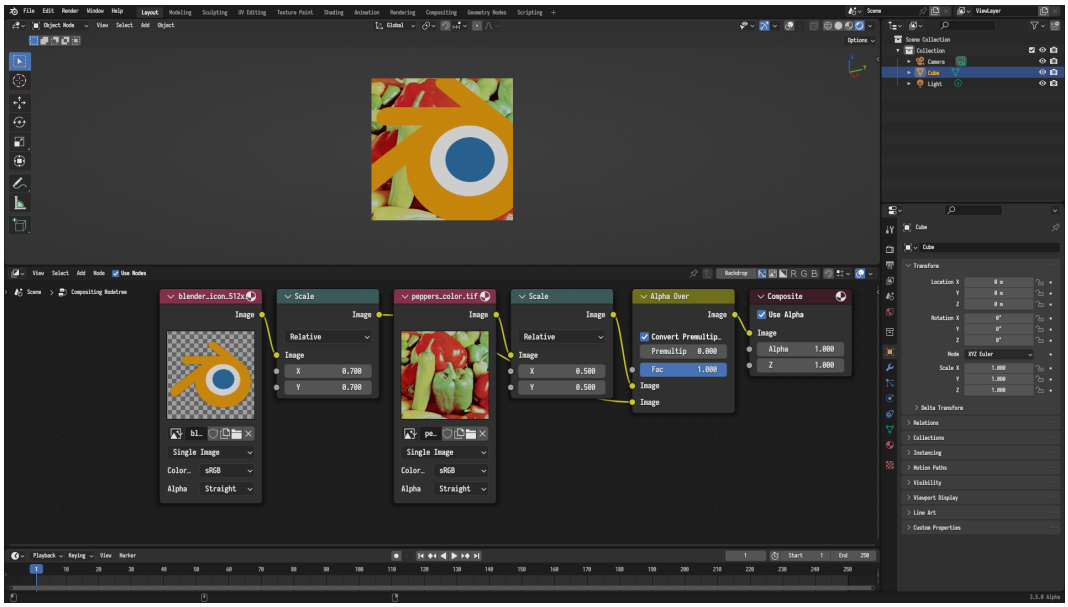
- Both the *Image* and factor inputs are connected to images. In this case, the *Image* input is the domain input because it has the highest priority and is connected to an image.
- The *Image* input is connected to an image, but the *Factor* input isn't. In this case, the *Image* input is the domain input because it it is the only input connected to an image regardless of its priority.
- The *Image* input is not connected to an image but the *Factor* input is. In this case, the *Factor* input is the domain input because it is the only input connected to an image regardless of its priority.
- Neither the *Image* nor the *Factor* inputs are connected to images, in this case, there isn't a domain input because the node is evaluated on single values.

Considerations

The aforementioned mechanism for determining the operation domain has a number of consequences that needs to be considered as they might be undesirable, each of which is presented in one of the following sections.

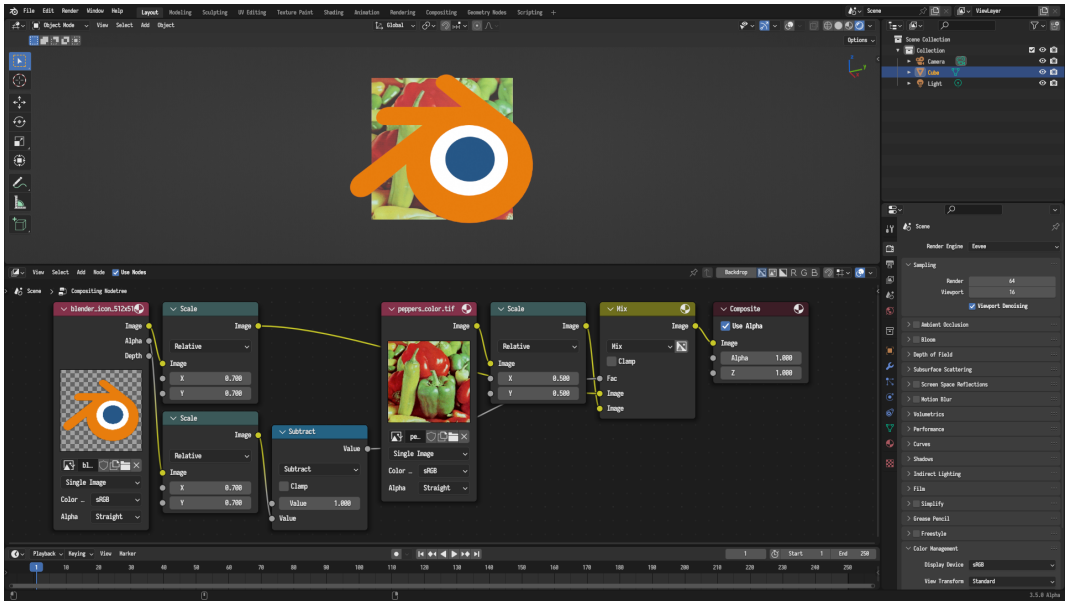
Clipping

The output of nodes will be intuitively clipped to the operation domain, or rather, the domain of the domain input. For instance, if the *Foreground* input is bigger than the *Background* input in the Alpha Over node, the output will be clipped to the *Background* input because it is the domain input, as shown in the following figure.



The *Foreground* input is bigger than the *Background* input in the *Alpha Over* node, so the output is intuitively clipped to the *Background* input because it is the domain input.

The Alpha Over node currently does not support changing the domain priority for its inputs, so as a workaround, one can use a Mix node to achieved the desired behavior, noting that the first *Image* input in the *Mix* node has the highest domain priority, as shown in the following figure.



Working around the clipping behavior of the *Alpha Over* node using a Mix node, noting that the first *Image* input in the *Mix* node has the highest domain priority

Output

The GPU compositor only supports a single active output target, that is, only one of the Composite nodes or Viewer nodes in the node tree will be considered active and the rest will be ignored. In particular, the compositor searches the *Active Node Tree Context* and falls back to the *Root Node Tree Context* if no active output was found in the active node tree context. The active node tree context is the node tree of an expanded node group, that is, when the users selects a node group node and edits its underlying tree, while the root node tree context is the top level node tree without any expanded node groups. The compositor searches for the active *Composite* node, if non was found, it searches for the active *Viewer* node, be it a *Viewer* or a *Split Viewer* node, if non was found, the compositor doesn't run altogether. Consequently, note that adding a *Viewer* node will have no effect if there is a *Composite* node, since the priority is given to *Composite* nodes.

Input Nodes

Input nodes produce information from a data source. For instance, an input can be:

- Taken directly from the active camera in a selected scene.
- A static image.
- A movie clip (such as an image sequence or video).
- A color or value.

These nodes generate the information that is passed to other nodes. As such, they have no input sockets; only outputs.

- Constant

- Bokeh Image Node
- Image Node
- Mask Node
- Movie Clip Node
- Texture Node

- Scene

Constant

- RGB Node
- Value Node

RGB Node



Inputs

This node has no input sockets.

Properties

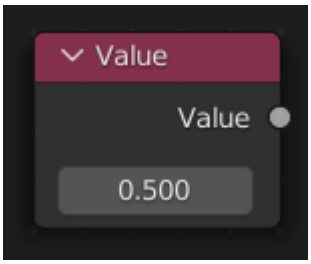
The RGB node uses the color picker widget.

Outputs

Color / RGBA
: A single RGBA color value.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.



Inputs

This node has no input sockets.

Properties

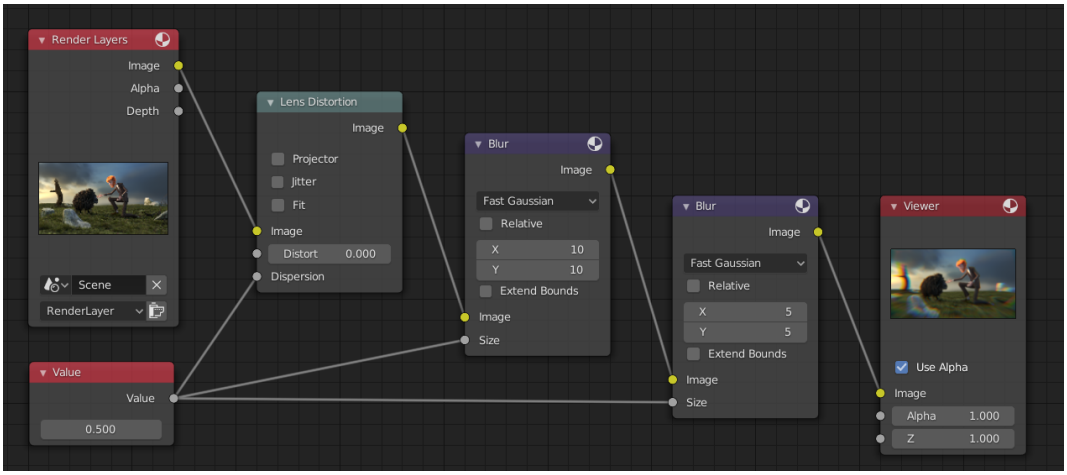
Single numerical value (floating-point).

Outputs

Value
: The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.



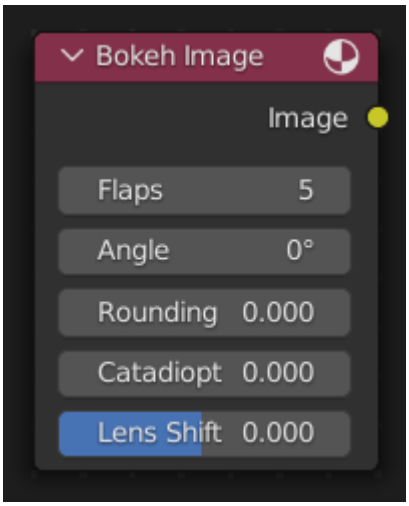
Example of the *Value* node.

Tip

From this you can also make different values proportional to each other by adding a Math Node in between the different links.

Bokeh Image Node

The *Bokeh Image* node generates a special input image for use with the Bokeh Blur filter node.



The *Bokeh Image* node is designed to create a reference image which simulates optical parameters such as aperture shape and lens distortions which have important impacts on bokeh in real cameras.

Inputs

This node has no input sockets.

Properties

The first three settings simulate the aperture of the camera.

Flaps
: Sets an integer number of blades for the cameras iris diaphragm.

Angle
: Gives these blades an angular offset relative to the image plane.

Rounding
: Sets the curvature of the blades with (0 to 1) from straight to bringing them to a perfect circle.

Catadioptric

Provides a type of distortion found in mirror lenses and some telescopes. This can be useful to produce a visual complex bokeh.

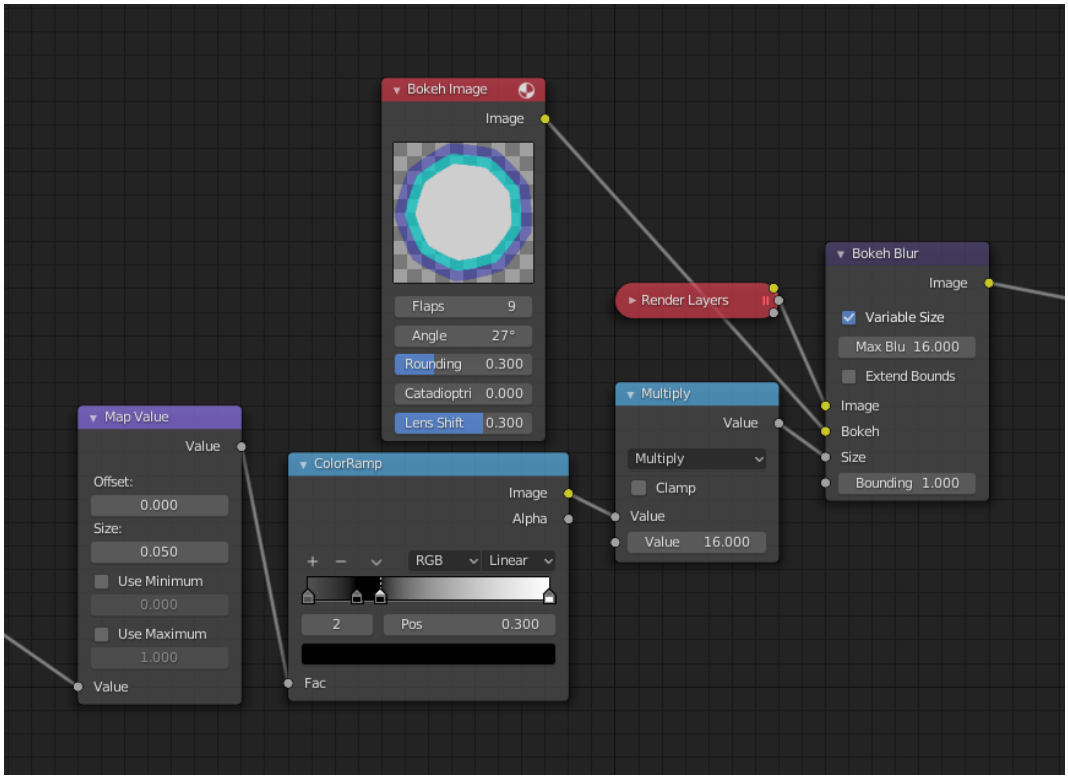
Lens Shift
: Introduces chromatic aberration into the blur such as would be caused by a tilt-shift lens.

Outputs

Image
: The generated bokeh image.

Example

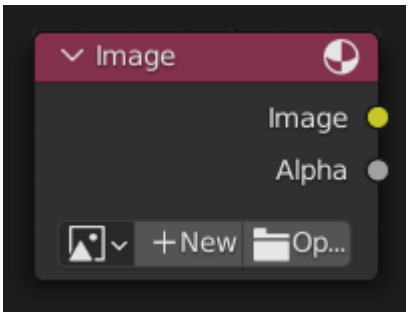
In the example below the *Bokeh Image* is used to define the shape of the bokeh for the Bokeh Blur node.



Example of *Bokeh Image* node.

Image Node

The *Image* node injects any image format that is supported by Blender.



Inputs

This node has no input sockets.

Properties

Image
: Selection of different types of media. For controls see Data-Block Menu. For the options see Image Settings.

Note

More options can be set in the Sidebar region.

Outputs

The first two sockets are the minimum.

Image
: Standard color output.

Alpha

Separate Alpha value.

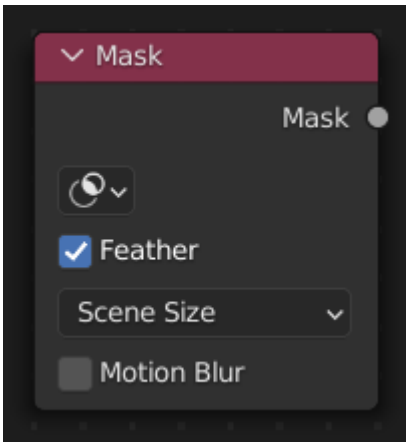
Note

Multi-Layer Format

When a multi-layer file format, like `EXR`, is loaded, each layer is made available as a socket.

Mask Node

The Mask node can be used to select a Mask data-block. This node can be used with other nodes, for example to Invert, Multiply or Mix, or used as a factor input.



Inputs

This node has no input sockets.

Properties

Masks
: The selectable mask data-block. If the label is left blank, the mask name will be set.

Feather
: Use or ignore feather points defined for splines see Mask Feathers for more details.

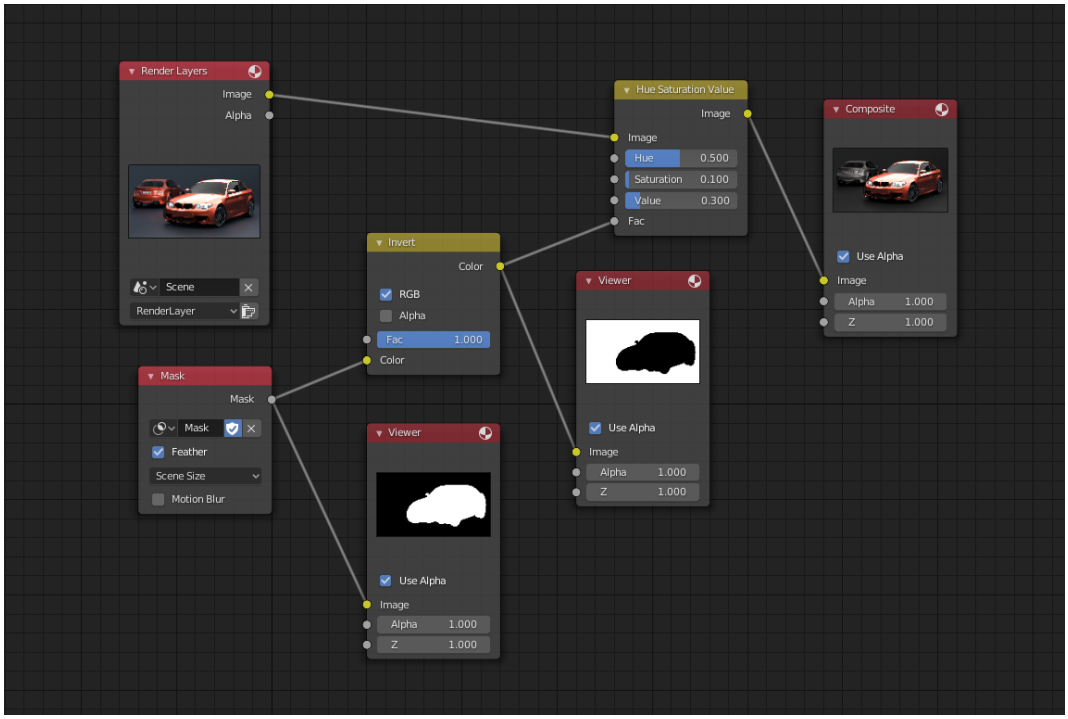
Size
: Scene Size will give an image the size of the render resolution for the scene, scaling along when rendering with different resolutions. Fixed gives a fixed size in pixels. Fixed/Scene gives a size in pixels that still scales along when changing the render resolution percentage in the scene.

Motion Blur
: For animated masks, creating a motion blurred mask from the surrounding frames, with a given number of samples (higher gives better quality), and a camera shutter time in seconds.

Outputs

Mask
: The black-and-white output of the mask.

Example

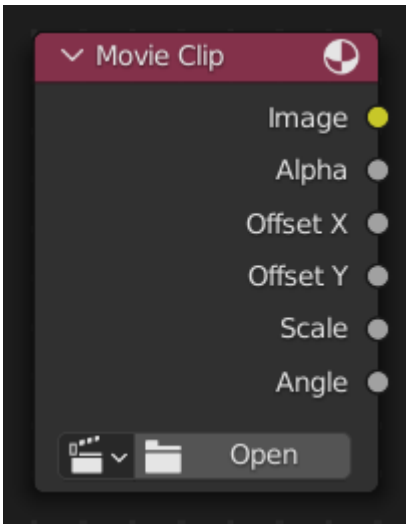


Example of the Mask node.

In the example above, the *Mask* node is used to isolate the object from the background to preserve it from being corrected.

Movie Clip Node

This node is a special node that uses some of the values taken from footage cameras and trackings and link them to the output. It is possible to load image sequences, but only Image and Alpha values will be available, because the other outputs will not have any values associated with them. When a tracked clip is chosen, Blender will fulfill the outputs using internal values taken from the tracking. So the controls for start and end frames will be defined in the Movie Clip editor.



Inputs

This node has no input sockets.

Properties

Movie Clip
: Used to select the movie clip. For controls see Data-Block Menu.

Outputs

The first two sockets are the minimum output.

Image
: Outputs the entire image in the specified color space.

Alpha
: The alpha value taken from the movie or image.

Offset X
: The X offset value from the footage camera or tracking.

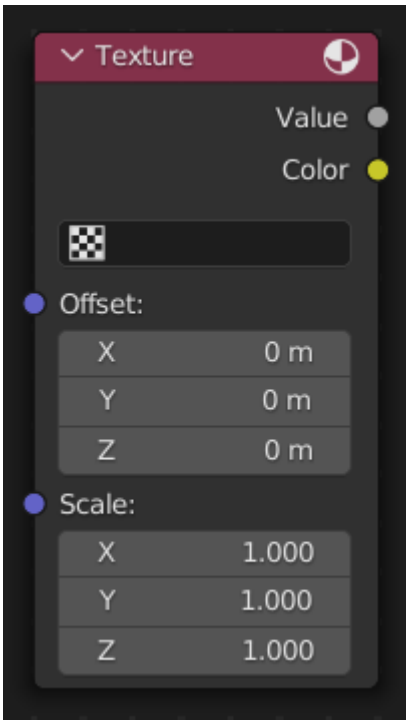
Offset Y
: The Y offset value from the footage camera or tracking.

Scale
: The scale of the image taken from the footage camera or tracking.

Angle
: The lens angle taken from the footage camera or tracking.

Texture Node

The Texture node makes 3D textures available to the Compositor.



Note

The Texture node is not supported by the viewport compositor.

Inputs

Offset
: A vector (XYZ) transforming the origin of the texture.

Scale
: A vector (XYZ) to scale the texture.

Properties

Texture
: The texture can be selected from a list of textures available in the current blend-file or linked-in textures. The textures themselves can not be edited in the Compositor, but in the Texture Node Editor.

Outputs

Value
: Gray-scale color values.

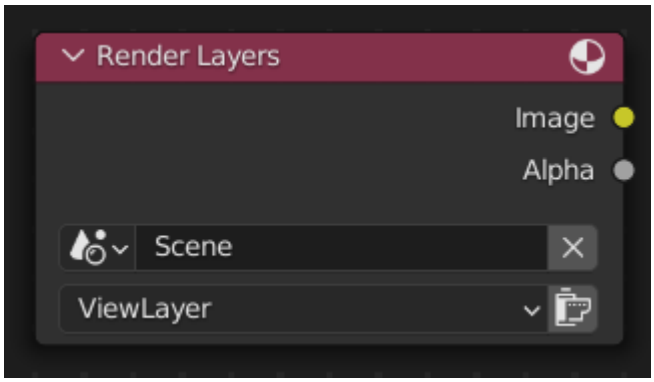
Color
: Color values.

Scene

- Render Layers Node
- Scene Time Node
- Time Curve Node

Render Layers Node

Read render layers and passes from a scene into the compositing node graph.



Inputs

This node has no input sockets.

Properties

Scene
: Select the scene within your blend-file. The scene information taken is the raw footage (pre-compositing and pre-sequencing).

  Hint

  To use composited footage from another scene, it has to be rendered into a multi-layer frameset (e.g. `OpenEXR`) as an intermediate file store and then imported with Image input node again.

Render Layer
: A list of available Render Layers. The render button is a shorthand to re-render the active scene.

Outputs

Image
: Rendered image.

Alpha
: Alpha channel.

Render Passes Sockets

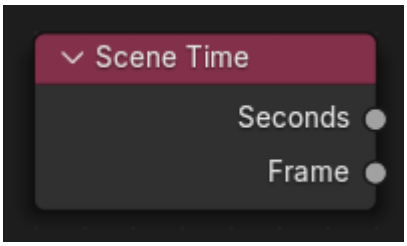
Depending on the Render passes that are enabled, other sockets are available. See render passes.

Note

In the viewport compositor, Render Passes are only supported in EEVEE. For other engines, the passes return a zero value, a zero vector, or a transparent color depending on their type.

Scene Time Node

The *Scene Time* node outputs the current time in the scene's animation in units of seconds or frames.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Seconds
: Current scene time in seconds.

Frames
: Current scene frame. As an input in geometry nodes, this output may also output non-round numbers, in order to support higher quality motion blur.

Time Curve Node

The *Time Curve node* generates a factor value (from 0.0 to 1.0) between the scene start and end time, using a curve mapping.



Inputs

This node has no inputs.

Properties

Curve
: The Y value defined by the curve is the factor output. For the curve controls, see Curve widget.

Tip

Flipping the curve around reverses the time input, but doing so is easily overlooked in the node setup.

Start, End
: Start frame and End frame of the range of time specifying the values the output should last. This range becomes the X axis of the graph. The time input could be reversed by specifying a start frame greater than the end frame.

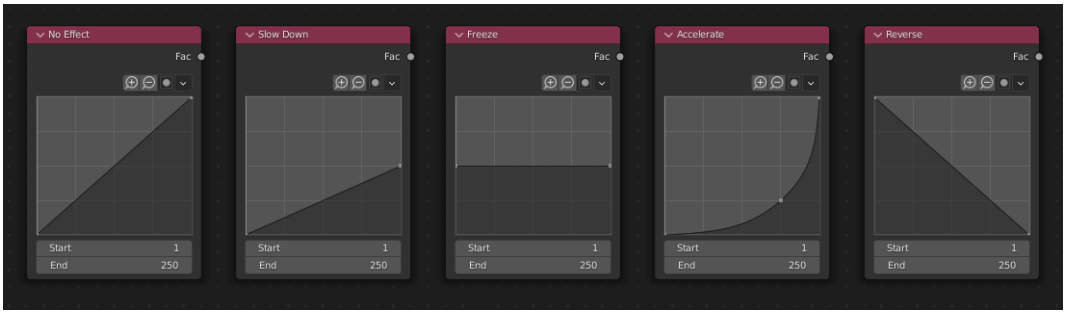
Outputs

Factor
: The Y value of the curve at the current frame.

Hint

The Map Value node can be used to map the output to a more appropriate value. With some curves, it is possible that the Time Curve node may output a number larger than one or less than zero. To be safe, use the Min/Max clamping function of the Map Value node to limit output.

Example



Time controls from left to right: no effect, slow down, freeze, accelerate, reverse.

Output Nodes

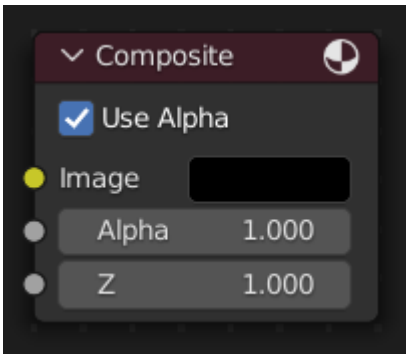
These nodes are used to output the composited result in some way.

- Composite Node
- Viewer Node

- File Output Node

Composite Node

The Composite node is where the actual output from the Compositor is connected to the renderer. This node is updated after each render, but also reflects changes in the node tree (provided at least one finished input node is connected).



Inputs

Connecting a node to the Composite node will output the result of the prior tree of that node to the Compositor.

Image
: RGB image. The default is black, so leaving this node unconnected will result in a black image.

Alpha
: Alpha channel.

Properties

Use Alpha
: Used alpha channel, colors are treated alpha *premultiplied*. If disabled, alpha channel gets set to 1, and colors are treated as alpha *straight*, i.e. color channels does not change.

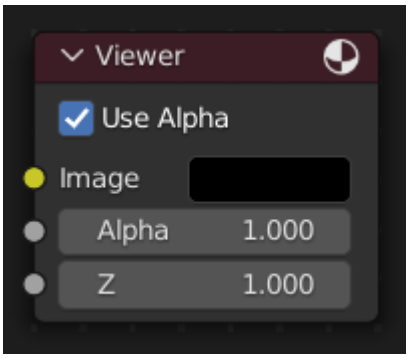
Outputs

This node has no output sockets.

Note

If multiple Composite nodes are added, only the active one (last selected, indicated by a red header) will be used.

Viewer Node



The *Viewer* node allows temporarily visualizing data from inside a node graph. It can be plugged in anywhere to inspect an image or value map in your node tree.

Select a view node with `LMB` to switch between multiple viewer nodes. It is possible to automatically plug any other node into a Viewer node by pressing `Shift-Ctrl-LMB` on it.

Inputs

Image
: RGB image. The default is black, so leaving this node unconnected will result in a black image.

Alpha
: Alpha channel.

Properties

Use Alpha
: Used alpha channel, colors are treated alpha *premultiplied*. If disabled, alpha channel gets set to 1, and colors are treated as alpha *straight*, i.e. color channels does not change.

Outputs

This node has no output sockets.

Note

It is possible to add multiple Viewer nodes, though only the active one (last selected, indicated by a red header) will be shown on the backdrop or in the Image editor.

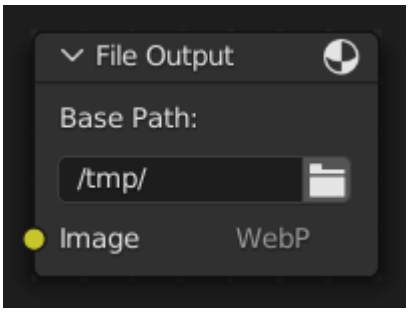
Using the Image Editor

The Viewer node allows results to be displayed in the Image Editor. The image is facilitated in the header by selecting *Viewer Node* in the linked *Image* data-block menu. The Image Editor will display the image from the currently selected Viewer node.

To save the image being viewed, use *Image* ▸ *Save As*…, `Alt-S` to save the image to a file.

The Image Editor also has three additional options in its header to view Images with or without Alpha, or to view the Alpha or Z itself. Click and holding the mouse in the Image displayed allows you to sample the values.

File Output Node



This node writes out an image, for each frame range specified, to the filename entered, as part of a frameset sequence.

This node can be used as a way to automatically save the image after a render; In addition, since this node can be hooked in anywhere in the node tree, it can also save intermediate images automatically.

Inputs

Image
: The image(s) will be saved on rendering, writing to the current frame. An entire sequence of images will be saved, when an animation is rendered.

Properties

Base Path
: Unlike the render output filepath, this node uses a base directory and an image name, by default the output path is composed of: `{base path}/{file name}{frame number}.{extension}`.

  Besides being split into two settings, in all other respects, this setting is treated the same as the render output path.

File Format
: Label that shows the selected file format.

Note

More options can be set in the Sidebar region.

Outputs

This node has no output sockets.

Color Nodes

These nodes adjust the image's colors, for example increasing the contrast, making it warmer, overlaying another image, etc.

- Adjust

- Mix

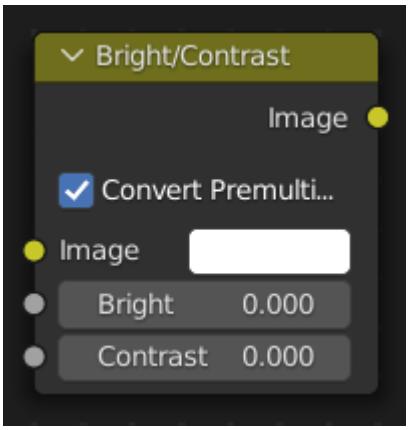
- Alpha Convert Node
- Color Ramp Node
- Convert Colorspace Node
- Set Alpha Node

- Invert Color Node
- RGB to BW Node

Adjust

- Brightness/Contrast Node
- Color Balance Node
- Color Correction Node
- Exposure Node
- Gamma Node
- Hue Correct Node
- Hue/Saturation/Value Node
- RGB Curves Node
- Tone Map Node

Brightness/Contrast Node



Inputs

Image
: Standard color input.

Brightness
: An additive-type factor by which to increase the overall brightness of the image. Use a negative number to darken an image.

Contrast
: A scaling type factor by which to make brighter pixels brighter, but keeping the darker pixels dark. Higher values make details stand out. Use a negative number to decrease the overall contrast in the image.

Properties

Convert Premultiplied
: By default, it is supposed to work in *premultiplied* alpha. If the *Convert Premul* checkbox is not enabled, it is supposed to work in *straight* alpha.

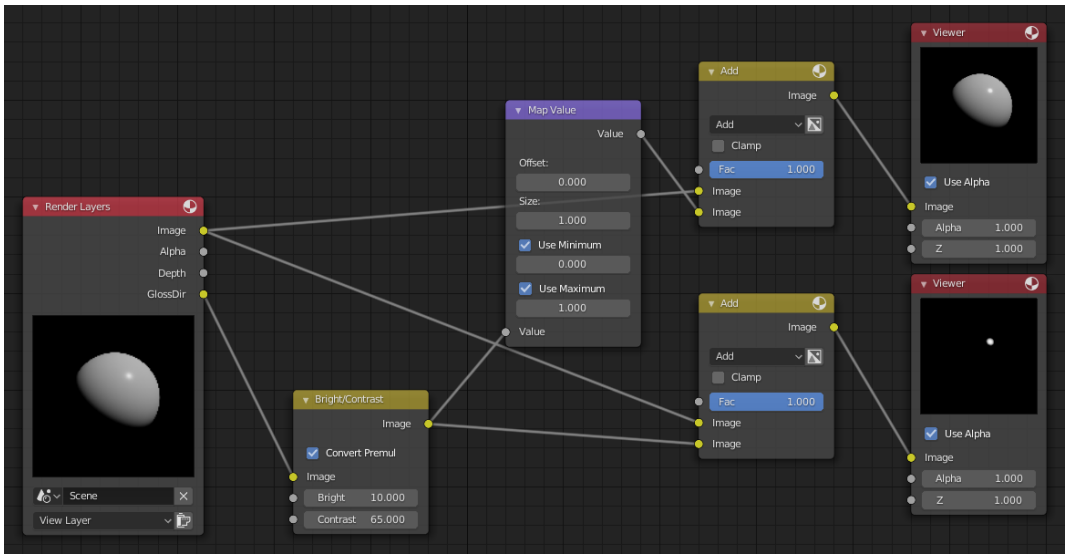
See Alpha Channel.

Outputs

Image
: Standard color output.

Notes

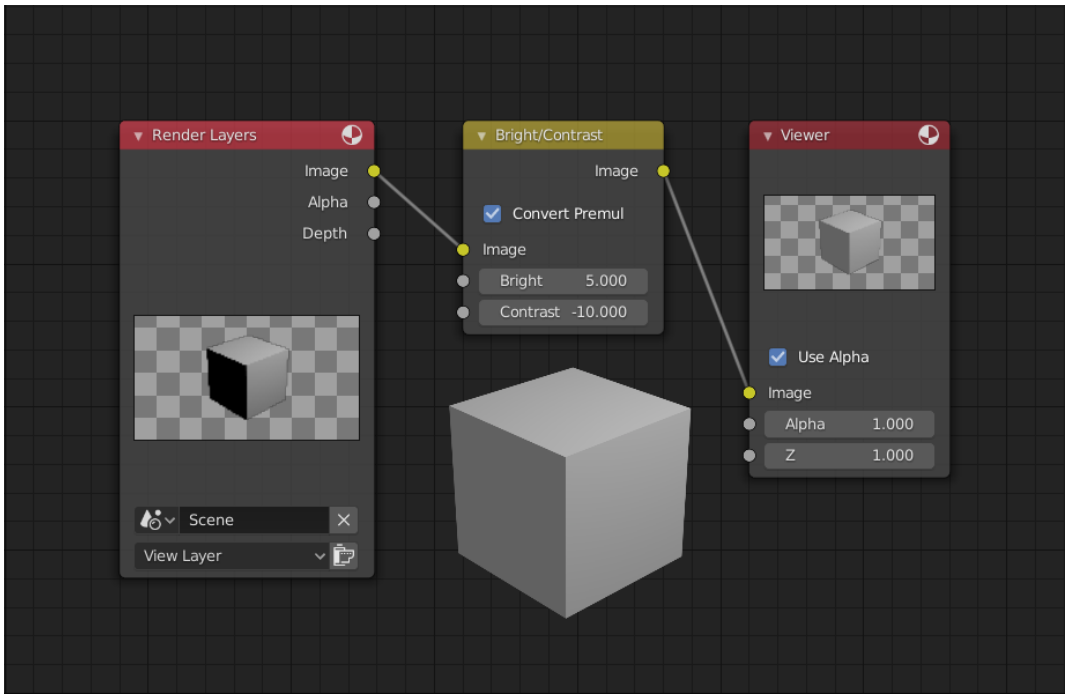
It is possible that this node will put out a value set that has values beyond the normal range, i.e. values greater than one and less than zero. If you will be using the output to mix with other images in the normal range, you should clamp the values using the Map Value node (with the Min and Max enabled), or put through a Color Ramp node (with all normal defaults).



Clamp the values to normal range.

Either of these nodes will scale the values back to normal range. In the example image, we want to intensify the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has a value much less than one in the dark areas; when added to the medium gray, it makes black. Passing the brightened image through either the Map Value or the Color Ramp node produces the desired effect.

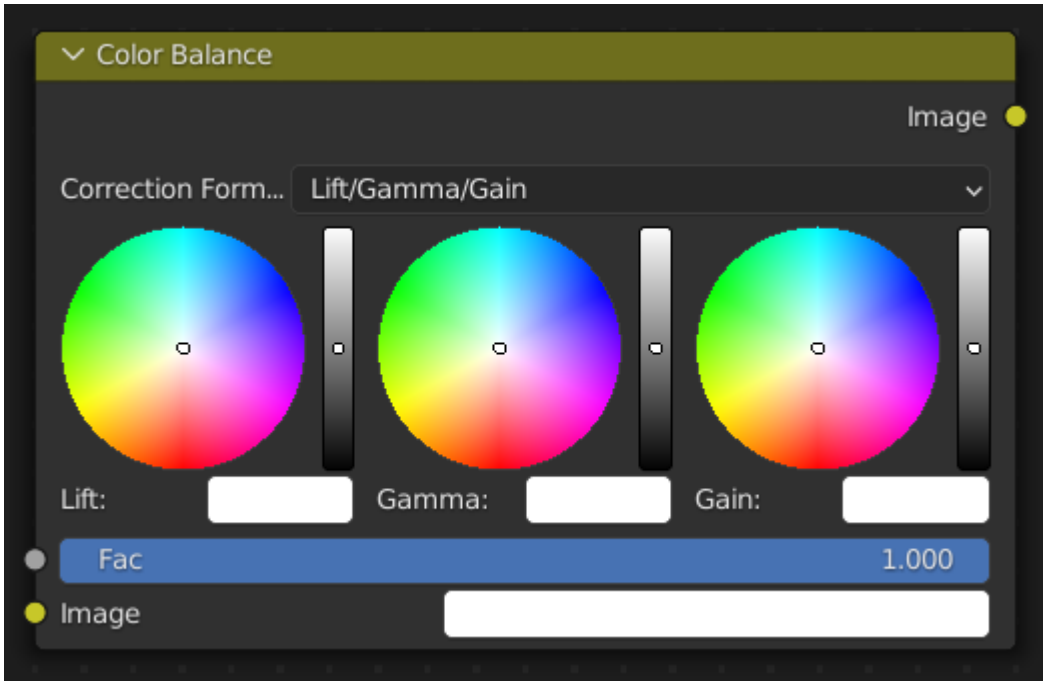
Example



A basic example.

Color Balance Node

The Color Balance node adjusts the color and values of an image.



Inputs

Factor
: Controls the amount of influence the node exerts on the output image.

Color
: Standard color input.

Properties

Correction Formula
: The mathematical method to adjust the image's colors.

Lift/Gamma/Gain:: Adjusts the colors and tonal range of an image by controlling the shadows, midtones, and highlights separately.

Lift
: Adjusts the darkest areas of the image (the shadows).

Gamma
: Primarily affects the midtones, the middle range of brightness in the image.

Gain
: Controls the brightest parts of the image (the highlights).

Offset/Power/Slope (ASC-CDL)::
A standardized model for adjusting the colors and tonal range of an image. This allows the same values to be used across different application to yield the same result. See Advanced for more details on the underlying implementation.

Offset
: Adjusts the darkest areas of the image (the shadows).

Basis
: Additional offset, allows to specify a negative offset value.

Power
: Primarily affects the midtones, the middle range of brightness in the image.

Slope
: Controls the brightest parts of the image (the highlights).

White Point:: Adjusts the color that should be considered white. The white point is specified as setting the inputs color temperature and then the desired output temperature.

Temperature

The blackbody temperature of the primary illuminant. By default a D65 white point is used.

Tint
: The amount of green/magenta shift of the blackbody curve.

Outputs

Color
: Standard output image.

Advanced

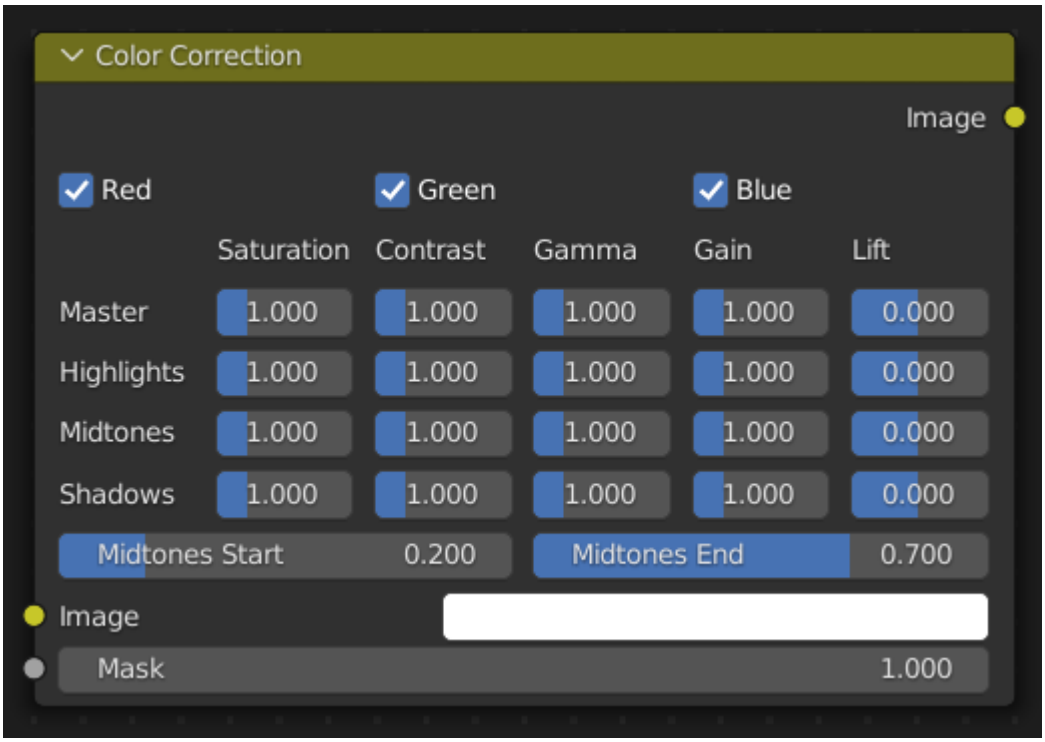
The Offset/Power/Slope Formula

$$\text{out} = (i \times s + o)^p$$

where:

- *out*: The color graded pixel code value.
- *i*: The input pixel code value (0 to 1) (black to white).
- *s*: Slope (any number 0 or greater, nominal value is 1.0).
- *o*: Offset (any number, the nominal value is 0).
- *p*: Power (any number greater than 0, nominal value is 1.0).

Color Correction Node



The Color Correction node adjusts the color of an image, separately in several tonal ranges (highlights, midtones and shadows).

Properties

Red, Green, Blue
: Specifies which RGB channels will be affected by the correction.

Correction Tools (Columns)

Saturation
: Adjusts the image's saturation.

Contrast
: Adjust image contrast.

Gamma
: Exponential gamma correction, affecting the midtones of the image. (Works like Power in the Color Balance node.)

Gain
: Multiplier, stronger influence on the highlights. (Works like Slope in the Color Balance node.)

Lift
: This value (can be negative) will be added (+), linear lightens or darkens the image. (Works like *Offset* in the Color Balance node.)

Tonal Ranges (Rows)

Master
: These sliders affect the entire tonal range.

Highlights
: These sliders only affect the highlights.

Midtones
: These sliders only affect the midtones.

Shadows
: Affects the dark tones of an image often affecting the shadows.

Midtones Start, Midtones End
: Defines the start and the end of midtones range, i.e. values where the whole tonal range is divided into the highlights, midtones and shadows (there is also a smooth transition between the ranges of width 0.2 units).

Inputs

Image
: Standard color input.

Mask

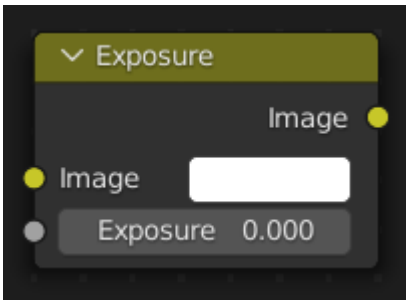
Controls the amount of influence the node exerts on the output image.

Outputs

Color
: Standard color output.

Exposure Node

The Exposure Node adjusts the brightness of an image using a camera exposure parameter.



See also

The exposure can also be adjusted in the scene Color Management.

Inputs

Image
: Standard color input.

Exposure
: Scalar factor to adjust the exposure of the image.

Properties

This node has no properties.

Outputs

Image
: Standard color output.

Examples

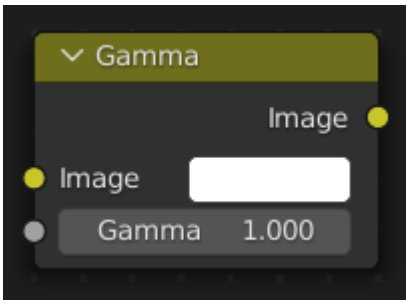
In the example below, the Exposure node is used to increase the brightness of the window area using a mask.



Example of an Exposure node.

Gamma Node

Use this node to apply a gamma correction.



Inputs

Image
: Standard color input.

Gamma
: An exponential brightness factor.

Properties

This node has no properties.

Outputs

Image
: Standard color output.

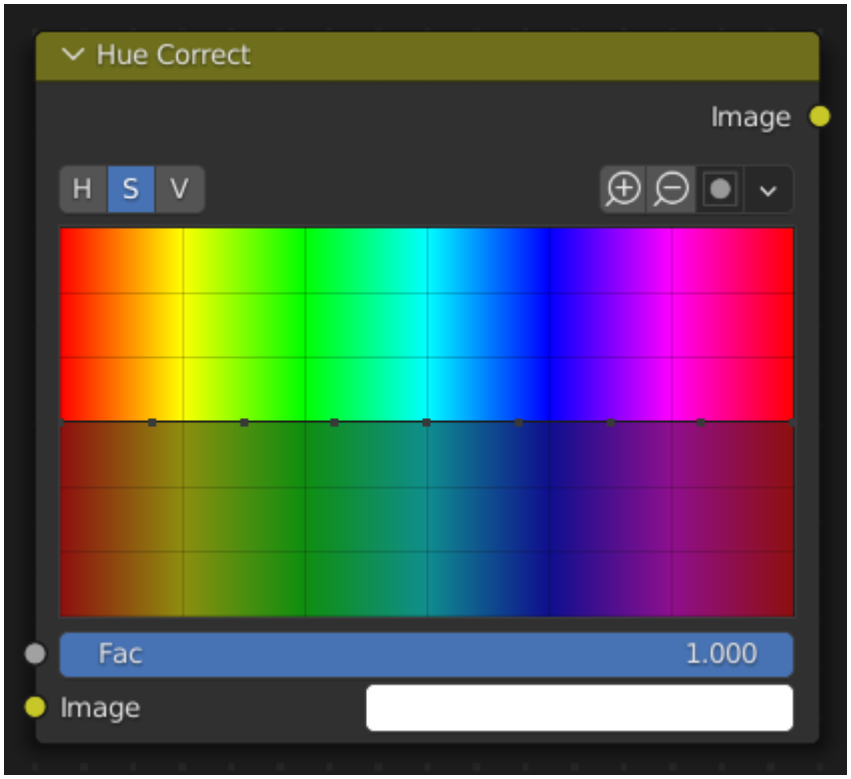
Examples



Example of a Gamma node.

Hue Correct Node

The *Hue Correct Node* adjusts the hue, saturation, and value of an image, with an input curve.



Inputs

Factor
: Controls the amount of influence the node exerts on the output image.

Image
: Standard color input.

Properties

Level

H (Hue), S (Saturation), V (Value)

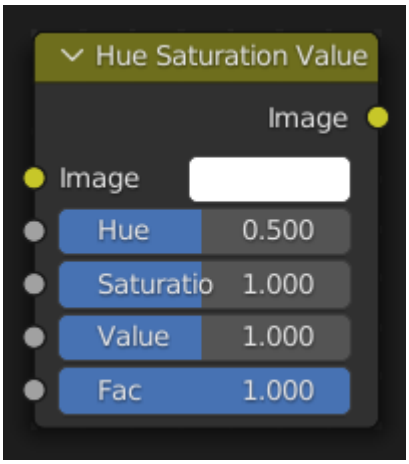
Curve
: For the curve controls see: Curve widget. By default, the curve is a straight line, meaning there is no change. The spectrum allows you to raise or lower HSV levels for each range of pixel colors. To change an H, S, or V level, move the curve points up or down. Pixels with hue values each point in the horizontal position of the graph will be changed depending on the shape of the curve.

Outputs

Image
: Standard color output.

Hue/Saturation/Value Node

The *Hue/Saturation/Value Node* applies a color transformation in the HSV Color Model.



Inputs

Factor
: The amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Hue
: The hue rotation offset, from 0 (-180°) to 1 (+180°). Note that 0 and 1 have the same result.

Saturation
: A value of 0 removes color from the image, making it black-and-white. A value greater than 1.0 increases saturation.

Value
: The value shift. 0 makes the color black, 1 keeps it the same, and higher values make it brighter.

Outputs

Image/Color
: Standard color output.

Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

Hues are laid out on a circle
: If you apply a Hue offset of 1 (+180°) to a blue image, you get the diametrically opposite color, which is yellow. If you apply a Hue offset of 1 to that yellow image, you get blue again.

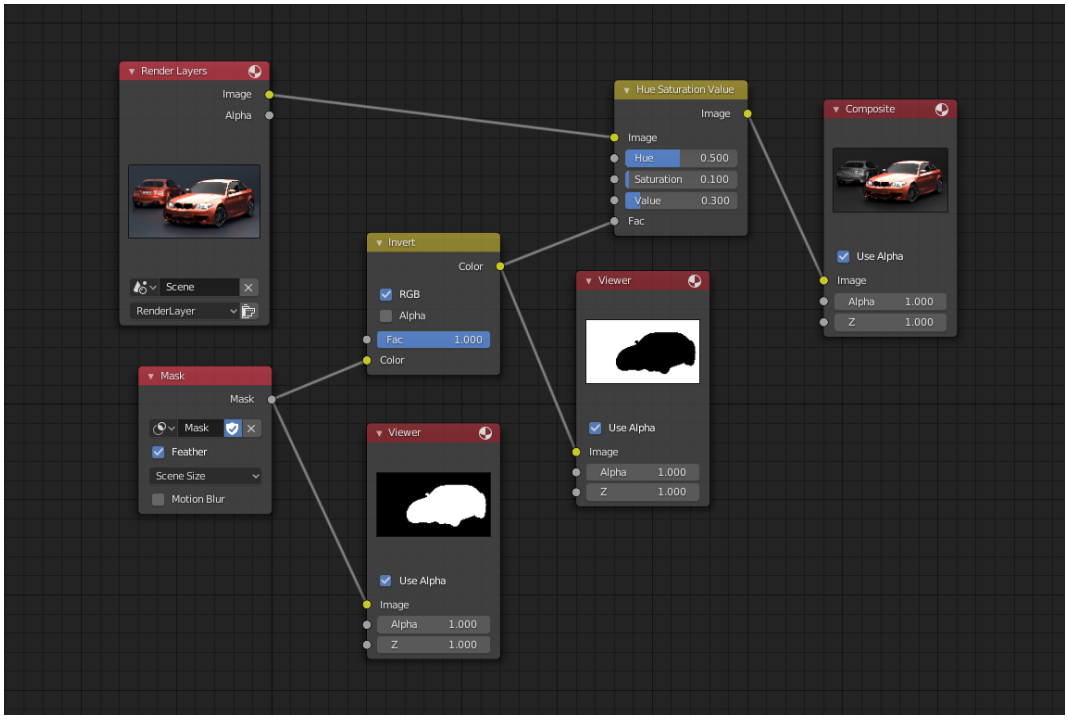
Grayscale images have no hue
: Trying to change the Hue or Saturation of a grayscale image has no effect. You can only brighten or darken it by adjusting the Value. To add color, use the Mix node instead.

Changing the effect over time
: The different values can be animated using a *Time Curve* node or by setting keyframes.

HSV Example



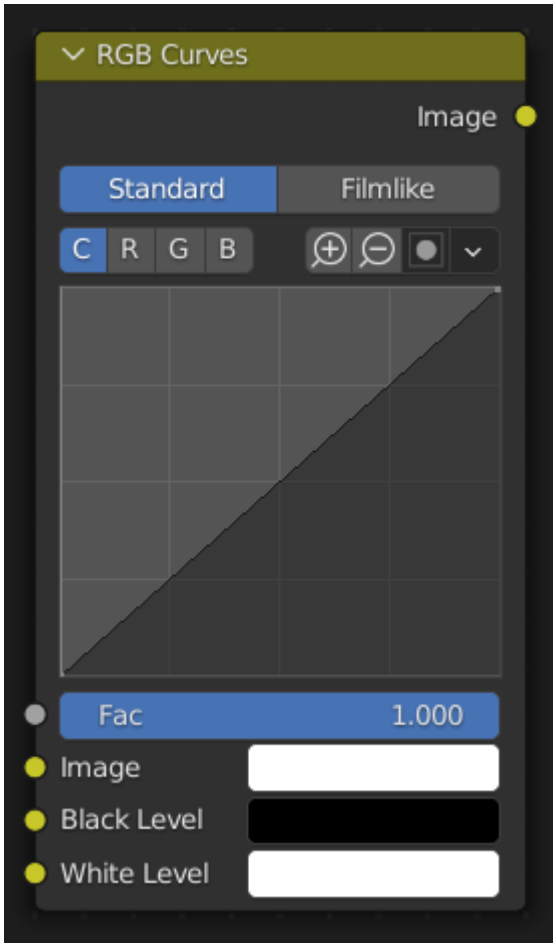
A basic example.



An example of using the Factor input for masking.

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.



Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*
: Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel
: The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve
: A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

Outputs

Image/Color
: Standard color output.

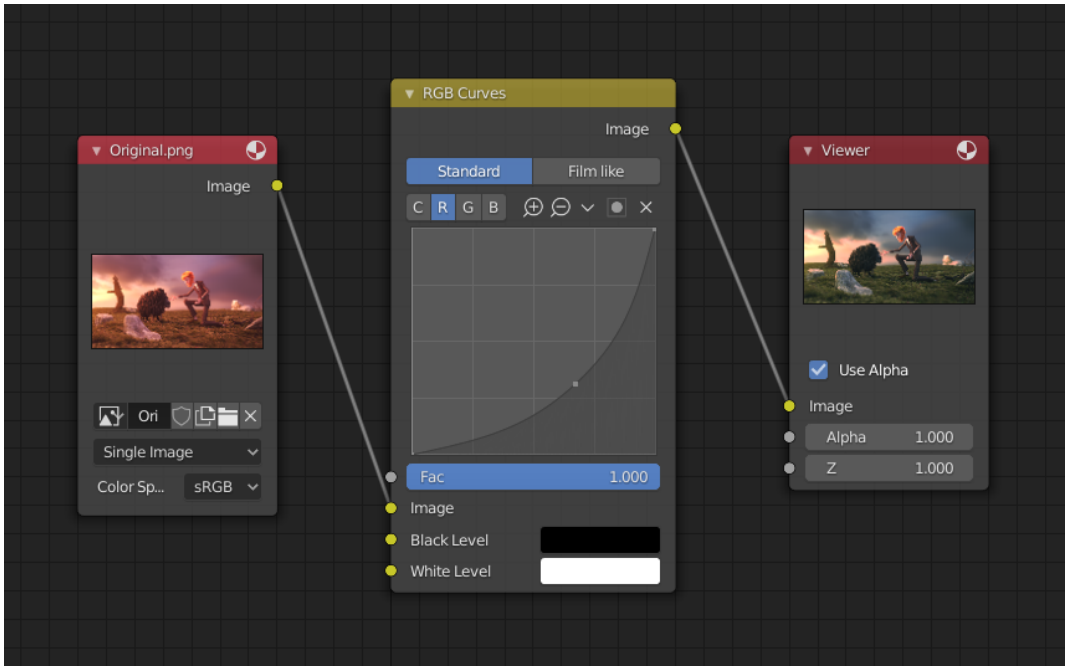
Examples

Below are some common curves you can use to achieve desired effects.



From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves

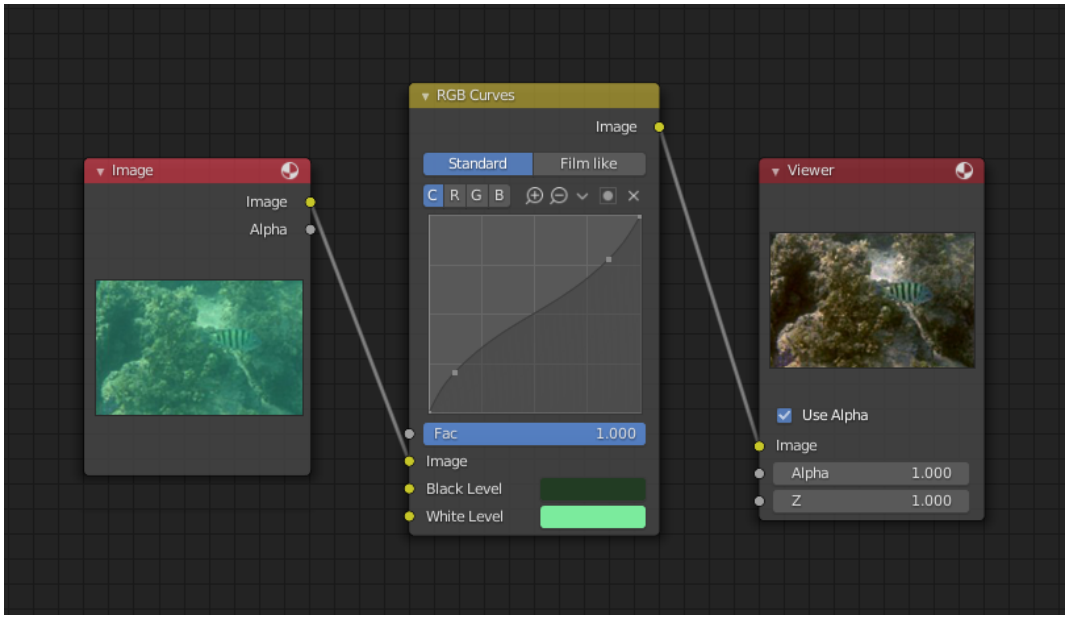


Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



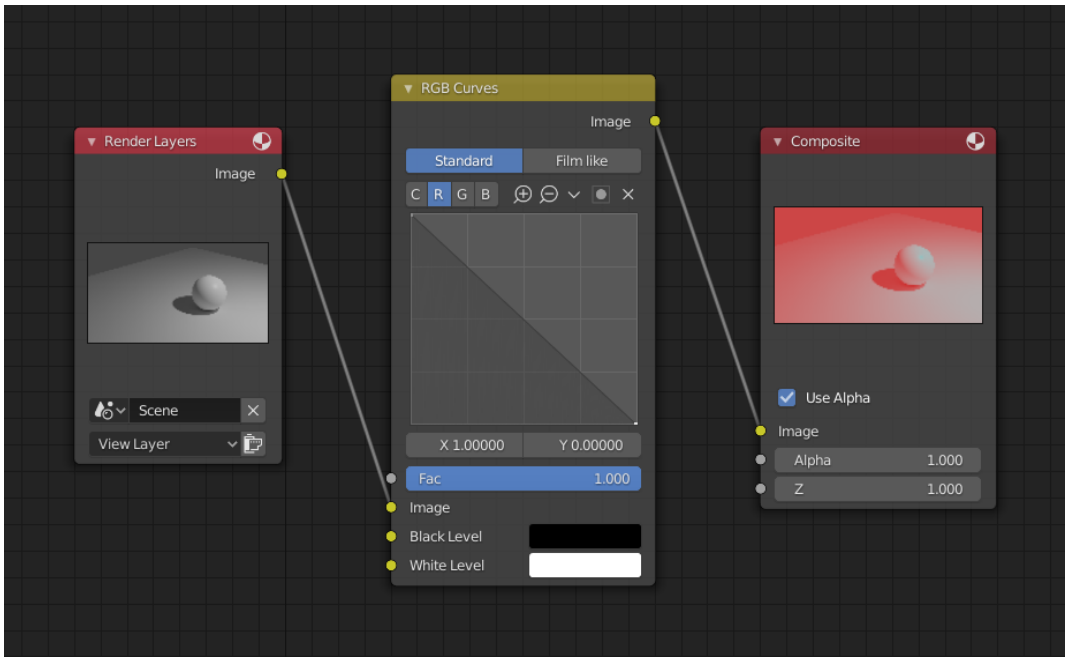
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

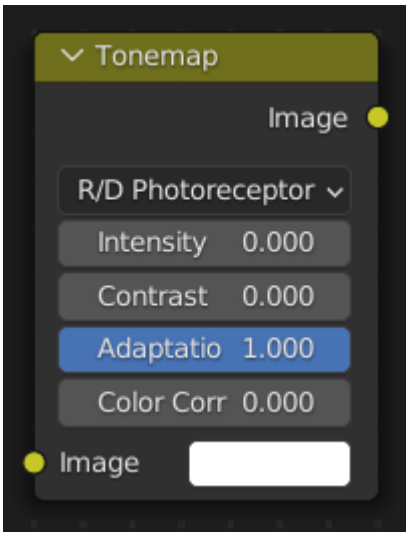
Effects



Changing colors by inverting the red channel.

Tone Map Node

Tone mapping is used to map high dynamic range colors into a more limited dynamic range supported by the display, while preserving the appearance as much as possible.



This is a legacy node. It is recommended to use view transforms in the color management settings instead, and output linear high dynamic range images from the compositor instead of low dynamic range.

Inputs

Image
: HDR image.

Properties

Type
: Rh Simple

  Key
  : The value the average luminance is mapped to.

  Offset
  : Normally always 1, but can be used as an extra control to alter the brightness curve.

  Gamma
  : If not used, set to 1.

R/D Photoreceptor

Intensity

If less than zero, darkens image; otherwise, makes it brighter.

Contrast

Set to 0 to use estimate from input image.

Adaptation

If 0, global; if 1, based on pixel intensity.

Color Correction

If 0, same for all channels; if 1, each independent.

Outputs

Image

LDR image.

Mix

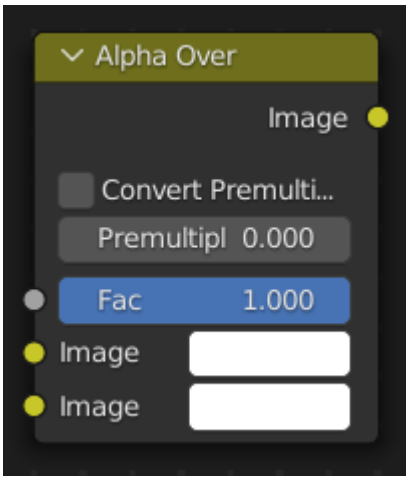
- Alpha Over Node

- Combine Color Node
- Separate Color Node

- Mix Color
- Z Combine Node

Alpha Over Node

The *Alpha Over* node is used to layer an image on top of another with alpha blending.



Inputs

Factor
: The alpha of the foreground image, going from 0 (fully transparent) to 1 (fully opaque).

Image
: The background image.

Image
: The foreground image.

Properties

Convert Premultiplied
: The *Alpha Over* node expects the foreground image to use Premultiplied Alpha. If it uses Straight Alpha instead, you can enable this checkbox to convert it.

Premultiplied

Interpolate between Premultiplied Alpha and Straight Alpha.

When set to 1, the foreground color values will be multiplied by the alpha; this is equivalent to enabling *Convert Premultiplied*. When set to 0, the color values do not change.

If *Premultiplied* is not zero, *Convert Premultiplied* will be ignored.

Note

This is a legacy option.

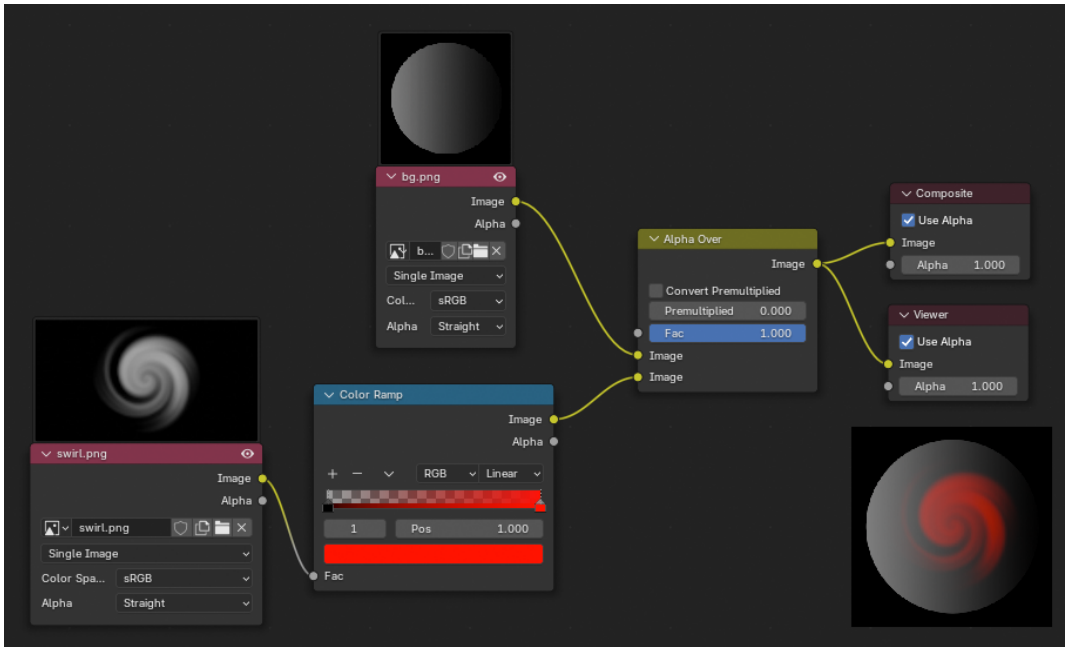
Outputs

Image
: The blended result.

Examples

Overlay

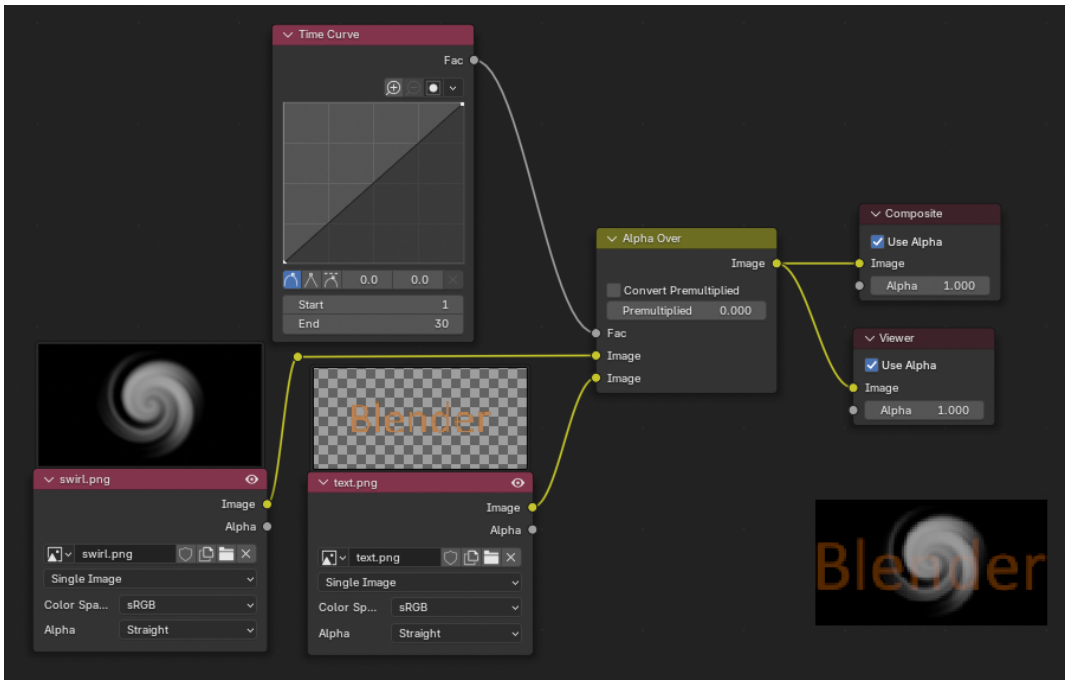
In the node tree below, the Color Ramp Node is used to convert an opaque, grayscale swirl image to a red one with transparency. Then, the *Alpha Over* node is used to overlay it on top of another image.



Assembling a composite image using Alpha Over.

Fade In

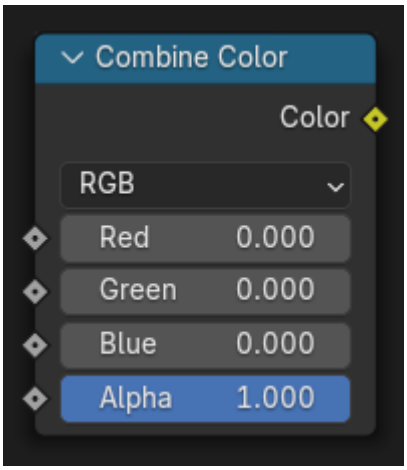
The example below uses the Time Curve Node to gradually increase the *Alpha Over* node's *Factor* from 0 to 1 over the course of 30 frames. This will result in the text fading in on top of the background image.



Animated fade in effect using Alpha Over.

Combine Color Node

The *Combine Color Node* combines an image from its composite color channels. The node can combine multiple Color Models depending on the Mode property.



Inputs

The outputs of this node depends on the Mode property (see below).

Alpha
: The color channel that is responsible for the image's transparency.

Properties

Mode
: The color model to output.

RGB:: Combine the three inputs: Red, Green, and Blue color channels into a single image.

HSV:: Combine the three inputs: Hue, Saturation, and Value color channels into a single image.

HSL:: Combine the three inputs: Hue, Saturation, and Lightness color channels into a single image.

YCbCrA::

Combine the three inputs: Luminance, Chrominance Blue, and Chrominance Red color channels into a single image.
Color Space
ITU 601, ITU 709, JPEG

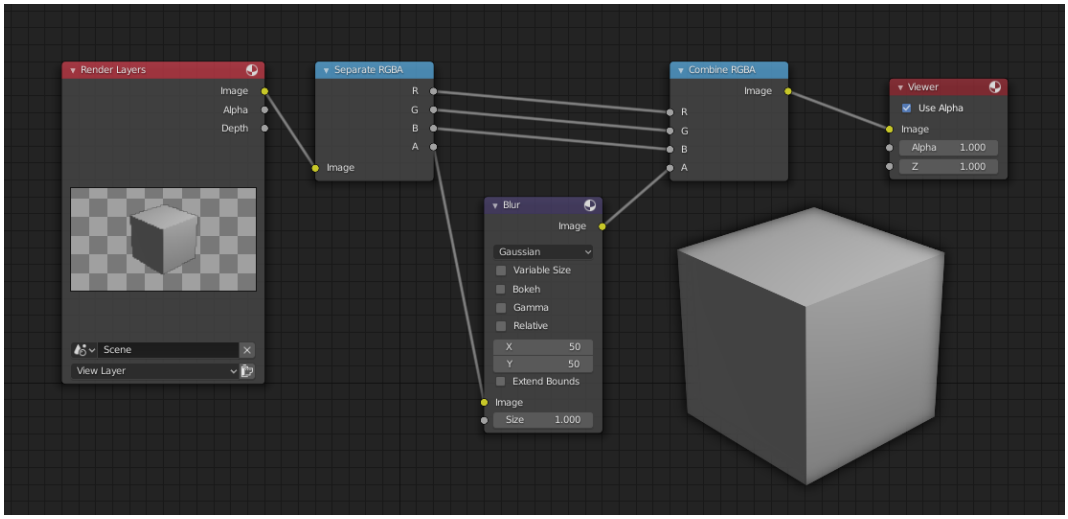
YUV:: Combine the three inputs: Luminance, U chrominance, and V chrominance color channels into a single image.

Output

Image
Standard image output.

Examples

Blur Alpha

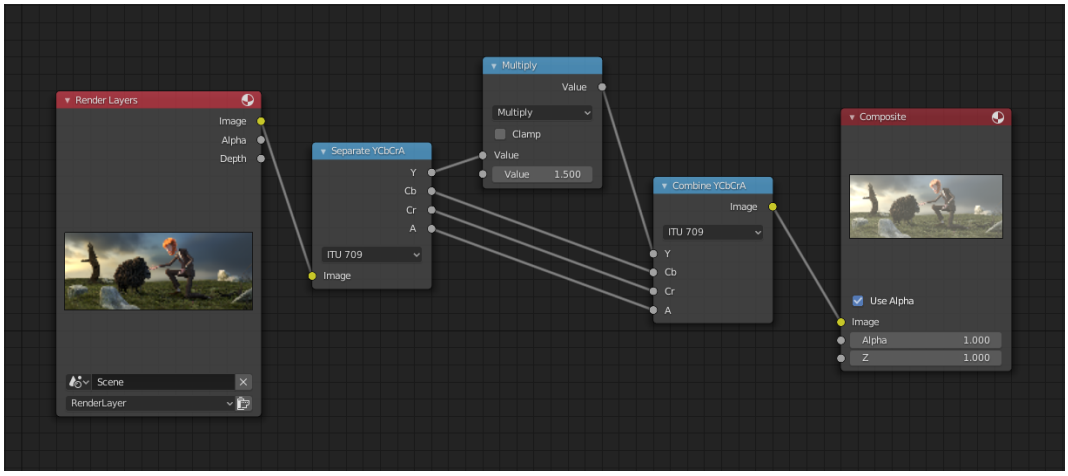


An example of blurring the alpha channel.

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like Anti-Aliasing

but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a broader scale will make the object appear to “phase” in and out, as an “out-of-phase” time-traveling sync effect.

Increase Luminance



An example of the scaling the Luminance channel.

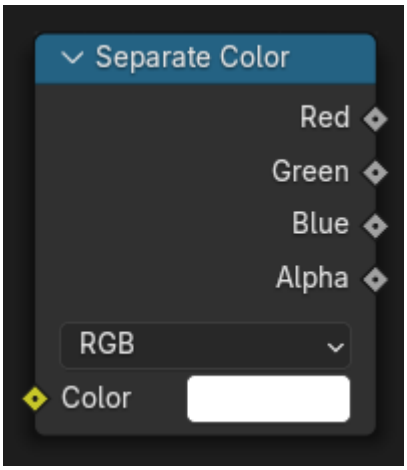
This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter.

Tip

If running these channels through a *Color Ramp* node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives a high-contrast effect.

Separate Color Node

The *Separate Color Node* splits an image into its composite color channels. The node can output multiple Color Models depending on the Mode property.



Inputs

Image
: Standard image input.

Properties

Mode
: The color model to output.

RGB:: Split the input image into it's three outputs: Red, Green, and Blue color channels.

HSV:: Split the input image into it's three outputs: Hue, Saturation, and Value color channels.

HSL:: Split the input image into it's three outputs: Hue, Saturation, and Lightness color channels.

YCbCrA:: Split the input image into it's three outputs: Luminance, Chrominance Blue, and Chrominance Red color channels.

Color Space
: ITU 601, ITU 709, JPEG

YUV:: Split the input image into it's three outputs: Luminance, U chrominance, and V chrominance color channels.

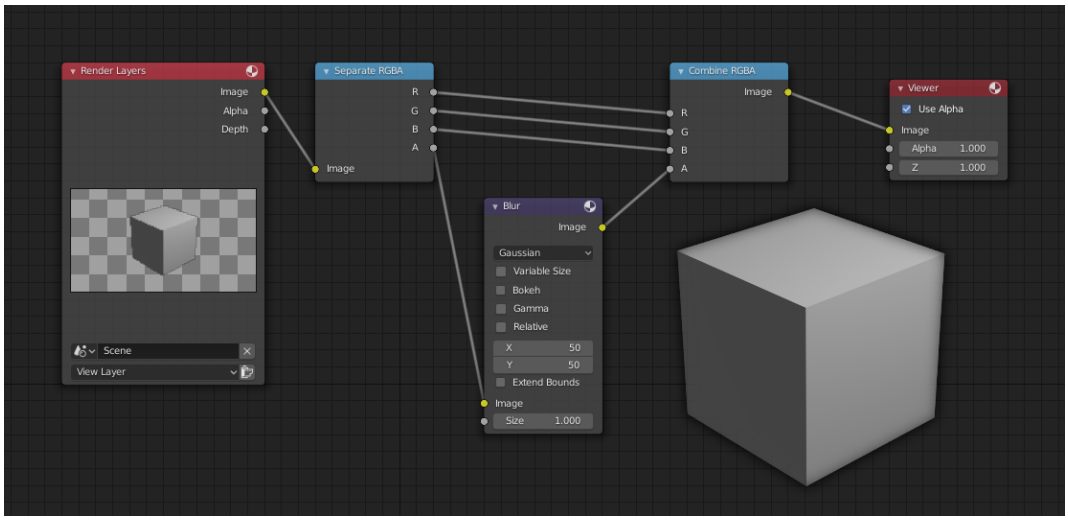
Outputs

The outputs of this node depends on the Mode property (see above).

Alpha
: The color channel that is responsible for the image's transparency.

Examples

Blur Alpha

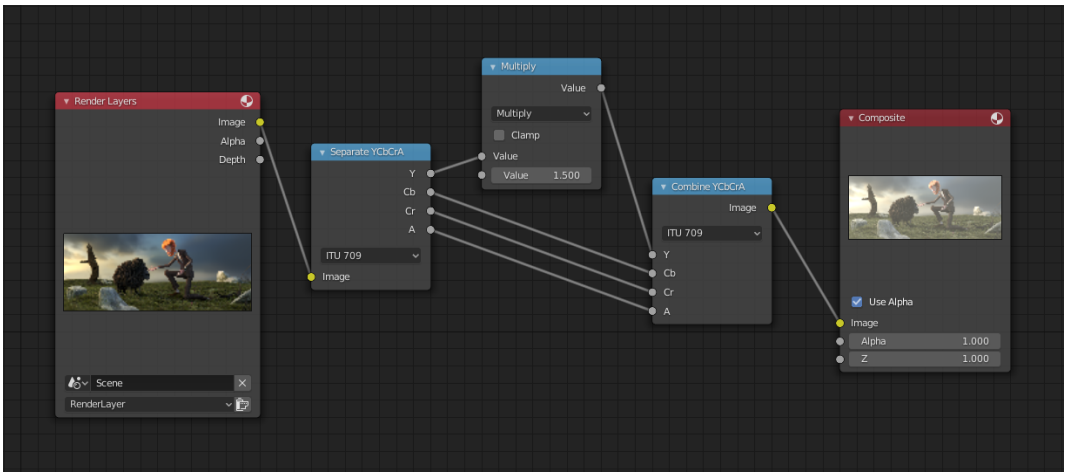


An example of blurring the alpha channel.

In this first example, we take the Alpha channel and blur it, and then combine it back with the colors. When placed in a scene, the edges of it will blend in, instead of having a hard edge. This is almost like Anti-Aliasing but in a three-dimensional sense. Use this node setup, when adding CG elements to live action to remove any hard edges. Animating this effect on a

broader scale will make the object appear to “phase” in and out, as an “out-of-phase” time-traveling sync effect.

Increase Luminance



An example of the scaling the Luminance channel.

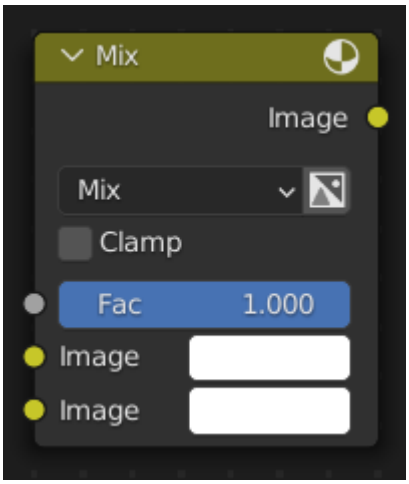
This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter.

Tip

If running these channels through a *Color Ramp* node to adjust value, use the Cardinal scale for accurate representation. Using the Exponential scale on the luminance channel gives a high-contrast effect.

Mix Node

Blends two images together, much like how an image editing program blends two layers.



Inputs

Fac
: The opacity of the foreground image.

Image
: The background image. Determines the dimensions of the output.

Image
: The foreground image.

Keep in mind that, unlike image editing programs where the foreground layer is on top, the foreground slot in Blender is on the bottom.

Properties

Blending Mode
: The blending mode to use.

  Mix

Regular alpha blending. Typically called *Normal* in image editing programs.

Darken
: For each color component, takes the smallest of the two values being blended.

Multiply
: Multiplies the colors component by component. Blending with a white pixel (value 1.0) has no effect, while blending with a black one (0.0) always results in black.

Color Burn
: Inverts the background color, divides it by the foreground color, and inverts the result.

Lighten
: For each color component, takes the largest of the two values being blended.

Screen
: Inverts both colors, multiplies them, and inverts the result.

Color Dodge
: Divides the background color by the inverted foreground color.

Add
: Adds the two colors together.

Overlay
: Applies Multiply blending if the foreground color's lightness is below 0.5, or Screen blending if it's above.

Soft Light
: Like Overlay, but more subtle.

Linear Light

Applies Linear Burn blending (background + foregound - 1) if the foreground color's lightness is below 0.5, or Linear Dodge (background + foreground) if it's above.

Difference
: For each component, subtracts the lower value from the higher value.

Exclusion
: Adds the two colors, then subtracts their multiple twice.

Subtract
: Subtracts the foreground color from the background color.

Divide
: Divides the background color by the foreground color.

Hue
: Combines the saturation and value of the background color with the hue of the foreground color.

Saturation
: Combines the hue and value of the background color with the saturation of the foreground color.

Color
: Combines the value of the background color with the hue and saturation of the foreground color.

Value
: Combines the hue and saturation of the background color with the value of the foreground color.

Use Alpha
: Whether to use the alpha channel of the foreground image during mixing. The alpha channel of the background image is always used.

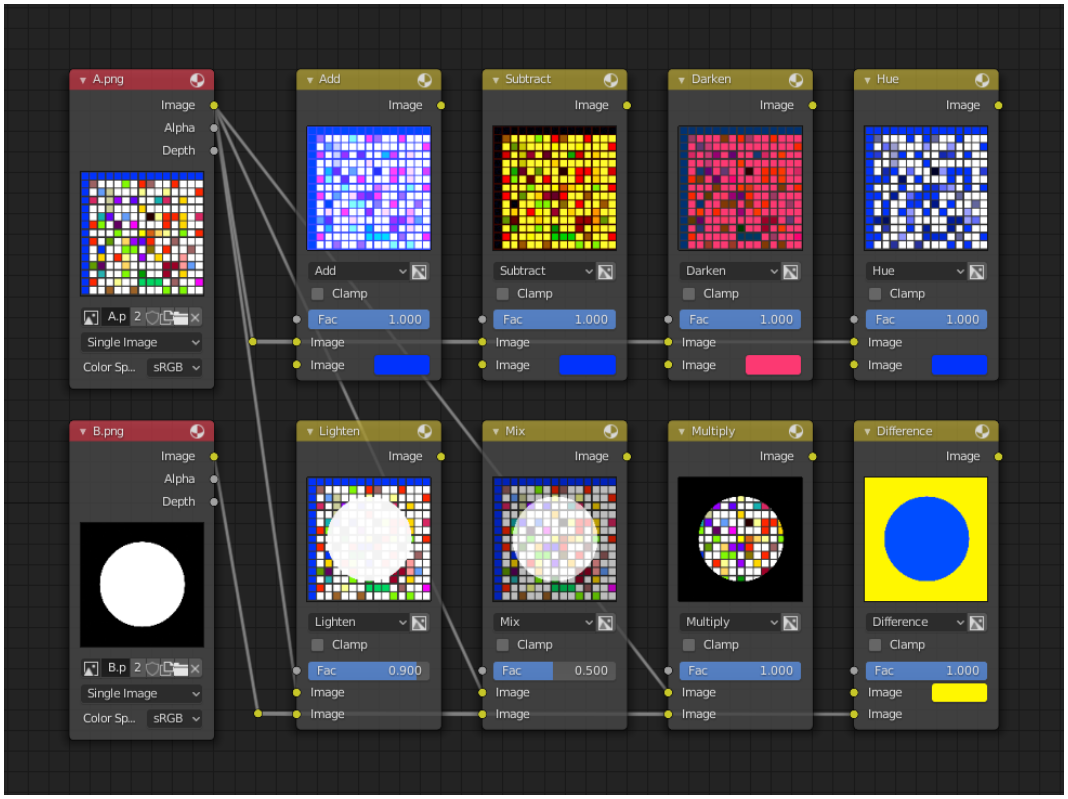
Clamp
: Clamp the output value to the [0.0, 1.0] range.

Outputs

Image
: The result of the mixing operation.

Examples

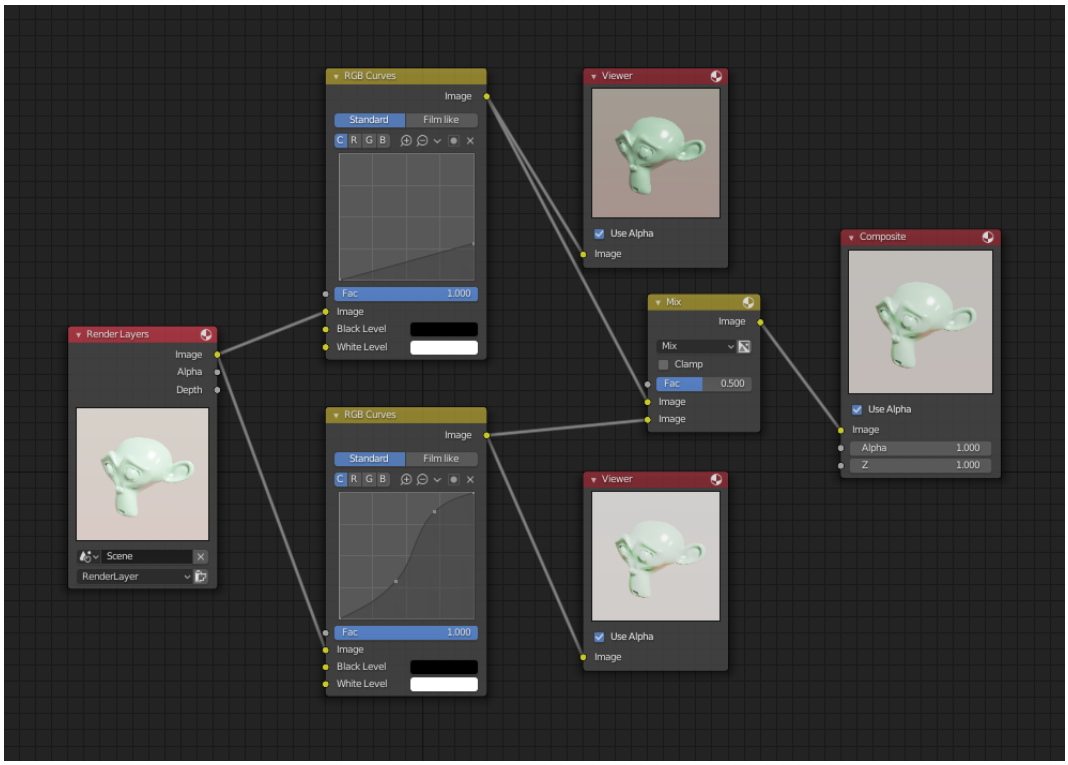
Below are examples of blending modes, as well as some practical use cases.



Blending a colored pattern with a flat color (top row) and a circular mask (bottom row).

Fixing overexposure

The Compositing setup below shows how to fix an overexposed render by darkening it and increasing contrast.



Example node setup showing two RGB Curves nodes and a Mix node for composition.

The top RGB Curves Node darkens the image by linearly scaling each color value to a smaller one.

The bottom curve node increases constract by making small values smaller and large values larger.

Finally, the Mix node blends the two together.

Watermark Images

In the old days, a pattern was pressed into the paper mush as it dried, creating a mark that identified who made the paper and where it came from. The mark was barely perceptible except in just the right light. Probably the first form of subliminal advertising.

Nowadays, people watermark their images to identify them as personal intellectual property, for subliminal advertising of the author or hosting

service, or simply to track their image's proliferation throughout the web.

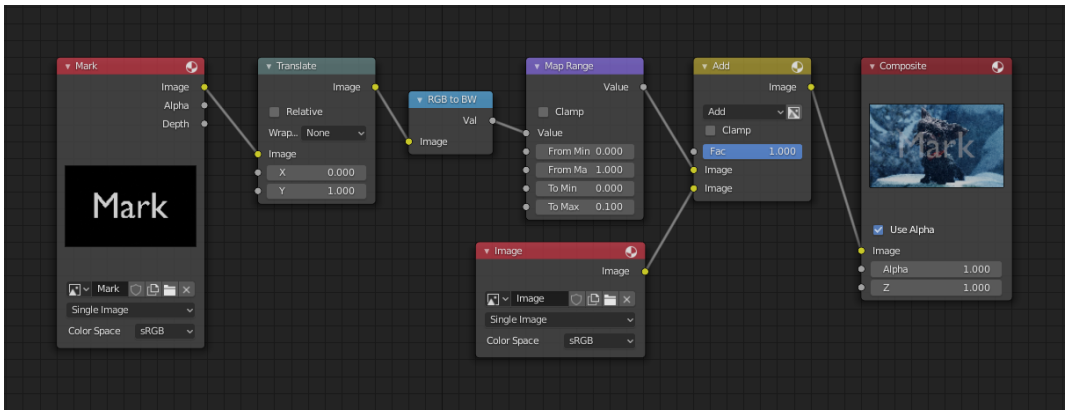
Blender provides a complete set of tools for you to both encode your watermark and to tell if an image has your watermark.

Encoding your Watermark in an Image

First, construct your own personal watermark. You can use your name, a word, or a shape or image not easily replicated. While neutral gray works best using the encoding method suggested, you are free to use other colors or patterns. It can be a single pixel or a whole gradient; it is up to you.

In the example below, we are encoding the watermark in a specific location in the image using the *Translate* node; this helps later because we only have to look at a specific location for the mark. We then use the *RGB to BW node* to convert the color image to grayscale numbers, which we then feed into the *Map Range* node to reduce the mark to one-tenth of its original intensity.

The *Add* node (*Mix* node with blending mode *Add*) adds the corresponding pixels, making the ones containing the mark ever-so-slightly brighter.

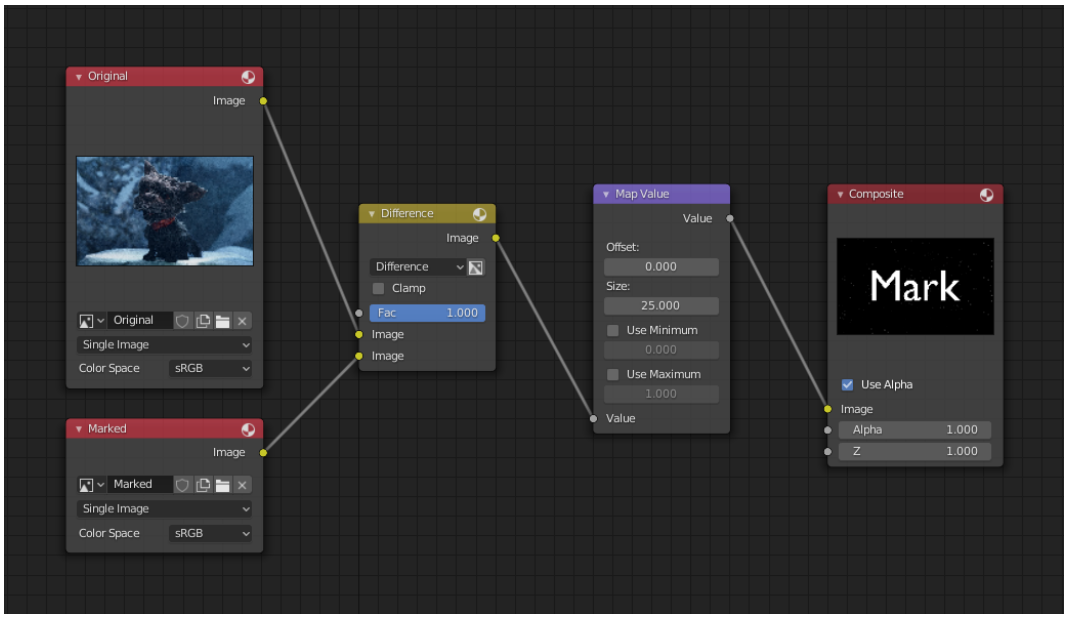


Embedding a watermark in an image.

Of course, if you *want* people to notice your mark, do not scale it so much, or make it a contrasting color. There are also many other ways, using other mix settings and fancier rigs. Feel free to experiment!

Decoding an Image for your Watermark

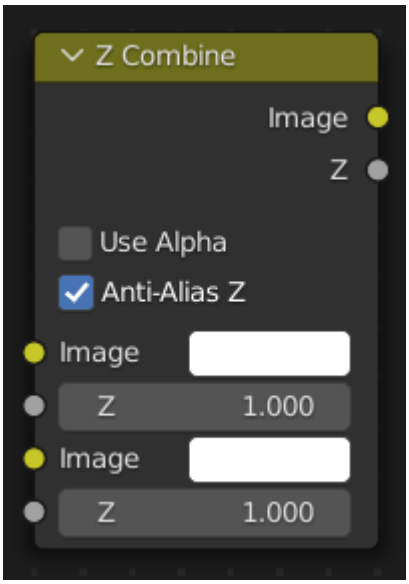
When you see an image that you think might be yours, use the node tree below to compare it to your stock image (pre-watermarked original). In this tree, the *Mix* node is set to Difference, and the *Map Value* node amplifies any difference. You can see how the original mark clearly stands out.



Checking an image for your watermark.

Z Combine Node

The Z Combine node combines two images based on their Z-depth maps. It overlays the images using the provided Z values to detect which parts of one image are in front of the other.



Inputs

Image
: The background image.

Z
: Z depth of the background image.

Image
: The foreground image.

Z
: Z depth of the foreground image.

Properties

Use Alpha

The chosen Image pixel alpha channel is also carried over. If a pixel is partially or totally transparent, the result of the Z Combine will also be partially transparent; in which case the background image will show through the foreground (chosen) pixel.

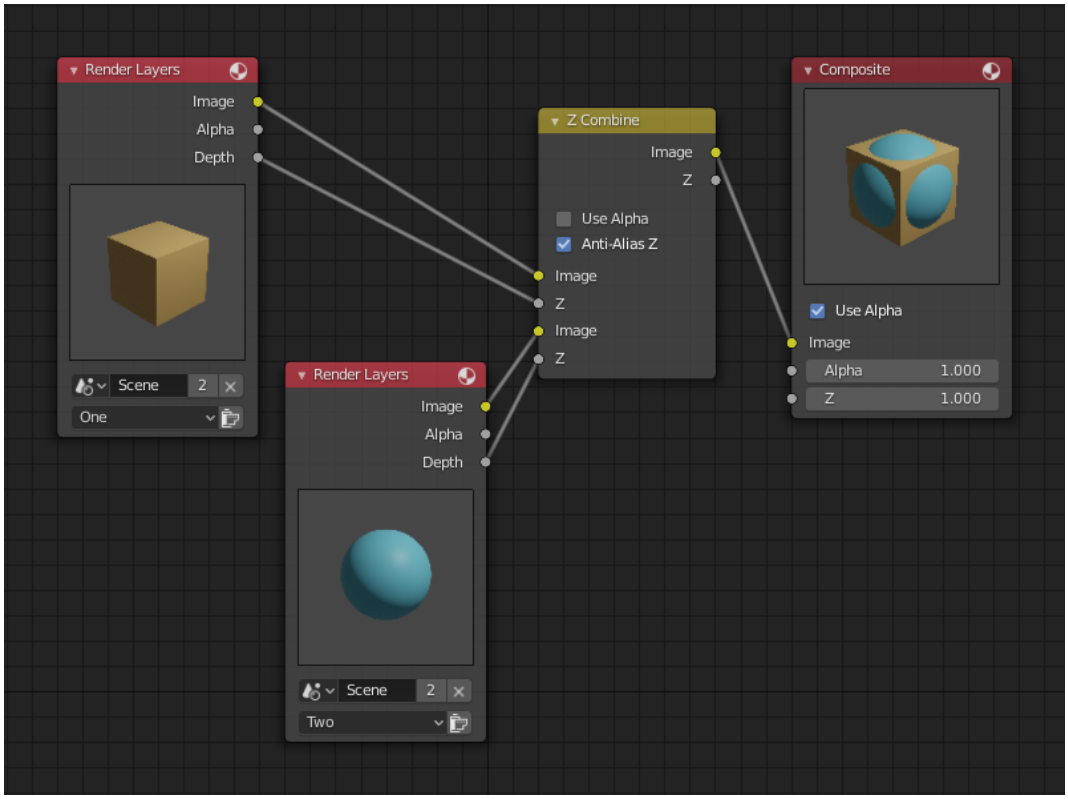
Anti-Alias Z
: Applies Anti-Aliasing to avoid artifacts at sharp edges or areas with a high contrast.

Outputs

Image
: If both Z values are equal, it will use the foreground image. Whichever Z value is less decides which image pixel is used. See Z-buffer.

Z
: The combined Z depth, which allows to thread multiple Z-combines together.

Examples



Choosing closest pixels.

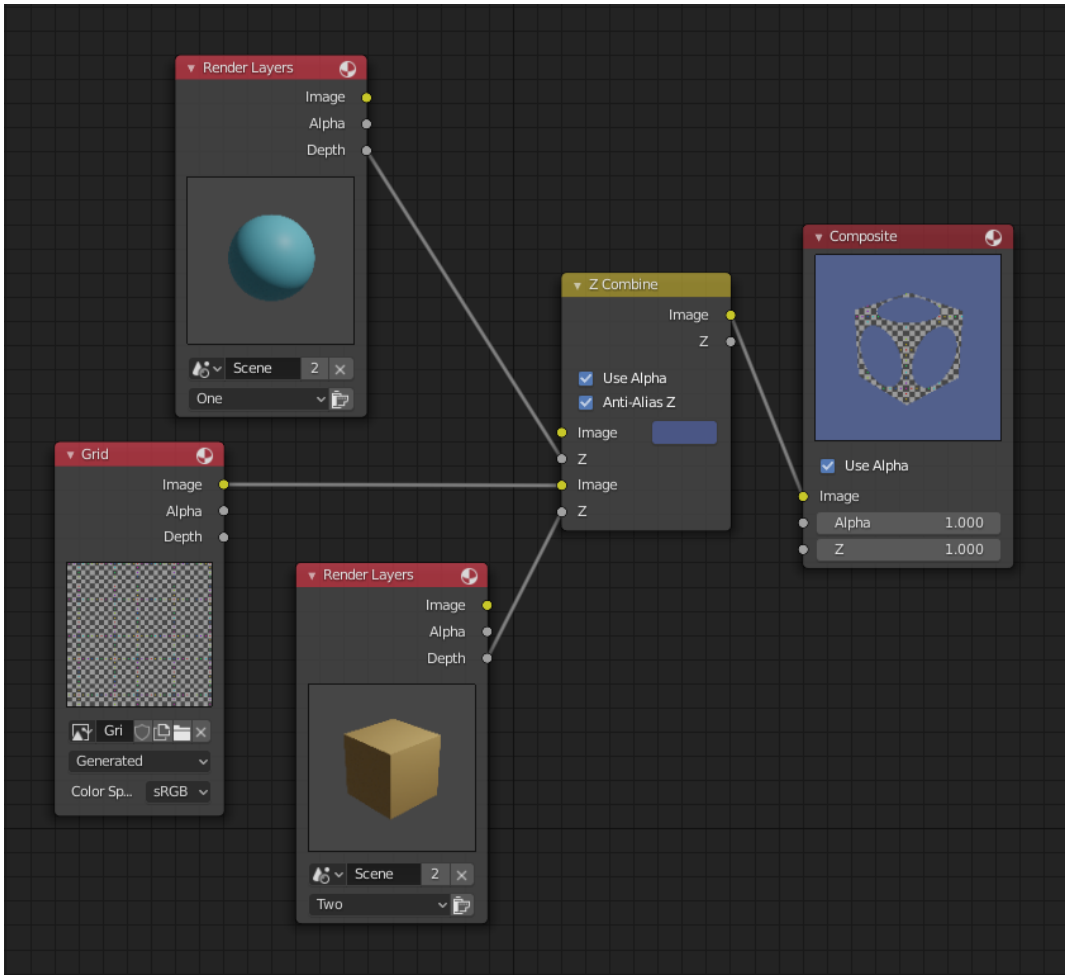
In the example above, the render output from two scenes are mixed using the Z Combine node, one from a sphere of size 1.3, and the other a cube of size 1.0. The sphere and square are located at the same place. The cube is tipped forward, so the corner in the center is closer to the camera than the sphere surface; so Z Combine chooses to use the cube's pixels. But the sphere is slightly larger (a size of 1.3 versus 1.0), so it does not fit totally inside the cube. At some point, as the cube's sides recede back away from the camera, the sphere's sides are closer. When this happens, Z Combine uses the sphere's pixels to form the resulting picture.

This node can be used to combine a foreground with a background matte painting. Walt Disney pioneered the use of multi-plane mattes, where three or four partial mattes were painted on glass and placed on the left and right at different Z positions; minimal camera moves to the right created the illusion of depth as Bambi moved through the forest.

Note

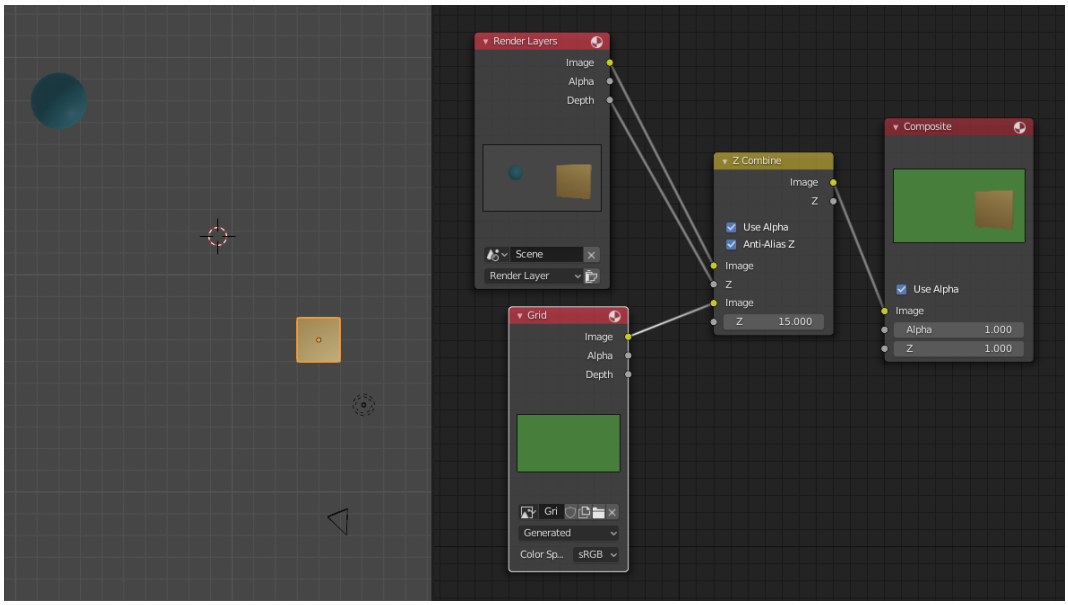
Valid Input

Z Input Sockets do not accept fixed values; they must get a vector set (see Map Value node). Image Input Sockets will not accept a color since they do not have UV coordinates.



Mix and match images.

The Z Combine can be used to merge two images as well. Using the Z values from the sphere and cube scenes above, but inputting different images, yields the example to the right.



Z Combine in action.

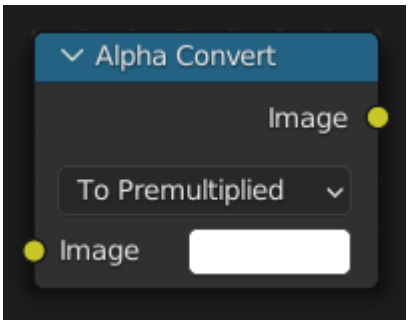
In this node setup a render scene is mixed with a flat image. In the side view of the scene, the orange cube is 10 units away from the camera, and the blue ball is 20. The 3D cursor is about 15 units away from the camera. The image is Z-in at a location of 15, thus inserting it in between the cube and the ball. The resulting image appears to have the cube on the green image.

Note

Invisible Man Effect

If a foreground image with a higher Alpha than the background, is then mixed in the Z Combine with a slightly magnified background, the outline of the transparent area will distort the background, enough to make it look like seeing a part of the background through an invisible yet Fresnel-lens object.

Alpha Convert Node



The *Alpha Convert Node* converts the alpha channel format of an image.

For compositing and rendering, Premultiplied Alpha is the standard in Blender. Render layers will be premultiplied alpha, and images loaded into rendering or compositing will be converted to this.

If you want to do a compositing operation with straight alpha, the *Alpha Convert* node can be used. Typically this would be a color correction operation where it might give better results working on RGB channels without alpha. If the alpha is converted to straight in the Compositor, it should be converted back to premultiplied before the *Composite Output* node, otherwise some artifacts might occur.

Inputs

Image
: Standard color input.

Properties

Mapping
: The direction of convert alpha. For details on the difference between both ways to store alpha values see Alpha Channel.

To Premultiplied:: Converts from Straight Alpha to Premultiplied Alpha.

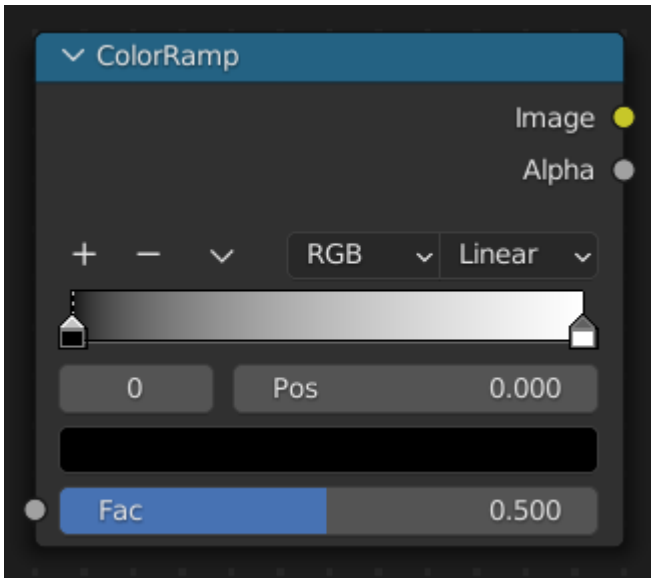
To Straight:: Converts from Premultiplied Alpha to Straight Alpha.

Outputs

Image
: Standard color output.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



Inputs

Factor
: The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp
: See Color Ramp Widget.

Outputs

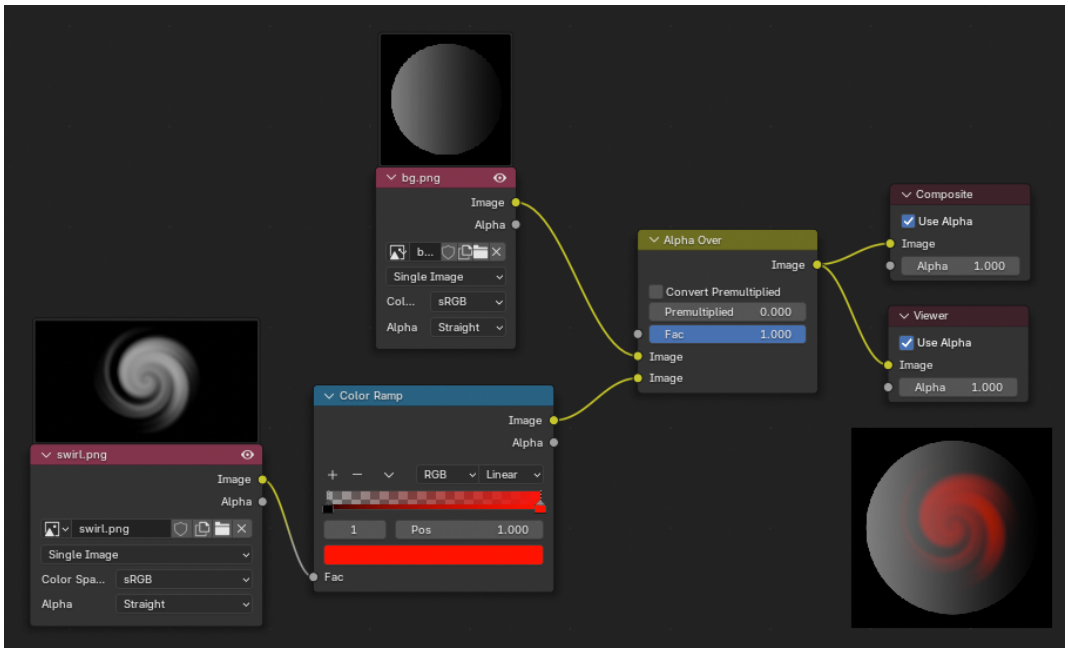
Image/Color
: Standard color output.

Alpha
: Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

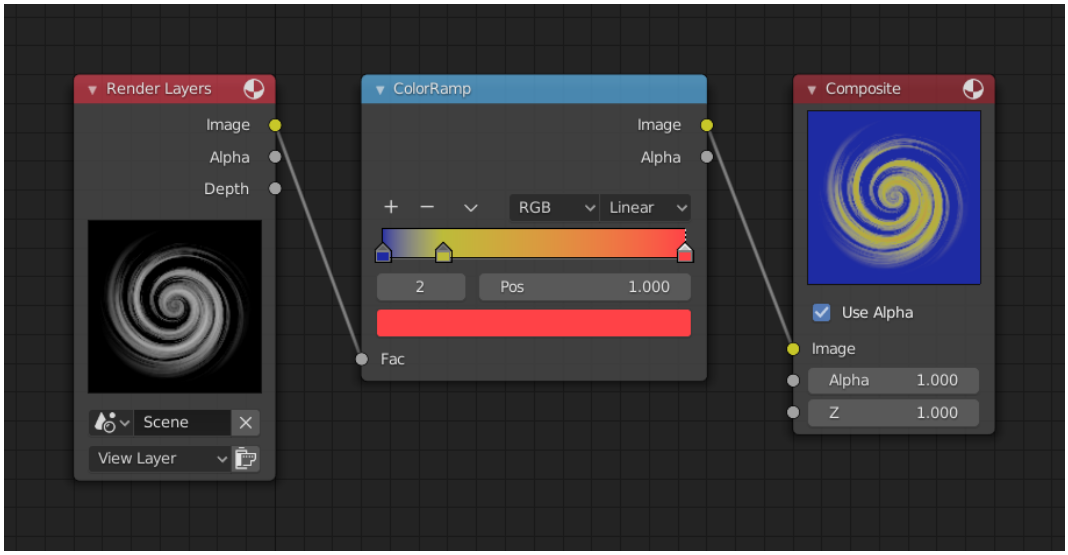


In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

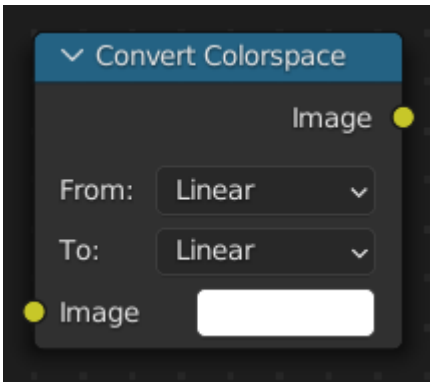
In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.



The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Convert Colorspace Node

The *Convert Colorspace* node converts images between color spaces.



Note

Images are already automatically converted into linear color space unless specified in the image's Color Space option.

Inputs

Image
: Standard color input.

Properties

From, To
: The color space of the input image and the color space to convert it to.

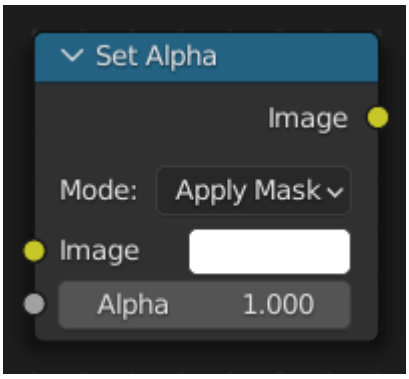
  The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Outputs

Image
: Standard color output.

Set Alpha Node

The *Set Alpha Node* adds an alpha channel to an image.



Inputs

Image
: Standard color input.

Alpha
: The amount of Alpha can be set for the whole image by using the input field or per pixel by connecting to the socket.

Properties

Mode
: Method to mix the alpha with the input *Image*.

Apply Mask:: Multiply the input images RGBA channels by the *Alpha* input value. In this cases, the output is using Premultiplied Alpha.

Replace Alpha:: Replace the inputs alpha channel with the *Alpha* input value. In this cases, the output is using Straight Alpha.

Outputs

Image
: Standard color output.

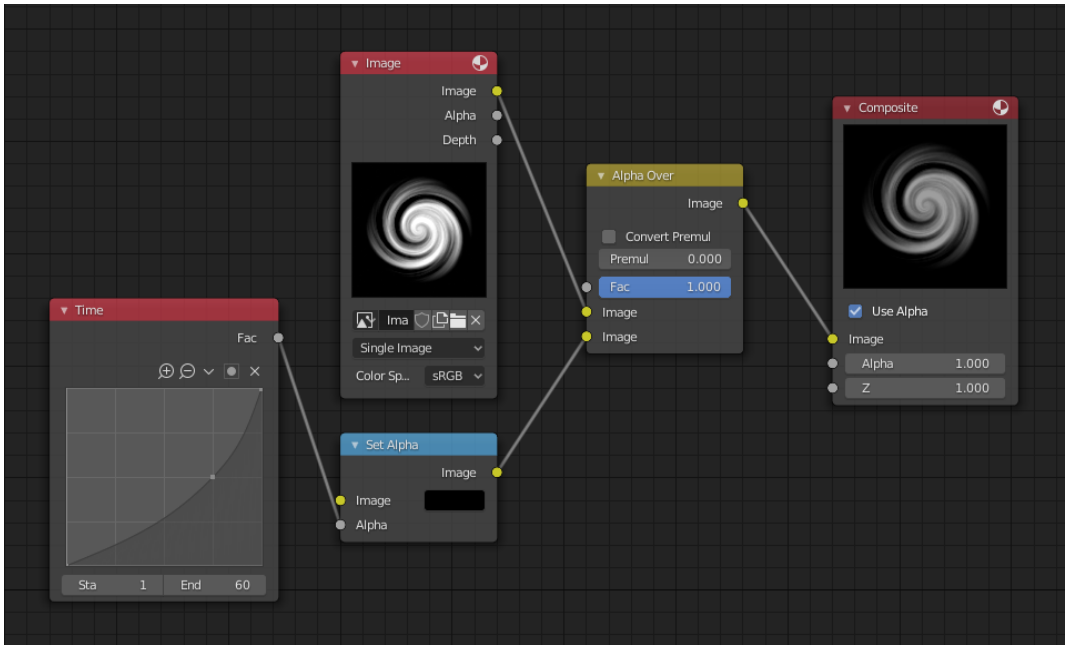
Note

This is not, and is not intended to be, a general-purpose solution to the problem of compositing an image that does not contain alpha information. You might wish to use “chroma keying” or “difference keying” (as discussed elsewhere) if you can. This node is most often used (with a suitable input being provided by means of the socket) in those troublesome cases when you *cannot*, for some reason, use those techniques directly.

Example

Fade To Black

To transition the audience from one scene or shot to another, a common technique is to “fade to black”. As its name implies, the scene fades to a black screen. You can also “fade to white” or whatever color you wish, but black is a good neutral color that is easy on the eyes and intellectually “resets” the viewer’s mind. The node tree below shows how to do this using the Set Alpha node.



Fade to black.

In the example above, the alpha channel of the swirl image is ignored. Instead, a Time Curve node introduces a factor from 0.0 to 1.0 over 60 frames, or about 2 seconds, to the Set Alpha node. Note that the time curve is exponentially-shaped, so that the overall blackness will fade in slowly and then accelerate toward the end. The Set Alpha node does not need an input image; instead, the flat (shadeless) black color is used. The Set Alpha Node uses the input factor and color to create a black image that has an alpha set which goes from 0.0 to 1.0 over 60 frames, or completely transparent to completely opaque. Think of alpha as a multiplier for how vivid you can see that pixel. These two images are combined by the Alpha Over node completely (a *Factor* of 1.0) to produce the composite image. The Set Alpha node will thus, depending on the frame being rendered, produce a black image that has some amount of transparency. Setup and animate, and you have an image sequence that fades to black over a two-second period.

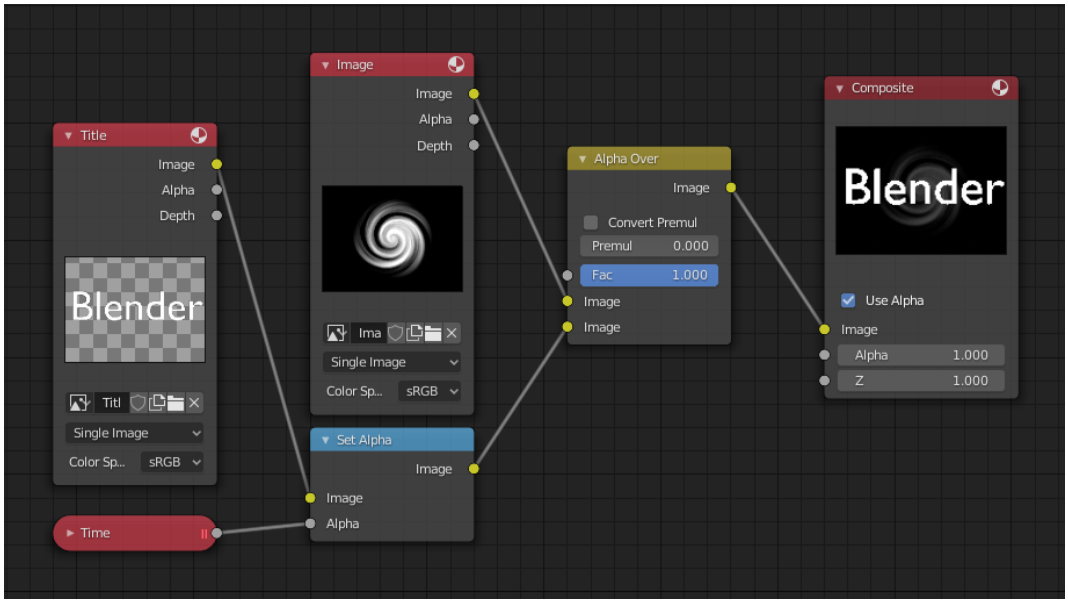
Note

No Scene Information Used

This example node tree does not use the Render Layer node. To produce this 2-second animation, no Blender scene information was used. This is an example of using Blender's powerful compositing abilities separate from its modeling and animation capabilities. (A Render Layer could be substituted for the Image layer, and the "fade-network" effect will still produce the same effect.)

Fade In a Title

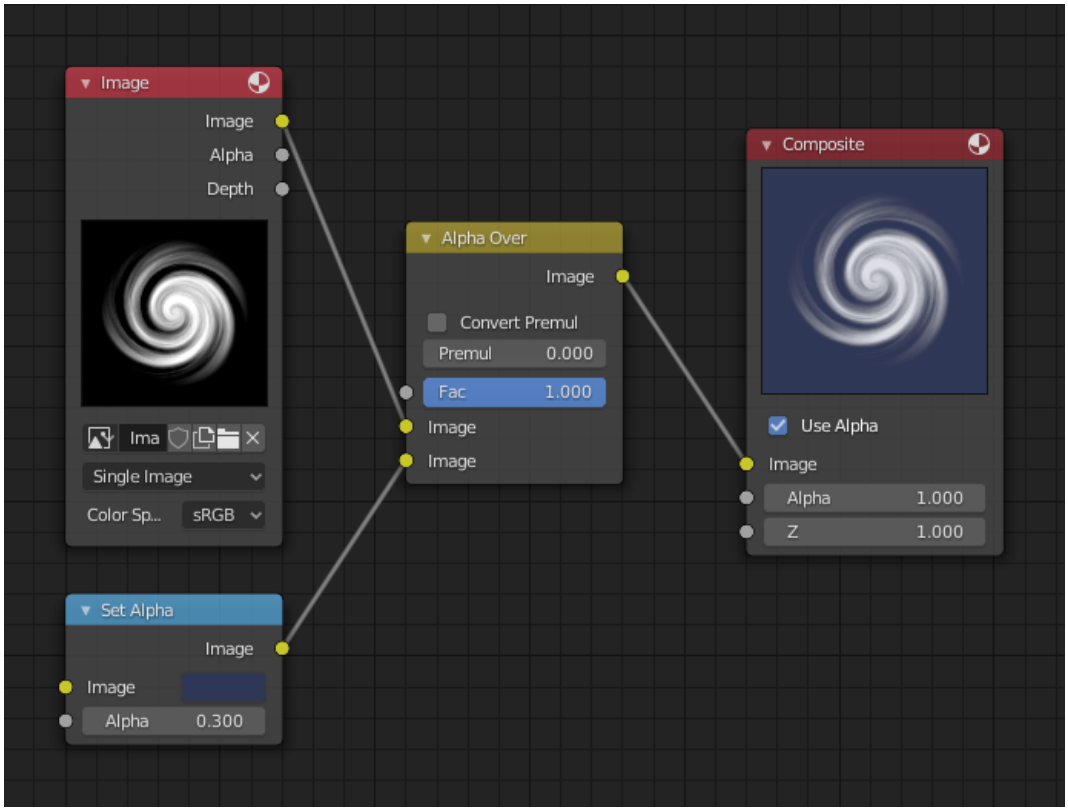
To introduce your animation, you will want to present the title of your animation over a background. You can have the title fly in, or fade it in. To fade it in, use the Set Alpha node with the Time Curve node as shown below.



Using Set Alpha to fade in a title.

In the above example, a Time curve provides the Alpha value to the input socket. The current Render Layer node, which has the title in view, provides the image. As before, the Alpha Over node mixes (using the alpha values) the background swirl and the alpha title to produce the composite image.

Colorizing a BW Image



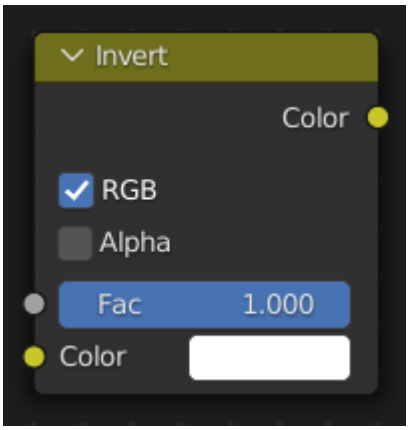
Using Set Alpha to colorize an image.

In the example above, notice how the blue tinge of the render input colors the swirl. You can use the Set Alpha node's color field with this kind of node tree to add a consistent color to a BW image.

In the example tree to the right, use the *Alpha* value of the Set Alpha node to give a desired degree of colorization. Thread the input image and the Set Alpha node into an Alpha Over node to colorize any black-and-white image in this manner.

Invert Color Node

Inverts the colors in the input image, producing a negative.



Inputs

Factor
: The amount of influence the node exerts on the image.

Color
: Standard color input.

Properties

In the compositing context, this node has the following properties:

RGB
: Invert the color channels.

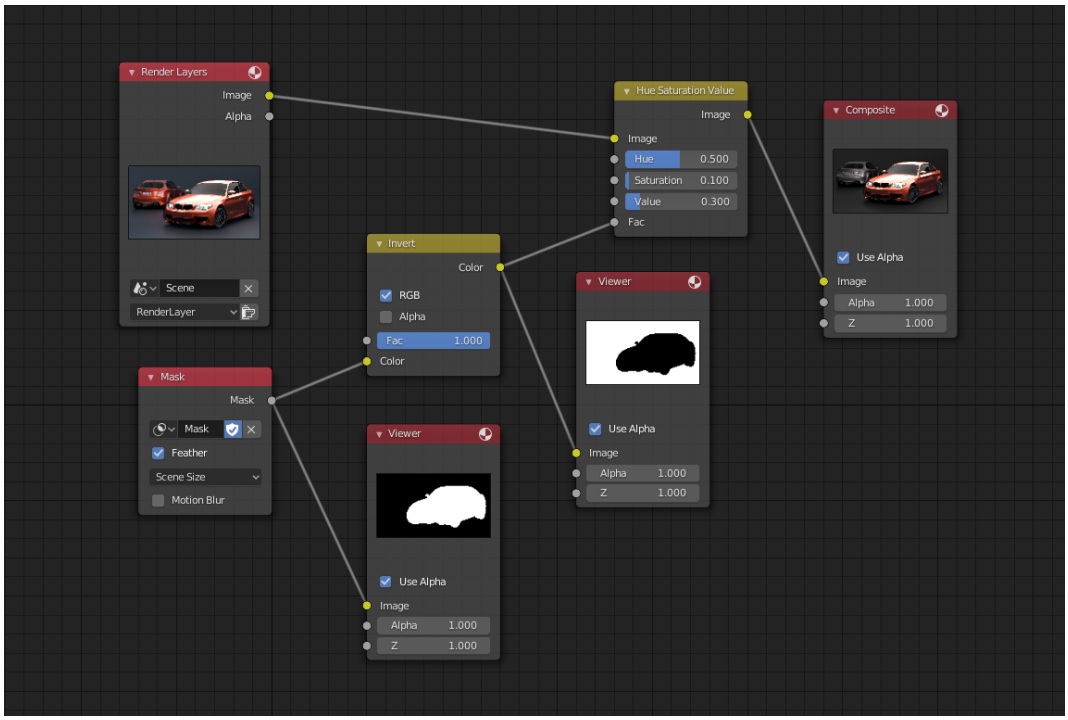
Alpha
: Invert the alpha channel.

Outputs

Color

Standard color output.

Example



The Invert node is used to invert the mask.

RGB to BW Node



The *RGB to BW Node* makes a color image black-and-white by outputting its luminance.

Note

You can directly connect Color sockets to Value sockets in node graphs, which also converts the image to black-and-white. As such, this node is not always necessary.

Inputs

Image
: Color image input.

Properties

This node has no properties.

Outputs

Value
: Grayscale value output.

Filter Nodes

Filters process the pixels of an image to highlight additional details or perform some sort of post-processing effect on the image.

- Blur Filter Nodes

- Anti-Aliasing Node
- Denoise Node
- Despeckle Node

- Dilate/Erode Node
- Inpaint Node

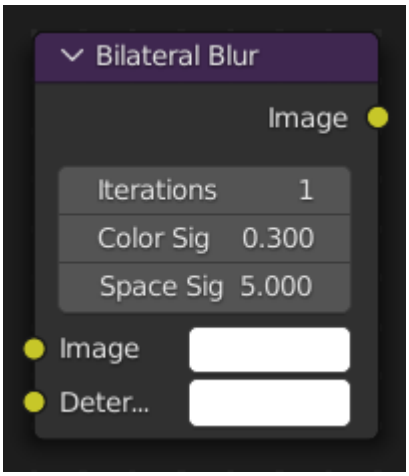
- Filter Node
- Glare Node
- Kuwahara Node
- Pixelate Node
- Posterize
- Sun Beams Node

Blur Filter Nodes

- Bilateral Blur Node
- Blur Node
- Bokeh Blur Node
- Defocus Node
- Directional Blur Node
- Vector Blur Node

Bilateral Blur Node

The Bilateral Blur node performs a high-quality adaptive blur, blurring the image while retaining sharp edges.



It can be used for various purposes like: smoothing noisy render passes to avoid longer computation times in example ray-traced ambient occlusion, blurry refractions/reflections, soft shadows, or to make non-photorealistic compositing effects.

Inputs

Image
: Standard color input. If only the image input is connected, the node blurs the image depending on the edges present in the source image.

Determinator
: Which is non-obligatory and if the Determinator is connected, it serves as the source for defining edges/borders for the blur in the image. This has great advantage in case the source image is too noisy, but normals in combination with Z-buffer can still define exact borders/edges of objects.

Properties

Iterations
: Defines how many times the filter should perform the operation on the image. It practically defines the radius of blur.

Color Sigma

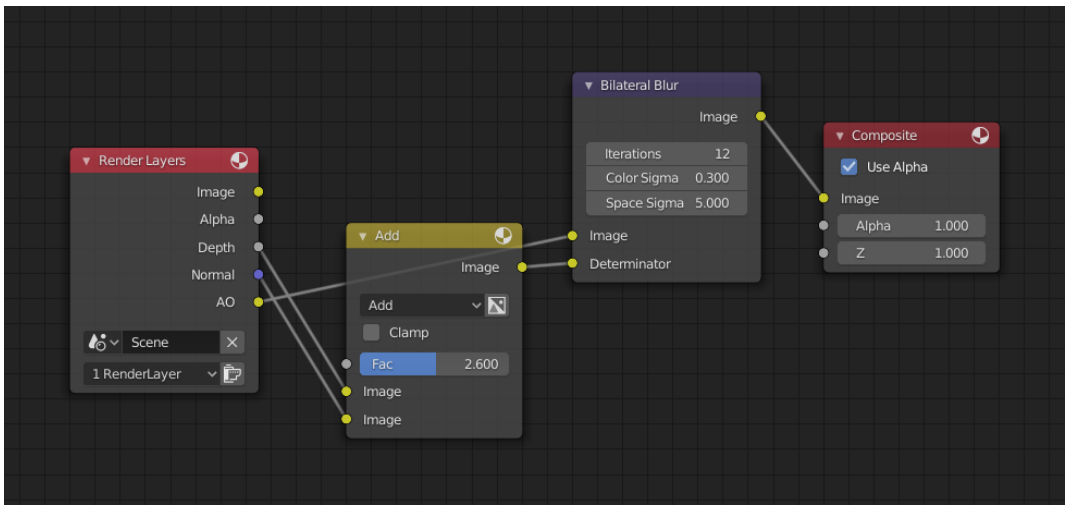
Defines the threshold for which color differences in the image should be taken as edges.

Space Sigma
: A fine-tuning variable for blur radius.

Outputs

Image
: Standard color output.

Example



Bilateral smoothed Ambient Occlusion. blend-file example



Render result.



Composite.

Blur Node

The Blur node blurs an image, providing several blur modes.



Inputs

Image
: Standard color input.

Size
: The optional Size input will be multiplied with the X and Y blur radius values. It also accepts a value image, to control the blur radius with a mask. The values should be mapped between (0 to 1) for an optimal effect.

Properties

Type

The difference between the types is in the way they handle sharp edges, smooth gradients and preserve the highs and the lows.

Flat::	Simply blurs everything uniformly.
Tent::	Preserves the high and the lows better by making a linear falloff.
Quadratic::	Looks similar to *Gaussian* but can be a little faster but slightly worse looking.
Cubic::	Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.
Gaussian::	Gives the best looking results but tends to be the slowest.
Fast Gaussian::	An approximation of the Gaussian.
Catmull-Rom::	Catmull-Rom keeps sharp contrast edges crisp.
Mitch::	Preserve the highs, but give an almost out-of-focus blur while smoothing sharp edges.

Variable Size

Allows a variable blur radius, if the size input is an image.

Bokeh

The Bokeh button will force the Blur node to use a circular blur filter. This gives higher quality results, but is slower than using a normal filter.

Gamma

The Gamma button applies a gamma correction on the image before blurring it.

Relative

Percentage Value of the blur radius relative to the image size.

Aspect Correction

None, Y, X

X, Y

Values set the ellipsoid radius in numbers of pixels over which to spread the blur effect.

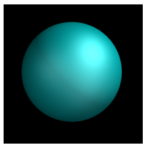
Extend Bounds
: Allows the image, that is being blurred, to extend past its original dimension.

Outputs

Image
: Standard color output.

Example

Blur node blur modes using 20% of image size as XY, no Bokeh/Gamma.



Original image.



Flat.



Tent.



Quadratic.



Cubic.



Gaussian.



Fast Gaussian.



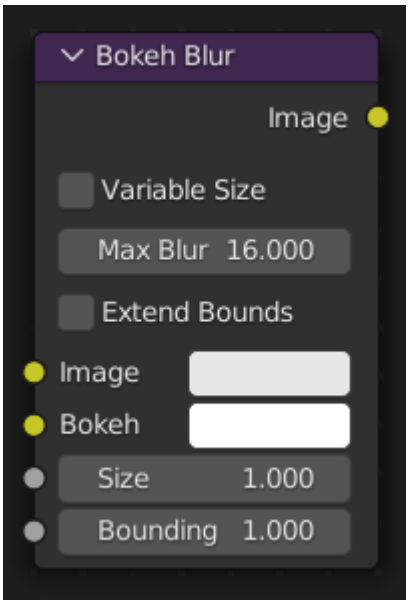
Catmull-Rom.



Mitch.

Bokeh Blur Node

The Bokeh Blur node generates a bokeh type blur similar to Defocus. Unlike defocus an in-focus region is defined in the Compositor. There is also more flexibility in the type of blur applied through the Bokeh Image node.



Several performance optimizations are also available such calculation area restriction and masking.

Inputs

Image
: Standard color input.

Bokeh
: This is an input for the Bokeh Image node.

Size
: Size controls the amount of blur. Size can either be a single value across the entire image or a variable value controlled by an input image. In order to use the latter, the Variable Size option must be selected. See the examples section below for more on how to use this.

Bounding Box
: This can be used with a Box Mask matte node or with a Mask input node to restrict the area of the image the blur is applied to. This could be helpful, for example, when developing a node system by allowing only

a small area of the image to be filtered thus saving composite time each time adjustments are made.

Properties

Variable Size
: Allows a variable blur radius, if the Size input is an image.

Max Blur
: Max Blur is intended to act as an optimization tool by limiting the number of pixels across which the blur is calculated.

Outputs

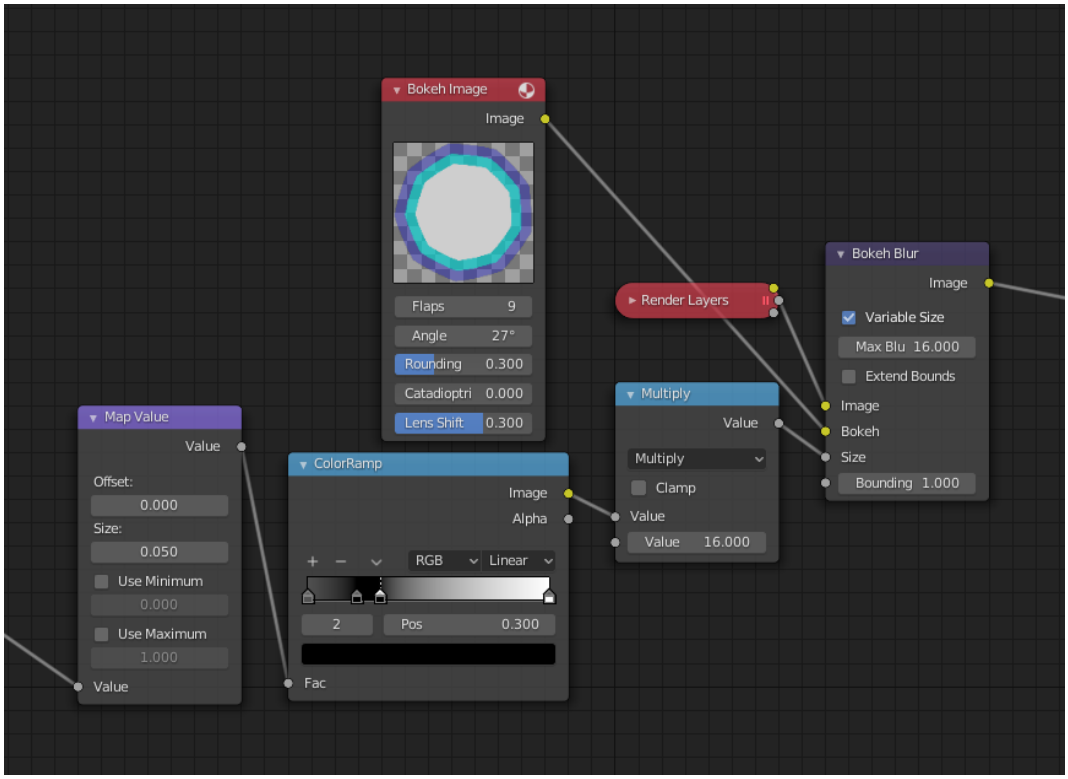
Image
: Standard color output.

Examples

Three examples of how the size input may be used follow.

An ID masked alpha image can be used so that a background is blurred while foreground objects remain in focus. To prevent strange edges the Dilate Node should be used.

The Z pass can be visualized using a Map Value node and a Color Ramp node as described in Render Layers. A *multiply* Math node can be used following the color ramp so that a blur value greater than one is used for objects outside the focal range.



Z pass used.

A manually created grayscale image can be used to define the sharp and blurry areas of a preexisting image. Again, a Multiply Node can be used so that a blur value greater than one is used.

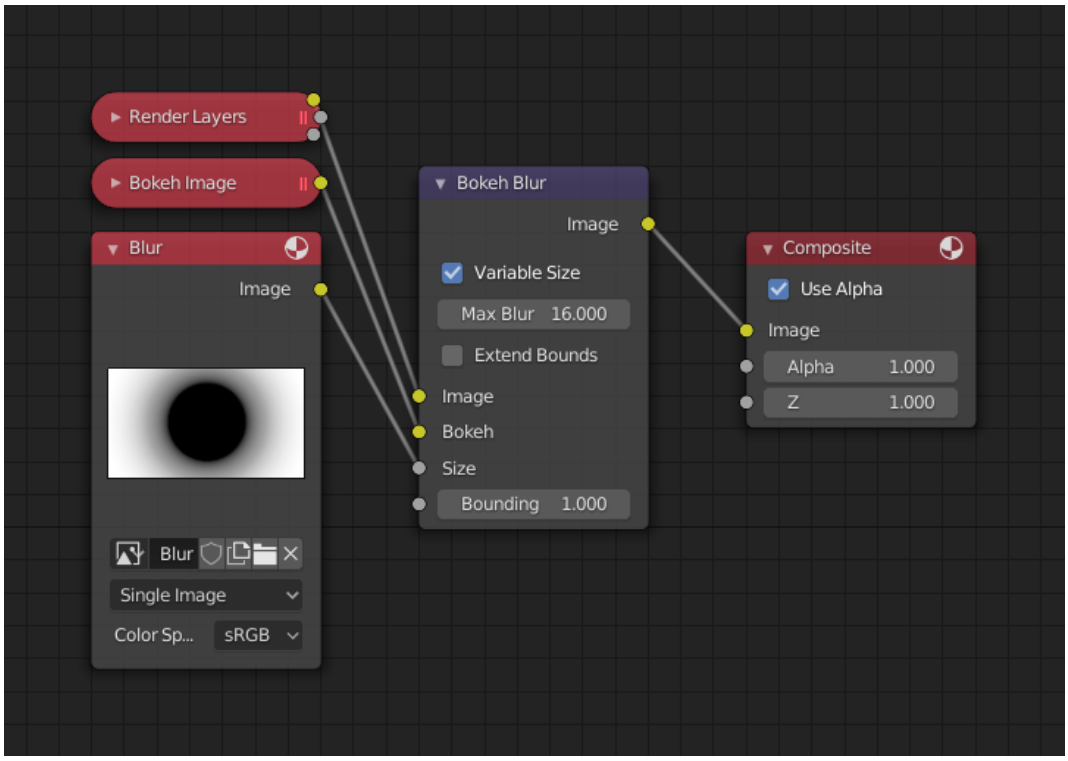
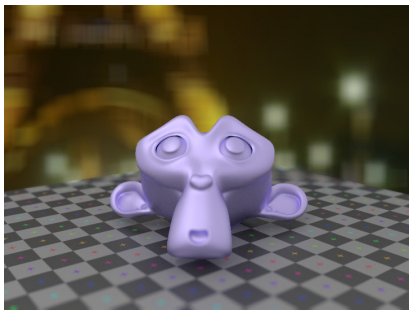


Image used.



Z pass used.

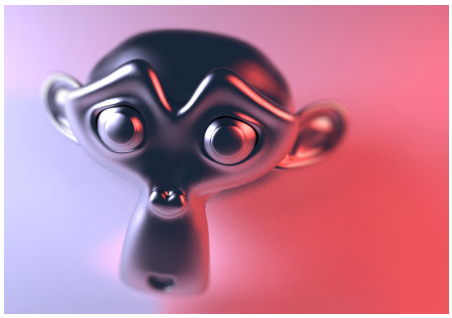
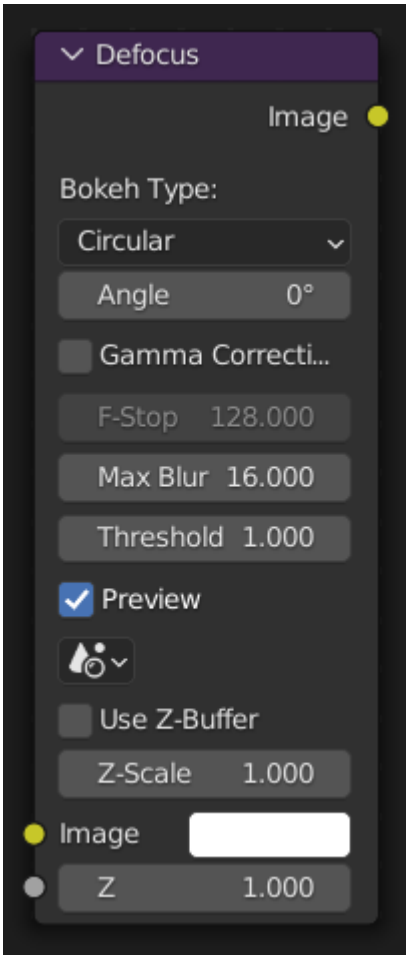


Image used.

Defocus Node

The *Defocus Node* blurs areas of an image based on a Z depth map or mask input.

It is typically used to emulate depth of field (DOF) using a post-processing method with a Z-buffer input. But also allows to blur images that are not based on Z depth too.



Inputs

Image
: Standard color input.

Z
: Z-buffer input, but could also be a (grayscale) image used as a mask, or a single value input.

Properties

Bokeh Type

The number of iris blades of the virtual camera's diaphragm.

Disk (to emulate a perfect circle), Triangle (3 blades), Square (4 blades), Pentagon (5 blades), Hexagon (6 blades), Heptagon (7 blades) or Octagon (8 blades).

Angle

This slider is deactivated, if the Bokeh Type is set to Disk. It can be used to add a rotation offset to the Bokeh shape. The value is the angle in degrees.

Gamma Correction

Applies a gamma correction on the image before and after blurring it.

F-Stop

This option controls the amount of focal blur in the same way as a real camera. It simulates the aperture *f* of a real lens' iris, without modifying the luminosity of the picture. The default value 128 is assumed to be infinity: everything is in perfect focus. Half the value will double the amount of blur. This slider is deactivated, if *No Z-buffer* is enabled.

Max Blur

This value limits the amount of blur by setting a maximum blur radius. Can be used to optimize the performance. The default value of 0 means no limit.

Threshold *CPU Compositor Only*

Some artifacts, like edge bleed, may occur, if the blur difference between pixels is large. This value controls how large that blur difference considered to be safe.

Tip

Only change this value, if there is an occurring problem with an in-focus object.

Preview *CPU Compositor Only*

If enabled a limited amount of (quasi-)random samples are used to render the preview. This way of sampling introduces additional noise, which will not show up in the final render.

Scene

To select the linked scene.

No Z-buffer

Should be activated for a non Z-buffer in the Z input. No Z-buffer will be enabled automatically whenever a node that is not image based is connected to the Z input.

Z Scale

Only active when *No Z-buffer* is enabled. When *No Z-buffer* is used, the input is used directly to control the blur radius (similar to *F-Stop* when using the Z-buffer). This parameter can be used to scale the range of the Z input.

Outputs

Image

Standard color output.

Examples



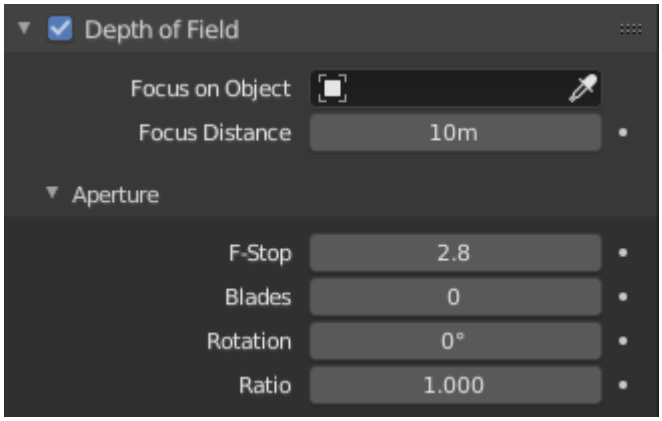
In this blend-file example, the ball array image is blurred as if it was taken by a camera with an f-stop of 2.8 resulting in a fairly narrow depth of field centered on 7.5 units from the camera. As the balls recede into the distance, they get blurrier.

No Z-Buffer Examples

Sometimes might want to have more control to blur the image. For instance, you may want to only blur one object while leaving everything else alone (or the other way around), or you want to blur the whole image uniformly all at once. The node, therefore, allows you to use something other than an actual Z-buffer as the Z input. For instance, you could connect an Image node and use a grayscale image where the color designates how much to blur the image at that point, where white is the maximum blur and black is no blur. Or, you could use a Time node to uniformly blur the image, where the time value controls the maximum blur for that frame. It may also be used to obtain a possibly slightly better DoF blur, by using a fake depth-shaded image instead of a Z-buffer. (A typical method to create the fake depth-shaded image is by using a linear blend texture for all objects in the scene or by using the "fog/mist" fake depth shading method.) This also has the advantage that the fake depth image can have Anti-Aliasing, which is not possible with a real Z-buffer.

The parameter *No Z-buffer*, becomes then the main blur control. The input has to be scaled, because usually the value of a texture is only in the numeric range 0.0 to 1.0.

Camera Settings



Distance setting in the Camera Depth of Field panel.

The *Defocus* node uses the actual camera data in your scene if supplied by a *Render Layer* node.

To set the point of focus, the camera now has a *Distance* parameter, which is shorthand for Depth of Field Distance. Use this camera parameter to set the focal plane of the camera (objects Depth of Field Distance away from the camera are in focus). Set *Distance* in the main *Camera* edit panel; the button is right below the *Depth of Field*.

To make the focal point visible, enable the camera *Limits* option, the focal point is then visible as a yellow cross along the view direction of the camera.

Hints

Preview
: In general, use preview mode, change parameters to your liking, only then disable preview mode for the final render.

Edge Artifacts
: For minimum artifacts, try to setup your scene such that differences in distances between two objects that may visibly overlap at some point are not too large.

“Focus Pull”

Keep in mind that this is not real DoF, only a post-processing simulation. Some things cannot be done which would be no problem for real DoF at all. A typical example is a scene with some object very close to the camera, and the camera focusing on some point far behind it. In the real world, using shallow depth of field, it is not impossible for nearby objects to become completely invisible, in effect allowing the camera to see behind them. Hollywood cinematographers use this visual characteristic to achieve the popular "focus pull" effect, where the focus shifts from a nearby to a distant object, such that the "other" object all but disappears. Well, this is simply not possible to do with the current post-processing method in a single pass. If you really want to achieve this effect, quite satisfactorily, here is how:

- Split up your scene into "nearby" and "far" objects, and render them in two passes.
- Now, combine the two results, each with their own "defocus" nodes driven by the same Time node, but with one of them inverted (e.g. using a Map Value node with a Size of -1). As the defocus of one increases, the defocus on the other decreases at the same rate, creating a smooth transition.

Aliasing at Low f-Stop Values

At very low values, less than 5, the node will start to remove any oversampling and bring the objects at DoF Distance very sharply into focus. If the object is against a contrasting background, this may lead to visible stair-stepping (aliasing) which OSA is designed to avoid. If you run into this problem:

- Do your own OSA by rendering at twice the intended size and then scaling down, so that adjacent pixels are blurred together.
- Use the Blur node with a setting of 2 for X and Y.
- Set DoF Distance off by a little, so that the object in focus is blurred by the tiniest bit.
- Use a higher f-stop, which will start the blur, and then use the Z socket to a Map Value to a Blur node to enhance the blur effect.
- Rearrange the objects in your scene to use a lower-contrast background.

No Z-Buffer

A final word of warning, since there is no way to detect if an actual Z-buffer is connected to the node, be **very** careful with the *No Z-buffer* switch. If the *Z scale* value happens to be large, and you forget to set it back to some low value, the values may suddenly be interpreted as huge blur radius values that will cause processing times to explode.

Directional Blur Node

Blurs an image along a specified direction. Can be used to fake motion blur.



Inputs

Image
: Standard color input.

Properties

Iterations
: Controls how may times the image is duplicated to create the blur effect. Higher values give smoother results.

Center X, Y

Sets the position where the blur center is. This makes a difference if the angle, spin, and/or zoom are used.

Distance
: How large the blur effect is.

Angle
: Image is blurred at this angle from the center.

Spin
: Rotates the image each iteration to create a spin effect, from the center point.

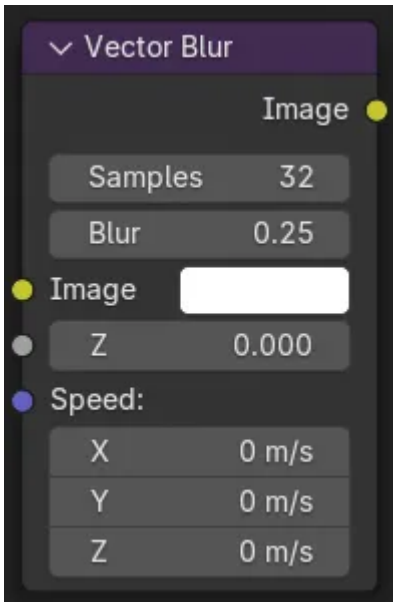
Zoom
: Scales the image each iteration, creating the effect of a zoom.

Outputs

Image
: Standard color output.

Vector Blur Node

The Vector Blur node is a fast method for simulating Motion Blur in compositing. It uses the vector speed render pass to blur the image pixels in 2D.



Inputs

Image
: Image input, to be linked to the “Combined” render pass.

Z
: Z depth, to be linked to the “Depth” render pass.

Speed
: Input for the “Vector” render pass. See Cycles render passes.

Properties

Samples
: Quality factor.

Blur

Scaling factor for the motion vector (actually the “shutter speed” in frames).

Outputs

Image
: Motion blurred image output.

Usage

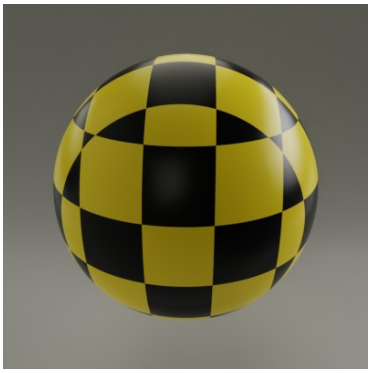
Even with a correct compositing setup with Image, Z and Speed nodes all linked to the appropriate passes, there may still be artifacts. The 2D render passes does not contain 3D information, and so the information what is behind a moving object or outside the camera view is lost.

Better results can be achieved by rendering the scene into multiple render layers, applying vector blur to each render layer, and then compositing the results together. Typically an animated character would be rendered in a separate render layer than the background set. Especially if hair or transparency is involved this is important.

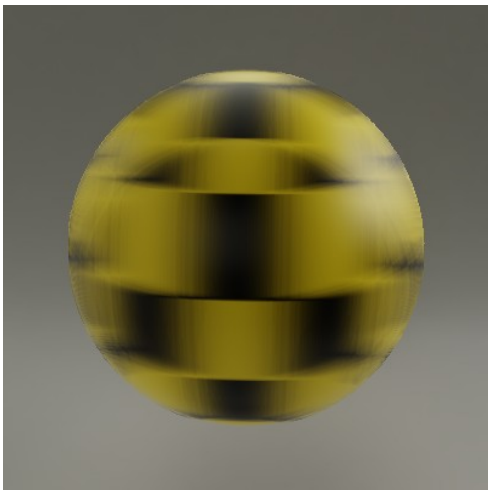
For other artifacts it can help to slightly blur the Speed pass or to set a Maximum Speed limit. This helps to smoothen out the motion, but too much blurring leads to its own problems.

Example

The *speed vector* in this example was created by animating the patterned sphere horizontally and using a frame at the mid-point of the sequence.



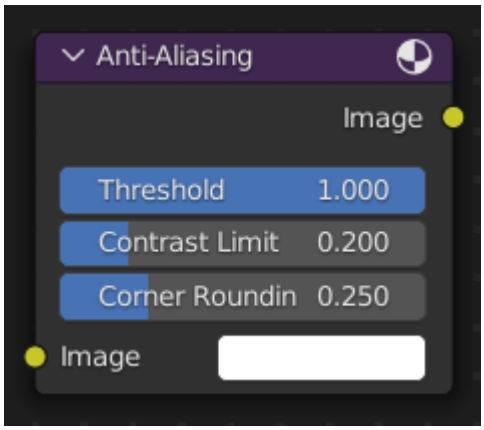
Render result, no post-processing.



Composite, with Samples set to 32 and Blur set to 1.0.

Anti-Aliasing Node

The Anti-Aliasing node smooths away jagged edges.



Inputs

Image
: Standard color input.

Properties

Threshold
: Controls edge detection sensitivity across the whole image.

Contrast Limit
: Controls contrast level to consider when detecting edges.

  The human eye does not perceive all edges equally. For instance, it tends to mask low contrast edges in the presence of much higher contrasts in the surrounding area. Therefore, applying anti-aliasing to unperceived edges will produce artifacts. This option quantifies the difference between low contrast and high contrast neighboring edges.

Corner Rounding

Detect corners to help preserve the original shape. Setting *Corner Rounding* to 0 means no corner detection and no corner rounding will take place. The higher the value the better corners will be preserved, i.e. resemble original image.

Outputs

Image
: Standard color output.

Examples

The Anti-Aliasing node has three properties shown here.



Changing threshold affects all edges.



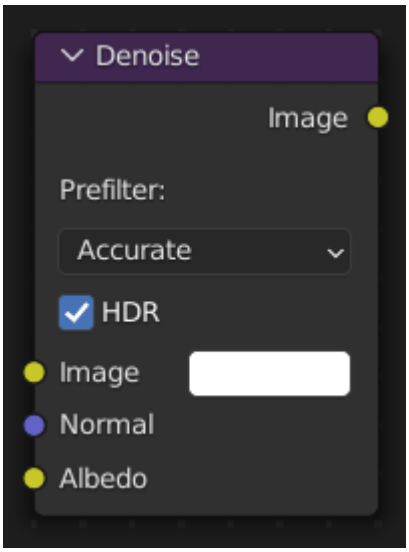
Changing contrast limit affects neighboring edges below contrast limit.

The effect of corner rounding.

Denoise Node

The Denoise node is used to denoise renders from Cycles and other ray tracing renderers. This helps to significantly reduce render time by rendering with fewer samples.

It uses Open Image Denoise, which transforms noisy images into clean images with machine learning.



Inputs

Image
: Noisy image input.

Normal
: Optional normal render pass to better preserve detail. For Cycles, it is recommended to use the Denoising Normal render pass, which is available when enabling the Denoising Data passes.

Albedo
: Optional albedo render pass to better preserve detail. For Cycles, it is recommended to use the Denoising Albedo render pass, which is available when enabling the Denoising Data passes.

Properties

Prefilter

None:: Does not apply any prefiltering to the input passes. This option retains the most detail and is the fastest, but assumes the input passes are noise free which may require a high sample count. If the input passes are not noise free, then noise will remain in the image after denoising.

Fast:: Assumes the input passes are not noise free, yet does not apply prefiltering to the input passes. This option is faster than *Accurate* but produces a blurrier result.

Accurate:: Prefilters the input passes before denoising to reduce noise. This option usually produces more detailed results than *Fast* with increased processing time.

HDR
: Preserve colors outside the 0 to 1 range.

Outputs

Image
: Denoised image output.

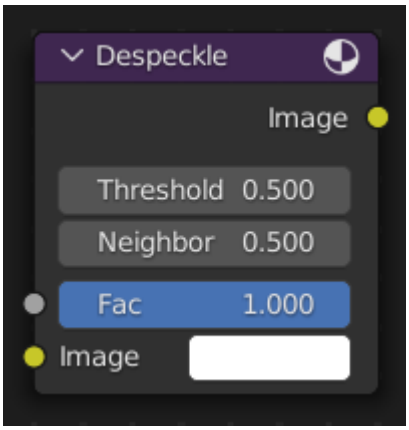
Examples



Render before and after denoising, with a very low number of samples as input. As more samples are used, the denoiser will be able to better preserve detail.

Despeckle Node

The *Despeckle node* is used to smooth areas of an image in which noise is noticeable, while leaving complex areas untouched.



This works by the standard deviation of each pixel and its neighbors is calculated to determine if the area is one of high complexity or low complexity. If the complexity is lower than the threshold then the area is smoothed using a simple mean filter.

Inputs

Factor
: Controls the amount the filter effects the image.

Image
: Standard color input.

Properties

Threshold
: The threshold to control high/low complexity.

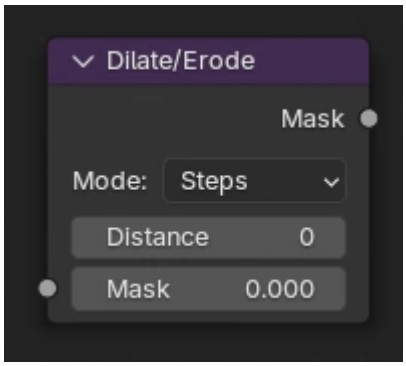
Neighbor
: The threshold to control the number of pixels that must match.

Outputs

Image
: Standard color output.

Dilate/Erode Node

Expands or shrinks a mask using a morphological operator.



Inputs

Mask
: A grayscale image.

Properties

Mode
: Steps
: Sets each pixel to the maximum (for dilation) or minimum (for erosion) value that's found within a square surrounding it. This approach keeps the original gray levels and is best suited for masks that contain sharp corners; rounded shapes such as circles will look more square-like in the output.

  Despite the name, this is not an iterative process; the dilation/erosion is only performed once regardless of the chosen *Distance*.

: Threshold
: Makes all the pixels fully black or white depending on whether they're darker or brighter than 50% gray. Then, sets each pixel to the

maximum (for dilation) or minimum (for erosion) value that's found within a circle surrounding it. This approach loses the original gray levels. Shape wise, it's well-suited for masks that contain rounded corners; sharp ones will be rounded off.

Distance
: Sets each pixel to the maximum (for dilation) or minimum (for erosion) value that's found within a circle surrounding it. This approach preserves the original gray levels and is well-suited for masks that contain rounded corners.

Feather
: Blurs the image.

Distance
: The size of the surrounding area to look at for each pixel; or in other words, how much to dilate (for positive values) or erode (for negative values) the mask.

Edge
: For the *Threshold* mode, determines how much to blur the edges after dilation/erosion.

Falloff
: For the *Feather* mode, determines the brightness curve of the blurred edges.

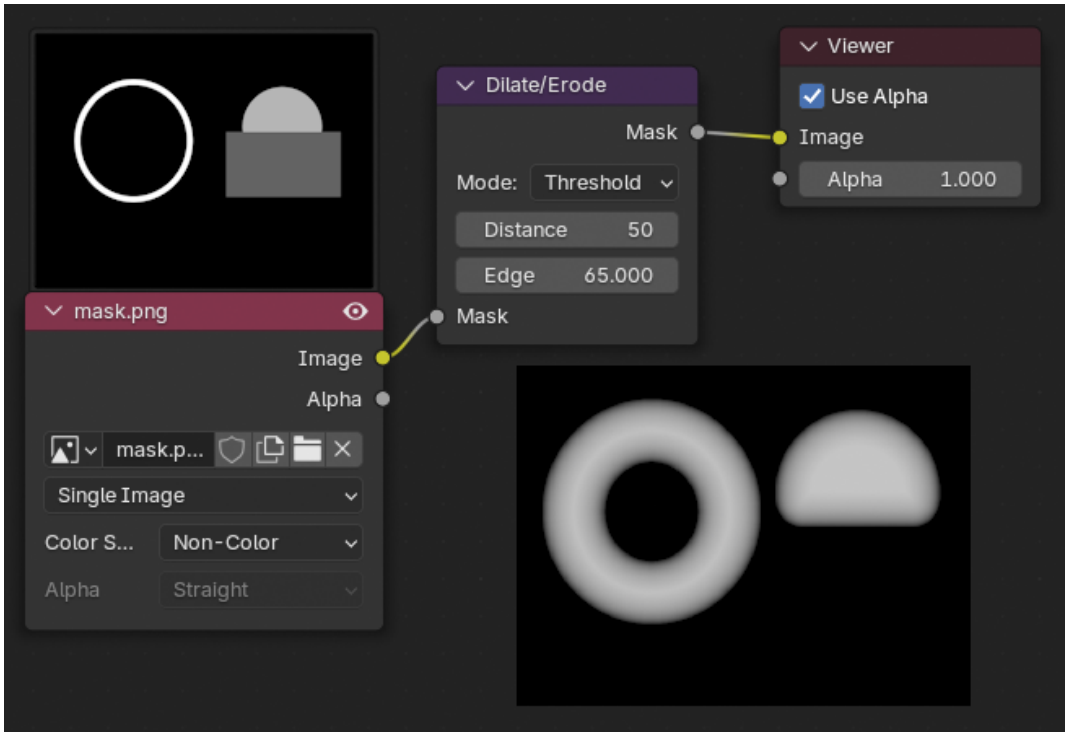
Outputs

Mask
: The resulting mask.

Example

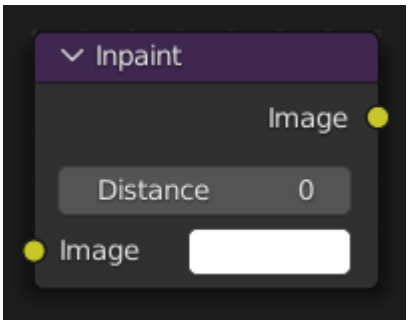
In the image below, notice that:

- The light gray disk has turned white and the dark gray rectangle has turned black because of the *Threshold* mode.
- The shapes have become thicker – dilated because of the positive *Distance*.
- The shapes appear blurred because of the positive *Edge*.



Inpaint Node

The *Inpaint node* is used to extend borders of an image into transparent or masked regions. This can be useful to solve problems like “wire removal” and holes created during chroma keying.



Inputs

Image
: Standard color input.

Properties

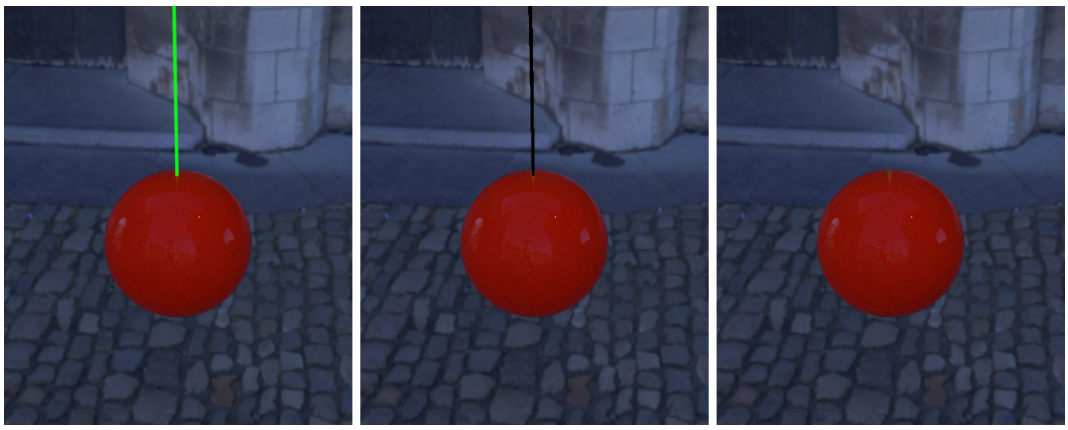
Distance
: The number of times to extend the image.

Outputs

Image
: Standard color output.

Examples

The left image shows the “wire” in place and after chroma keying has been applied. You will see you are left with a blank space – it’s shown as a black line here but it will be alpha in your Blender output.



Inpaint Node example.

Inpainting fills in a couple of pixels using the surrounding image and voilà… your wire is removed.

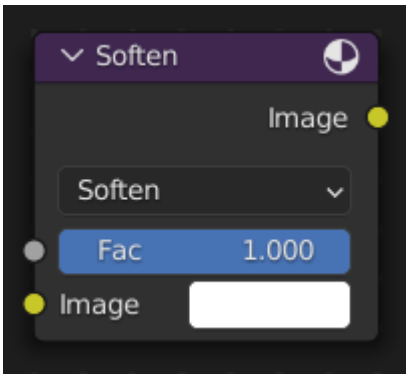
Note

The wider the “hole” is, the more noticeable this effect is! If you use more than a few pixels of infill, the effect is almost as irritating as the wire and your viewers won’t be impressed.

Inpainting can also cover up a multitude of other minor sins such as control points for motion capture: use it sparingly and it will amaze.

Filter Node

The Filter node implements various common image enhancement filters.



Inputs

Factor
: Controls the amount of influence the node exerts on the output image.

Image
: Standard color input.

Properties

Type
: The Soften, Laplace, Sobel, Prewitt and Kirsch all perform edge detection (in slightly different ways) based on vector calculus and set theory equations.

Soften::	Slightly blurs the image.
Box Sharpen::	Increases the contrast, especially at edges.
Diamond Sharpen::	Less aggressive than box sharpen, reducing sharpening artifacts.
Laplace::	Edge highlighting filter susceptible to highlighting visual noise.
Sobel::	Creates a negative image that highlights edges.
Prewitt::	Produces a similar results to Sobel.

Kirsch:: Gives better blending than Sobel or Prewitt, when approaching an edge.

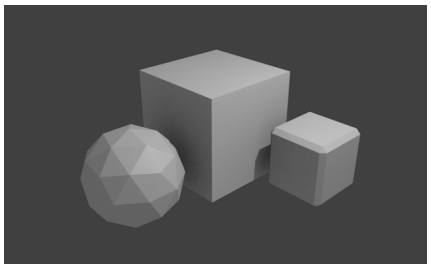
Shadow:: Performs a relief, emboss effect, darkening outside edges.

Outputs

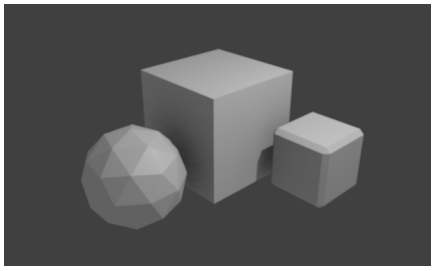
Image
: Standard color output.

Example

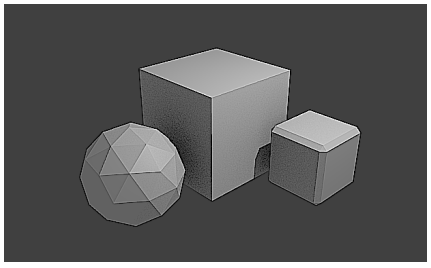
The Filter node has eight modes, shown here.



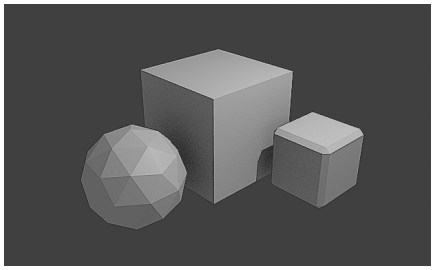
Original image.



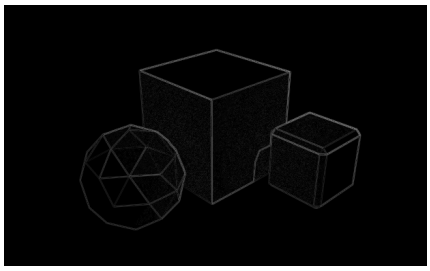
Soften.



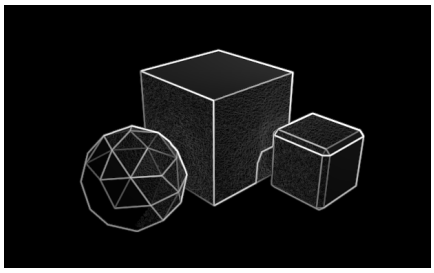
Box Sharpen.



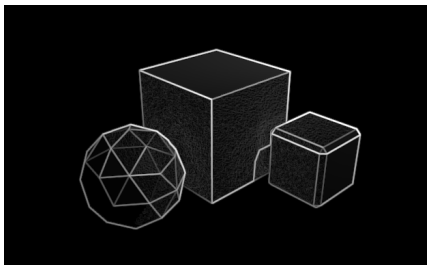
Diamond Sharpen.



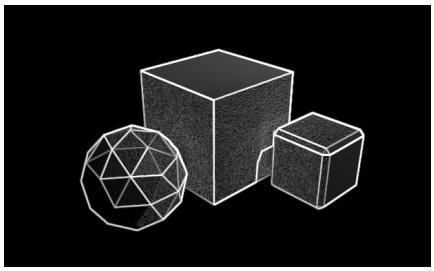
Laplace.



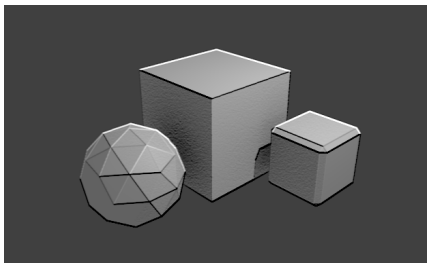
Sobel.



Prewitt.



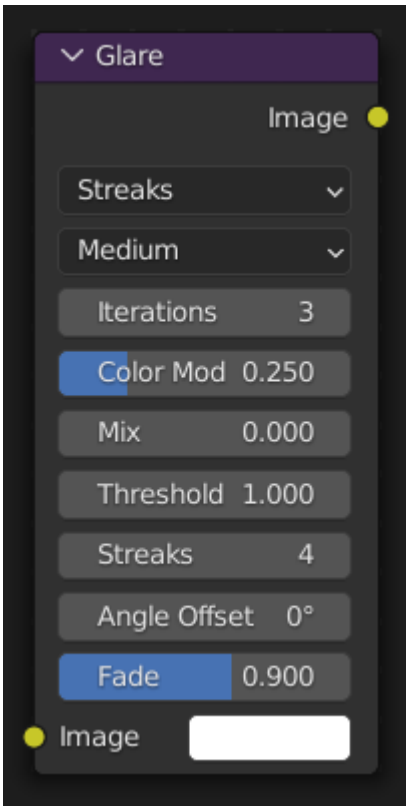
Kirsch.



Shadow.

Glare Node

The *Glare node* is used to add lens flares, fog, glows around bright parts of an image.



Inputs

Image
: Standard color input.

Properties

Glare Type

Bloom:: Simulates the glow around bright objects caused by light scattering in eyes and cameras.

Size
: Scale of the glow relative to the size of the image. 9 means the glow can cover the entire

image, 8 means it can only cover half the image, 7 means it can only cover quarter of the image, and so on.

Ghosts:: Creates a haze over the image.

Streaks:: Creates bright streaks used to simulate lens flares.

Streaks
: Total number of streaks.

Angle Offset
: The rotation offset factor of the streaks.

Fade
: Fade out factor for the streaks.

Fog Glow:: Simulates the glow around bright objects caused by light scattering in eyes and cameras. This is similar to the *Bloom* mode, but is more physically accurate, at the cost of much slower computation time.

Size
: Scale of the glow relative to the size of the image. 9 means the glow will cover the entire image, 8 means it will cover half the image, 7 means it will cover quarter of the image, and so on.

Simple Star:: Works similar to *Streaks* but gives a simpler shape looking like a star.

Fade
: Fade out factor for the streaks.

Rotate 45
: Rotate the streaks by 45°.

Quality
: If not set to something other the *High*, then the glare effect will only be applied to a low resolution copy of the image. This can be helpful to save render times while only doing preview renders.

Iterations

The number of times to run through the filter algorithm. Higher values will give more accurate results but will take longer to compute. Note that, this is not available for *Fog Glow* as it does not use an iterative-based algorithm.

Color Modulation

Used for *Streaks* and *Ghosts* to create a special dispersion effect.

Johannes Itten describes this effect, Color Modulation, as subtle variations in tones and chroma.

Mix

Value to control how much of the effect is added on to the image. A value of -1 would give just the original image, 0 gives a 50/50 mix, and 1 gives just the effect.

Threshold

Pixels brighter than this value will be affected by the glare filter.

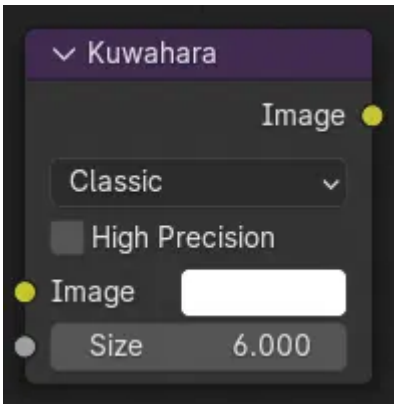
Outputs

Image

Standard color output.

Kuwahara Node

The Kuwahara node implements the Kuwahara filter as well as its anisotropic variant. The Kuwahara filter is a smoothing filter that tries to preserve the edges in the image. The smoothing effect of the anisotropic variant is similar to brush strokes, so the node can be used to create stylized painting effects.



Inputs

Image
: Standard color input.

Size
: Controls the size of the smoothing neighborhood. Large values may introduce artifacts for highly detailed areas. For the anisotropic method, the larger the size, the slower the filter.



Original.



Size: 3.



Size: 6.



Size: 9.

Properties

Type

Classic:: A simple smoothing method that averages the local square neighborhood of the image while preserving edges. Produces blocky results due to the square neighborhood and provides no tuning parameters, but is faster to compute.

Anisotropic:: A complex smoothing method that averages the local neighborhood of the image in the direction of the flow of the edges, thus preserving the edges in the output. Produces painterly-like results and provides multiple turning parameters, while being slower to compute.

High Precision

Uses a more precise but slower method. Use if the output contains undesirable noise.

Uniformity

Controls the uniformity of the directions of the edges of the image. Non uniform directions are nearly never desirable, so this should typically be increased until the user notices the result is no longer changing in a significant way. Further increases would produces worst results and increase compute time.

Sharpness

Controls the sharpness of the edges of the image.



Original.



Sharpness: 0.



Sharpness: 0.5.



Sharpness: 1.

Eccentricity

Controls how thin and directional the filter is. Low eccentricity corresponds to circular omnidirectional features while high eccentricity corresponds to thin directional features.



Original.



Eccentricity: 0.



Eccentricity: 1.



Eccentricity: 2.

Outputs

Image
: Standard color output.

Notes

Iterations
: The filter can be applied multiple times by chaining the node multiple times. This chaining can produce more flat filtering.



Original.



Iterations: 1.



Iterations: 2.



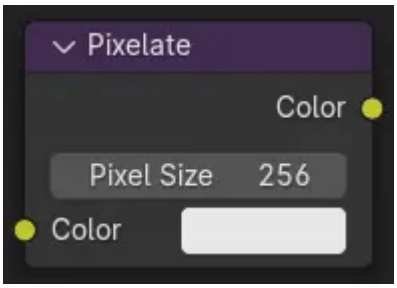
Iterations: 3.

Performance

The filter can be expensive to compute for high size input and high resolution images. To improve performance, consider scaling down the image, applying the filter, then scaling it up again. This can work well because the filter already attenuates low frequency details.

Pixelate Node

The Pixelate node reduces the detail in an image by making individual pixels more prominent. It results in a blocky or mosaic-like appearance, where the fine details of the image are obscured, and the overall image is represented using larger, more noticeable pixels.



Inputs

Color
: Standard color input.

Properties

Pixel Size
: The size of the pixels in the output image, measured in pixels.

Outputs

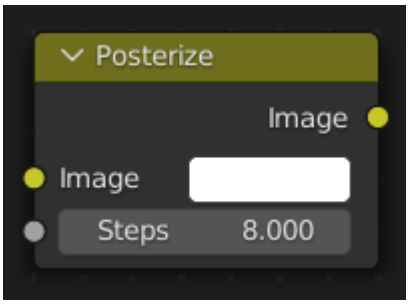
Color
: Standard color output.

Example



Posterize

The *Posterize Node* reduces the number of colors in image, converting smooth gradients into sharp transitions. This node is useful for generating masks in particular for rotoscoping.



Inputs

Image
: Standard color input.

Steps
: The number of colors per channel; A value of 8 will result in \(8^3 = 512\) total colors.

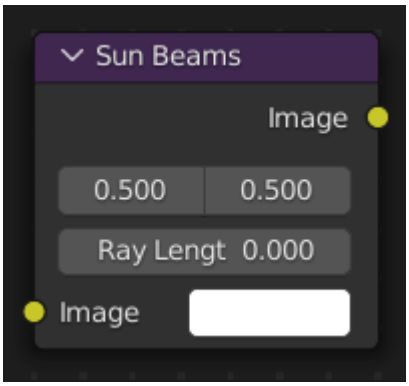
Properties

This node has no properties.

Outputs

Image
: Standard color output.

Sun Beams Node



The Sun Beams node provides a cheap way of adding sun beams based on image brightness alone.

Sun Beams is a 2D effect for simulating the effect of bright light getting scattered in a medium (Crepuscular Rays). This phenomenon can be created by renderers, but full volumetric lighting is a rather arduous approach and takes a long time to render.

Inputs

Image
: Standard color input.

Properties

Source Width, Height
: Source point of the rays as a factor of the image dimensions.

Ray Length
: Length of the rays as a factor of the image size.

Outputs

Image
: Standard color output.

Example

Usually, the first step is to define the area from which rays are cast. Any diffuse reflected light from surfaces is not going to contribute to such scattering in the real world, so should be excluded from the input data. Possible ways to achieve this are:

- Entirely separate image as a light source.
- Brightness/contrast tweaking to leave only the brightest areas.
- Muting shadow and midtone colors, which is a bit more flexible.
- Masking for ultimate control.

After generating the sun beams from such a light source image they can then be overlaid on the original image. Usually, a simple “Add” Mix node is sufficient, and physically correct because the scattered light adds to the final result.



Keying Nodes

These nodes give you the essential tools for creating a Matte for images that do not already have their own Alpha Channel. One usage scenario is blue-screen or green-screen footage, where live action is shot in front of a blue or green backdrop for replacement by a matte painting or virtual background.

In general, hook up these nodes to a viewer, set your Image Editor to show the Viewer node, and play with the sliders in real-time using a sample image from the footage, to get the settings right. In some cases, small adjustments can eliminate artifacts or foreground image degradation. Taking out too much green can result in foreground actors looking flat or bluish/purplish.

You can and should chain these nodes together, improving your masking and color correction in successive refinements, using each node's strengths to operate on the previous node's output. Keying Node is the closest to a "does-it-all" node for green screens, but the best results stem from a combination of techniques.

Note

Garbage Matte is not a node, but a technique selecting what to exclude from an image. It is a Mask used to identify content to be removed from an image that cannot be removed by an automatic process like chroma keying. It is used either to select specific content to be removed, or it is the inverse of a rough selection of the subject; removing everything else.

Some nodes accept a garbage matte directly. For those that don't, you can still apply one by subtracting the garbage matte from the matte generated by the node.

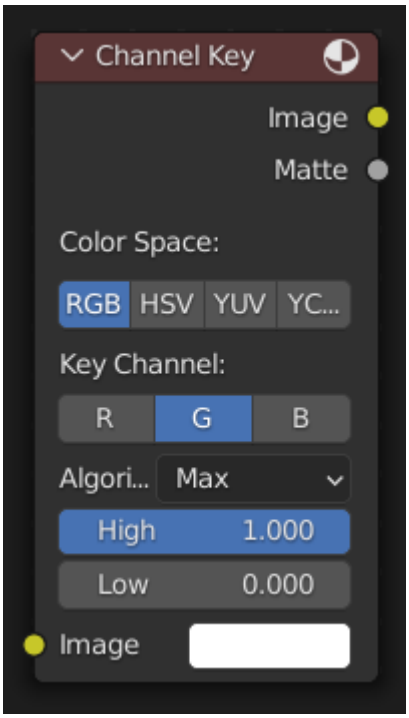
Simple garbage mattes can be created with the Box Mask or the Ellipse Mask. More complicated matte shapes using a Double Edge Mask or using a Mask.

- Channel Key Node
- Chroma Key Node
- Color Key Node
- Color Spill Node
- Difference Key Node
- Distance Key Node
- Keying Node
- Keying Screen Node
- Luminance Key Node

Channel Key Node

The *Channel Key* node determines background objects from foreground objects by the difference in the selected channel's levels.

For example in the YUV Color Model, this node is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.



Inputs

Image
: Standard color input.

Properties

Color Space
: This button selects what color model the channels will represent.

  RGB, HSV, YUV, YCbCr

Key Channel
: This button selects the channel, defined by the *Color Space*, to use to determine the matte.

Algorithm
: Method to calculate the difference between levels.

Max::	Limit by the maximum of the other two channels other than the *Key Channel*.
Single::	Limit by the maximum of the selected *Limiting Channel*.

Limiting Channel
: The channel to use when computing the maximum, the options are defined by the *Color Space*.

High
: Determines the lowest values that are considered foreground. (Which is supposed to be – relatively – high values: from this value to 1.0.)

Low
: Determines the highest values that are considered to be background objects. (Which is supposed to be – relatively – low values: from 0.0 to this value.)

Tip

It is possible to have a separation between the *High* and *Low* values to allow for a gradient of transparency between foreground and background objects.

Outputs

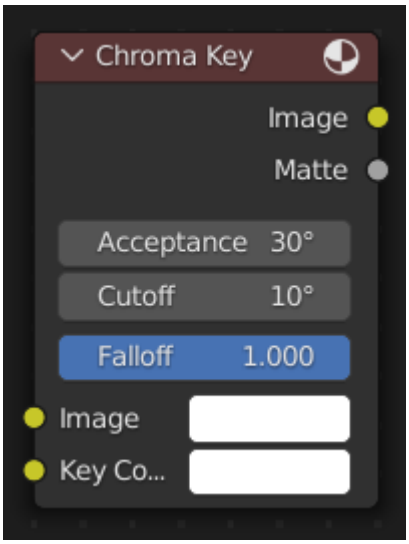
Image
: Image with an alpha channel adjusted for the keyed selection.

Matte
: A black-and-white alpha mask of the key.

Chroma Key Node

The *Chroma Key* node determines if a pixel is a foreground or background (and thereby should be transparent) based on its chroma values.

Use this, for example, to composite images that have been shot in front of a green or blue screen.



Inputs

Image
: Standard color input.

Key Color
: The background color usually selected using the color picker and the original image.

Properties

Acceptance
: An angle on the color wheel that represents how tolerant the keying color is. Larger angles allow for larger variation in the keying color to be considered background pixels.

Cutoff
: Controls the level that is considered the pure background. Higher cutoff levels mean more pixels will be 100% transparent if they are within the

angle tolerance.

Falloff
: Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

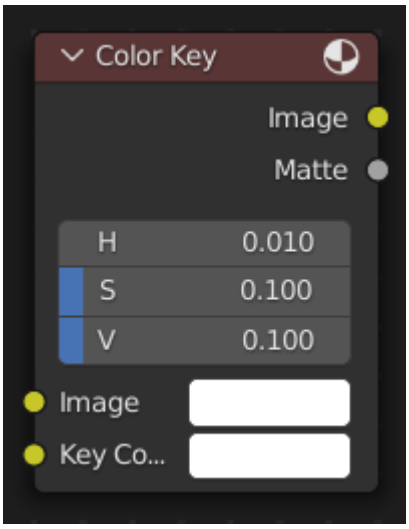
Outputs

Image
: Image with its alpha channel adjusted for the keyed selection.

Matte
: A black-and-white alpha mask of the key.

Color Key Node

The Color Key node creates a matte based on a specified color of the input image.



Inputs

Image
: Standard color input.

Properties

Color
: The sliders represent threshold values. Higher values in this node's context mean a wider range of colors from the specified will be added to the matte.

  Hue, Saturation, Value

Outputs

Image
: Image with its alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Color Spill Node



The *Color Spill* node reduces one of the RGB channels so that it is not greater than any of the others.

This is common when compositing images that were shot in front of a green or blue screen. In some cases, if the foreground object is reflective, it will show the green or blue color; that color has "spilled" onto the foreground object. If there is light from the side or back, and the foreground actor is wearing white, it is possible to get "spill" green (or blue) light from the background onto the foreground objects, coloring them with a tinge of green or blue. To remove the green (or blue) light, you use this fancy node.

Inputs

Image
: Standard color input.

Factor
: Standard Factor.

Properties

Despill Channel
: R, G, B

Algorithm
: Simple, Average

Limiting Channel
: R, G, B

Ratio
: Scale limit by value.

Unspill
: Allows you to reduce the selected channel's input to the image greater than the color spill algorithm normally allows. This is useful for exceptionally high amounts of the color spill.

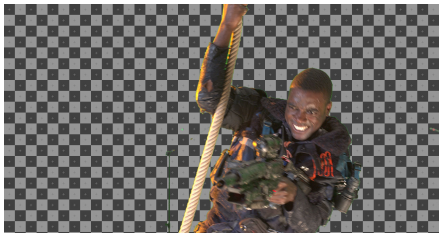
  R, G, B

Outputs

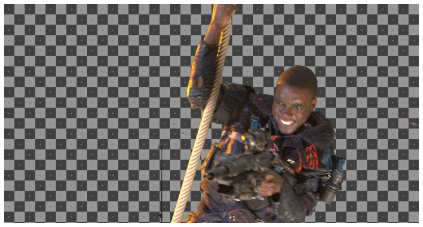
Image
: The image with the corrected channels.

Example

Results with the nodes applied to an image from the Mango Open Movie.



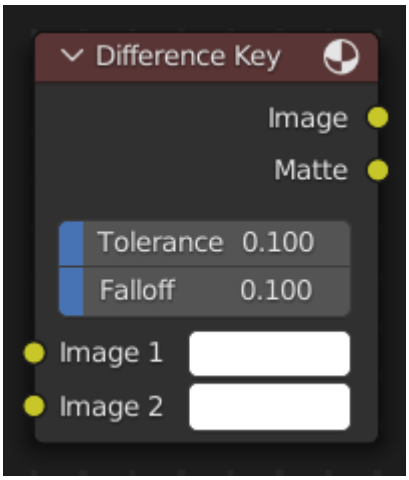
Before: green border and green reflections.



After: no unwanted green.

Difference Key Node

This node produces a matte that isolates foreground content by comparing it with a reference background image.



Inputs

Image
: Contains foreground content against the background that is to be removed.

Image
: The reference background image.

Properties

Tolerance
: Where pixels match the reference background to within the specified threshold, the matte is made transparent.

Falloff
: Increase to make nearby pixels partially transparent producing a smoother blend along the edges.

Outputs

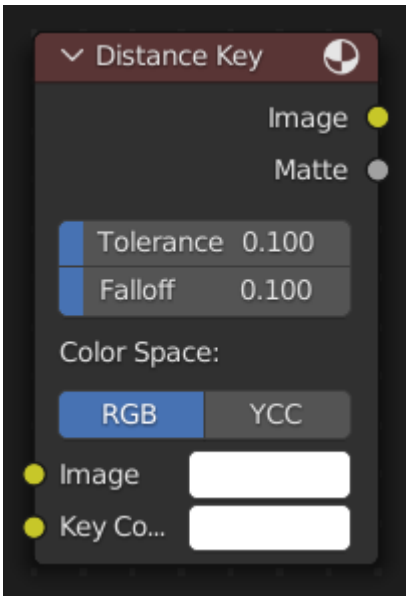
Image
: Image with its alpha channel adjusted for the keyed selection.

Matte
: A black-and-white alpha mask of the key.

Distance Key Node

The Distance Key node determines a pixel's alpha value based on the three-dimensional distance between the image pixel color and the key color in a 3D color space.

This key works well when trying to single out a specific color in a background (not necessarily green).



Inputs

Image
: Standard color input.

Key Color
: The color that is to be keyed.

Properties

Tolerance
: A threshold what the node considers a match between the key color and the foreground pixel. The tolerance affects how close a pixel needs to be to the background pixel to be considered an absolute match.

Falloff
: When the Falloff value is high, pixels that are close to the Key Color are more transparent than pixels that are not as close to the Key Color (but

still considered close enough to be keyed). When the Falloff value is low, it does not matter how close the pixel color (Image) is to the Key Color, it is transparent.

Color Space

It is also possible to work with YCbCr color space, but only the Cb and Cr channels are taken into consideration for determining the distance between the foreground and background pixels.

RGB, YCC

Outputs

Image

The image with an alpha channel adjusted for the keyed selection.

Matte

A black-and-white alpha mask of the key.

Keying Node

The *Keying* node is a one-stop-shop for “green screen” / “blue screen” removal. It performs both chroma keying to remove the backdrop and despill to correct color cast from the backdrop. Additionally, you can perform common operations used to tweak the resulting matte.



Inputs

Image
: Standard color input.

Key Color
: The color of content to be removed. This may be a single color, or a reference image such as generated by the Keying Screen Node.

Garbage Matte
: An optional mask of area(s) to always *exclude* from the output. This is removed from the chroma key generated matte.

Core Matte
: An optional mask of area(s) to always *include* in the output. This is merged with the chroma key generated matte.

Properties

Pre Blur
: Reduce the effects of color noise in the image by blurring only color by the given amount, leaving luminosity intact. This will affect matte calculation only, not the result image.

Screen Balance
: This is the balance between color channels compared with the key color. 0.5 will average the other channels (red and blue in the case of a green screen).

  This may be tweaked in tandem with *Clip Black* and *Clip White* while checking the *Matte* output to create a mask with optimal separation.

Despill Factor
: Controls how much color bleed from the key color is removed from the input image: 0 means no despilling, 1 means all possible spilling will be removed. The underlying implementation is the same as adjusting the *Unspill* amount of the Color Spill Node.

Despill Balance
: This controls how the color channels are compared when computing spill, affecting the hue and shade of the corrected colors. It is similar to setting the *Limiting Channel* in the Color Spill Node.

Edge Kernel Radius

Defines the radius in pixel used to detect an edge.

Edge Kernel Tolerance

Defines threshold used to check if pixels in radius are the same as current pixel: if the difference between pixel colors is higher than this threshold then the point will be considered an edge.

Clip Black

This sets the threshold for what becomes fully transparent in the output (black in the matte). It should be set as low as possible. Uneven backdrops will require this value to be increased. Use of the Keying Screen Node can help keep this value low. You may also use a *Garbage Matte* to exclude problematic areas.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Clip White

This sets the threshold for what becomes fully opaque in the output (white in the matte). It should be set as high as possible. Colors close to green in the foreground may require reducing this value and/or adjusting the *Screen Balance*. Particularly problematic parts can fixed with a *Core Matte* instead of a low *Clip White*.

This value does not impact areas detected as edges to ensure edge detail is preserved.

Dilate/Erode

Enlarge (positive numbers) or shrink (negative numbers) the matte by the specified number of pixels. This is similar to using the Dilate/Erode Node on the matte.

This a simple way to include more or less along the edges of the matte, particularly combined with *Post Blur*.

Feather Falloff

The rate of the falloff at the edges of the matte when feathering, to manage edge detail.

Feather Distance
: Controls how much the matte is feathered inwards (negative number) or outwards (positive number).

Post Blur
: Make the matte less sharp, for smoother transitions to the background and noise reduction.

Outputs

Image
: Processed image with the *Matte* applied to the images' Alpha Channel.

Matte
: Output matte to use for checking the quality of the key, or to manually apply using a Set Alpha Node or Mix Node.

Edges
: Shows what edges were detected on the matte. Useful for adjusting the *Edge Kernel Radius* and *Edge Kernel Tolerance*.

Tip

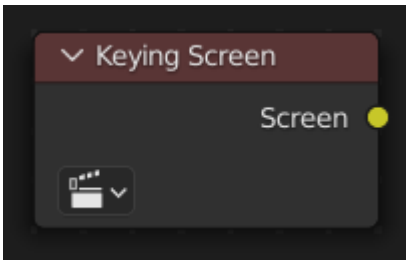
If there are problems with the edges of the matte, it may help to start with adjusting the *Edge Kernel* parameters before adjusting feathering. Detected edges are not subject to *Clip Black* / *Clip White* thresholds to preserve fine edge detail. You can check edge detection by connecting a Viewer Node to the *Edges* output.

Sharper detected edges (smaller *Edge Kernel Radius*, like 2 / larger *Edge Kernel Tolerance*, like 0.4) will create a sharper matte, but may loose some detail like stray hairs. A sharp matte is good, but disappearing or flickering hairs are distracting.

Fat edges (larger *Edge Kernel Radius*, like 8 / smaller *Edge Kernel Tolerance*, like 0.05) will capture more edge detail, but may also produce a halo around the subject. The halo can be adjusted with *Feather* controls along with *Dilate/Erode*.

Keying Screen Node

The Keying Screen node creates plates for use as a color reference for keying nodes. It generates gradients from sampled colors on motion tracking points on movie clips. It can be used to deal with uneven colors of green screens.



Inputs

This node has no input sockets.

Properties

Movie Clip
: The selectable clip data-block used as input for the gradient colors.

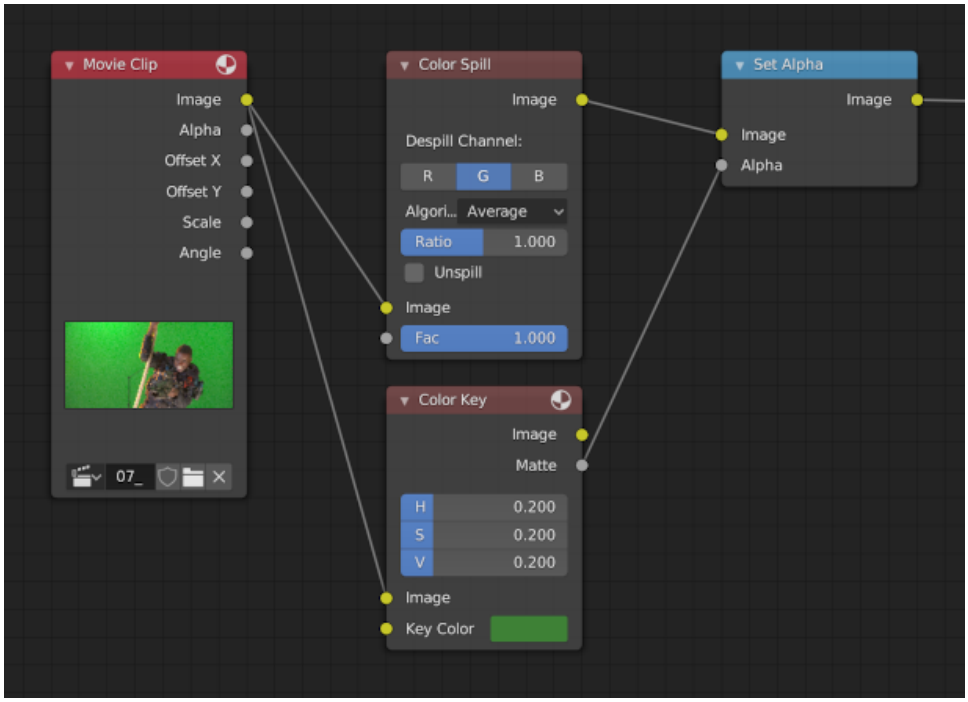
Tracking Object
: Tracking Object to generate the gradient. You will probably want to create new a tracking object in the Object panel, because tracks used for gradients can not actually be used for camera/object tracking. After this tracks might be placed in places where gradient colors should be sampled. These tracks could be tracked or moved manually, so gradients would be updating automatically along the movie. Tracks might have an offset for easier tracking of feature-less screens.

Outputs

Screen
: Gradient image output.

Example

Consider a node setup for green screen removal, using a Color Key:

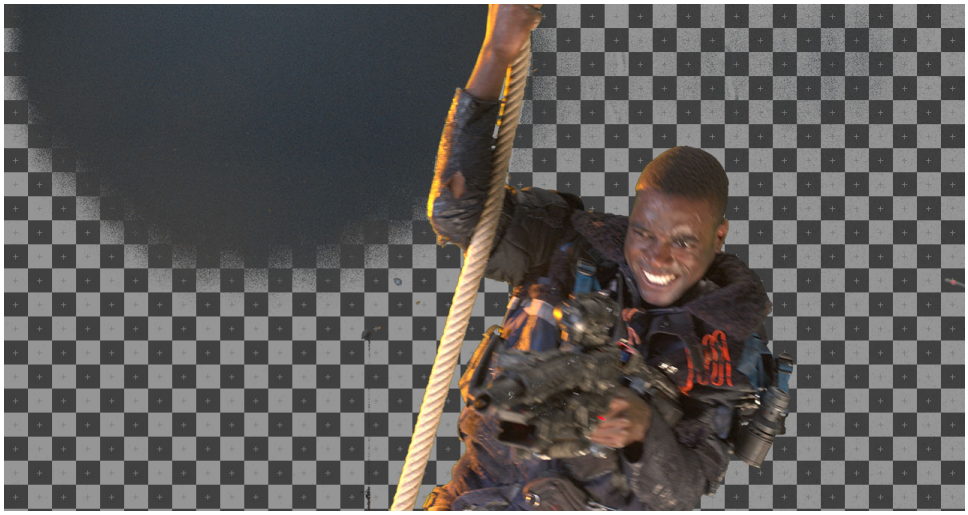


Often, lighting is uneven across the backdrop.



Example from the Mango Open Movie, Tears of Steel.

That can result in a bad matte.

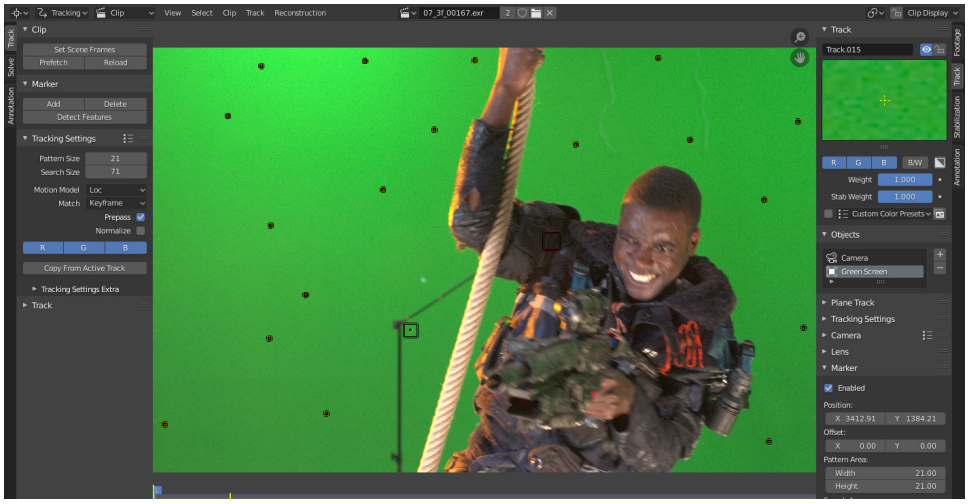


Example of a poor mask: Some of the backdrop is opaque, and some parts of the gun in the foreground are transparent.

If you increase the tolerances on the Color Key node, it will accept more shades of green to mask out. But it may also incorrectly mask out more of the foreground.

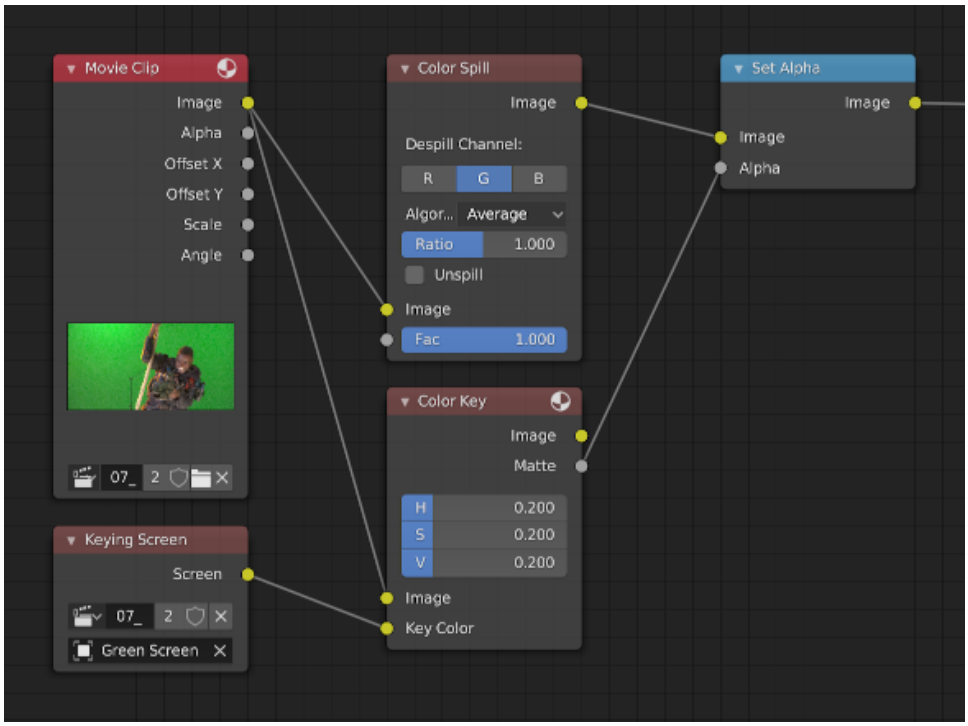
Instead of increasing the range of accepted shades to be masked out, the Keying Screen node lets you change what shade of green (or other color) to used for different parts of the image.

Start in the Movie Clip Editor. Open the Sidebar region and Toolbar to show tracking configuration. Tracks used for gradients are not useful for camera solving, because they do not track well. So create a new object track in the *Objects* selector. Place tracking markers on the clip to sample different parts of the backdrop.



These tracks may be tracked or moved manually, so gradients can be updated over time. If the marker is not enabled for a frame, it will not be used creating the gradient. (Such as the red-colored marker on the arm in the screenshot above)

Once the tracks are created, add the node to your compositing setup, and select the tracking object used for the backdrop.

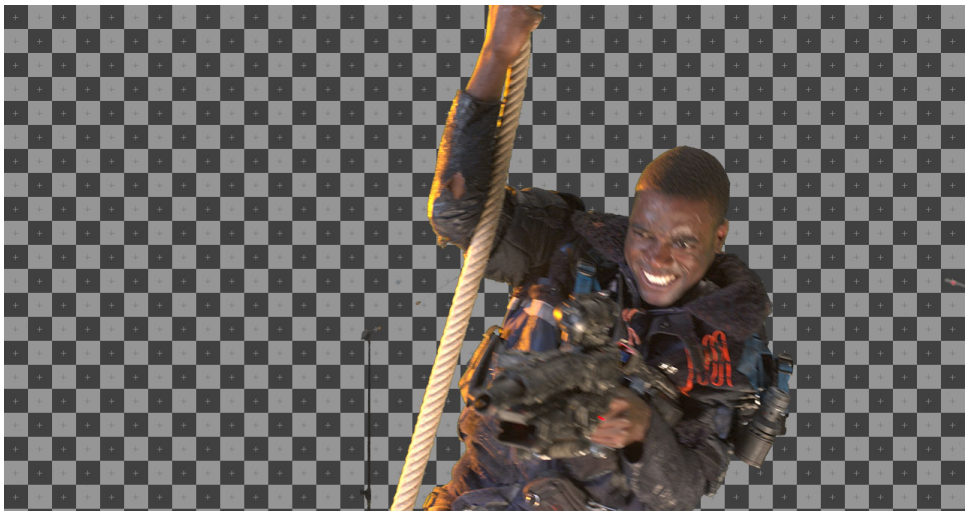


Node configuration with *Keying Screen*'s generated gradient plate connected to the Color input of the Keying node.

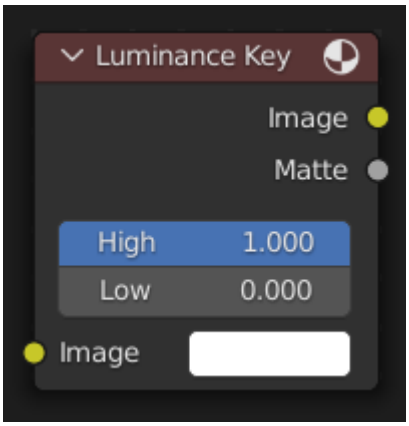


Gradient plate generated by *Keying Screen*.

The resulting image now has a better matte.



Luminance Key Node



The *Luminance Key* node determines background objects from foreground objects by the difference in the luminance (brightness) levels.

Stock footage of explosions, smoke or debris are normally shot against a solid, dark background rather than a green screen. This node can separate the foreground effect from the background. It can also be used for sky replacement for overexposed or gray skies that aren't suitable for chroma keying.

Tip

When compositing footage of something that emits light and has a dark background, like fire, a Mix Node using a *Screen* or *Add* operator will produce better results.

Inputs

Image
: Standard color input.

Properties

Limit
: High
: Determines the lowest values that are considered foreground. (Which is supposed to be – relatively – light: from this value to 1.0.)

Low
: Determines the highest values that are considered to be background objects. (Which is supposed to be – relatively – dark: from 0.0 to this value.)

Note

Brightness levels between the two values form a gradient of transparency between foreground and background objects.

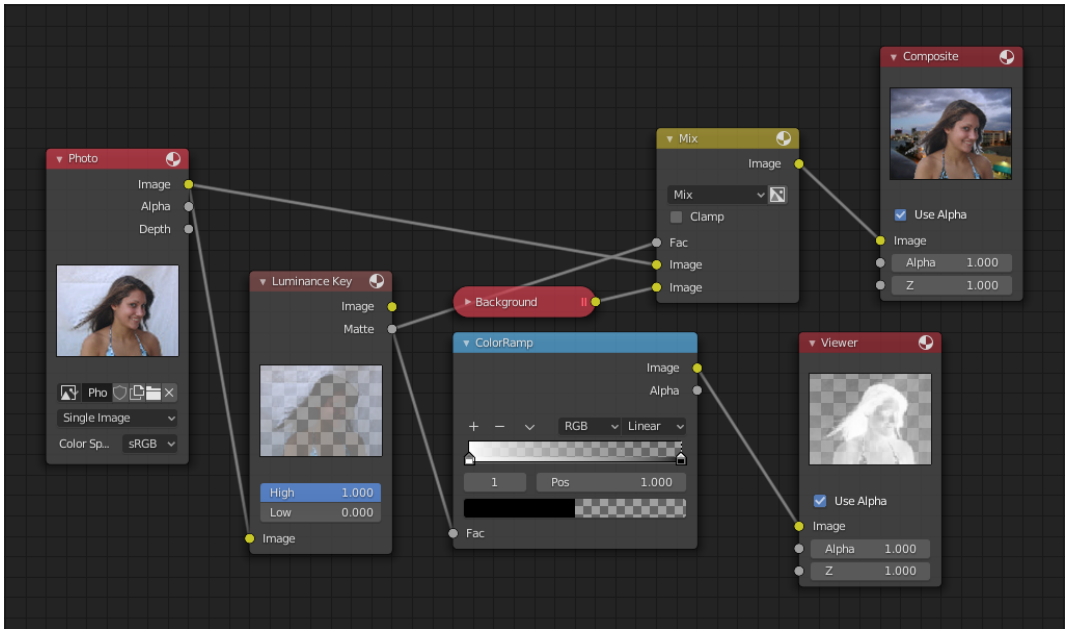
Outputs

Image
: Image with an alpha channel adjusted for the keyed selection.

Matte
: A black-and-white alpha mask of the key.

Example

For this example the model was shot against a *white* background. Using the Luminance Key node, we get a matte out where the background is white, and the model is black; the opposite of what we want. If we wanted to use the matte, we have to switch the white and the black. How to do this? Color Ramp node to the rescue – we set the left color to White Alpha 1.0, and the right color to be Black Alpha 0.0. Thus, when the Color Ramp gets in black, it spits out white, and vice versa. The reversed mask is shown; its white outline is usable as an alpha mask now.



Using Luma Key with a twist.

Now to mix, we do not really need the *Alpha Over* node; we can just use the mask as our Factor input. In this kinda weird case, we can use the matte directly; we just switch the input nodes. As you can see, since the matte is white (1.0) where we do not want to use the model picture, we feed the background photo to the bottom socket (recall the Mix node uses the top socket where the factor is 0.0, and the bottom socket where the factor is 1.0). Feeding our original photo into the top socket means it will be used where the Luminance Key node has spit out Black. Voilà, our model is teleported from Atlanta to aboard a cruise ship docked in Miami.

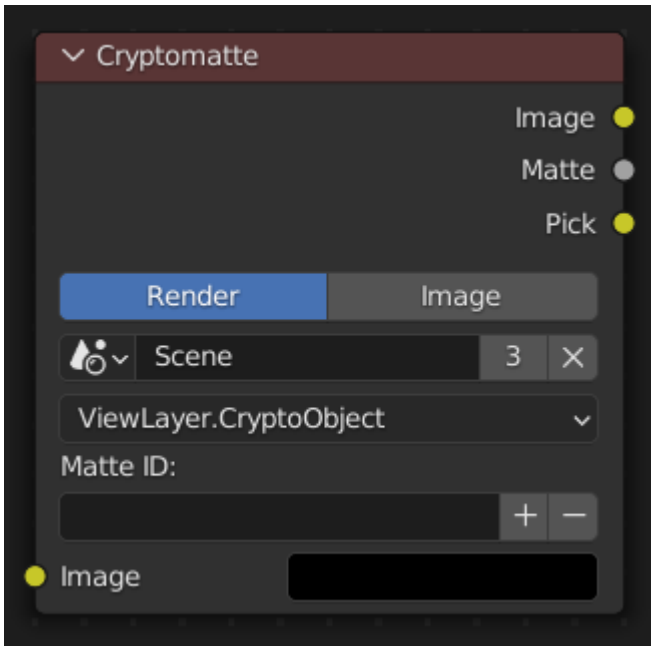
Mask Nodes

- Cryptomatte Node
- Cryptomatte Node (Legacy)

- Box Mask Node
- Ellipse Mask Node

- Double Edge Mask Node
- ID Mask Node

Cryptomatte Node



The Cryptomatte node uses the Cryptomatte standard to efficiently create mattes for compositing. Cycles and EEVEE output the required render passes, which can then be used in the Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing, and mattes will be anti-aliased and take into account effects like motion blur and transparency.

Inputs

Image
: Standard color input.

Properties

Source
: The source of the Cryptomatte data.

Render:: Use Cryptomatte data that are stored as part of the render.

Image:: Use Cryptomatte data that are stored inside a multilayered OpenEXR image.

Scene
: Scene selector. Only available when Render Source is selected.

Image
: Image selector. Only available when Image Source is selected.

Cryptomatte Layer
: Selector of the Cryptomatte layer.

Matte ID
: List of object and material crypto IDs to include in matte. This can be used for example to quickly clear all mattes by deleting the text or used to copy-and-paste crypto IDs from other software.

Outputs

Image
: A colored output of the input image with the matte applied to only include selected layers.

Matte
: A black-and-white alpha mask of all the selected crypto layers.

Pick
: A colored representation of the Cryptomatte pass which can be used as a higher contrast image for color picking.

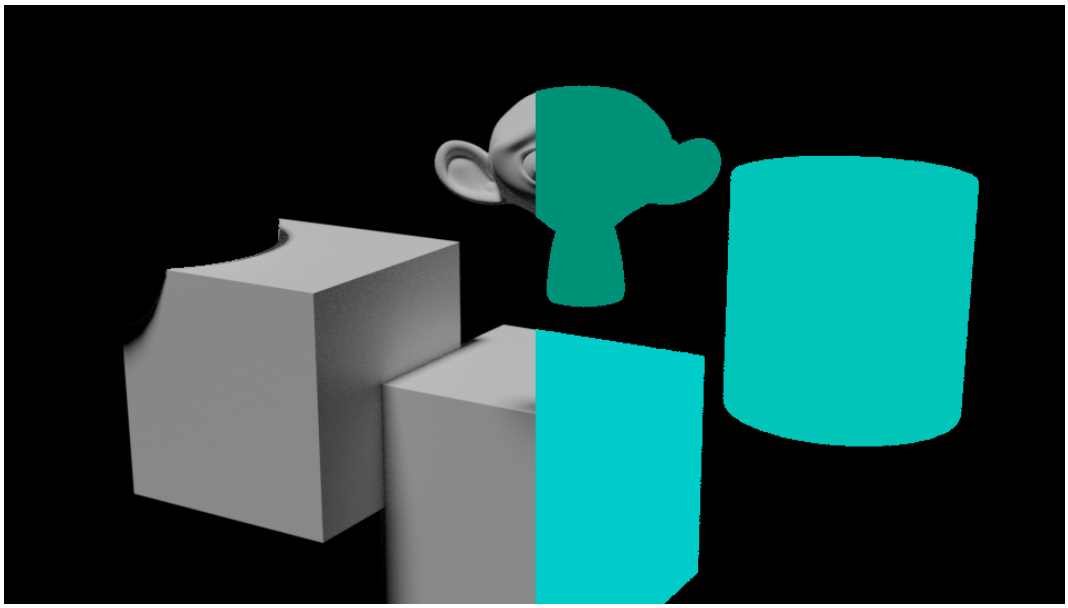
Usage

1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and select the Cryptomatte layer.
3. Attach a Viewer node to the combined pass of the render layers.
4. Use the Cryptomatte Add/Remove button to sample objects from the Compositor backdrop.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

The Image editor, UV editor, node backdrop or Movie Clip editor can be used to pick a Cryptomatte sample. They don't need to show any Cryptomatte layer. The node will use the sample image coordinate to sample in the Cryptomatte layer that is selected in the node.

Example

In the example below, you can see the pass output on the right side. On the left side you can see a couple of objects that were selected through the *Cryptomatte* node. Notice how the cube on the left has a sphere-shaped cut-out from a sphere that was not selected in the node.



Limitations

- Cryptomatte sidecars (metadata files) are not supported.
- Cryptomatte node cannot be used in node groups.
- Volume Objects are not supported.

Cryptomatte Node (Legacy)



The Cryptomatte node uses the Cryptomatte standard to efficiently create mattes for compositing. Cycles and EEVEE output the required render passes, which can then be used in the Compositor or another compositor with Cryptomatte support to create masks for specified objects.

Unlike the Material and Object Index passes, the objects to isolate are selected in compositing, and mattes will be anti-aliased and take into account effects like motion blur and transparency.

Important

The Cryptomatte Legacy node is deprecated and replaced by Cryptomatte Node. The legacy node will be removed in a future Blender release.

Inputs

Image
: Standard color input.

Crypto Passes
: Each crypto layer will be given its own render pass; each of these render passes must be connected to one of these crypto layer inputs. By default there are only four layers, see Adding/Removing Layers to add more.

Properties

Add/Remove
: Adds/Removes an object or material from matte, by picking a color from the *Pick* output.

Matte ID
: List of object and material crypto IDs to include in matte. This can be used for example to quickly clear all mattes by deleting the text or used to copy-and-paste crypto IDs from other software.

Outputs

Image
: A colored output of the input image with the matte applied to only include selected layers.

Matte
: A black-and-white alpha mask of the all the selected crypto layers.

Pick
: A colored representation of the Cryptomatte pass which can be used with a Viewer node to select which crypto passes are used to create the matte image.

Usage

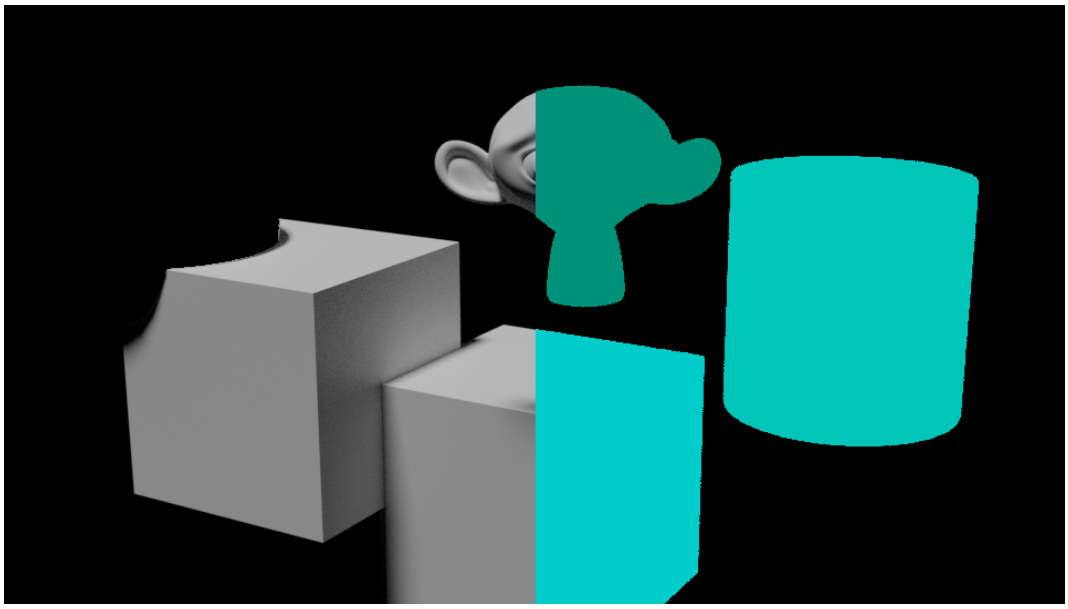
1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and link the Render Layer matching Image and Cryptomatte passes to it.
3. Attach a Viewer node to the Pick output of the Cryptomatte node.
4. Use the Cryptomatte Add/Remove button to sample objects in the Pick Viewer node.
5. Use the Matte output of the Cryptomatte node to get the alpha mask.

Adding/Removing Layers

By default there are only four crypto layers available as inputs to the Cryptomatte node. You can add or remove layer inputs through *Sidebar ▸ Item ▸ Properties ▸ Add/Remove Crypto Layer*. These operators will add/remove layers from the bottom of the pass inputs.

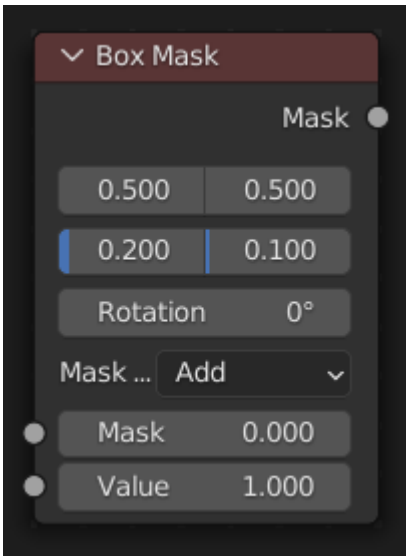
Example

In the example below, you can see the pass output on the right side. On the left side you can see a couple of objects that were selected through the *Cryptomatte* node. Notice how the cube on the left has a sphere-shaped cut-out from a sphere that was not selected in the node.



Box Mask Node

The *Box Mask* node creates an image suitable for use as a simple matte.



Inputs

Mask
: An optional mask to use as the base for mask operations.

Value
: Intensity of the generated mask.

Properties

X, Y
: Position of the center of the box as a fraction of the total width or height. (0.5, 0.5 creates a centered box; 0.0, 0.0 creates a box in the lower left.)

Width
: Width of the box as a fraction of the total image width.

Height

Height of the box as a fraction of the total image *width*, not height.

Rotation
: Rotation of the box around its center point.

Mask Type
: Operation to use against the input mask.

Add::	This yields the *union* of the input mask and the generated mask: Areas covered by the generated mask are set to the specified *Value*. Other parts of the input masked are passed through unchanged, or set to black if there is no input mask.
Subtract::	Values of the input mask have the specified *Value* subtracted from them.
Multiply::	This yields the *intersection* of this generated mask and the input mask: Values of the input mask are multiplied by the specified *Value* for the area covered by the generated mask. All other areas become black.
Not::	Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified *Value*. Areas uncovered by the generated mask remain unchanged.

Outputs

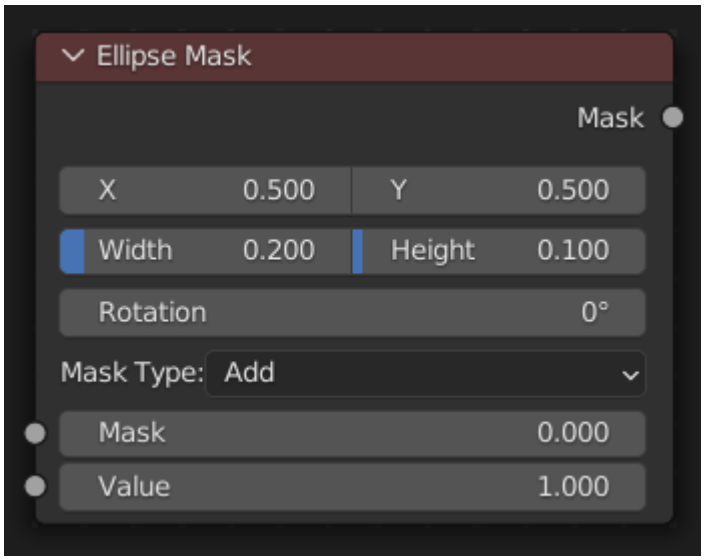
Mask
: A generated rectangular mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip

For soft edges, pass the output mask through a slight Blur node.

Ellipse Mask Node

The *Ellipse Mask* node creates an image suitable for use as a simple matte or vignette mask.



Inputs

Mask
: An optional mask to use as the base for mask operations.

Value
: Intensity of the generated mask.

Properties

X, Y
: Position of the center of the ellipse as a fraction of the total width or height. (0.5, 0.5 creates a centered ellipse; 0.0, 0.0 creates an ellipse with its center in the lower left.)

Width
: Width of the ellipse as a fraction of the total image width.

Height

Height of the ellipse as a fraction of the total image *width*, not height. Equal *Width* and *Height* values with produce a circle.

Rotation
: Rotation of the ellipse around its center point.

Mask Type
: Operation to use against the input mask.

Add::	This yields the *union* of the input mask and the generated mask: Areas covered by the generated mask are set to the specified *Value*. Other parts of the input masked are passed through unchanged, or set to black if there is no input mask.
Subtract::	Values of the input mask have the specified *Value* subtracted from them.
Multiply::	This yields the *intersection* of this generated mask and the input mask: Values of the input mask are multiplied by the specified *Value* for the area covered by the generated mask. All other areas become black.
Not::	Any area covered by both the input mask and the generated mask becomes black. Areas covered by the generated mask that are black on the input mask become the specified *Value*. Areas uncovered by the generated mask remain unchanged.

Outputs

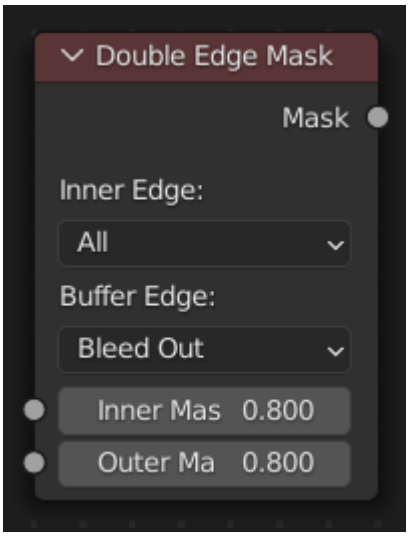
Mask
: A generated elliptical mask merged with the input mask. The created mask is the size of the current scene render dimensions.

Tip

For soft edges, pass the output mask through a slight Blur node. For a vignette, pass the output of this through a heavy blur.

Double Edge Mask Node

The *Double Edge Mask* node creates a gradient between two masks.



Inputs

Inner Mask
: A mask representing the inside shape, which will be fully white.

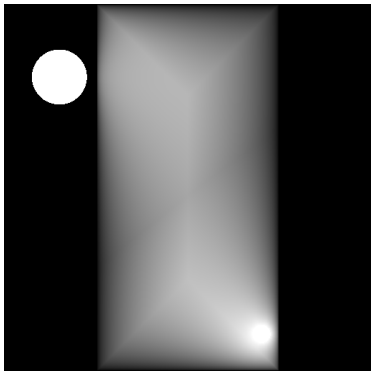
Outer Mask
: A mask representing the outside shape, which will fade from black at its edges to white at the *Inner Mask*.

Properties

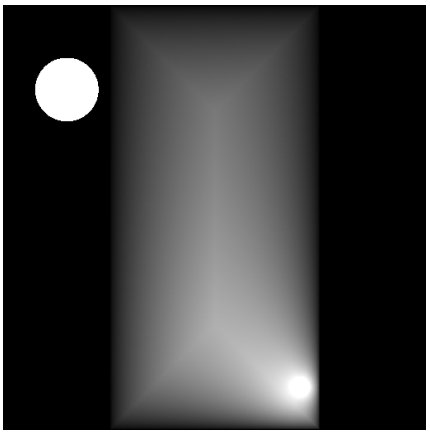
Inner Edge

All:: All shapes in the *Inner Mask* contribute to the gradient, even ones that do not touch the *Outer Mask* shape.

Adjacent Only:: Only shapes in the *Inner Mask* that overlap with the *Outer Mask* contribute to the gradient.



All.

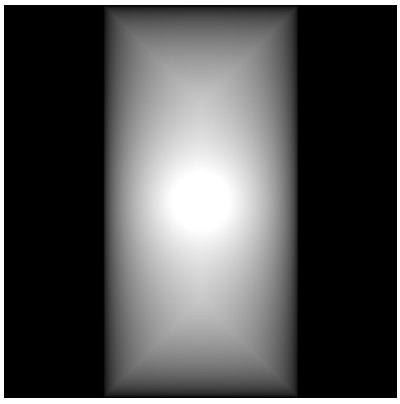


Adjacent Only.

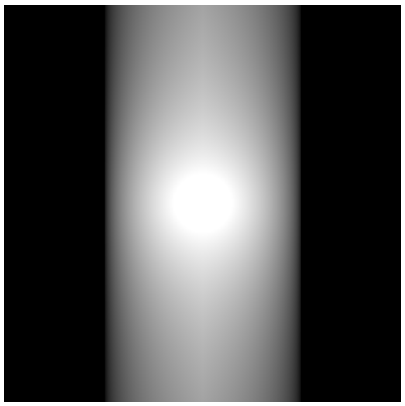
Buffer Edge

Keep In:: Parts of the *Outer Mask* that touch the edge of the image are treated as if they stop at the edge.

Bleed Out:: Parts of the *Outer Mask* that touch the edge of the image are extended beyond the boundary of the image.



Keep In.



Bleed Out.

Outputs

Mask

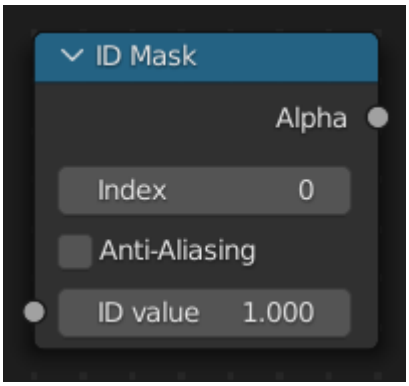
Standard mask output.

Example

Double Edge Mask Example Video

ID Mask Node

The *ID Mask Node* can be used to access an alpha mask per object or per material.



See also

The ID Mask node is superseded by the Cryptomatte Node. Cryptomatte is more feature complete and supported by Cycles and EEVEE. It is recommended to use this feature moving forward.

Inputs

ID Value
: Input for the *Object Index* or *Material Index* render pass. Which is an output of the Render Layers node or the Image node with a multi-layer format.

Properties

Index
: Selection of the previously specified index.

Anti-Aliasing
: This post-processing filter smooths the mask edges. See Anti-Aliasing.

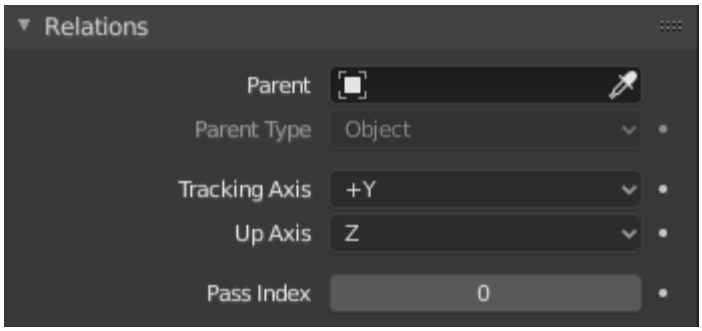
Outputs

Alpha

The mask is white where the object is and black where it is not. If the object is transparent, the alpha mask represent that with gray values.

Setup

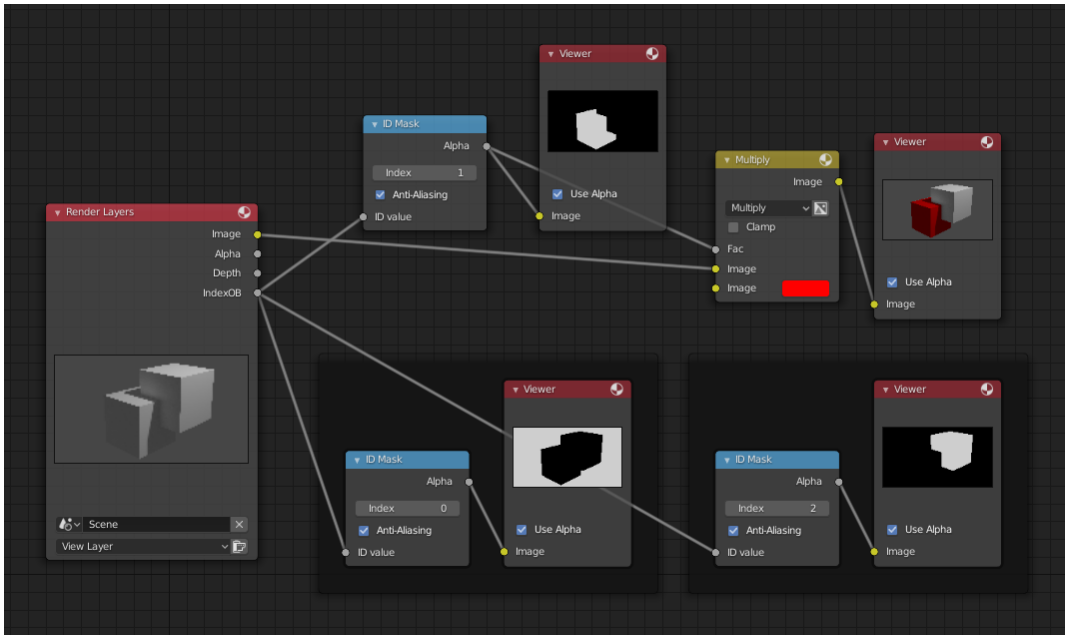
An index can be specify for any object or material in the scene. The Object Index can be set in *Properties ▸ Object Properties ▸ Relations ▸ Pass Index* and *Material ▸ Settings ▸ Pass Index* for the Material Index. To be accessible after rendering, the Render Engine must be Cycles, and *Object Index* or *Material Index* render pass has to be enabled.



Object Pass Index.

Example

In this example, the left rear red cube is assigned Pass Index 1, and the right cube Pass Index 2. Where the two cubes intersect, there is going to be noticeable pixelation because they come together at a sharp angle and are different colors. Using the mask from object 1, which is smoothed (anti-aliased) at the edges, we use a *Mix Node* set on *Multiply* to multiply the smoothed edges of the image, thus removing those nasty lines, thus, being smoothed out.



ID Mask node example.

Limitations

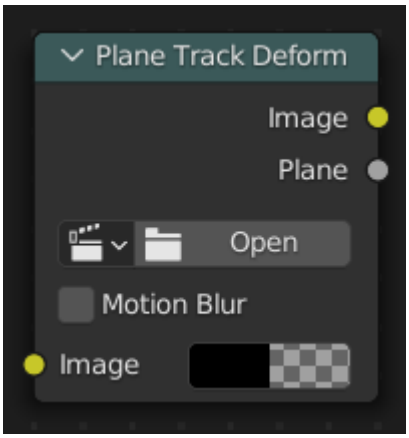
- Volume Objects are not supported.

Tracking Nodes

- Plane Track Deform Node
- Stabilize 2D Node
- Track Position Node

Plane Track Deform Node

The Plane Track Deform Node is used replace flat planes in footage by another image, using plane tracks from motion tracking.



Plane Track

Before using this node, plane track for the footage should be made in the *Movie Clip Editor*.

Inputs

Image
: Image to put in place of the plane track, and thus, override that area in the movie clip.

Properties

Movie Clip
: Used to select the movie clip whose plane track to use. For controls see Data-Block Menu.

Object
: Used to select the object to which the plane track is linked.

Track

Used to select the plane track to use.

Motion Blur
: Specify whether to use blur caused by motion of the plane track or not.

  Samples
  : Set the number of samples to take for each frame. The higher this number, the smoother the blur effect, but the longer the render, as each virtual intermediate frame has to be rendered.

    Note

    Samples are taken only from the *next* frame, not the previous one. Therefore the blurred object will appear to be slightly ahead of how it would look without motion blur.

  Shutter
  : Time (in frames) the shutter is open. If you are rendering at 24 fps, and the Shutter is set to 0.5, the time in between frames is 41.67 ms, so the shutter is open for half that, 20.83 ms.

Outputs

Image
: The output by perspective wrapping the image to that plane track.

Plane
: Produces a black-and-white mask of the plane track.

Examples

Using Image Output

This can simply be achieved by using the Alpha Over node.

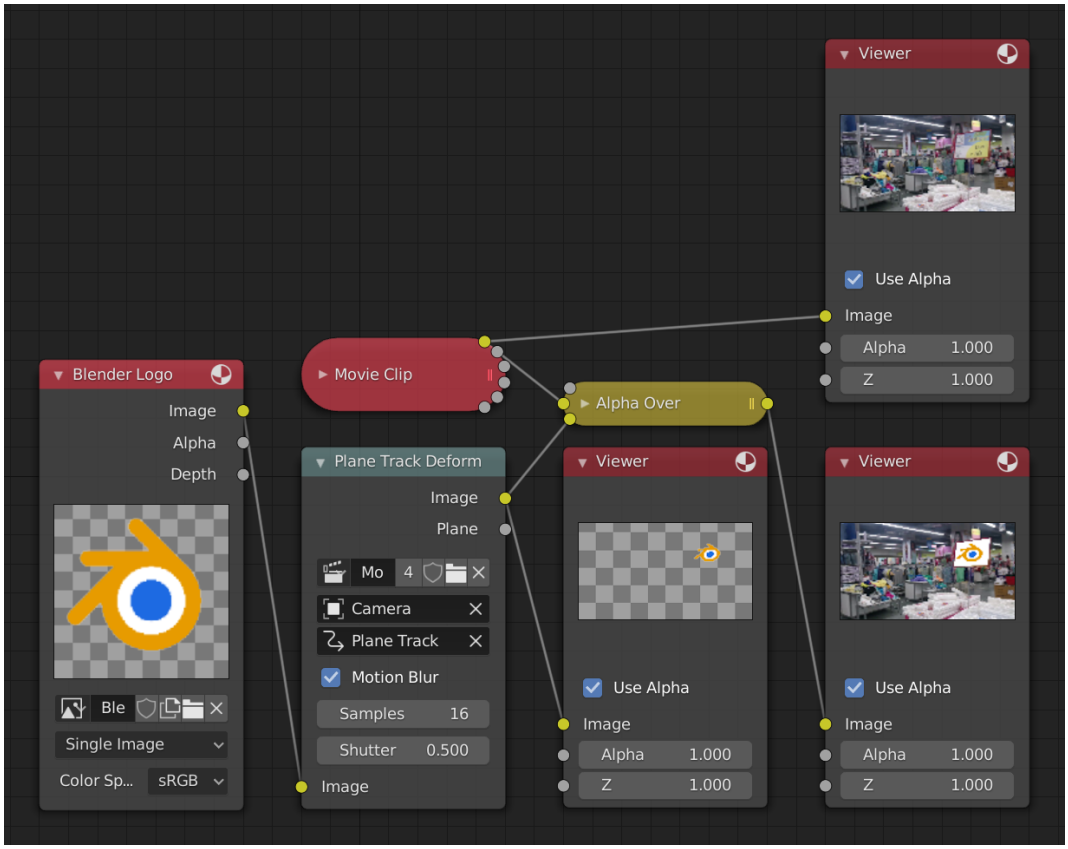
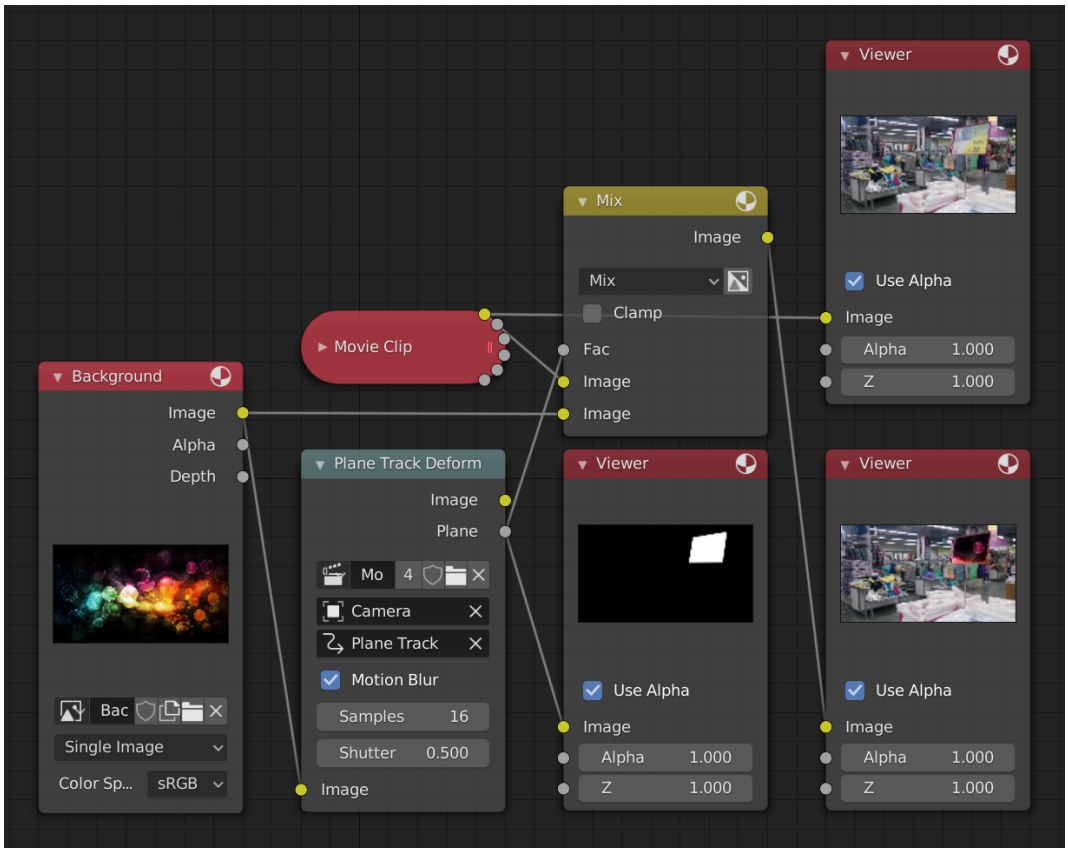


Image output.

Using Plane Output

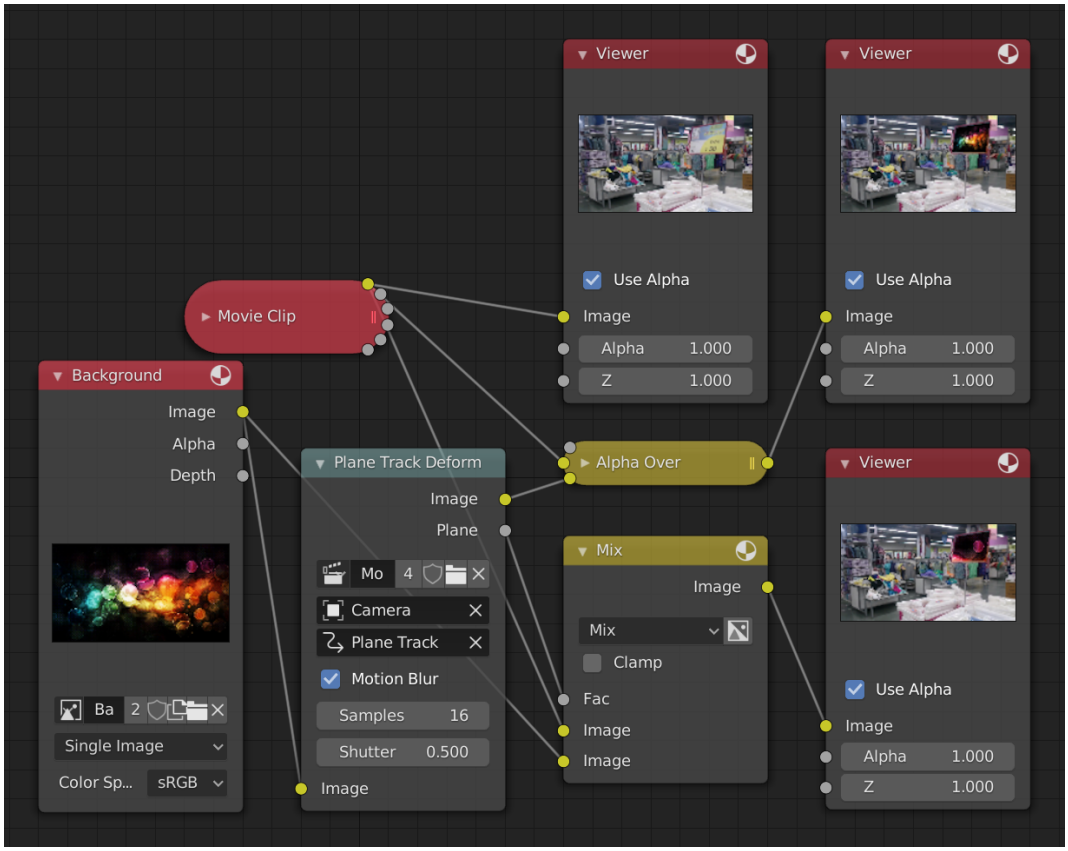
This can be achieved by mixing the movie clip and the image by using the plane output as the factor.



Plane output.

Using Image Output vs Using Original Image

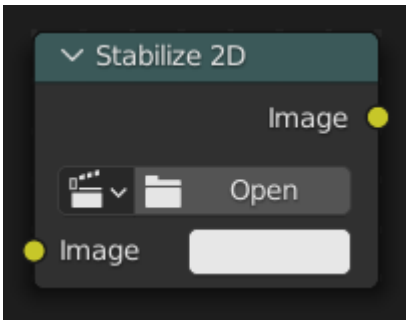
Using Image output scales, moves, and skews the input image according to the track while using the original image and mixing it with the movie clip using Plane output as factor will display the part of the image that lies inside that mask. This image shows the difference:



Comparison between image output and original image (see Viewer nodes carefully).

Stabilize 2D Node

Stabilizes the footage according to the settings set in *Movie Clip Editor* ‣ *Properties* ‣ *Stabilization* ‣ *2D Stabilization*. For more information, see 2D Stabilization.



Inputs

Image
: Standard color input.

Properties

Movie Clip
: The movie clip whose stabilization to use.

Interpolation
: Various methods for the stabilization. Usually, the same as used in *Movie Clip Editor* ‣ *Properties* ‣ *Stabilization* ‣ *2D Stabilization* ‣ *Interpolate*. For technical details on their difference, see this. But for most purposes, default of Bilinear should suffice.

Invert
: Invert the stabilization. If the stabilization calculated is to move the movie clip up by 5 units, this will move the movie clip down by 5 units.

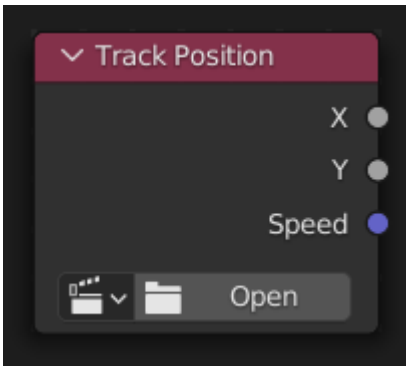
Outputs

Image

Standard color input.

Track Position Node

The *Track Position node* is used to return information about a tracking marker to the Compositor.



Inputs

This node has no inputs.

Properties

Movie Clip
: Used to select a Movie Clip data-block to use, for controls see Data-Block Menu.

Tracking Object
: Camera object to get track information from.

Track Name
: The name of the track to get track information from.

Position
: Which marker position to use for output.

Absolute:: Outputs an absolute position of a marker.
Relative Start:: Outputs the positions of a marker relative to the first marker of a track.
Relative Frame::

Outputs the positions of a marker relative to the markers of the given *Frame*.

Absolute Frame:: Outputs the absolute positions of a marker at the given *Frame*.

Outputs

X/Y
: The marker's X and Y location.

Speed
: The velocity of the marker, measured in pixels per frame. This could be used to fake effects like motion blur by connecting it to the Vector Blur Node.

Transform Nodes

These nodes distort the image in some fashion, operating either uniformly on the image, or by using a mask to vary the effect over the image.

- Rotate Node
- Scale Node
- Transform Node
- Translate Node

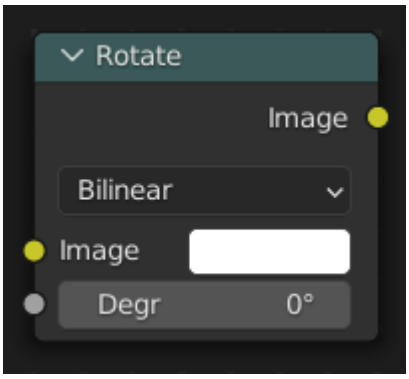
- Corner Pin Node
- Crop Node

- Displace Node
- Flip Node
- Map UV Node

- Lens Distortion Node
- Movie Distortion Node

Rotate Node

This node rotates an image.



Inputs

Image
: Standard color input.

Degr
: Rotation angle in degree. Positive values rotate clockwise and negative ones counterclockwise.

Properties

Filter
: Interpolation Methods.

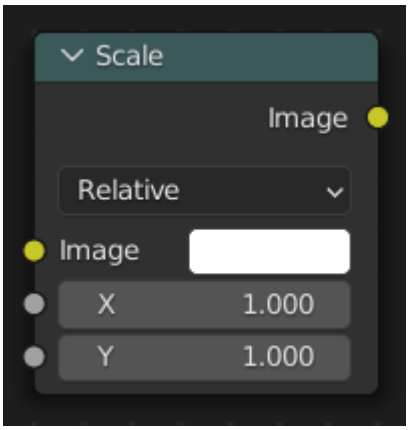
Nearest::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

Outputs

Image
: Standard color output.

Scale Node

This node scales the size of an image.



Inputs

Image
: Standard color input.

X, Y
: Scale in the axis directions, only available if *Space* is set to *Relative* or *Absolute*.

Properties

Space
: Coordinate Space to scale relative to.

  Relative
  : Percentage values relative to the dimensions of the image input.

  Absolute
  : Size of an image by using absolute pixel values.

  Scene Size

Sizes an image to the size of the final render resolution for the scene. For example, rendering a scene at the standard 1080p resolution but setting the render percentage at 50%, will produce a 1080p image with the scene scaled down 50% and leaving the rest of the image as alpha.

Render Size
: Image dimensions set in the Render panel.

  Stretch, Fit, Crop
  : Stretch distorts the image so that it fits into the render size. Fit scales the image until the bigger axis “fits” into the render size. Crop cuts the image so that it is the same aspect ratio as the render size.

  X, Y
  : Offset factor for the final scaled image.

Outputs

Image
: Standard color output.

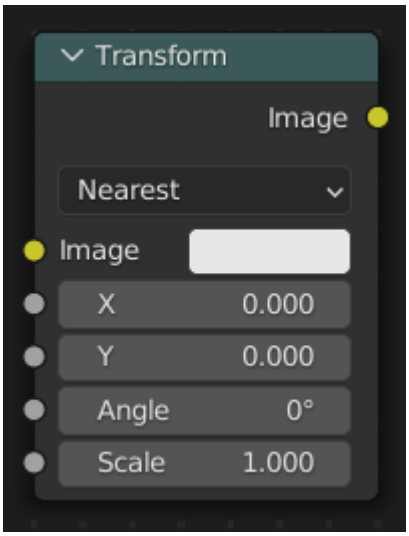
Examples

For instance X: 0.5 and Y: 0.5 would produce an image which width and height would be half of what they used to be.

Use this node to match image sizes. Most nodes produce an image that is the same size as the image input into their top image socket. To uniformly combine two images of different size, the second image has to be scaled up to match the resolution of the first.

Transform Node

This node combines the functionality of three other nodes: Scale, translate, and rotate nodes.



Inputs

Image
: Standard color input.

X, Y
: Used to move the input image horizontally and vertically.

Angle
: Used to rotate an image around its center. Positive values rotate counter-clockwise and negative ones clockwise.

Scale
: Used to resize the image. The scaling is relative, meaning a value of 0.5 gives half the size and a value of 2.0 gives twice the size of the original image.

Properties

Filter
: Interpolation Methods.

Nearest::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

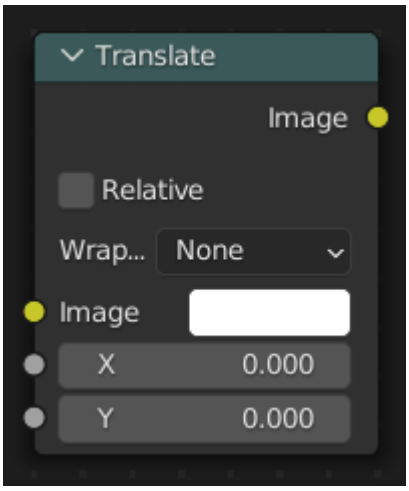
Outputs

Image
: Standard color output.

Translate Node

The Translate node moves an image.

Could also be used to add a 2D camera shake.



Inputs

Image
: Standard color input.

X, Y
: Used to move the input image horizontally and vertically.

Properties

Relative
: Percentage translation values relative to the input image size.

Wrapping
: Repeat pixels on the other side when they extend over the image dimensions, making endless translating possible.

  None, X Axis, Y Axis, Both Axis

Filter

Interpolation Methods.

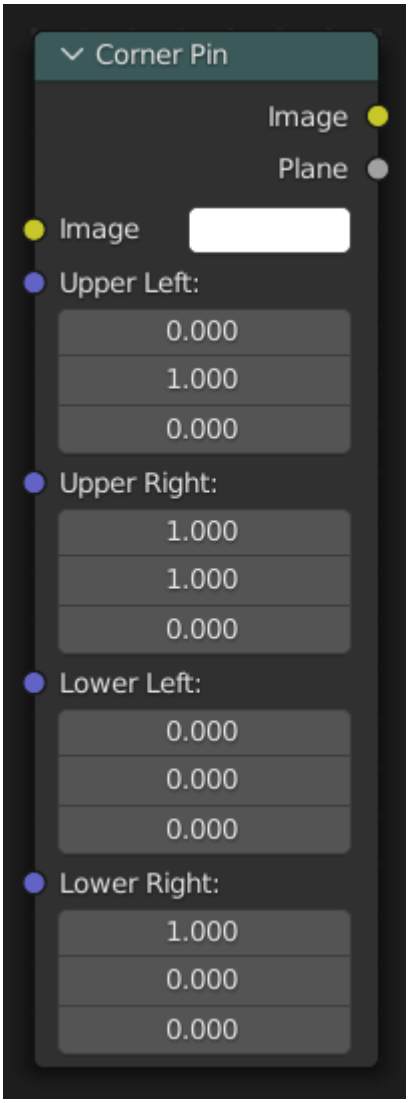
Nearest::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

Outputs

Image
: Standard color output.

Corner Pin Node

The Corner Pin node uses explicit corner values for a plane warp transformation. It works like the Plane Track Deform node, but without using "plane track" data from the Movie Clip Editor.



Inputs

Image
: Standard color input.

Corners

Four vector inputs to define the plane warping. (Z component of vector inputs is ignored.)

Properties

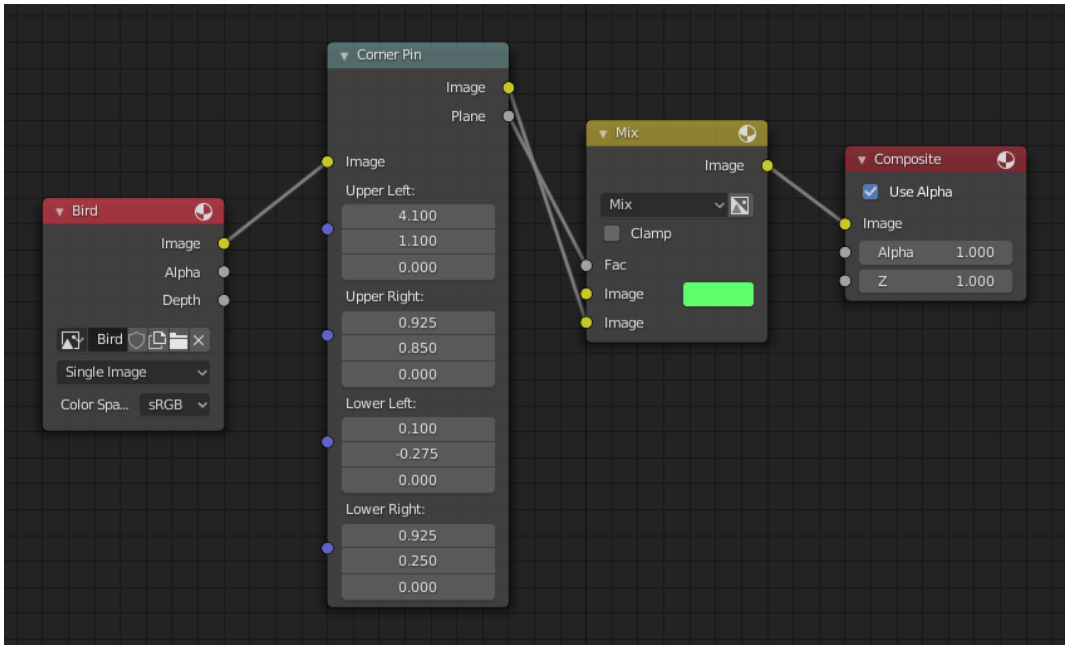
This node has no properties.

Outputs

Image
: Standard color output. (The image after distorting.)

Plane
: A black-and-white alpha mask of the plane.

Example



An example of the Corner Pin node.



An example of the distorted image.

In the example above, the image of the bird is distorted by the vectors specified by the Corner Pin node.

Crop Node

The Crop node crops an input image to a selected region by either making the cropped area transparent or by resizing the input image.



Inputs

Image
: Standard color input. If no image is selected, an image filled with the selected color is used. You can use and crop this image in combination with another background image.

Properties

Crop Image Size
: When disabled, the image remains the same size, but the cropped areas become transparent pixels. When enabled, the image size is cropped to the specified region and gets a new width or height or both.

  Note that this will probably reposition the image in the render output because the cropped image is automatically centered.

Relative

When enabled, crop dimensions are a percentage of the input image's width and height. When disabled, the range of the *Crop Region Values* are the width and height of the image in pixels.

Crop Region Values
: Define borders of the crop region; *Left* or *Right* can vary between 0 and the width of the image. *Up* or *Down* can vary between 0 and the height of the image.

Note

The terminology (*Left*, *Right*, *Up*, *Down*) can be misunderstood easily. First, the numbers do not represent the amount of cropping, e.g. *Left* is set to 50 and *Right* to 50 does not mean that you will be cropping the image for 50 pixels on both the left and right side. In fact, it will result in zero-sized image because you are cropping from pixel 50 to pixel 50. So, the numbers defines a position in the input image.

Secondly, depending on which one is bigger, *Left* should be interpreted as *Right* and vice versa. If *Left* is greater than *Right* then both values are switched and *Left* gets the value of *Right* and vice versa. The same operation is done for *Up* and *Down*, where you can think of them as the top and bottom of the image.

Thirdly, the terms *Up* and *Down* are ambiguous and suggest an action; e.g. "Crop down". The values, however, are not amounts but positions. The term *Down* should be interpreted as "Bottom" and *Up* as "Top".

Outputs

Image
: Standard color output.

Usage

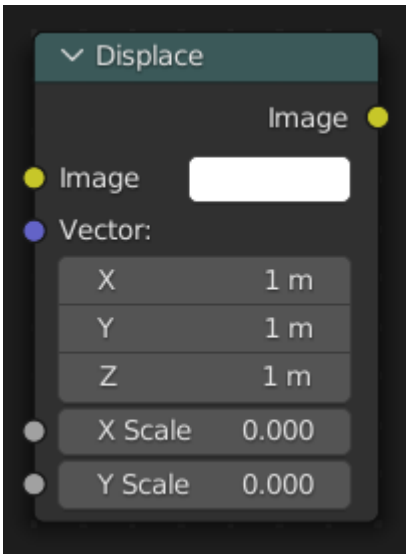
The following workflow removes some possible confusion:

1. Uncheck *Crop Image Size* for this step, so that you can see the borders of the input image. To see this border, you have to select the Viewer node.
2. If you don't need pixel-perfect cropping, check *Relative* so that you do not have to consider the exact dimensions of the input image.
3. Set *Left* and *Down* to zero. Set *Right* and *Up* to one, or to the width and height of the input image. You should see now the entire input image in the backdrop. *Up* is thus interpreted as the top of the image. The origin of the image (0, 0) is at the bottom (down) left corner.
4. To crop from the left, change the *Left* value. To crop from the right, change the *Right* value. To crop from the top, change the *Up* value. To crop from the bottom, change the *Down* value.

Displace Node

The *Displace Node* displaces the pixel position based on an input vector.

This node could be used to model phenomena, like hot air distortion, refractions of uneven glass or for surreal video effects.



Inputs

Image
: Standard color input.

Vector
: Input of the displacement map. If the a color output is implicitly converted in the vector input, the first channel (red) value determines displacement along X axis. The second channel (green) the displacement along Y axis. If the input is a grayscale image, where both the channel values are equal, the input image will be displaced equally in both X and Y directions.

Scale X, Y
: Separate scaling of the vector input in X and Y direction. Acting as multipliers by increasing or decreasing the strength of the displacement along their respective axes.

Properties

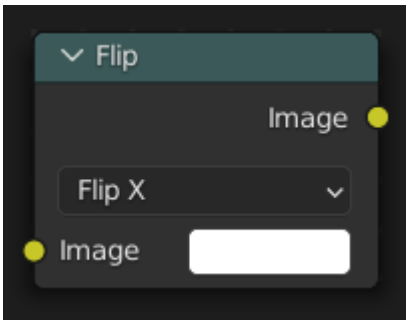
This node has no properties.

Outputs

Image
: Standard color output.

Flip Node

This node flips an image along a defined axis.



You can use this node to just flip or use it as a part of mirror setting. Mix half of the image to be mirrored with its flipped version to produce mirrored image.

Inputs

Image
: Standard color input.

Properties

Axis
: This can be either X or Y. Also, flipping can be done on both X and Y axis simultaneously.

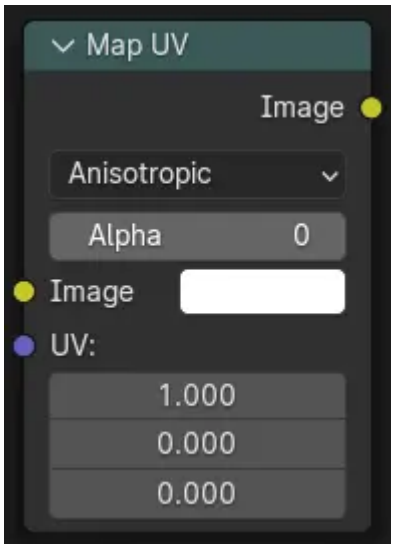
  Flip X, Flip Y, Flip X & Y

Outputs

Image
: Standard color output.

Map UV Node

Map a texture using UV coordinates, to apply a texture to objects in compositing.



May be used in combination with Cryptomatte Node to apply the texture only to specific objects.

Inputs

Image
: The new 2D texture.

UV
: The input for UV render pass. See Cycles render passes.

Hint

To store the UV pass a multi-layer OpenEXR format could be used.

Properties

Filter Type
: Interpolation Methods.

Anisotropic:: Enhances the clarity of textures viewed at oblique angles, addressing issues like blurring and distortion.

Nearest:: No interpolation, uses nearest neighboring pixel.

Alpha

Alpha threshold is used to fade out pixels on boundaries.

Outputs

Image

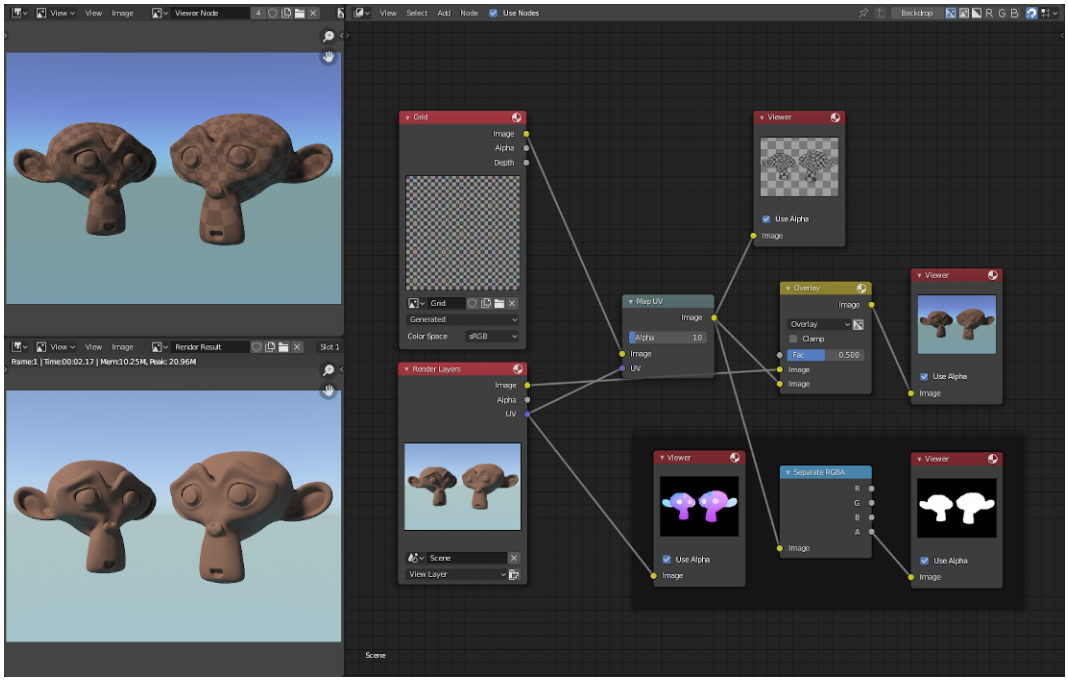
The resulting image is the input image texture distorted to match the UV coordinates. That image can then be overlay mixed with the original image to paint the texture on top of the original. Adjust alpha and the mix factor to control how much the new texture overlays the old.

Hint

When painting the new texture, it helps to have the UV maps for the original objects in the scene, it is recommended to keep those UV texture outlines around even, when shooting is done.

Examples

In the example below, we have overlaid a grid pattern on top of the two heads after they have been rendered. During rendering, we enabled the UV layer in the Properties *Render Layer ▸ Passes*. Using a Mix node ("Overlay" in figure), we mix that new UV texture over the original face. We can use this grid texture to help in any motion tracking that we need to do.

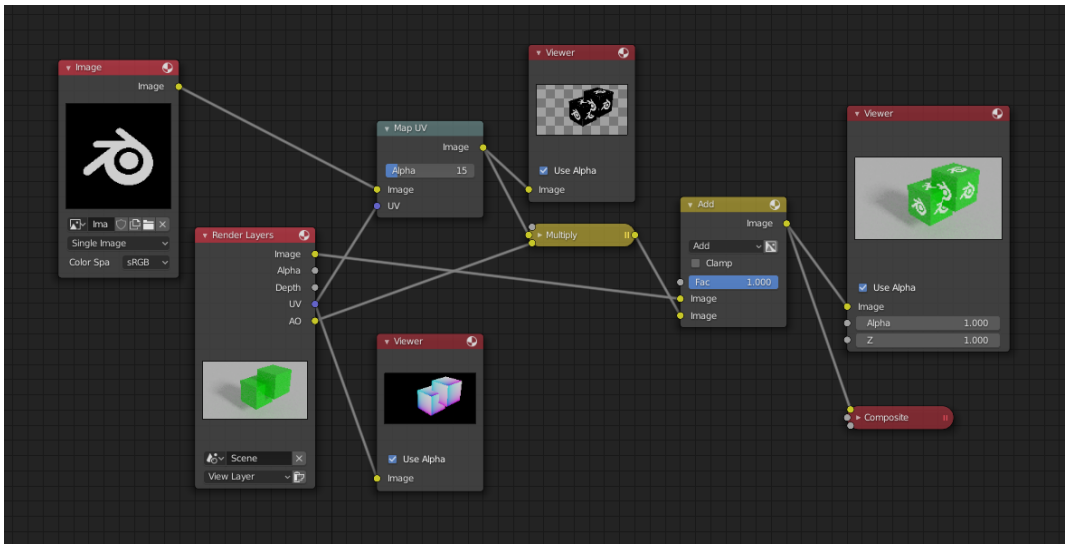


Adding a grid UV textures for motion tracking.

In the next example, we overlay a logo on top of a mesh composed of two intersecting cubes, and we ensure that we Enable the Alpha premultiply button on the Mix node. The logo is used as additional UV texture on top of the existing texture. Other examples include the possibility that there was used an unauthorized product box during the initial animation, and it is needed to substitute in a different product sponsor after rendering.

Hint

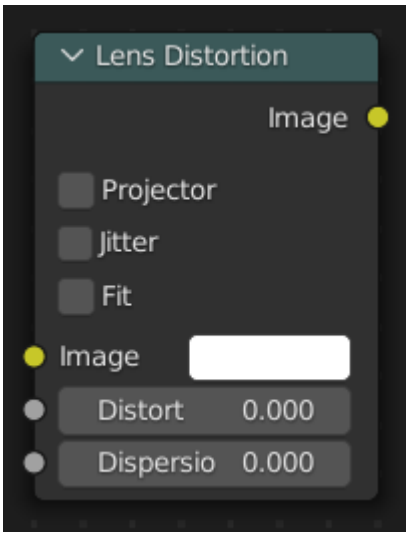
Due to limits of this node, it is not recommended to rush pre-production rendering under the guise of “fixing it later”.



Adding UV textures in post-production.

Lens Distortion Node

Use this node to simulate distortions that real camera lenses produce.



Inputs

Image
: Standard color input.

Distortion
: This creates a bulging or pinching effect from the center of the image.

Dispersion
: This simulates chromatic aberrations, where different wavelengths of light refract slightly differently, creating a rainbow colored fringe.

Properties

Projector
: Enable or disable slider projection mode. When on, distortion is only applied horizontally. Disables *Jitter* and *Fit*.

Jitter

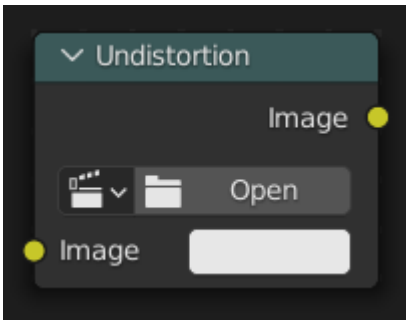
Adds jitter to the distortion. Faster, but noisier.

Fit
: Scales image so black areas are not visible. Only works for positive distortion.

Outputs

Image
: Standard color output.

Movie Distortion Node



In the real world, all camera lenses produce some or the other sort of lens distortion. But, whatever we render has got no distortion. So, this node helps in removing distortion from movies or adding distortion to render to make our render blend in with the movie clip.

Usually, it is used while motion tracking.

Calculating Distortion

Before using this node, one has to calculate the lens distortion of the clip. This can be done by adjusting K1, K2 and K3 values in *Movie Clip Editor* ‣ *Properties* ‣ *Lens*. For more information on how to edit those values, check this out.

Inputs

Image
: Standard color input.

Properties

Movie Clip
: Used to select the movie clip whose distortion is to be used. This can be useful if more than one movie clips are present, each having a different distortion setting. For controls see Data-Block Menu.

Distortion Method

Undistort::	Used to undistort the image received, and is usually used for the raw distorted movie clip.

Distort:: Used to distort the image received, and is usually used for rendered images.

Outputs

Image
: The image after distorting/undistorting.

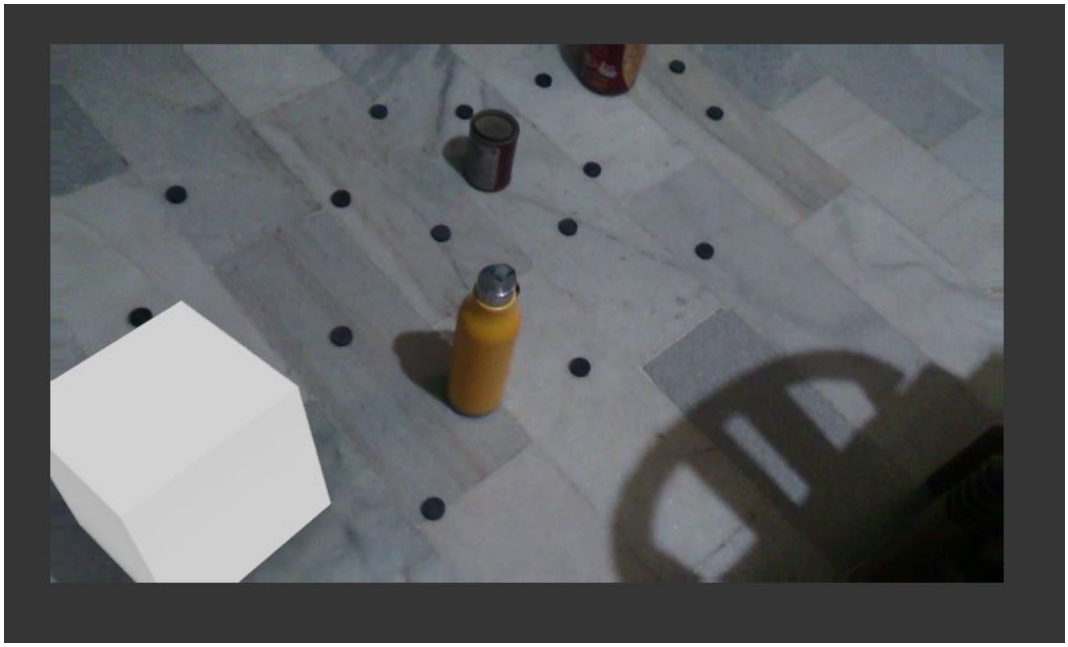
Distortion vs Undistortion

Although, both, distortion of render and undistortion of movie clip are possible, and produce similar results, there is a difference between these two methods.

There are two kinds of lens distortion possible and, in simple terms, they can be said as:

1. When the movie clip is bulging out.
2. When the movie clip is bulging in.

For the first case, it is recommended to distort the render and leave the movie clip as it is, because, undistorting the movie clip will require extra pixel information, which is not available to Blender. Similarly, in the second case, it is recommended to undistort the movie clip and leave the render as it is, because, distorting the render will require those extra unavailable pixels. Doing the wrong method in the wrong case can create weird results around the edges, such as in the image shown.



Problems (notice the edges?)

Utilities Nodes

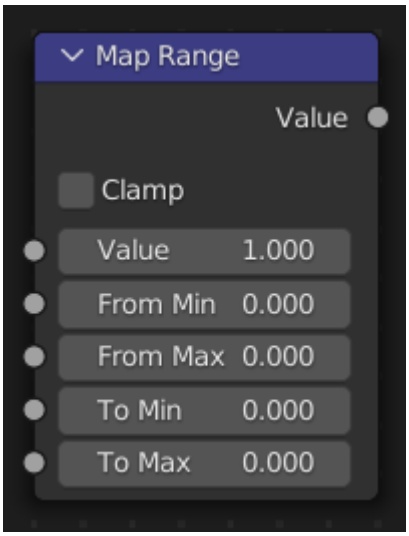
- Map Range Node
- Map Value Node
- Math Node

- Levels Node
- Normalize Node

- Split Node
- Switch Node
- Switch View Node

Map Range Node

This node converts (maps) an input value range into a destination range. By default, values outside the specified input range will be proportionally mapped as well. This node is similar to *Map Value* node but provides a more intuitive way to specify the desired output range.



Inputs

Value
: The input value to be remapped.

From Min
: The lower bound of the range to remap from.

From Max
: The higher bound of the range to remap from.

To Min
: The lower bound of the target range.

To Max
: The higher bound of the target range.

Properties

Clamp
: Clamps values to Min/Max of the destination range.

Outputs

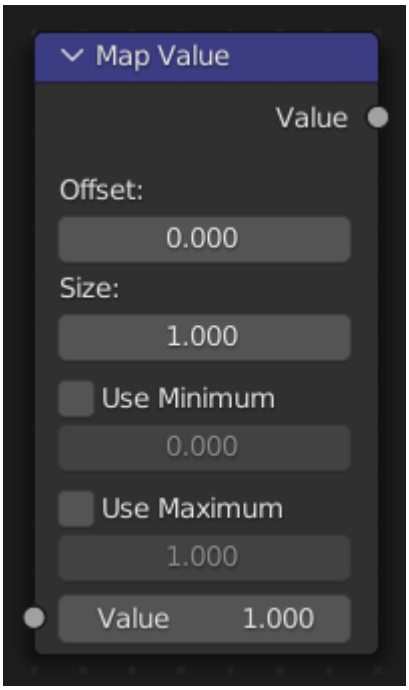
Value
: Standard value output.

Usage

One important use case is to easily map the original range of the Z-depth channel to a more usable range (i.e: 0.0 - 1.0) for use as a matte for colorization or filtering operations.

Map Value Node

Map Value node is used to scale, offset and clamp values.



Inputs

Value
: Standard Value input. (Value refers to each vector in the set.)

Properties

Offset
: Factor added to the input value.

Size
: Scales (multiply) the input value.

Use Minimum, Maximum
: Enable this to activate their related operation.

Min, Max
: Defines a range between minimum and maximum to Clamp the input value to.

Outputs

Value
: Standard value output.

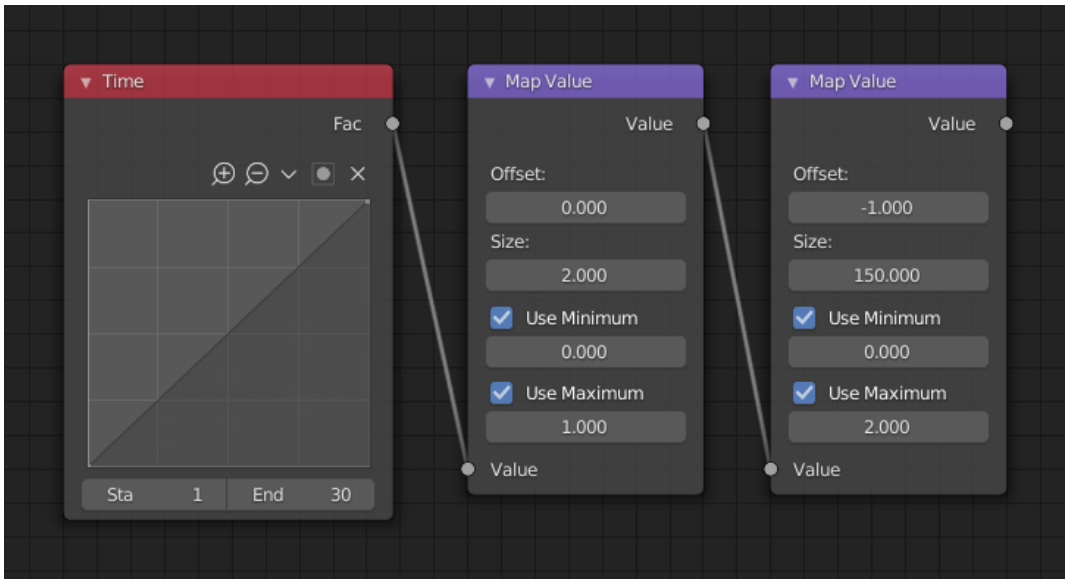
Example

Z-Depth Map

This is particularly useful in achieving a depth of field effect, where the Map Value node is used to map a Z value (which can be 20 or 30 or even 500 depending on the scene) to the range between (0 to 1), suitable for connecting to a Blur node.

Multiplying Values

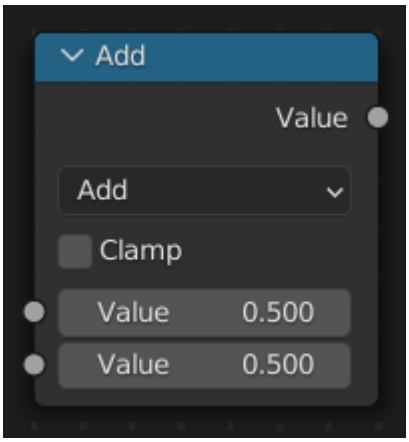
The Map Value node can also be used to multiply values to achieve a desired output value. In the mini-map to the right, the Time node outputs a value between 0.0 and 1.0 evenly scaled over 30 frames. The *first* Map Value node multiplies the input by 2, resulting in an output value that scales from 0.0 to 2.0 over 30 frames. The *second* Map Value node subtracts 1 from the input, giving working values between (-1.00 to 1.0), and multiplies that by 150, resulting in an output value between (-150 to 150) over a 30-frame sequence.



Using Map Value to multiply.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value
: Input Value. Trigonometric functions read this value as radians.

Addend
: Input Addend.

Base
: Input Base.

Exponent
: Input Exponent.

Epsilon
: Input Epsilon.

Distance

Input Distance.

Min
Input Minimum.

Max
Input Maximum.

Increment
Input Increment.

Scale
Input Scale.

Degrees
Input Degrees.

Radians
Input Radians.

Properties

Operation
The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

One divided by the square root of the value.

Absolute:: The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises Euler's number to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: Smooth Minimum.

Smooth Maximum:: Smooth Maximum.

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo:: Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.

Snap:: Rounds the input value down to the nearest integer multiple of *Increment*.

Ping-pong:: Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine:: The Sine of the input value.

Cosine:: The Cosine of the input value.

Tangent:: The Tangent of the input value.

Arcsine:: The Arcsine of the input value.

Arccosine:: The Arccosine of the input value.

Arctangent:: The Arctangent of the input value.

Arctan2:: Outputs the Inverse Tangent of the first value divided by the second value measured in radians.

Hyperbolic Sine:: The Hyperbolic Sine of the input value.

Hyperbolic Cosine:: The Hyperbolic Cosine of the input value.

Hyperbolic Tangent:: The Hyperbolic Tangent of the input value.

Conversion

To Radians:: Converts the input from degrees to radians.

To Degrees:: Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

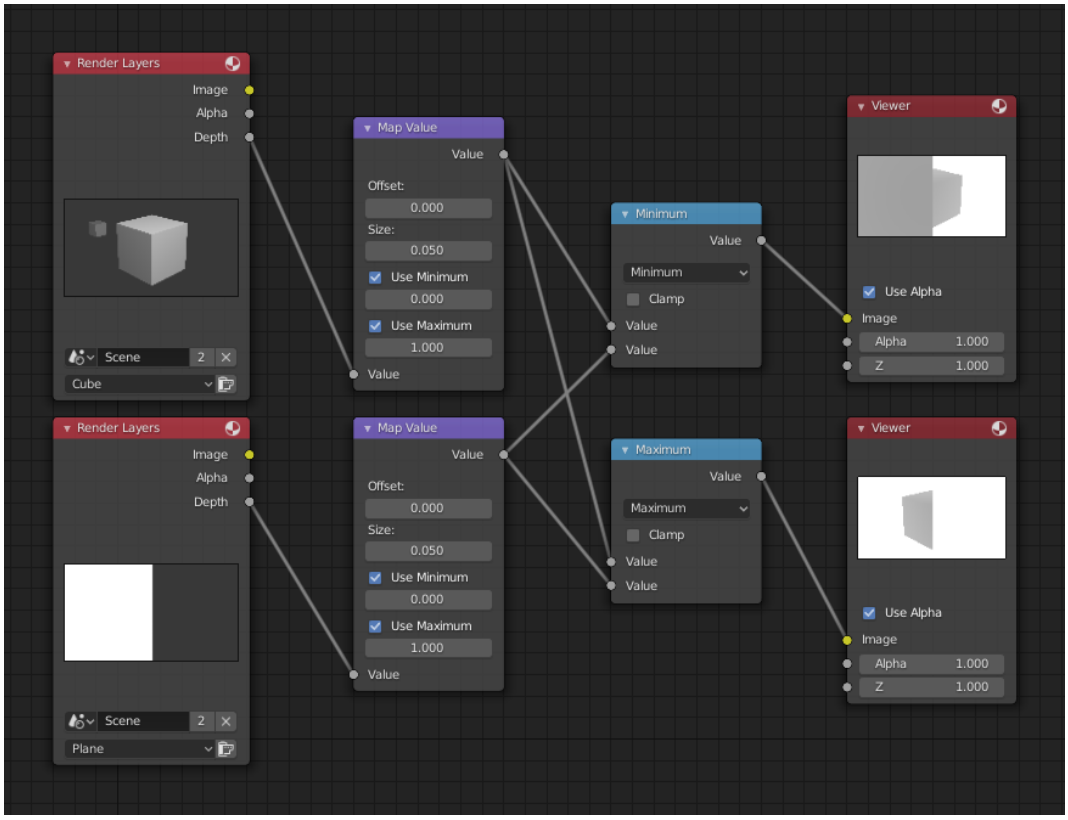
Outputs

Value

Numerical value output.

Examples

Manual Z-Mask



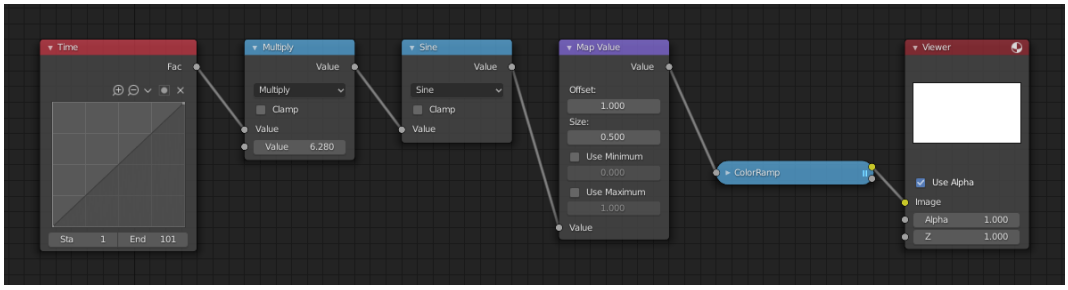
Minimum and maximum function example.

The top *Render Layers* node has a cube that is about 10 units from the camera. The bottom *Render Layers* node has a plane that covers the left half of the view and is 7 units from the camera. Both are fed through their respective *Map Value* nodes to multiply the depth value by 0.05 and clamp it to [0.0, 1.0], bringing it into a suitable range for displaying it as a color.

The Minimum node selects the smallest of the two depth values at each pixel. In the left half, it chooses the plane (because it's closer than the cube), and in the right half, it chooses the cube (because it's closer than the background, which is infinitely far away).

The Maximum node selects the largest of the two depth values at each pixel. In the left half, it chooses the cube (because it's farther away than the plane), and in the right half, it chooses the background (because it's farther away than the cube).

Using Sine Function to Pulsate



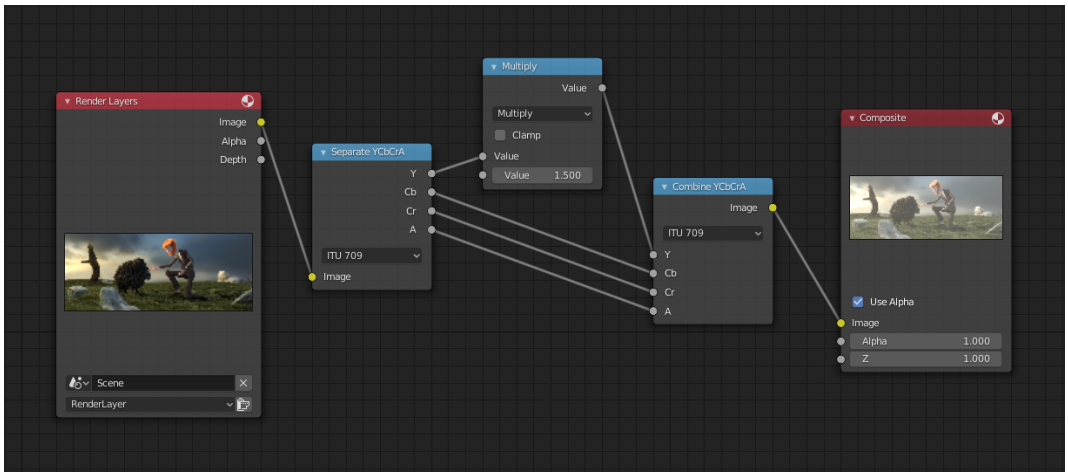
Using sine function example.

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. At frame 25, the output value is 0.25. That value is multiplied by 2 × pi (6.28) and converted to 1.0 by the Sine function, since \(sin(2 × pi/ 4) = sin(pi/ 2) = +1.0\).

Since the sine function can output values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by 0.5 (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a gray-scale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, \(sin(pi/ 2) = 1.0\). Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color “pulse”.

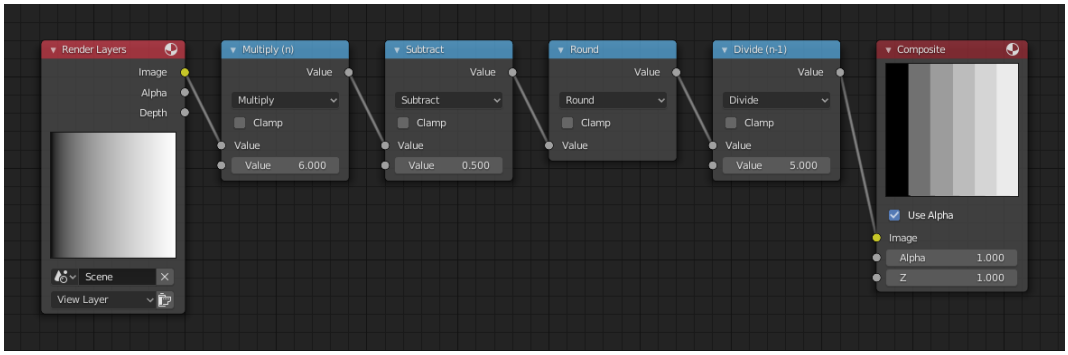
Brightening (Scaling) a Channel



Scaling a channel example.

This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high dynamic range image. For this particular example, there is also a *Brightness/Contrast node* that might give simpler control over brightness.

Restrict Color Selection (Posterization)



Posterization example.

In this example, we restrict the color values to be one of the six values: 0, 0.2, 0.4, 0.6, 0.8, 1.

To split up a continuous range of values between 0 and 1 to certain set of values, the following function is used: \(round(x × n - 0.5) / (n - 1)\), where

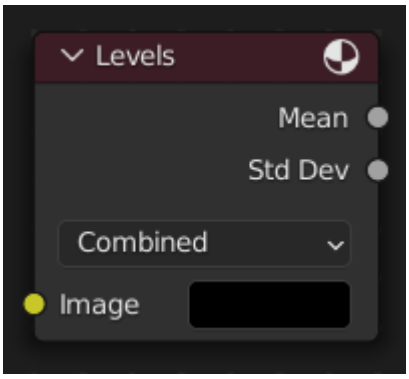
“n” is the number of possible output values, and “x” is the input pixel color. Read more about this function.

To implement this function in Blender, consider the node setup above. We string the Math nodes into a function that takes each color (values from 0 to 1), multiplies it up by six, the desired number of divisions (values become from 0 to 6), offsets it by 0.5 (-0.5 to 5.5), rounds the value to the nearest whole number (produces 0, 1, 2, 3, 4, 5), and then divides the image pixel color by five (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

In the case of a color image, you need to split it into separate RGB channels using *Separate/Combine Color* nodes and perform this operation on each channel independently.

Levels Node

The Levels Node read the input color channels and outputs analytical values. The output is one-dimensional meaning the visualization will be a uniform gray color.



Inputs

Image
: Standard color input.

Properties

Channel
: Selects which color values are used to calculate analytics.

Combined::	Calculate values based on the red, green, and blue channels.
Red::	Calculate values based on the red channel.
Green::	Calculate values based on the green channel.
Blue::	Calculate values based on the blue channel.
Luminance::	Calculate values based on the Luminance of the image.

Outputs

Mean

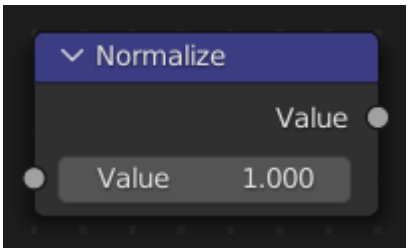
The mean is the average value of all image pixels in specified channel. It represents the overall brightness of the image and can be used as such for setups that depend on how “bright” or “dark” the input is.

Standard Deviation

How much pixel values differ from the mean. A low standard deviation indicates that the pixel values tend to be very close to the mean. A high standard deviation indicates that the values are spread out over a large range of values.

Normalize Node

Find the minimum and maximum values of a single channel. Then map the values to a range of 0 and 1. This is mostly useful for the Z-buffer.



Inputs

Value
: Standard value input.

Properties

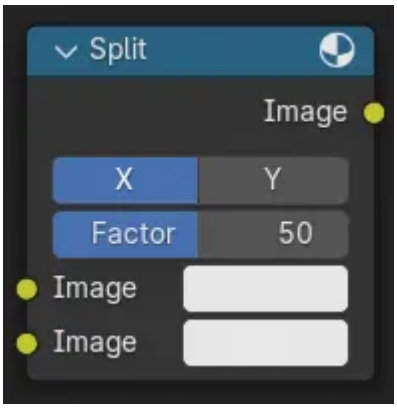
This node has no properties.

Outputs

Value
: Standard value output.

Split Node

The *Split* node combines two images for side-by-side display. Typically used in combination with the Viewer Node.



Inputs

Image
: Shown on the right or top half set by the axis.

Image
: And respectively the left or bottom half.

Properties

Axis
: X or Y used as the split axis.

Factor
: Percentage factor setting the space distribution between the two images.

Outputs

This node has no output sockets.

Hint

This node could be used to plan scene transitions by comparing the end frame of one scene with the start frame of another to make sure they align.

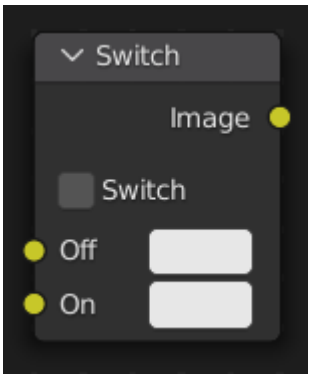
Examples



Example of a Split Viewer node.

Switch Node

Switch between two images using a checkbox.



Inputs

Image
: First image input.

Image
: Second image input.

Properties

Switch
- When it is unchecked, the first input labeled “Off” is passed to the output.
- When checked, the second input labeled “On” is passed to the output.

Outputs

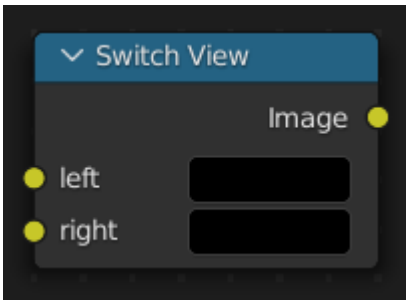
Image
: Standard color output.

Tip

Switch state may be animated by adding a keyframe. This makes the Switch node useful for bypassing nodes which are not wanted during part of a sequence.

Switch View Node

The *Switch View* node combines the *views* (left and right) into a single stereo 3D output. This can be useful if for example, you need to treat the view as separate images by combining each of the views.



See also

The multi-view workflow.

Inputs

Left
: Left-eye image input.

Right
: Right-eye image input.

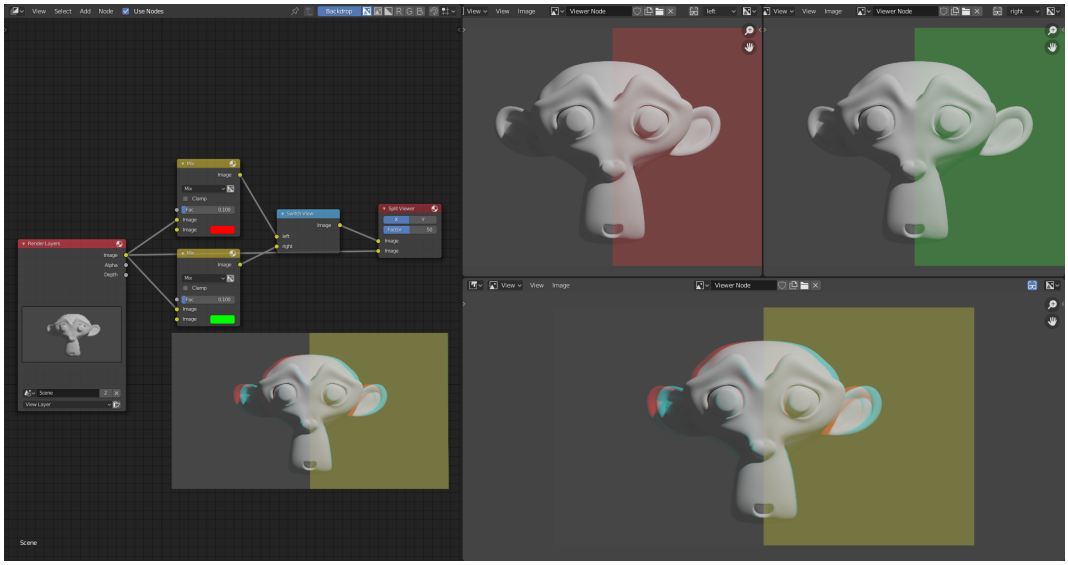
Properties

This node has no properties.

Outputs

Image
: Stereo 3D image output.

Example



Compositor, Backdrop and Split Viewer Node.

The views to render are defined in the current scene views, in a similar way as you define the composite output resolution in the current scene render panel, regardless of the Image nodes resolutions or Render Layers from different scenes.

Vector Nodes

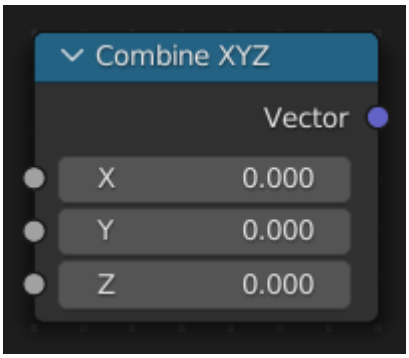
These nodes can be used to manipulate various types of vectors, such as surface normals and speed vectors.

- Combine XYZ Node
- Separate XYZ Node

- Normal Node
- Vector Curves Node

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.



Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

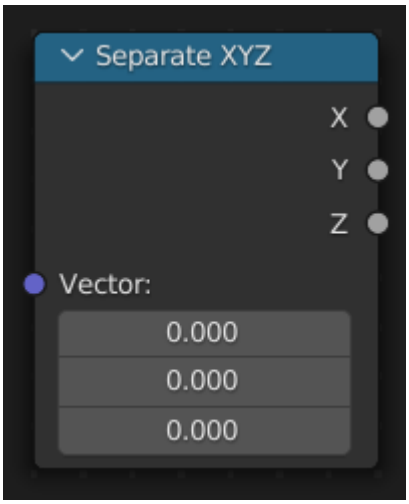
Vector
: Standard vector output.

Note

The vector is not normalized.

Separate XYZ Node

The *Separate XYZ Node* splits a vector into its individual components.



Input

Vector
: Standard vector input.

Properties

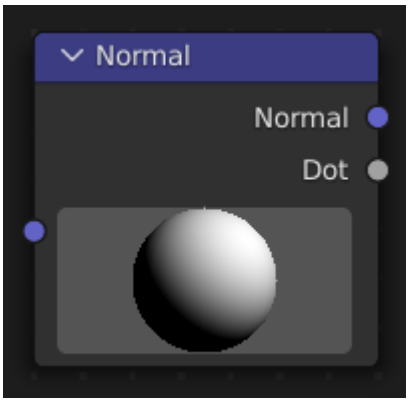
This node has no properties.

Outputs

- X
- Y
- Z

Normal Node

The Normal node generates a normal vector and a dot product.



Inputs

Normal
: Normal vector input.

Properties

Normal Direction
: To manually set a fixed normal direction vector. `LMB` click and drag on the sphere to set the direction of the normal. Holding `Ctrl` while dragging snaps to 45 degree rotation increments.

Outputs

Normal
: Normal vector output.

Dot
: Dot product output. The dot product is a scalar value.

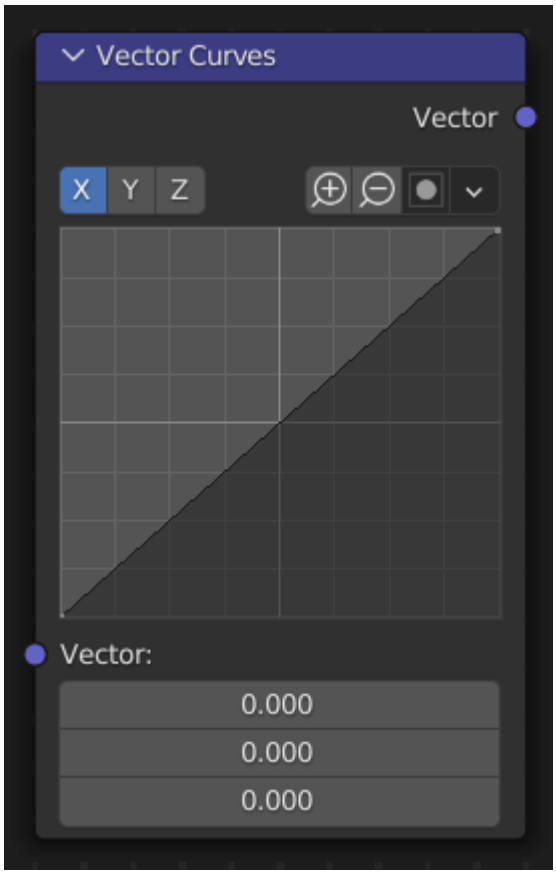
  - If two normals are pointing in the same direction the dot product is 1.

- If they are perpendicular the dot product is zero (0).
- If they are antiparallel (facing directly away from each other) the dot product is -1.

Vector Curves Node

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.



Inputs

In the shader context the node also has an additional Factor property.

Factor
: Controls the amount of influence the node exerts on the output vector.

Vector
: Standard vector input.

Properties

Channel
: X, Y, Z

Curve
: For the curve controls see: Curve widget.

Outputs

Vector
: Standard vector output.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

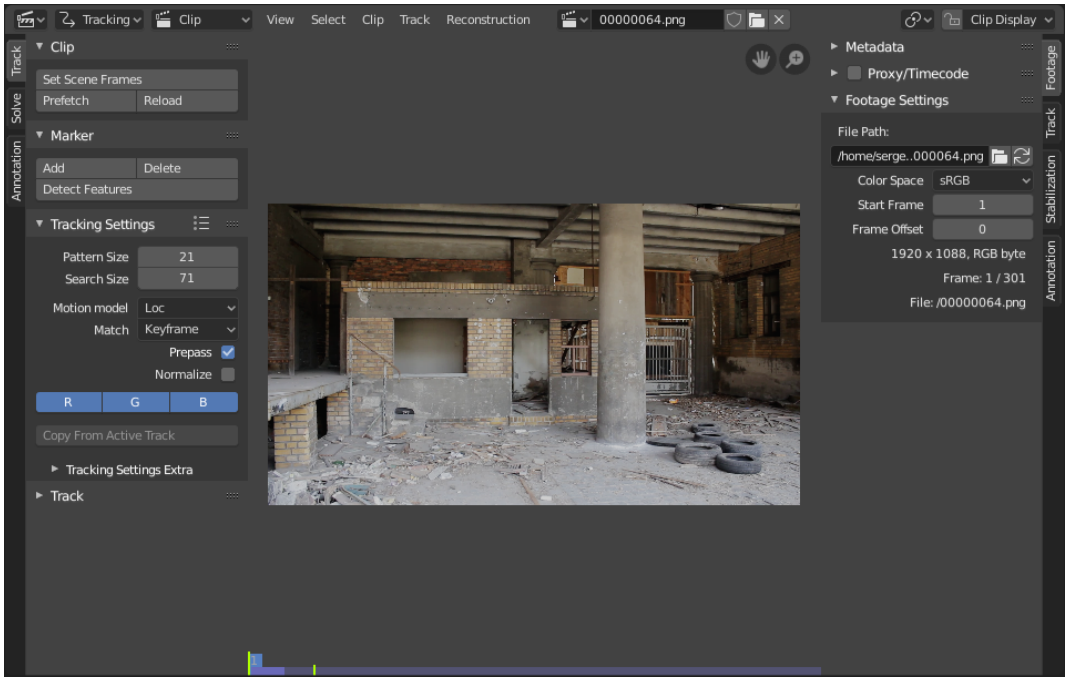
This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Layout Nodes

These are nodes which help you control the layout and connectivity of nodes within the Compositor.

Motion Tracking & Masking

You perform masking and tracking with the Movie Clip Editor.



The Movie Clip Editor.

See Movie Clip Editor for more information on the Movie Clip Editor.

- Motion Tracking
  - Introduction
  - Clip View
  - Graph View
  - Dope Sheet View
- Masking
  - Introduction
  - S-Curves
  - Selecting
  - Editing
  - Sidebar

Motion Tracking

- Introduction
  - Views
  - Manual Lens Calibration
  - Camera and Object Motion Solving
  - Tools for Scene Orientation and Stabilization
- Clip View
  - Introduction
  - Tracking Marker
  - Toolbar
  - Selecting
  - Editing
  - Sidebar
- Graph View
  - Introduction
  - Header
  - Usage
- Dope Sheet View
  - Header
  - Usage

Introduction

Motion Tracking is used to track the motion of objects and/or a camera and, through the constraints, to apply this tracking data to 3D objects (or just one), which have either been created in Blender or imported into the application. Blender's motion tracker supports a couple of very powerful tools for 2D tracking and 3D motion reconstruction, including camera tracking and object tracking, as well as some special features like the plane track for compositing. Tracks can also be used to move and deform masks for rotoscoping in the Mask Editor, which is available as a special mode in the Movie Clip Editor.

Views

In Tracking Mode there are three different views available. You can toggle between view modes using the View selector, which is located in the header. When you selected a view in the whole area of the Movie Clip editor will change. Hence, to display a curve or dope sheet view, the editor must be split into two, with one switched to the curve or dope sheet view.

Manual Lens Calibration

All cameras record distorted video. Nothing can be done about this because of the manner in which optical lenses work. For accurate camera motion, the exact value of the focal length and the "strength" of distortion are needed.

Currently, focal length can be automatically obtained only from the camera's settings or from the EXIF information. There are some external tools which can help to find approximate values to compensate for distortion. There are also fully manual tools where you can use a grid which is getting affected by distortion model and deformed cells defines straight lines in the footage.

Within Blender you can use the Annotation tool for this – just draw a line which should be straight on the footage using poly line brush and adjust the distortion values to make the annotations match lines on the footage.

To calibrate your camera more accurately, use the Grid calibration tool from OpenCV. OpenCV is using the same distortion model, so it should not be a problem.

Camera and Object Motion Solving

Blender not only supports the solving of camera motion, including tripod shots, but also the solving of object motion in relation to the motion of the camera. In addition to that there is the Plane Track, which solves the motion of all markers on one plane.

Tools for Scene Orientation and Stabilization

After solve, you need to orient the real scene in the 3D scene for more convenient compositing. There are tools to define the floor, the scene origin, and the X/Y axes to perform scene orientation.

Sometimes, the video footage includes spurious jumps and tilting movements, like e.g. when using a hand-held camera. Based on some tracked image elements, the 2D Stabilization is able to detect and compensate such movements to improve the quality of the final result.

Clip View

- Introduction
  - Main View
- Tracking Marker
  - Point
  - Plane
- Toolbar
  - Track
  - Solve
- Selecting
  - Box Select
  - Circle Select
  - Lasso Select
  - Select Grouped
  - Select Stabilization Tracks
  - Select Stabilization Rotation Tracks
- Editing
  - Clip
  - Track
  - Reconstruction
- Sidebar
  - Track
  - 2D Stabilization
  - View

Introduction

The Clip View is the main part of the Movie Clip editor; almost all motion tracking tools are concentrated within the Clip View.

It should be mentioned that the camera solver consists of three quite separate steps:

1. 2D tracking of footage.
2. Camera intrinsics (focal length, distortion coefficients) specification/estimation/calibration.
3. Solving camera, scene orientation, and scene reconstruction.

Main View

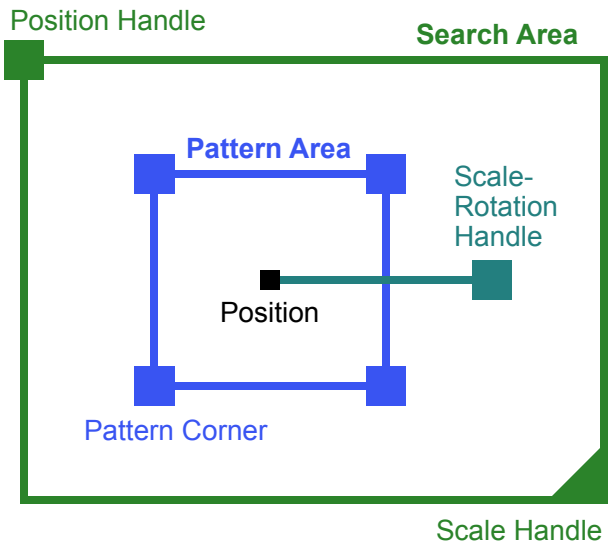
When a clip is loaded a Timeline is shown at bottom of the Preview. It expands over the full area limited by the animation range. You can move the Playhead by dragging with `LMB`.

The Timeline is composed of the following visual elements:

- Blue line: Playhead
- Yellow: Motion track
- Yellow line: Keyframe
- Orange line: Shape keyframe
- Purple: Prefetched frames
- Light green line: Solve start/end keyframe

Tracking Marker

Point



Marker schematic.

The whole marker can be moved with `RMB` or by dragging the anchor point (black dot) with `LMB`. Pressing `G` also moves the whole marker. When pressing `G` twice the marker will be moved while keeping the anchor in place. Note that the anchor point outside the pattern area is shown as a cross connected with marker position with a dashed line.

`S` scales the whole marker. The whole pattern area only will be scaled by pressing `S` twice; The Pattern can also be rotated using the `R` key which, depending on the used pivot point, will either rotate patterns around their own centers or rotate the whole markers around e.g. the median point.

To match the perspective transformation of a marker on a plane, the individual corners must be edited manually. Each marker corner can deform individually to define the shapes. Corner positions can be edited by

dragging them with a mouse. Dragging with LMB will change the position of an individual corner.

Note

Note that deforming a pattern is not only useful for planar / affine tracking. Since only pixels within the pattern will be considered this can help to specify a better pattern to track even for simple position tracking.

The Search area can not be rotated; this is intentional. It doesn't make sense to deform the search area.

Plane

The left bottom corner of the plane does have X/Y axis (X is red, Y is green) to help distinguishing orientation of the plane in space.

It is likely that corner of the plane object need to be manually adjusted. To do this sliding individual corners with mouse LMB or general transform tools G, R, S could be used.

Adjusting plane corners will keep it following the plane defined by tracks it was originally created from.

Toolbar

- Track
  - Clip
  - Marker
  - Tracking Settings
  - Track
- Solve
  - Plane Track
  - Solve
  - Cleanup
  - Geometry
  - Orientation
  - Scene Setup

Track

Clip

Set Scene Frames
: See Set Scene Frames.

Prefetch
: See Prefetch.

Reload
: See Reload Clip.

Marker

Add
: See Add Marker.

Delete
: See Delete Track.

Detect Features
: See Detect Features.

Tracking Settings

This panel contains all settings for the 2D tracking algorithms. Preset tracking settings can be configured in the panel header. These presets are based on tracking experience of real footage and provides good start values to begin working with a specific footage.

Pattern Size, Search Size
: Defines size of a newly created tracks.

Motion Model

Defines which possible motions tracking feature has. This option should be set depending on which motion a particular feature has and it will make tracking most accurate for such a motion.

Location, Location & Rotation, Location & Scale, Location, Rotation & Scale, Affine

Perspective

Is usually used to track a planar feature, but often *Affine* is a good enough approximation and may have more stable tracks.

Match

Controls which patterns get tracked; to be more precise, the pattern from which frame is getting tracked. Here is an example which should make things clearer.

The tracker algorithm receives two images inside the search area and the position of a point to be tracked in the first image. The tracker tries to find the position of that point from the first image in the second image.

Now, this is how tracking of the sequence happens. The second image is always from a frame at which the position of marker is not known (next tracking frame). But a different first image (instead of the one that immediately precedes the second image in the footage) can be sent to the tracker.

Keyframe

An image created from a frame on which the track was keyframed. This configuration prevents sliding from the original position (because the position which best corresponds to the original pattern is returned by the tracker), but it can lead to small jumps and can lead to failures when the feature point is deformed due to camera motion (perspective transformation, for example).

Previous Frame

Keyframes for tracks are creating every frames, and tracking between keyframed image and next image is used. In this configuration the pattern is tracking between two neighboring frames. It allows dealing with cases of large transformations of the feature point but can lead to sliding from the original position, so it should be controlled.

Prepass
: Enables a two pass tracking, where the first pass is a brute force tracking of location only, and the second pass will use tracking of the full motion model refining the first pass.

Normalize
: Means patterns will be normalized by their average intensity while tracking, to make them invariant to illumination changes. An example where this is useful is a scene where a marker moves in the shadow of an object.

R, G, B
: Defines color channels which will be used by a tracking algorithm. Disabling some colors might increase the contrast to enhance the feature detection.

Copy from Active Track
: Copies all settings from active track. Allows to ease creation of new tracks with the same setting.

Tracking Extra Settings

Weight
: See Track Weight.

Correlation
: This value defines the minimal correlation between a matched pattern and a reference to be considered a successful tracking. If the tracker is stops too early, decrease this value, or if the track is slipping too much when it should stop sooner, increase this value.

Margin
: Can be used disable tracks when they become too close to the image boundary. This slider sets “too close” in pixels.

Use Mask
: Allows to use annotation tool to mask part of a pattern, narrowing down what the tracker algorithm is attempting to match across frames.

Frames Limit
: Controls how many frames can be tracked when the Track Sequence operator is called. So, each Track Sequence operation would track maximum *Frames Limit* frames. This also helps to notice a slide-off of tracks and correct them.

Speed
: Marker settings only – Can be used to control the speed of sequence tracking. This option does not affect the quality of tracking; it just helps to control if tracking happens accurately. In most cases tracking happens much faster than real-time, and it is difficult to notice when a track began to slide out of position. In such cases *Speed* can be set to Double or Half to add some delay between tracking two frames, so a slide-off would be noticed earlier and the tracking process can be canceled to adjust positions of tracks.

Track

Track
: See Track Motion.

Clear
: See Before.

Refine
: See Refine.

Merge
: Join Tracks

See Join Tracks.

Solve

Plane Track

See Create Plane Track.

Solve

Tripod
: Tripod Motion can be used for footage where the camera does not move and only rotates. Such footage can't be tracked with a generic solver approach, and it is impossible to determine the actual feature points in space due to a lack of information. So this solver will solve only the relative camera rotation and then reproject the feature points into a sphere, with the same distance between feature and camera for all feature points.

  Note

  This is special type of camera solver and it behaves different from regular solver. It means using more tracks doesn't imply more accurate solution. Having 5-10 tracks on frame is likely what shall be commonly used for this kind of solver.

Keyframe
: Automatically select keyframes for initial reconstruction. This option enables complex algorithms which tries to find a keyframe pair with minimal reconstruction error and best scene scale guess.

Keyframe A/B
: Start (A) and End (B) frame of the range used for reconstruction.

Refine

Specifies which parameters should be refined during solve. Such refining is useful when you are not sure about some camera intrinsics, and solver should try to find the best parameter for those intrinsics. But you still have to know approximate initial values – it will fail to find correct values if they were set completely incorrectly initially.

Focal Length
: Refine the camera's Focal Length.

Optical Center
: Refine the camera's Optical Center.

Radial Distortion
: Refine the camera's Radial Distortion Parameters.

Tangential Distortion
: Refine the camera's Tangential Distortion Parameters.

Solve Camera/Object Motion
: See Solve Solution.

Cleanup

This panel contains operators and their settings which are needed to clean up bad tracks: tracks which are not tracked long enough or which failed to reconstruct accurately.

Frames
: Tracks or tracked segments shorter than this number of frames will be removed.

Error
: Tracks which has reprojection error higher than this value will be removed.

Type
: Several actions can be performed for bad tracks:

Select
: They can simply be selected.

Delete Track
: The whole track can be deleted.

Delete Segments
: Bad segments of tracked sequence can be removed.

Clean Tracks
: See Clean Tracks.

Filter Tracks
: See Filter Tracks.

Geometry

3D Markers to Mesh
: See 3D Markers to Mesh.

Link Empty to Track
: See Link Empty to Track.

Orientation

Scene orientation tools can be used for orienting object to bundles.

Floor
: See Set Origin.

Wall
: See Set Floor.

Set Origin
: See Set Floor.

Set X, Y Axis

See Set X/Y Axis.

Set Scale
See Set Scale.

Apply Scale
See Apply Solution Scale.

Distance
Distance in active scene units which is used by Set/Apply scale.

Scene Setup

Set as Background
See Set as Background.

Setup Tracking Scene
See Setup Tracking Scene.

Selecting

All `A`
: Selects all items.

None `Alt-A`
: Deselects all items.

Inverse `Ctrl-I`
: Selects non-selected items and deselects existing selection.

Box Select

Reference

Mode::	All modes
Menu::	*Select ▸ Box Select*
Shortcut::	`B`

See Select Box.

Circle Select

Reference

Mode::	All modes
Menu::	*Select ▸ Circle Select*
Shortcut::	`C`

See Select Circle.

Lasso Select

Reference

Mode::	All modes
Menu::	*Select ▸ Lasso Select*
Shortcut::	`Ctrl-Alt-RMB`

See Select Lasso.

Select Grouped

Reference

Mode::	All modes
Menu::	*Select ▸ Select Grouped*
Shortcut::	`Shift-G`

Select all tracks from specified group.

Action
: The group of tracks to select.

Keyframed Tracks::
: Select all keyframed tracks.

Estimated Tracks:: Select all estimated tracks.

Tracked Tracks:: Select all tracked tracks.

Locked Tracks:: Select all locked tracks.

Disabled Tracks:: Select all disabled tracks.

Track with Same Color::
: Select all tracks with same color as active track.

Failed Tracks:: Select all tracks which failed to be reconstructed.

Select Stabilization Tracks

Reference

Mode:: Tracking mode
Menu:: *Select ▸ Select Stabilization Tracks*

Select tracks which are used for translation stabilization.

Select Stabilization Rotation Tracks

Reference

Mode:: Tracking mode
Menu:: *Select ▸ Select Stabilization Rotation Tracks*

Select tracks which are used for rotation stabilization.

Editing

- Clip
  - Open Clip
  - Set Scene Frames
  - Set Principal to Center
  - Prefetch
  - Reload Clip
  - Proxy
  - Set as Background
  - Setup Tracking Scene
- Track
  - Transform
  - Track Motion
  - Clear
  - Refine
  - Add Marker
  - Detect Features
  - Create Plane Track
  - Solve Solution
  - Join Tracks
  - Average Tracks
  - Copy Tracks
  - Paste Tracks
  - Animation
  - Show/Hide
  - Clean Up
  - Delete Track
  - Delete Marker
- Reconstruction
  - Set Origin
  - Set Floor
  - Set Wall
  - Set X/Y Axis
  - Set Scale

- Apply Solution Scale
- Link Empty to Track
- 3D Markers to Mesh

Clip

Open Clip

Reference

Mode::	All modes
Menu::	*Clip* ▸ *Open Clip*
Shortcut::	`Alt-O`

Todo.

Set Scene Frames

Reference

Mode::	Tracking
Menu::	*Clip* ▸ *Set Scene Frames*

Sets end scene frame to match current clip duration.

Set Principal to Center

Reference

Mode::	Tracking
Menu::	*Clip* ▸ *Set Principal to Center*

Changes the Optical Center values to the center of image.

Prefetch

Reference

Mode::	All modes
Menu::	*Clip ‣ Prefetch*
Shortcut::	P

Fills cache with frames. As many frames as fits into cache are load form the drive. This allows to fill in the cache as fast as possible when you really need to track something, but this keeps CPU and drive bandwidth idle if you've got a Clip editor opened but not actually interacting with it.

Reload Clip

Reference

Mode::	All modes
Menu::	*Clip ‣ Open Clip*

Force reload the currently loaded movie clip. Is mainly useful when the clip gets edited outside of Blender.

Proxy

Todo.

Set as Background

Reference

Mode::	Tracking
Menu::	*Clip ‣ Set as Background*

Sets the clip currently being edited as the camera background for all visible 3D Viewports. If there is no visible 3D Viewports or the Clip Editor is open

in full screen, nothing will happen.

Setup Tracking Scene

Reference

Mode::	Tracking
Menu::	*Clip ▸ Setup Tracking Scene*

Performs all usual steps to set up a VFX scene:

- Create reference objects for floor and test object.
- Create node set up for combining CG with an actual clip.

Track

Transform

Todo.

Track Motion

The Track Motion menu is used to perform tracking of selected tracks (i.e. following the selected feature from frame to frame).

This operator depends on settings from the Tracking Settings panel. If during sequence tracking the algorithm fails to track some markers, they will be disabled and tracking will continue for the rest of the markers. If the algorithm fails when tracking frame-by-frame, the marker is not disabled, and the most likely position of the feature on the next frame is used.

Backwards

Reference

Mode::	Tracking
Menu::	*Track ▸ Track Motion ▸ Backwards*
Shortcut::	`Shift-Ctrl-T`

Tracks the motion backward along the sequence.

Frame Backwards

Reference

Mode::	Tracking
Menu::	*Track ▸ Track Motion ▸ Frame Backwards*

Shortcut:: `Alt-Left`

Tracks the motion backward by one frame.

Forwards

Reference

Mode:: Tracking
Menu:: *Track ▸ Track Motion ▸ Forwards*
Shortcut:: `Ctrl-T`

Tracks the motion forward along the whole sequence.

Frame Forwards

Reference

Mode:: Tracking
Menu:: *Track ▸ Track Motion ▸ Frame Forwards*
Shortcut:: `Alt-Right`

Tracks the motion forward one frame.

Clear

Before

Reference

Mode:: Tracking
Menu:: *Track ▸ Clear ▸ Before*
Shortcut:: `Shift-T`

Deletes all tracked and keyframed markers after the current frame for all selected tracks.

Clear Active
: Limits clear action to only active track (as opposite to all selected ones).

After

Reference

Mode::	Tracking
Menu::	*Track ▸ Clear ▸ After*
Shortcut::	`Alt-T`

Deletes all tracked and keyframed markers before the current frame for all selected tracks.

Clear Active
: Limits clear action to only active track (as opposite to all selected ones).

Track Path

Reference

Mode::	Tracking
Menu::	*Track ▸ Clear ▸ Track Path*
Shortcut::	`Shift-Alt-T`

Clears all markers except the current one from all selected tracks.

Clear Active
: Limits clear action to only active track (as opposite to all selected ones).

Clear Solution

Todo.

Refine

This operator will run a tracker from previous keyframe to current frame for all selected markers. Current markers positions are considering initial position guess which could be updated by a tracker for better match.

Useful in cases when feature disappears from the frame and then appears again. Usage in this case is the following:

- When feature point re-appeared on frame, manually place marker on it.
- Use Refine Markers operation to allow tracker to find a better match.

Depending on direction of tracking use either *Forwards* or *Backwards* refining. Accordingly if tracking happens forwards, use *Refine Forwards*, otherwise use *Refine Backwards*.

Backwards

Reference

Mode:: Tracking
Menu:: *Track ▸ Refine ▸ Backwards*

Refine the track backwards.

Forwards

Reference

Mode:: Tracking
Menu:: *Track ▸ Refine ▸ Forwards*

Refine the track forwards.

Add Marker

Reference

Mode:: Tracking
Menu:: *Track ▸ Add Marker*

Places a new marker at the position of the mouse (which is under the button in this case, not ideal but it is just how things work) and then it can be moved to the needed location. When it is moved to the desired position, `LMB` can be used to finish placing the new marker. Also, `Return` and `Spacebar` can be used to finish placing the marker. But it is faster to use `Ctrl-LMB` to place markers directly on the footage. This shortcut will place the marker in the place you have clicked.

In addition to this until you have released the mouse button, you can adjust the marker position by moving the mouse and using the track preview widget to control how accurately the marker is placed.

Detect Features

Reference

Mode:: Tracking
Menu:: *Track ▸ Detect Features*

Detects all possible features on the current frame and places markers at these features. This operator does not take other frames into account, so it might place markers on features which belong to moving objects. If the camera is turning away from this shot, no markers could be present within the frames after the camera moved away.

There are several properties for this operator:

Placement
: Controls where to place markers.

  Whole Frame

Places markers throughout the whole frame.

Inside Annotated Area
: Places markers inside the area outlined with the Annotation Tools. This can be used to outline some areas with interesting features and place markers only inside the outlined area.

Outside Annotated Area
: Places markers outside the area outlined with the Annotation Tools. This can be used to outline areas of no interest (like trees, humans, etc.) and place markers outside of these areas.

Margin
: Controls the distance from the image boundary for created markers. If markers are placed too close to the image boundary, they will fail to track really quickly and they should be deleted manually. To reduce the amount of manual clean-up, this parameter can be used.

Threshold
: Limits minimal threshold for placing markers. This value comes from the feature detection algorithm and it means: low values means most probably this feature would fail to track very soon, high value means it is not much such track. Amount of markers to be added can be controlled with this value.

Distance
: Defines the minimal distance between placed markers. It is needed to prevent markers from being placed too close to each other (such placement can confuse the camera solver).

Create Plane Track

Reference

Mode::	Tracking
Menu::	*Track ▸ Create Plane Track*

The *Create Plane Track* operator creates a new plane track. Planar tracking takes advantage of the fact that there are often planar surfaces in footage, by attaching markers to points on these flat planes. It can be used to replace things like billboards and screens on the footage with another image or video. It also might be used for masking.

This button will create a plane object which is deforming in the same way as plane defined by all selected point tracks. At least four feature points tracked across the footage which belongs to the plane you want to replace are needed. More tracks will give better estimation of plane motion.

Feature points used to estimate plane motion could be used from any place on the plane, meaning it's not necessarily need to be corners. Corners are not always easy to be tracked, they might be occluded. In this case you can position tracked features that lay on the same plane far away from the actual plane which should be replaced.

This provides more information about the possible deformation of the marker in following frames, and such markers can be tracked even if partially occluded (appear and disappear during the time). It is only required that two neighbor frames have at least four common tracks.

An image can be projected onto the plane with the Plane Track Deform Node compositing node.

Solve Solution

Reference

Mode::	Tracking
Menu::	*Track* ▸ *Solve Solution*

The *Camera Motion* operator solves the motion of camera using all tracks placed on the footage and two keyframes specified on this panel. There are some requirements:

- There should be at least eight common tracks on the both of the selected keyframes.
- There should be noticeable parallax effects between these two keyframes.

If everything goes smoothly during the solve, the average reprojection error is reported to the information space and to the Clip editor header. Reprojection error means the average distance between reconstructed 3D position of tracks projected back to footage and original position of tracks. Basically, reprojection error below 0.3 means accurate reprojection, (0.3 - 3.0) means quite nice solving which still can be used. Values above 3 means some tracks should be tracked more accurately, or that values for focal length or distortion coefficients were set incorrectly.

Join Tracks

Reference

Mode::	Tracking
Menu::	*Track ▸ Join Tracks*
Shortcut::	`Ctrl-J`

This operator joins all selected tracks into one. Selected tracks should not have common tracked or keyframed markers at the same frame.

Average Tracks

Reference

Mode::	Tracking
Menu::	*Track ▸ Average Tracks*

The Average Tracks operator creates a new tracking marker by averaging the data from the selected tracks. This can be used to improve stability of tracking on blurry or non-very-sharp feature shapes. The operator takes into

account all Marker properties however, disabled markers do not affect the averaging.

Gaps in the original tracks will be linearly interpolated, to reduce result track jump. Note that this only applies to gaps “in between”. This means that if a track does not have markers in the beginning or end of it, there is nothing to interpolate with and the resulting track will jump.

Keep Original
: When enabled, the selected tracks are *not* deleted.

Copy Tracks

Todo.

Paste Tracks

Todo.

Animation

Todo.

Show/Hide

Todo.

Clean Up

Clean Tracks

Reference

Mode:: Tracking

Menu:: *Track ▸ Clean Up ▸ Clean Tracks*

Identifies all tracks which matches settings from above and performs desired action on them.

Tracked Frames
: Tracks or tracked segments shorter than this number of frames will be removed.

Reprojection Error
: Tracks which has reprojection error higher than this value will be removed.

Action
: Several actions can be performed for bad tracks.

  Select
  : They can simply be selected.

  Delete Track
  : The whole track can be deleted.

  Delete Segments
  : Bad segments of tracked sequence can be removed.

Filter Tracks

Reference

Mode:: Tracking
Menu:: *Track ▸ Clean Up ▸ Filter Tracks*

This operator deletes obviously bad tracks (for example, the ones which are too short). Additionally, it identifies tracks which has suspicious spikes in their motion and selects them.

Delete Track

Reference

Mode::	Tracking
Menu::	*Track ▸ Delete Track*
Shortcut::	`X`

Delete all selected tracks.

Delete Marker

Reference

Mode::	Tracking
Menu::	*Track ▸ Delete Marker*
Shortcut::	`Shift-X`

Todo.

Reconstruction

Scene orientation tools can be used for orienting object to bundles.

Set Origin

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Set Origin*

Transform camera in a way which makes active track to be moved to a scene origin. Only translation is applied to the camera.

Set Floor

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Set Floor*

Use selected three markers to define a floor. Camera will be transformed in a way which makes the selected markers to be flat (have Z = 0).

Set Wall

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Set Wall*

Similar to the floor orientation, but defines a wall (selected tracks are placed onto the XZ plane).

Set X/Y Axis

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Set X/Y Axis*

Transform camera in a way which makes active track to become on X or Y axis. No translation is applied, meaning scene origin which was specified before will be preserved.

Set Scale

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Set Scale*

Scale camera or tracking object in a way which makes distance between two selected tracks match the given value in *Distance*.

Apply Solution Scale

Reference

Mode:: Tracking
Menu:: *Reconstruction ▸ Apply Solution Scale*

Similar to Set Scale, but actually modifies the tracking data.

Link Empty to Track

Reference

Mode::	Tracking
Menu::	*Reconstruction ‣ Link Empty to Track*

Creates new empty in 3D Viewport and appends constraint which parts it to the active track.

3D Markers to Mesh

Reference

Mode::	Tracking
Menu::	*Reconstruction ‣ 3D Markers to Mesh*

Creates a mesh which vertices matches positions of reconstructed tracks. It is required to have motion solved first before using this operator. Only tracks from the current tracking object will be used. The intention of this operator is to give a nice starting point for a manual mesh reconstruction.

Sidebar

- Track
  - Track
  - Objects Panel
  - Plane Track
  - Camera
  - Marker
- 2D Stabilization
  - Introduction
  - 2D Stabilization Panel
  - Workflow
- View
  - Annotations

Track

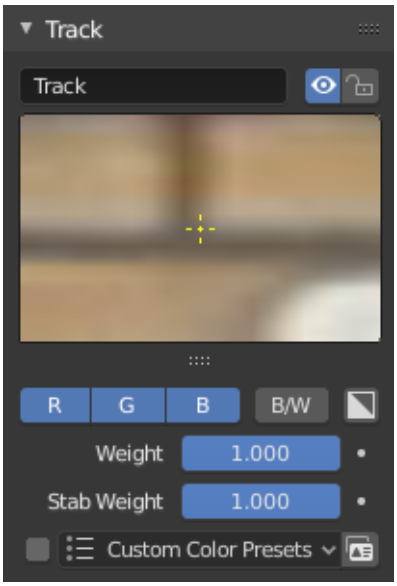
- Track
  - Track Preview Widget
  - Further Options
- Objects Panel
- Plane Track
- Camera
  - Lens
- Marker

Track

Name
: The track name can be changed with this field. Track names are used for linking tracking data to other areas, like a Follow Track constraint.

Enable (eye icon)
: This toggle controls if the marker is enabled. If a marker is disabled, its position is not used either by solver nor by constraints.

Lock (padlock icon)
: The toggle controls whether the track is locked. Locked tracks cannot be edited at all. This helps to prevent accidental changes to tracks which are “finished” (tracked accurate along the whole footage).



Track panel.

Track Preview Widget

The widget in this panel is called “Track Preview” and it displays the content of the pattern area. This helps to check how accurately the feature is being tracked (controlling that there is no sliding off original position) and also helps to move the track back to the correct position. The track can be moved directly using this widget by mouse dragging.

If an anchor is used (the position in the image which is tracking is different from the position which is used for parenting), a preview widget will display the area around the anchor position. This configuration helps in masking some things when there is no good feature at position where the mask corner should be placed. Details of this technique will be written later.

There is small area below the preview widget which can be used to enlarge the vertical size of preview widget (the area is highlighted with two horizontal lines).

Further Options

R, G, B
: Tracking happens in gray-scale space, so a high contrast between the feature and its background yields more accurate tracking. In such cases disabling some color channels can help.

Grayscale Preview (B/W)
: Display the preview image as gray-scale even if all channels are enabled.

Mask Preview (black/white icon)
: Applies mask defined by an annotation tool in the preview widget.

Weight
: When several tracks are used for 3D camera reconstruction, it is possible to assign a reduced weight to some tracks to control their influence on the solution result. This parameter can (and often need to) be animated.

  Altering the weights of problem tracking markers can correct or greatly reduce undesirable jumps as feature disappear or become difficult to track.

  Another use of Track Weights is when you want to reconstruct a scene from your camera solution. In that case you can first carefully track and solve your scene, and once you are done, lock all your markers with `Ctrl-L`, set the tracker weight in the Extra Settings of the tracker settings to zero and use the feature detection to quickly add lots of markers. Now track them and solve the scene again. Since their weight is zero they will not influence your solution at all, but you will have lots of good reference points in your scene.

Stabilization Weight

While *Weight* parameter is used for 3D reconstruction, the *Stabilization Weight* is used to control 2D stabilization.

Custom Color Presets

The preset for the *Custom Color*.

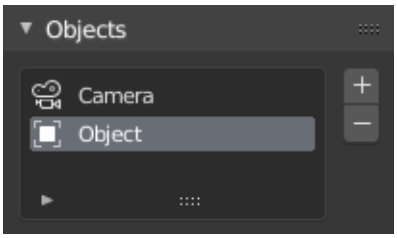
Custom Color

This setting overrides the default marker color used in the Clip editor and 3D Viewport, and it helps to distinguish different type of features (for example, features in the background vs. foreground and so on). Color also can be used for "grouping" tracks so a whole group of tracks can be selected by color using the Select Grouped operator.

Tip

To select good points for tracking, use points in the middle of the footage timeline and track backwards and forwards from there. This will provide a greater chance of the marker and point staying in the camera shot.

Objects Panel



Objects panel.

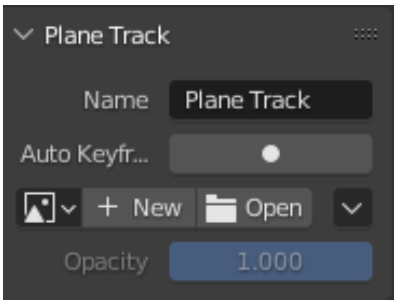
This panel contains a list view with all objects which can be used for tracking, camera or object solving. By default there is only one object in this list which is used for camera solving. It cannot be deleted and other objects cannot be used for camera solving; all added objects are used for object tracking and solving only. These objects can be referenced from Follow Track and Object Solver constraints. Follow Track uses the camera object by default.

If some tracks were added and tracked to the wrong object, they can be copied to another object using *Track ▸ Copy Tracks* and *Track ▸ Paste Tracks*.

The usage for all kind of objects (used for camera and object tracking) is the same: track features, set camera data, solve motion. Camera data is sharing between all objects and refining of camera intrinsics happens when solving camera motion only.

Plane Track

Its properties are shown only when a plane track is selected.



Plane Track panel.

Name
: The name of the selected plane track is shown. It can also be changed from here.

Auto Keyframe
: Toggles the auto-keyframing for corners of the plane track. With this enabled, keyframes will automatically get inserted when any corner is moved.

Image
: Field to select or create an image which will be displayed inside the plane track. This image is for preview purposes in the Movie Clip editor only. To include it in your final render, see Plane Track Deform node.

New Image from Plane Track
: Creates an image from the pixels of the movieclip that the plane marker "sees" at the current frame. This allows to create an unwarped texture of any flat surface in the footage. The resulting image can then be used for editing and retouching, for example to paint out certain parts of the footage.

Update Image from Plane Track
: Updates the pixels of the active Plane Track's image.

Opacity
: Used to set the opacity of this image. Again, this is for display purposes only, and will not affect your final render.

Camera

This panel contains all settings of the camera used for filming the movie which is currently being edited in the Clip editor. Different predefined settings can be used here and can be chosen from the panel header. But such settings as distortion coefficients and principal point are not included in the presets and should be filled in even if camera presets are used.

Sensor Width
: Is the width of the CCD sensor in the camera. This value can be found in camera specifications.

Pixel Aspect
: Is the pixel aspect of the CCD sensor. This value can be found in camera specifications, but can also be guessed. For example, you know that the footage should be 1920×1080, but the images themselves are 1280×1080. In this case, the pixel aspect is: 1920 / 1280 = 1.5.

Lens

Focal Length
: Is self-explanatory; it is the focal length with which the movie was shot. It can be set in millimeters or pixels.

Optical Center
: Is the optical center of the lens used in the camera. In most cases it is equal to the image center, but it can be different in some special cases. Check camera/lens specifications in such cases.

Tip

Optical Center also know as the principal point in photogrammetry.

Set Center

See Set Principal to Center.

Lens Distortion

Mathematical function to convert distorted to undistorted coordinates.

Polynomial:: Polynomial radial distortion. Uses three distortion coefficients: K1, K2, and K3.

Division:: It defines high distortions, which makes this model suitable much better for cameras with fisheye lenses. Use two distortion coefficients: K1, K2.

Nuke:: Distortion model used by the Nuke compositor. Use two distortion coefficients K1, K2.

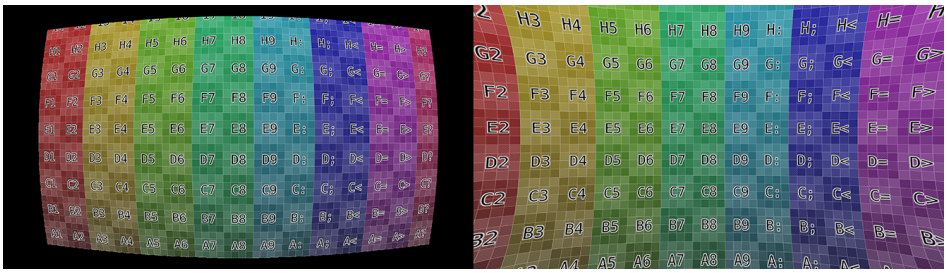
Brown:: Brown-Conrady is one of most advanced mathematical lens distortion models. Used to model both radial and tangential distortion. Can use up to four radial distortion coefficients: K1 - K4 and up to two tangential distortion coefficients: P1 and P2.

Coefficients

Coefficients are used to compensate for lens distortion when the movie was shot. Currently these values can be tweaked by hand only (there are no calibration tools yet) using tools available in Distortion mode. To do this tweak K1 until the solving is the closest to the known focal length (but also take grid and annotations into account to prevent “impossible” distortion).

Radial Distortion Coefficients (K1 - K4)

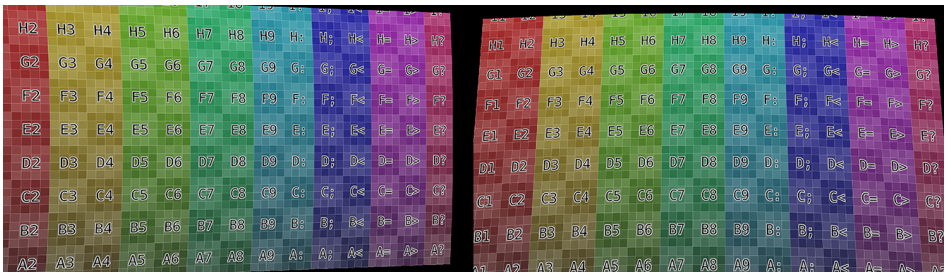
The coefficients in lens distortion models work independent from each other. Positive values will give a barrel distortion while negative values give a pincushion distortion. With a mixture of both negative and positive coefficients you can define more complicated mustache distortions or other complex distortions, that are less common but not rare.



Example of radial distortion for positive and negative K coefficients.

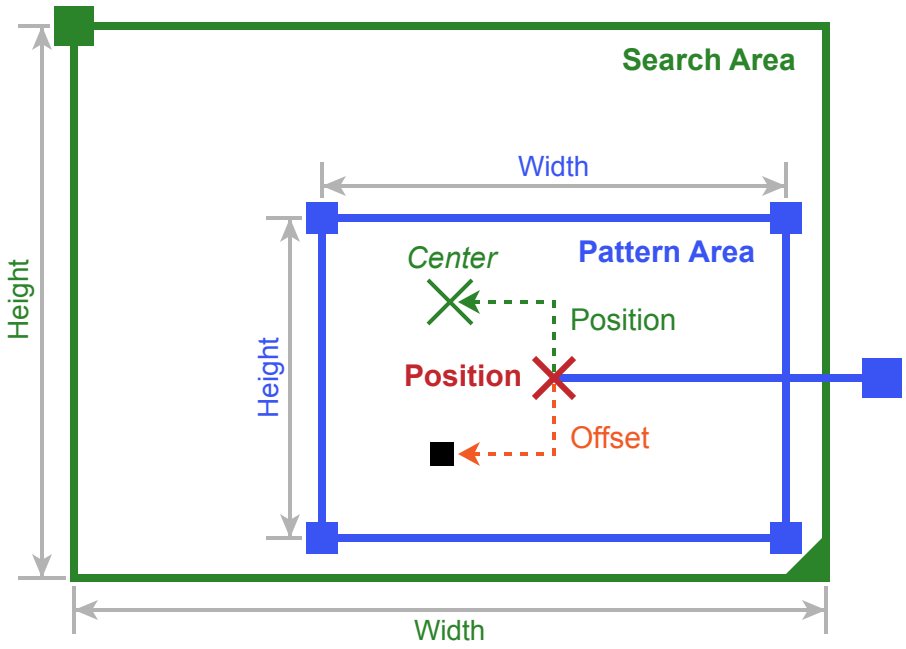
Tangential Distortion Coefficients (P1, P2)

Works independent and allow to compensate for situations when the sensor is not perpendicular to a group of lens. The optical center (also called principal point) will be shifted (distorted) from the center of the sensor. P1 is used to compensate for sensor rotation in Z (vertical) axes, while P2 is for compensating sensor rotation in X (horizontal) axes. Such distortions can be found in sources from cameras with a sensor stabilization system.



Example of tangential distortion for P coefficients.

Marker



Marker schematic.

This panel contains numerical settings for marker position, pattern and search area dimensions, and offset of anchor point from pattern center.

Enabled
: Toggles the markers affect on the current frame.

Position X, Y
: The X/Y positions of the marker at frame in screen coordinates.

Offset X, Y
: The X/Y offset to parenting point.

Pattern Area Width, Height
: The width/height of a marker's pattern in screen coordinates.

Search Area X, Y

The X/Y position of search at frame relative to the marker's position.

Search Area Width, Height

The width/height of the markers search in screen coordinates.

2D Stabilization

- Introduction
  - How It Works
- 2D Stabilization Panel
  - Options
- Workflow
  - The Simple Case
  - Avoid Problematic Footage
  - Elaborate Movements
  - Animating Stabilization Parameters
  - Irregular Track Setup

Introduction

The 2D video stabilization is a feature built on top of Blender's image feature tracking abilities: You can use some tracking points to remove shakiness, bumps and jerks from video footage. Typically, image stabilization is part of a 2D workflow to prepare and improve footage prior to further processing or modeling steps. This page helps to understand how it works, introduces related terms and concepts, describes the available interface controls in detail and finally gives some hints about usage in practice.

Typical usage scenarios of the stabilizer:

- Fix minor deficiencies (shaky tripod, jerk in camera movement).
- "Poor man's steadycam" (when a real steadycam was not available, affordable or applicable).
- As a preparation for masking, matching and rotoscoping.

It is not uncommon for 2D stabilization to have to deal with somewhat imperfect and flawed footage.

How It Works

To detect spurious movement in a given shot, we'll assume a simplified model about this movement. We then try to fit the movement of tracked features with this simplified model to derive a compensation. Of course, this works only to the degree our model is adequate – yet in practice, this simplified approach works surprisingly well even with rather complicated shots, where our basic assumption was just an approximation of much more elaborate movements.

This simplified model underlying the 2D stabilization as implemented here assumes movement by an affine-linear transform:

- The camera is pushed up/down/sideways by some translation component.
- The image is then tilted and scaled around a pivot point (rotation center).

To compensate movement according to this simplified model, the 2D stabilizer proceeds in two steps. First we try to detect the translation offset from the weighted average of all *translation* tracking points. After compensating this translation component, we then use additional *rotation/scale* tracking points to detect rotation around a given pivot point. Again, we detect rotation and scale changes through a weighted average of all the rotation/scale tracking points given.

In the current version, the pivot point is anchored to the weight center of the translation tracking points. So effectively the detected translation is already factored out. In some cases this is not optimal, especially when tracks have gaps or do not cover the whole duration of the footage – we plan further options to better control the pivot point in future releases.

Stabilization Tracks

Thus, as foundation for any image stabilization, we need tracked image features to derive the movements. These tracking points or "tracks" can be established with Blender's image feature tracking component The right choice of points to track is somewhat tricky, yet crucial for successful image stabilization. Often, we're here because we'll have to deal with imperfect footage. In such cases, the averaging of tracks helps to work around image or tracking errors at some point. Moreover, when the footage contains perspective induced movements, symmetrically placed tracking points above and below the horizon can be used to cancel out spurious movement and get stabilization to the focal area in between.



Diverging movements caused by perspective.

Tracks can be added in two groups:

1. First of all is the list of tracks to be used to compensate for jumps in the camera location. From all the tracking points added to this group, we calculate a weighted average. We then try to keep this average location constant during the whole shot. Thus it is a good idea to use tracking markers close to and centered around the most important subject.
2. A second selection of tracks is used to keep the rotation and scale of the image constant. You may use the same tracks for both selections. But usually it is best to use tracking points with large distance from the image center, and symmetrically, on both sides, to capture the angular movements more precisely. Similar to the “location” case, we calculate an average angular contribution and then try to keep this value constant during the whole shot.

Footage, Image & Canvas

When talking about the movement stabilization of video, we have to distinguish several frames of reference. The image elements featured by the footage move around irregularly within the footage’s original image boundaries – this is the very reason why we are using the stabilizer. When our attempt at stabilization was successful, the image elements can be considered stable now, while in exchange the footage’s image boundaries have taken on irregular movement and jump around in the opposite way. This is the immediate consequence of the stabilizer’s activity.

Since the actual image elements, i.e. the subject of our footage can be considered stable now, we may use these as a new frame of reference: we consider them attached to a fixed backdrop, which we call the canvas. Introducing this concept of a "canvas" helps to deal with deliberate movements of the camera. And beyond that, it yields an additional benefit: It is very frequent for the pixels of video footage to be non-square. So we have to stretch and expand those pixels, before we're able to perform any sensible rotation stabilization. Thus the canvas becomes, by definition, the reference for an undistorted display of the image contents.

But when the camera was moved intentionally, we have to consider yet another frame of reference beyond the canvas: namely the frame (or "cadre") of the final image we want to create. To understand this distinction, let's consider a hand-held, panning shot to the right: Since our camera was turned towards the right side, the actual image contents move towards the left side *within* the original image frame. But let's assume the stabilizer was successful with "fixing" any image contents relative to the canvas – which in turn means, that the original image boundaries start to move irregularly towards the right side, and the contents of the image will begin to disappear gradually behind the left boundary of the original image. After some amount of panning, we'll have lost all of our original contents and just see an empty black image backdrop. The only solution to deal with that problem is to move the final image frame along to the right, thus following the originally intended panning movement. Of course, this time, we do want to perform this newly added panning movement in a smooth and clean way.



Stabilizing a panning shot.

To allow for such a compensation and to reintroduce deliberate panning, or tilting and zoom of the resulting image, the stabilizer offers a dedicated set of controls: *Expected position*, *Expected rotation* and *Expected scale*. These act like the controls of a virtual camera filming the contents we have fixed onto the canvas. By animating those parameters, we're able to perform all kinds of deliberate camera movements in a smooth fashion.



Restoring the expected camera movement.

The “Dancing” Black Borders

As explained above, when we succeed with stabilizing the image contents, the boundaries of the original footage start to jump around in the opposite direction of the movements compensated. This is inevitable – yet very annoying, since due to the irregular nature of these movements, these “dancing black borders” tend to distract attention from the actual subject and introduce an annoying restlessness. Thus our goal must be to hide those dancing borders as good as possible. A simple solution is to add a small amount of zoom. Sometimes we’ll also need to animate the parameter *Expected* position in order to keep the image centered as good as we can – this helps to reduce the amount of zoom necessary to remove those annoying borders.

The *Autoscale* function can be used to find the minimal amount of zoom just sufficient to remove those black borders completely. However, if the camera jumps a lot, the autoscale function often zooms in too much, especially since this calculation aims at finding a single, static zoom factor

for the whole duration of the footage. When this happens, you'll typically get overall better results with animating both the zoom factor and the expected position manually.

2D Stabilization Panel

The purpose of 2D stabilization is to smooth out jerky camera handling on existing real-world footage. To activate the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Show Stable* in the Clip Display pop-over. Then you'll need to set up some tracking points to detect the image movements.

The 2D Stabilization panel is used to define the data used for 2D stabilization of the shot. Several options are available in this panel: you may add a list of tracks to determine lateral image shifts and another list of tracks to determine tilting and zooming movements. Based on the average contribution of these tracks, a compensating movement is calculated and applied to each frame.

When the footage includes panning and traveling movements, the stabilizer tends to push the image out of the visible area. This can be compensated by animating the parameters for the intentional, "expected" camera movement.

Note

To *activate* the 2D stabilizer, you need to set the toggle in the panel, and additionally you need to enable *Show Stable* in the *Clip Display* pop-over.

Options

Anchor Frame

Reference point to anchor stabilization: other frames will be adjusted relative to this frame's position, orientation and scale. You might want to select a frame number where your main subject is featured in an optimal way.

Stabilization Type

Rotation
: In addition to location, stabilizes detected rotation around the *rotation pivot point*, which is the weighted average of all location tracking points.

Scale
: Compensates any scale changes relative to center of rotation.

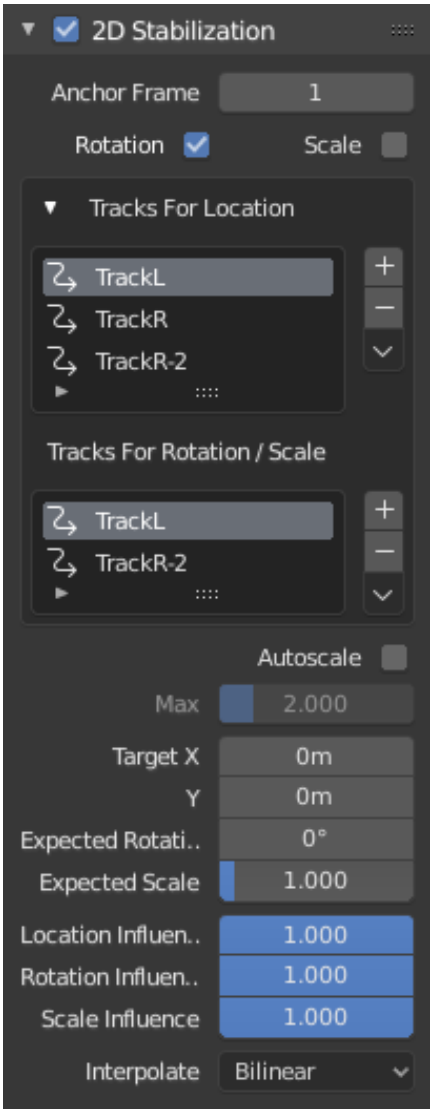
Tracks for Stabilization

Location
: List of tracks to be used to compensate for camera jumps, or location movement.

Rotation/Scale
: List of tracks to be used to compensate for camera tilts and scale changes.

Autoscale
: Finds smallest scale factor which, when applied to the footage, would eliminate all empty black borders near the image boundaries.

Max
: Limits the amount of automatic scaling.



2D Stabilization panel.

Expected Position X/Y
: Known relative offset of original shot, will be subtracted, e.g. for panning shots.

Expected Rotation
: Rotation present on original shot, will be compensated, e.g. for deliberate tilting.

Expected Zoom

Explicitly scale resulting frame to compensate zoom of original shot.

Influence

The amount of transformation applied to the footage can be controlled. In some cases it is not necessary to fully compensate camera jumps. The amount of stabilization applied to the footage can be controlled. In some cases you may not want to fully compensate some of the camera's jumps. Please note that these "* *Influence*" parameters do control only the *compensation movements* calculated by the stabilizer, not the deliberate movements added through the "*Expected* *"-parameters.

Interpolate

The stabilizer calculates compensation movements with sub-pixel accuracy. Consequently, a resulting image pixel needs to be derived from several adjacent source footage pixels. Unfortunately, any interpolation causes some minor degree of softening and loss of image quality.

Nearest

No interpolation, uses nearest neighboring pixel. This setting basically retains the original image's sharpness. The downside is we also retain residual movement below the size of one pixel, and compensation movements are done in 1 pixel steps, which might be noticeable as irregular jumps.

Bilinear

Simple linear interpolation between adjacent pixels.

Bicubic

Highest quality interpolation, most expensive to calculate.

Workflow

Depending on the original footage's properties, achieving good stabilization results might be simple and easy, or it might require some work, dedication and careful planning. This section covers some practical considerations to help improving the results.

The Simple Case

Whenever the camera is basically fixed, or at least "almost" stationary, and the footage is crisp and without motion blur, perfect stabilization is easy to achieve. This might be the case when a tripod was used, but wind or vibrations on the floor (e.g. on a stage) caused some minor shakes. Shoulder camera shots done by an experienced operator also frequently fall into this category.

- Use as few points as possible. Start with a single point right on the main subject.
- Track this single point as accurate as possible. Beware of movements and shape changes of the tracked feature. Proceed in small increments (e.g. 50 frames), zoom in and readjust the target point manually when it drifts away. Another option is to use a larger target area for tracking; since we're tracking only a single point, the slower tracking speed might be acceptable.
- After enabling the basic (location) stabilization, consider if you really need rotation stabilization. Often, some minor, slow swinging movements are not really noticeable and do not warrant the additional working time and quality loss caused by rotation and scale stabilization.
- For rotation, start with one extra point, well spaced but preferably still attached to the main subject.
- Consider to fix some slow residual motion by manually animating the "*Expected* *" parameters, before you even think of adding more tracking markers. Because doing so is often not worth the effort.

- If you need to add more points, the most important goal is to achieve *symmetry*. Place location tracking points symmetrically above and below the horizon. Place rotation tracking points into diagonally opposed direction, always centered around the main focal area.

Avoid Problematic Footage

The 2D stabilizer can not work miracles; some flaws simply can not be fixed satisfactory. Notorious issues are motion blur, rolling shutter, pumping autofocus and moving compression artifacts. Especially if you do succeed with basic stabilization, such image flaws become yet the more noticeable and annoying. When on set or on location, it might be tempting to "fix matters in post production". Resist that deception, it rarely works out well.

- Prefer a short exposure time to avoid motion blur. While motion blur is good to render filmed movements more smooth and natural, it seriously impedes the ability to track features precisely. As a guideline, try to get at least to 1/250 s.
- Prefer higher frame rates. The more *temporal resolution* the stabilizer has to work on, the better the results. If you have the option to choose between progressive and interlaced modes, by all means use interlaced and deinterlace the footage to the *doubled frame rate*. This can be done with the yadif filter of FFmpeg: use the mode 1 (`send_field`).
- Beware of the Rolling Shutter effect. Avoid fast lateral movements. If you can, prefer a camera which produces less rolling shutter. Also, using a higher frame rate reduces the amount of rolling shutter; another reason to prefer interlaced over progressive for the purpose at hand.
- Switch off autofocus. Better plan your movement beforehand, set a fixed focus and rely on depth of field through using a small aperture. Pumping movements might not be so noticeable to the human observer, but the feature tracking tends to slide away on defocused image elements; fixing this manually after the fact can cause a huge waste of time.
- Increase the lighting level, at least use a higher sensitivity. This helps to set a fast shutter speed plus a small aperture. Better lighting and good exposure also help to reduce the impact of compression artifacts. If you can, also select a codec with less data reduction, better color

space, etc. Inevitably, we're loosing some quality through the interpolation necessary for stabilization. Plus we're loosing some quality due to color space conversion.

Elaborate Movements

When the footage builds on elaborate intended movement of the camera, the process of stabilization becomes more involved – especially when there is a shift in the main area of interest within the shot. When working with many tracks and fine-grained animation, it is easy to get into a situation where additional manipulations actually decrease the quality, while it might be hard to spot and locate the root cause of problems. Recommendation is to proceed systematically, starting from the general outline down to tweaking of specific aspects.

1. Understand the nature of the movements in the shot, both the intended and the accidental.

2. Track some relevant features for location.

3. Establish the basic location stabilization. This includes the decision, which feature to use for what segment of the shot. Work with the track weights to get an overall consistent movement of the weight center, in accordance with the inherent focus of the shot.

4. Define the panning movements of the virtual camera (through animation of the *Expected Position* parameter).

5. Add tracking for rotation and zoom stabilization.

6. Fine-tuning pass:

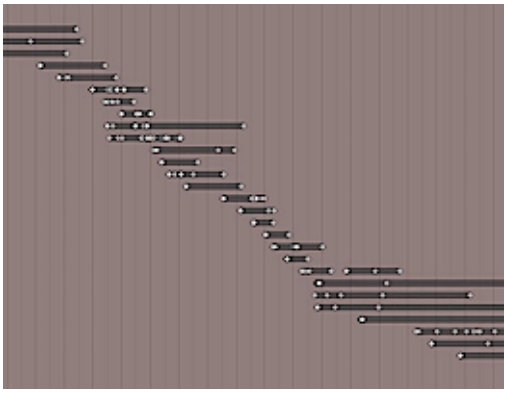
   Break down the whole duration of the shot into logical segments to define the intended camera movement. Then refine those segments incrementally step-by-step, until the overall result looks satisfactory…

Animating Stabilization Parameters

Animating some parameters over duration of the shot is often necessary, at least to get the final touch, including control of the scale factor to hide the dancing black borders. Unfortunately there is a **known limitation** in the current version: it is not possible to open the generic animation editors (Graph editor and Dope Sheet) for animation data beyond the 3D scene. So, while it *is possible* to set keyframes *right within the UI controls* of the stabilizer (either through pressing the `I` key or with the help of the context menu), it is not possible to manipulate the resulting curves graphically. The only way to readjust or remove a misguided keyframe is to locate the timeline to the very frame and then use the context menu of the animated UI control. (Hint: the color of the UI control changes when you have located at precisely the frame number of the keyframe.)

Irregular Track Setup

It might not be possible to track a given feature over the whole duration of the shot. The feature might be blurred or obscured; it might even move out of sight entirely, due to deliberate camera movement. In such a situation, we need *another tracked feature* to take on its role, and we need some *overlap time* to get a smooth transition without visible jump.



Irregular Tracks.

The stabilizer is able to deal with gaps and partial coverage within the given tracks. However, the basic assumption is that each track covers a single, fixed reference point whenever there is any usable/enabled data. Thus, you must not "reuse" a given track to follow several different points, rather you should disable and thus end one track, when tracking this feature is no longer feasible. You may include "gaps", when a tracking point is temporarily disabled or unavailable, but you should start a new track for each distinct new feature to be tracked.

Each track contributes to the overall result by the degree controlled through its *Stab Weight* parameter. It is evaluated on a per-frame basis, which enables us to control the influence of a track by *animating* this *Stab Weight*. You may imagine the overall working of the stabilizer as if each tracking point "drags" the image through a flexible spring: When you turn down the *Stab Weight* of a tracking point, you decrease the amount of "drag" it creates. Sometimes the contribution of different tracks has to work partially counter each other. This effect might be used to cancel out spurious movement, e.g. as caused by perspective. But when, in such a situation, one of the involved tracks suddenly goes away, a jump in image position or rotation might be the result. Thus, whenever we notice a jump at the very frame where some partially covered track starts or ends, we need to soften the transition. We do so by animating the *Stab Weight* gradually down, so that it reaches zero at the boundary point. In a similar vein, when we plan a "handover" between several partially covered tracks, we define a *cross-fade* over the duration where the tracks overlap, again by animating the *Stab Weight* parameters accordingly. But even with such cross-fade smoothing, some residual movement might remain, which then needs to be corrected with the *Expected Position* or *Expected rotation* parameters. It is crucial to avoid "overshooting" movements in such a situation – always strive at setting the animation keyframes onto precisely the same frame number for all the tracks and parameters involved.

View

Annotations

Annotation tool strokes can be enabled/disabled with the checkbox in the panel header. It is a standard annotations panel where annotation layers and frames can be controlled. There is one difference in the behavior of the annotation tools from other areas – when a new layer is created “on-demand” (when making a stroke without adding a layer before this) the default color for the layer is set to pink. This heightens the color contrast to make the stroke more noticeable on all kinds of movies.

Data Source

Determines the data-block type the current annotation layer is stored.

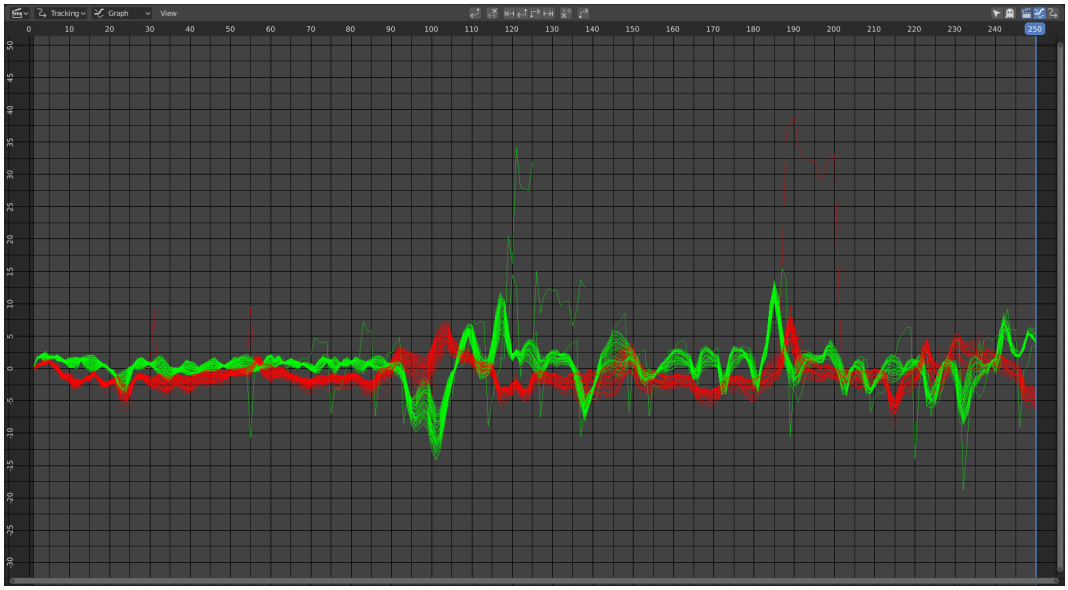
Clip:: Store the current annotation layer with the active *Movie Clip* data-block.

Track:: Store the current annotation layer with the active *Track* data-block.

See also

Annotation Tools for more information on general annotation layers.

Graph View



Graph View.

Introduction

The graph or curves view has numerous purposes based on the color of the lines. The red and green lines on the graph show you the speed of the trackers at a given frame. Green is vertical movement, Red is horizontal. Therefore the first frames will always be at zero.

The blue line is the line that comes out when you click on the film strip is the average per-frame error. This curve is available only after pressing camera solve and is not editable. This is the one line that you want to be as flat as possible and as closer to zero as you can. The high points will show you where in your shot you are having inaccurate tracking.

Frames outside of scene frame range are darkened.

Header

Show Selected (mouse cursor icon)
: Displays the graph for only selected trackers.

Display Hidden (ghost icon)
: Displays channels from objects that are hidden.

Filter
: Display options, defines what curves are visible.

  Frames
  : Visualizes per-frame average reprojection error of all tracks in the active tracking object.

  Motion
  : Shows curves for X and Y speed of tracks.

  Error
  : Per-frame reprojection error of tracks.

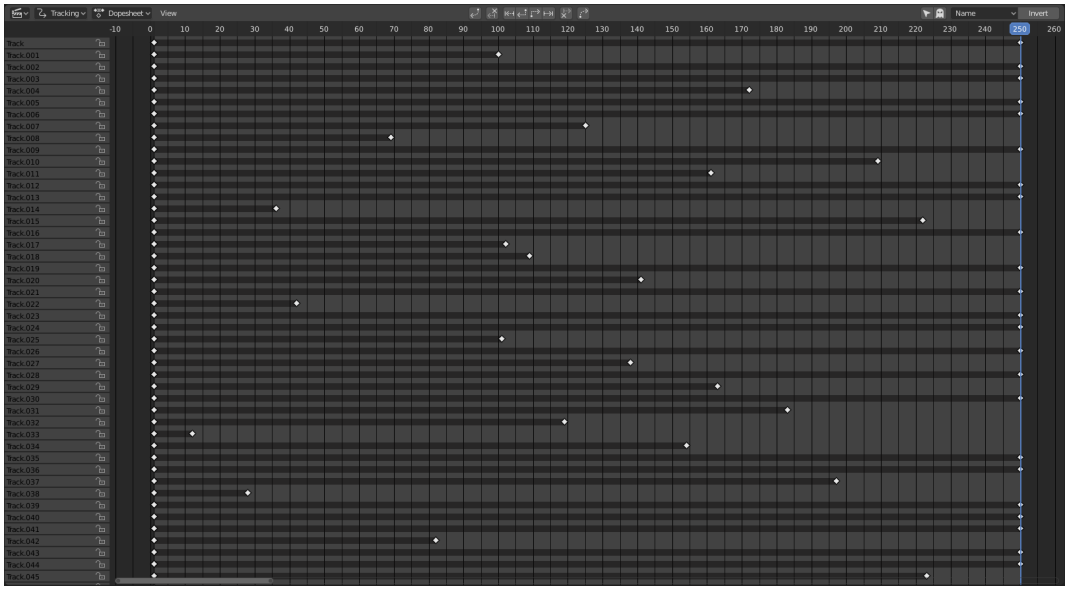
Usage

The curves are useful to see if particular trackers are moving differently than the average. A line that spikes from the rest of the curve usually means a tracking error.

You can manually edit the curve by selecting a point in the curve and dragging it or deleting, that will affect the corresponding tracker on that particular frame.

Lock to Selection L
: Locks the view to selected markers during playback.

Dope Sheet View



Dope Sheet View.

The Dope Sheet View is used to visualize motion tracking data, it is implemented as separate view of the Movie Clip editor just like the Graph View.

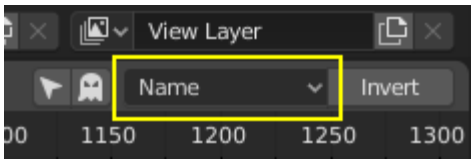
It displays channels for selected tracks and each channel visualizes tracked segments of tracks as dark bars and keyframed positions of tracks as small diamonds.

The background is highlighted depending on the number of tracks in a frame. This means that if for a frame (or sequence of frames) there are less than eight tracks, the background will turn red; if there are from eight to sixteen tracks, the background will be yellow.

This is only a visual feedback, which doesn't mean that the camera motion will not reconstruct with less than eight tracks. It only means that you should pay attention to those frames and check if all possible good feature points are tracked there. Remember, if there are no good feature points in

the frame and there are less than 16 tracks in the frame, it doesn't mean the solution won't be accurate. Rather, adding more tracks on bad feature points will reduce the accuracy of solution.

Header



Sort order of the channels.

Show Only Selected (mouse cursor icon)
: Limits Dope Sheet channels to only information about selected tracks.

Hidden (ghost icon)
: Includes information from hidden tracks.

Sort Method
: Sort order of the tracks.

Name::	Sort selected tracks in alphabetical order based on their names.
Longest::	Sort tracks by longest tracked segment length.
Total::	Sort tracks by overall amount of frames.
Average Error::	Sort tracks by their average reprojection error after solving camera or object motion.
Start Frame::	Sort channels by first frame number.
End Frame::	Sort channels by last frame number.

Invert
: To change the sort order from ascending to descending.

Usage

The Dope Sheet View is for visualization and does not have any tools to actually edit data.

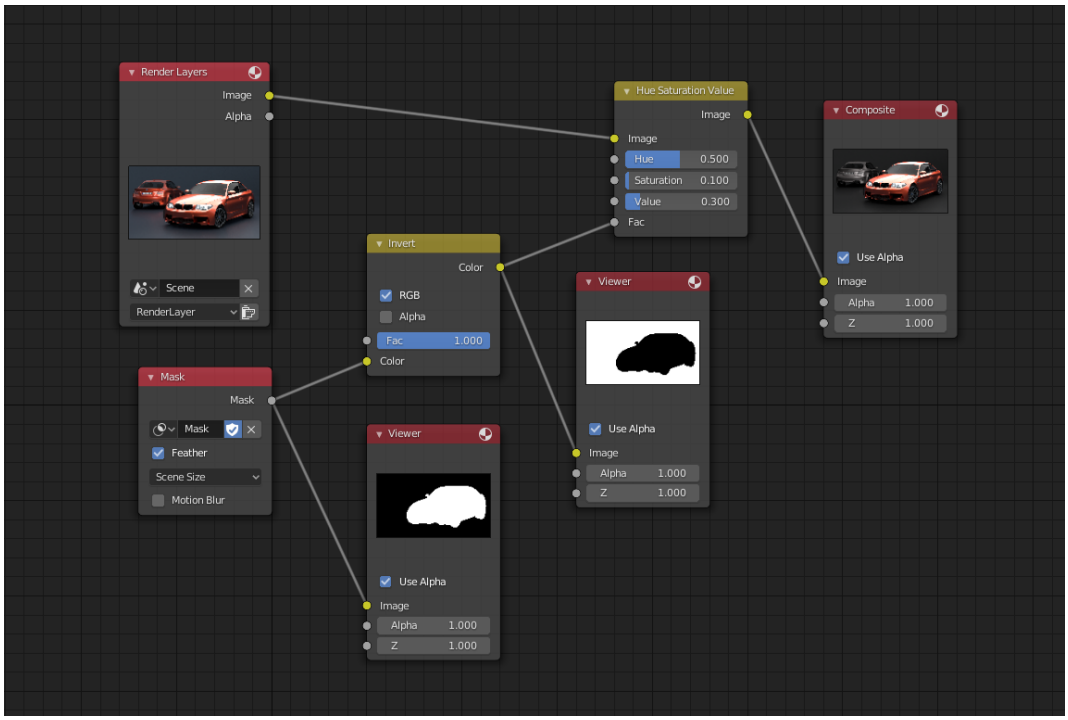
Masking

- Introduction
  - Mask Data-block
  - Header
- S-Curves
  - Primitives
- Selecting
  - All
  - None
  - Invert
  - Box Select
  - Circle Select
  - Lasso Select
  - Select Linked
- Editing
  - Transform
  - Clear Feather Weight
  - Toggle Cyclic
  - Set Handle Type
  - Recalculate Handles
  - Switch Direction
  - Copy Paste
  - Clear Parent
  - Make Parent
  - Animation
  - Show/Hide
  - Delete
  - Miscellaneous
- Sidebar
  - Mask Settings
  - Mask Layers
  - Active Spline
  - Active Point

Introduction

Masks can be created in the *Image* and *Movie Clip* editors, by changing the mode to *Mask* in the header. This will add various tools and properties to the editor panels, while hiding others that are not needed for interacting with masks.

Masks have many purposes. They can be used in a motion tracking workflow to mask out, or influence a particular object in the footage. They can be used for manual rotoscoping to pull a particular object out of the footage, or as a rough matte for green-screen keying. Masks are independent from a particular image of movie clip, and so they can just as well be used for creating motion graphics or other effects in the Compositor.



Using the Mask node to isolate an object in compositing.

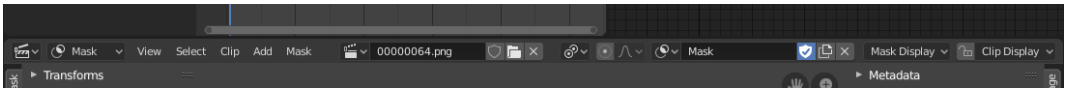
While the *Movie Clip Editor* and *Image Editor* are used to edit masks, the *Compositor* and *Sequencer* are just using already created mask.

Masks can be animated over the time so that they follow some object from the footage, e.g. a running actor. This can be achieved with shape keys or parenting the mask to tracking markers.

Mask Data-block

Mask data-block containing multiple mask layers and splines. They are the most high-level entities used for masking purposes. Masks can be reused in different places, and hold global parameters for all the entities they consist of.

Header



The Movie Clip Editor header in Mask mode.

Menus

View
: Center View to Cursor
: Move the view so that the 2D cursor is at the center of the editor.

Add
: Use to add primitive shapes.

Mask
: Operators used to Edit masks.

Controls

Mask
: Once set to Mask mode, a mask data-block can be added with a data-block menu. Any image, movie clip, render or compositing result can be used as a backdrop to display masks over.

New + `Alt-N`

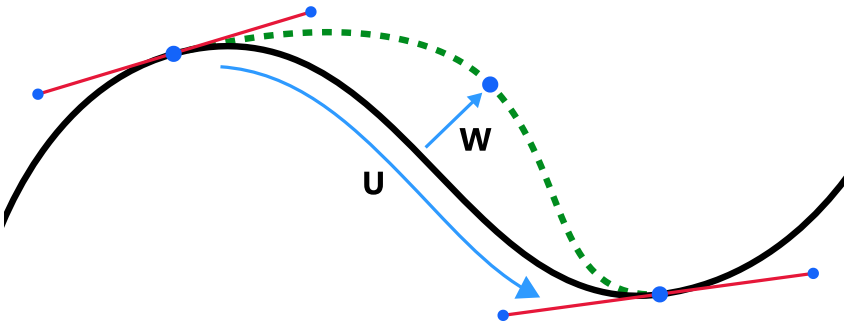
Mask Display

See Mask Display.

S-Curves

The curve type used for creating mask splines is almost a Bézier curve, but with some differences. Smooth edges of the mask are defined by feathering. The curve needed to support feathering in a way that stuck to the curve as you edited it, for ease of editing an animation. These are called S-curves.

Besides the handles, every control point also has points that define the feather between the current point and the next point on the spline. Each feather point is stored in UW space, where U means position across spline segment, and W (weight) means distance between main spline and feather points.



S-Curve explained.

This allows for deforming the main spline in almost any way, and the feather will be updated automatically to reflect that change.

For example if there is just rotation of the spline, feather would stay completely unchanged. If one point's feather is moved, the other feathers will be automatically stretched uniformly along that segment and the overall shape will be almost the same as artists would want it to be.

Primitives

Reference

Mode::	Mask Mode
Tool::	*Add*
Shortcut::	`Shift-A`

There are two primitives available: a Bézier Circle and a Square with vector handles.

Selecting

All

Reference

Mode::	All modes
Menu::	*Select ▸ All*
Shortcut::	`A`

Selects all items.

None

Reference

Mode::	All modes
Menu::	*Select ▸ None*
Shortcut::	`Alt-A`

Resets the selection to nothing.

Invert

Reference

Mode::	All modes
Menu::	*Select ▸ Inverse*
Shortcut::	`Ctrl-I`

Selects non-selected items and deselects existing selection.

Box Select

Reference

Mode::	All modes
Menu::	*Select ▸ Box Select*
Shortcut::	`B`

See Select Box.

Circle Select

Reference

Mode::	All modes
Menu::	*Select ▸ Circle Select*
Shortcut::	`C`

See Select Circle.

Lasso Select

Reference

Mode::	All modes
Menu::	*Select ▸ Lasso Select*
Shortcut::	`Ctrl-Alt-LMB`

See Select Lasso.

Select Linked

Reference

Mode::	All modes
Menu::	*Select ▸ Select Linked*
Shortcut::	`Ctrl-L`

Select all curve points linked to already selected ones.

Editing

The tools and panels available to edit masks are the same in both editors. Editing of mask splines happens in a way similar to editing Bézier curves or paths in GIMP or other curve editors.

Tip

To get interactive feedback on the resulting mask, a Mask node can be connected directly to a Viewer node in the Compositor, which will then keep updating the compositing result while editing.

Transform

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Transform*

Move `G`
: Change the location of control points. Control points can also be moved with `LMB`. The whole spline can be moved by dragging the center dot with `LMB`.

Rotate `R`
: Change the location of control points by rotating about a pivot point.

Scale `S`
: Change the location of control points by expanding the distance between points.

To Sphere `Shift-Alt-S`
: Morphs the control points to the shape of a circle.

Shear `Shift-Ctrl-Alt-S`
: Shifts control points along a defined axis so parallel control points move past one another.

Push/Pull
: Moves the control points closer together (Push) or further apart (Pull).

Scale Feather `Alt-S`
: Will scale the feather size.

Clear Feather Weight

Reference

Mode::	Mask Mode
Menu::	*Mask ▸ Clear Feather Weight*

Resets the feather weight to zero.

Toggle Cyclic

Reference

Mode::	Mask Mode
Menu::	*Mask ▸ Toggle Cyclic*
Shortcut::	`Alt-C`

Toggle to create a closed curve or open it again. Close the mask by joining the last control point to the first.

Set Handle Type

Reference

Mode::	Mask Mode

Menu:: *Mask ▸ Set Handle Type*
Shortcut:: `V`

Set handle type for selected spline points.

Recalculate Handles

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Recalculate Handles*
Shortcut:: `Shift-N`

Make normals (handle directions) consistent.

Switch Direction

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Switch Direction*

Switch Direction handle directions in/out.

Copy Paste

Todo.

Clear Parent

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Clear Parent*

Shortcut:: `Alt-P`

Clears any parenting relationship for the selected spline points.

Make Parent

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Make Parent*
Shortcut:: `Ctrl-P`

Parents one or more selected spline points to the active motion tracker.

Animation

Reference

Mode:: Mask Mode
Menu:: *Mask ▸ Animation*

Masks can be animated with the shape keying system. This can be useful when there are not enough good feature points to track in the footage, or the mask is not based on footage. Mask animation timing can be edited from the *Dope Sheet's* Mask Mode.

Insert Shape Key `I`
: Will insert a shape key for the active mask layer at the current frame. This works on the level of mask layers, so inserting a shape key will keyframe all the splines and points contained in it.

Clear Shape Key `Alt-I`
: Will clear the shape key for the active mask layer at the current frame.

Feather Reset Animation

Resets the feather offset across all animated frames.

Re-Key Points of Selected Shapes
: Re-interpolate selected points on across the range of keys selected in the *Dope Sheet*.

Show/Hide

Reference

Mode::	Mask Mode
Menu::	*Mask* ‣ *Show/Hide*

- Hide Selected `H`
- Hide Unselected `Shift-H`
- Clear Restricted View `Alt-H`

Delete

Reference

Mode::	Mask Mode
Menu::	*Mask* ‣ *Delete*
Shortcut::	`X`

Removes control points.

Miscellaneous

Slide Spline Curvature `LMB`
: Moves the curve and/or control points by clicking on them and dragging.

Add Vertex and Slide `Ctrl-LMB`

Inserts new control points and defines handle orientations by a continued mouse drag. If the last point was selected, double-click will also close the curve.

Add Feather Vertex and Slide `Shift-Ctrl-LMB`
: Inserts new feather control points that can be transformed independently of the main spline curve. If no feather mask is in use this will create a basic feather mask to the curve.

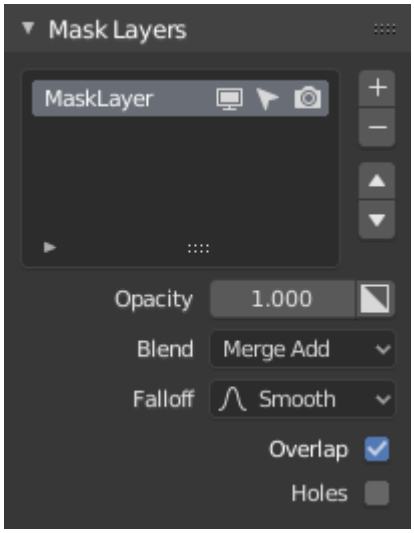
Sidebar

Mask Settings

Start Frame, End Frame
: Set the frame range of the mask for *Sequencer*.

Mask Layers

Mask layers consists of one or several splines and used to “grouped” operation on splines. Layers can be used to create complex shapes and to define how the splines interact with each other. Splines belonging to the same layer can be animated together, for example by an item from motion tracker footage. Example of such tools might be parenting the whole set of splines to single motion tracking data or simple to transform all of them together.



Mask Layer panel.

Opacity
: Used to set the opacity of the mask layer.

Invert (black/white icon)
: Inverts the values (colors) in the mask layer.

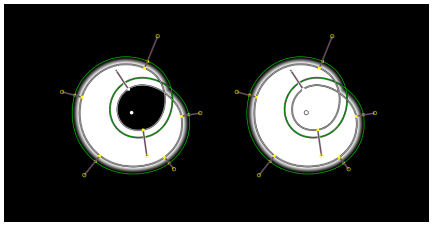
Blend
: The layer blending operation to perform. See Color Blend Modes.

  Modes *Merge Add* and *Merge Subtract* give better results when using a *Feather* on overlapping masks than straightforward mathematical addition and subtraction.

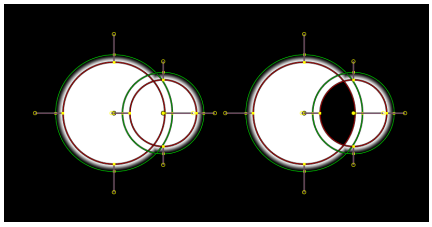
Falloff
: Type of the *Feather* falloff, controls the shape of the transition between black and white.

Overlap
: Fills the self-intersecting areas.

Holes
: Overlapping splines from the same layer will generate holes in the mask.



The Overlap option example.



The Holes option example.

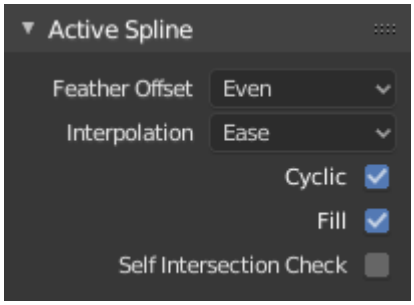
Example

The purpose of mask layers can be explained with an example. Suppose there are two unwanted people in the footage, and one of them goes from left to right, and the other in the opposite direction. Two mask layers can then be used to mask them separately by using a single mask data-block. At the point of intersection of these shapes they will be added together rather than creating a hole, as would happen if they were on the same layer. If the motion is simple enough, a single motion tracked point can be used to drive the location of the entire mask layer.

Active Spline

Feather Offset
: The method used for calculating the offset of the mask spline feather.

Even::



Active Spline panel.

Smooth::	Preserves the thickness of the feather, but can give undesirable loops of the feather curve. Gives a nicer and smoother shape, but can also give an undesirable sharp feather when a curve segment forms an S-shape.

Weight Interpolation
: The type of weight (thickness of feather) interpolation between points. *Linear* or *Ease* (i.e. changes occur slowly at the beginning and at the end).

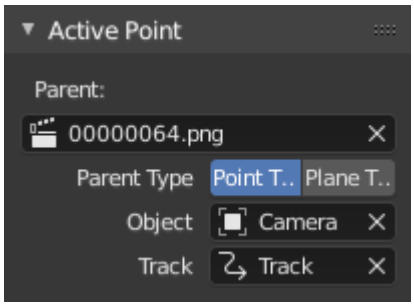
Cyclic
: If the spline is closed or not.

Fill
: Creates splines with filled areas. If disabled, Blender will create curves with a thickness to mask out thin objects such as wires or hair.

Self Intersection Check
: Prevent the feather (not the curve itself) from intersecting with itself.

Active Point

This panel is shown when both a tracking marker and mask is selected.



Active Point panel.

Parent

In the *Movie Clip Editor* it is possible to link the whole mask or its points to motion tracks. This way the mask or points will follow the tracks.

Parent
: Data ID to which the mask or spline is parented to in case of parenting to movie tracking data set to Movie Clip data-block.

Parent Type
: Point Track, Plane Track

Object
: Object to parent to.

Track
: Name of individual tracks.

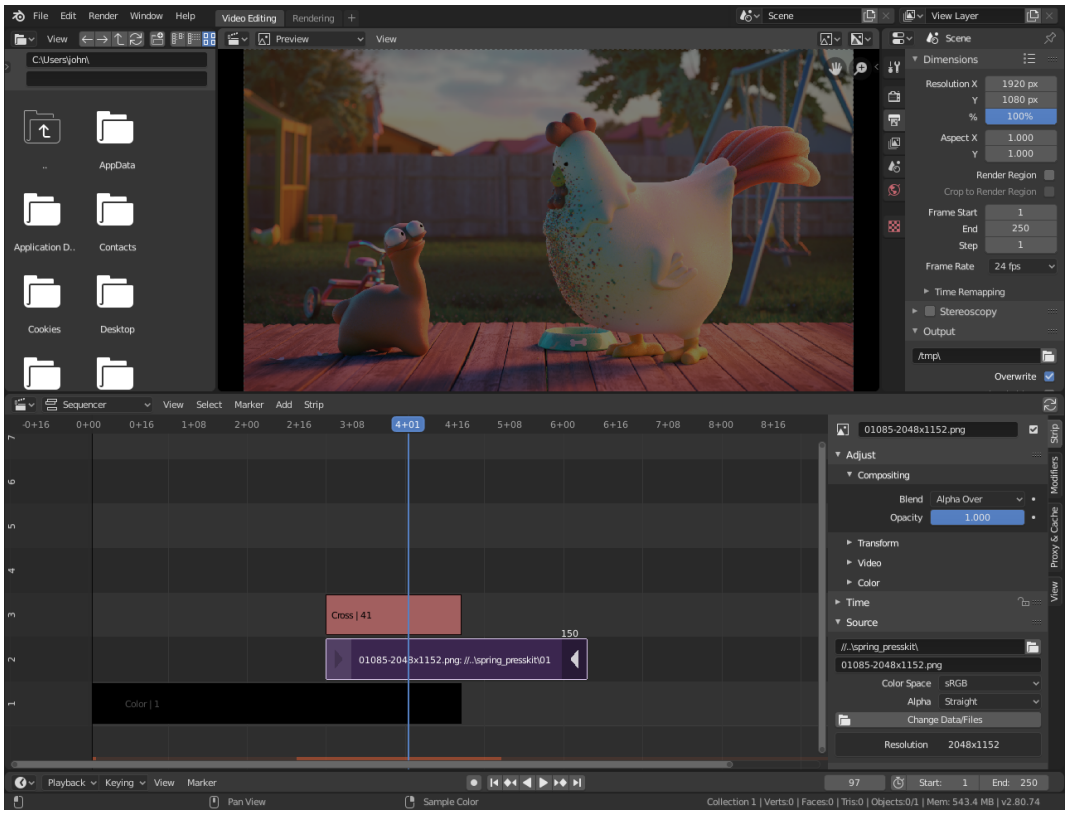
Video Editing

- Introduction
- Setup Your Project
  - Introduction
  - Directory Structure
- Edit Your Project
  - Introduction
  - Montage

Introduction

In addition to modeling and animation, Blender can be used to edit videos. There are two possible methods for this, one being the Compositor and the other, described in this chapter, being the Video Sequencer. The Video Sequencer within Blender is a complete video editing system that allows you to combine multiple video channels and add effects to them. You can use these effects to create powerful video edits, especially when you combine it with the animation power of Blender!

To use the Video Sequencer, you load multiple video clips and lay them end-to-end (or in some cases, overlay them), inserting fades and transitions to link the end of one clip to the beginning of another. Finally, you can add audio and synchronize the timing of the video sequence to match it.



Default Video Editing screen layout.

Setup Your Project

- Introduction
- Directory Structure

Introduction

The proverb “A good start is half the battle”, certainly applies to video editing. Setting up your *work environment* and *project* to your needs is key to success. In setting up your video project, you have to distinguish between:

- Work Environment related settings and activities:

  These settings apply to all of your projects and are set at “Blender level”; for example the installation of add-ons. In fact, they can also influence your non-video editing projects. Most of these settings remain more or less stable throughout your projects and should probably only set once.

- Project related settings and activities:

  These settings vary from project to project and are very specific for your project; for example the output media format. For each new project, you have to evaluate these settings and activities.

Of course, many settings and activities occur at both levels. For example, automatic proxies can be enabled globally but changed on a per project or even strip basis. The layout of the Video Editing Workspace is defined at Blender level but can be tweaked per project.

Directory Structure

A video project is most likely a combination of several different assets. You can separate them into three broad categories.

- Video files: video clips (or movies in Blender-talk), photos, graphic files (charts, logos, …), and Visual Effects (VFX) such as masks, lens flares, animation.
- Audio files: recorded dialog, voice-over, music, and Sound Effects (SFX) such as environmental sounds, swooshes, …
- Project files: the blend files and backups, (partial) render results, documentation such as scripts and storyboards.

Together they can form rapidly an intangible heap of files. It's good practice to collect all those assets in *one* project directory with appropriate subdirectories. Why? It will lower the probability that you accidentally delete a file (which will result in a 'file not found' error) or that you forget to include a necessary file when transferring the project ('file is missing' error). And most of all, an appropriate directory structure, will help you to keep a clear overview of your assets.

See also

Blender can incorporate some files within the Blend file (see Packed Data). However, this doesn't work with video files, which can have very huge file sizes. So, it's better to assure that your project directory contains all necessary files.

It's also good practice to use some kind of naming convention and add metadata. Figure 1 shows a possible example, based on our categorization of the assets above. The directories are numbered so that they mimic a normal workflow.

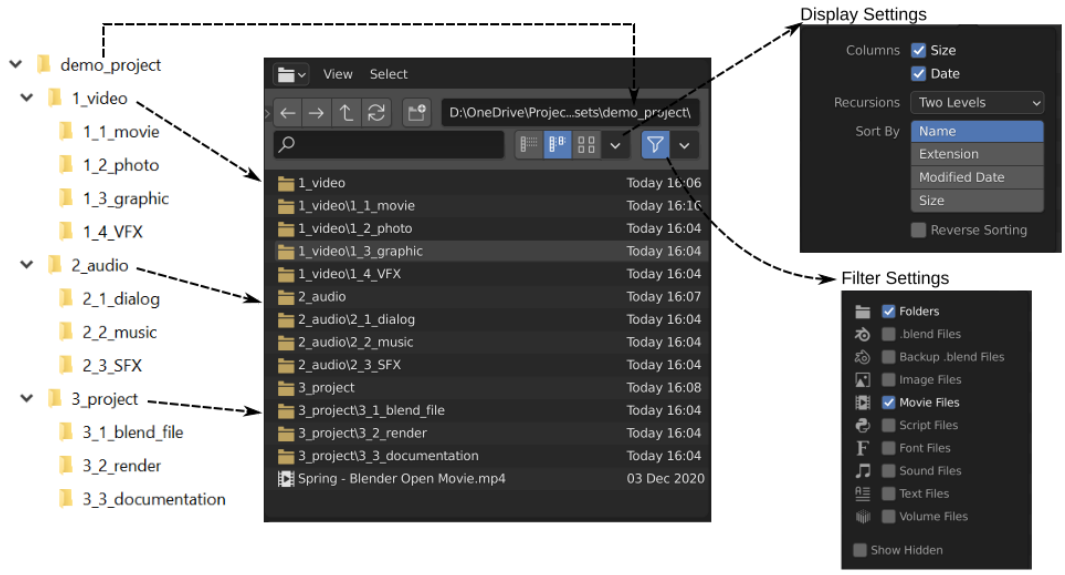


Figure 1: Organizing your project.

Edit Your Project

- Introduction
- Montage
  - Introduction
  - Strips
  - Selecting
  - Editing
  - Meta Strips

Introduction

There is no single optimal workflow for editing your video projects that fits every case. Although there certainly are some specific use cases; e.g. tutorial editing, wedding videos, …, some agreement exists about distinguishing four basic activities.

1. Montage: starting with your raw footage, you'll have to deliver a coherent and fluent succession of strips that tell your story. Clips have to be rearranged, combined, split (cut) and trimmed.
2. Effects: these can be simple transition effects between strips, e.g. fade or full-fledged animations, e.g. rolling end credits.
3. Color grading: because your assembled timeline consists of different shots taken with different cameras under different lighting conditions, colors can vary widely between them. With color grading and color correcting you can harmonize the color perception.
4. Sound: this ranges from adding background music and voice-over to creating special sound effects and using third-party software such as Audacity.

Montage

- Introduction
- Strips
  - Introduction
  - Types
- Selecting
  - Select Menu
- Editing
  - Transform
  - Image Transform
  - Split
  - Hold Split
  - Duplicate Strips
  - Delete
  - Update Scene Frame Range
  - Separate Images
  - Movie Strip
  - Effect Strip
  - Lock/Unlock
  - Mute/Unmute
  - Inputs
  - Context Menu
  - Fades
  - Retiming
- Meta Strips

Introduction

Montage is the technique of assembling separate clips (video, audio, text, effects) into a coherent sequence. The importance of montage was first demonstrated by the Russian filmmaker Lev Kuleshov in the 1910s and 1920s. Famous is the Kuleshov effect: viewers derive more meaning from the interaction of two sequential shots than from a single shot in isolation (see the Wikipedia article for a nice illustration).

Obviously, the first thing in assembling your timeline is importing or adding strips. There are several ways to do this and each has its own advantages and disadvantages.

Before you can do anything with a strip, you have to select them. There are multiple ways of selecting strips.

Strips can be moved in time (left to right on the X axis) or in the display stack (bottom to top on the Y axis).

Splitting or cutting a strip will create two parts of the strip (before and after the split). There are two variants: Split and Hold Split.

Trimming is the process of removing or adding a portion of the video at its head or tail. This will result in a decrease or increase of the duration of the video.

Grouping is the creating of a meta strip whereby several strips are grouped together.

Hint

Creating a good montage is a time consuming process. Therefore, all basic operations (add, split, trim, …) have an associated shortcut. It pays off to

learn these shortcuts, even though the same result could be obtained with the menu and/or mouse.

Strips

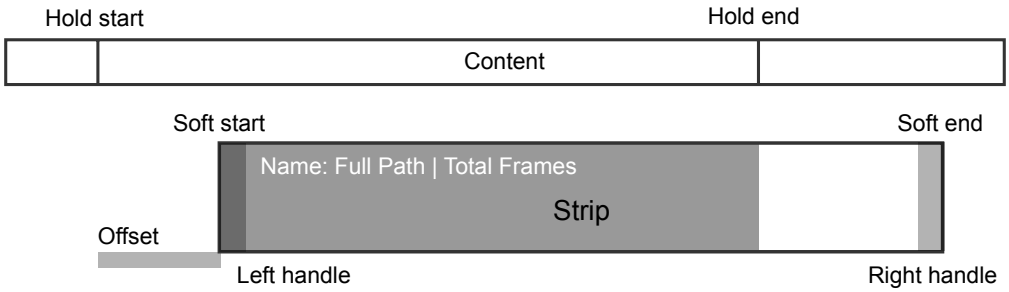
- Introduction
  - Adding Strips
  - Visualization

Types

- Scene Strip
- Clip Strip
- Mask Strip
- Movie Strip
- Sound Strip
- Image/Sequence Strip
- Color Strip
- Text Strip
- Adjustment Layer Strip
- Effect Strips
  - Add Strip
  - Subtract Strip
  - Multiply Strip
  - Alpha Over, Under & Over Drop Strips
  - Color Mix Strip
  - Multicam Selector Strip
  - Transform Strip
  - Speed Control Strip
  - Glow Strip
  - Gaussian Blur Strip
- Transitions
  - Sound Crossfade
  - Cross Strip
  - Gamma Cross Strip
  - Wipe Strip

Introduction

A strip is a container which carries frames provided by one or more sources (input). It is defined by a *Start Frame* and a *Length*, and is displayed as a colored horizontal rectangle.



Strip schematic.

Adding Strips

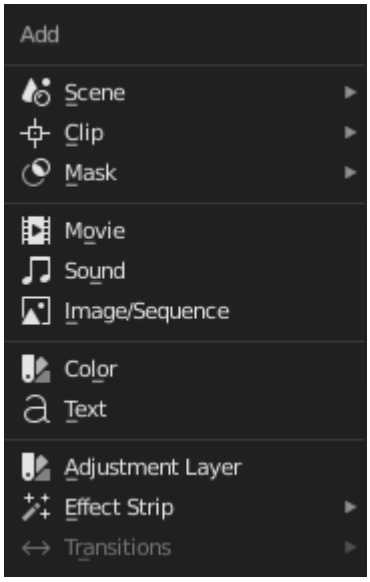
Reference

Menu:: *Add*
Shortcut:: `Shift-A`

The Add menu is the main menu you will be using to add content to the Video Sequencer. In general, you load up your strips, create strips of special transition effects, and then animate out your sequence by selecting “Do Sequence” and clicking the *Animation* button. You can use the Add menu in the header, or hover your mouse cursor over the Sequence workspace and press `Shift-A`.

Blender does not care which of these you use; you can freely mix and match any of them. When you choose to add one of these, it lets you either choose a data-block or the editor area will switch to a File Browser for you to select what you want to add. Supported files are filtered by default.

The start frame of the newly created strips will be placed at the position of the frame indicator. When loading multiple files (movie and sound) at the same time each will be added one after the other.



The Add Menu.

Adding Effects & Transitions

Blender offers a set of effects that can be added to your sequence.

To add an effect strip, select one base strip (image, movie, or scene) by `LMB` clicking on it. For some effects, like the Cross transition effect, you will need to `Shift-LMB` a second overlapping strip (it depends on the effect you want). From Add menu pick the effect you want. When you do, the Effect strip will be shown above the source strips. If it is an independent effect, like the Color Generator, it will be placed at the position of the frame indicator.

Note

Since most Effects strips depend on one or two source strips, their frame location and duration depends on their source strips. Thus, you may not be able to move it; you have to move the source strips in order to affect the effect strip.

With some effects, like the Alpha Over, the order in which you select the strips is important. You can also use one effect strip as the input or source strip with another strip, thus layering effects on top of one another.

If you picked the wrong effect from the menu, you can always exchange it using Effect Strip.

Visualization

They all become a color-coded strip in the Video Sequencer:

- Scene strip: Light green.
- Clip strip: Dark blue.
- Mask strip: Red.
- Movie strip: Aquamarine.
- Image strip: Purple.
- Image strip: Turquoise.
- Sound strip: Turquoise.

Each of the effect strips has its own color.

Besides each of these default colors you can also assign individual strips an alternative color in the Strip Properties.

Note

These colors are dependent on the user interface Theme. The colors described above are in reference to Blender's default theme.

Scene Strip

Scene strips are a way to insert the render output of another scene into your sequence. Instead of rendering out a video, then inserting the video file, you can insert the scene directly.

The strip length will be determined based on the animation settings in that scene.

Note

Scene strips cannot be used to reference the sequence's own scene; a secondary scene must be used instead.

Adding Scene Strips

Existing scenes strips can be added from the *Add ▸ Scene ▸ "Scene Name"*. New scenes can also be created directly from the add menu with *Add ▸ Scene ▸ New Scene*.

Options

Start Frame
: The first frame to start the scene strip.

Channel
: The channel to place the strip in.

Replace Selection
: Replace the active strip with the new scene strip.

When creating a new scene you have the following options:

Type
: How the new scene is created.

New::	Add new Strip with a new empty Scene with default settings.
Copy Settings::	Add a new Strip, with an empty scene, and copy settings from the current scene.
Linked Copy::	Add a Strip and link in the collections from the current scene (shallow copy).
Full Copy::	Add a Strip and make a full copy of the current scene.

Options

Scene
: A Data-Block Menu to select or create the scene to render from.

Input
: Input type to use for the Scene strip.

Camera::	Use the Scene's 3D camera as input.
Sequencer::	Use the scene's Sequencer timeline as input, allowing one scene to reuse another scene's edit (instead of taking the render output from the scene). This is similar to how Meta Strips work, with the added advantage of supporting multiple instances of the same data.

Camera
: This can be used to override the scene's camera with any other object.

  It is useful to support switching views within a single scene.

Show
: Annotations

Shows Annotations while in non-render Preview Shading Modes i.e. *Solid* or *Wireframe* mode.

Transparent
: Creates a transparent background. This is useful for doing overlays like rendering out Grease Pencil films via the Sequencer.

Sound

Strip Volume
: Volume of the audio taken from the chosen scene.

Limitations

Scene strips do not render individual Render Passes; only the *Combined* render pass will be used.

Clip Strip

Clip can be modified within the Movie Clip Editor.

Options

Reference

Panel:: *Sidebar region ▸ Strip ▸ Movie Clip*

Movie Clip
: Used to select the movie clip. For controls see Data-Block Menu.

Use: 2D Stabilized Clip
: Use the 2D stabilized version of the clip.

Use: Undistorted Clip
: Use the undistorted version of the clip.

Below the properties, shows the frame range of the movie clip (before strip adjustments).

Mask Strip

The Mask strip generates a mask image from the selected mask data-block generated in the Movie Clip Editor. This works similar to the Mask Node but without the options available for finer control. The mask image is always generated at the render resolution, scaling along with different proxy levels.

Options

Mask
: Data-block menu to select a mask.

Movie Strip

To add a movie (with or without audio) select a movie file(s) in the File Browser e.g. in the Audio-Video Interleaved format (`*.avi` file).

Note

Clips can be Huge

A three minute QuickTime `.mov` file can be 140MB. Loading it, even over a high-speed LAN can take some time. Do not assume your computer or Blender has locked up if nothing happens for awhile.

Tip

Movie strips can display thumbnails in the Sequencer overlaid on their strips by enabling the Thumbnails overlay.

Add Movie Strip

Reference

Menu:: *Add ▸ Movie*

Relative Path
: Store the location of the image file relative to the blend-file.

Start Frame
: The Start Frame to place the left handle of the strip.

Channel
: The Channel to place the strip.

Replace Selection

Replaces the currently selected strips with the new strip.

Fit Method

Determines how images with an aspect ratio different than the scene's Resolution are scaled to fit inside the render area.

Scale to Fit:: Adjusts the strips Scale Transforms so the visual contents of the strip to fit exactly within the project's Resolution while maintaining the original aspect ratio.
This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill:: Adjusts the strips Scale Transforms so the visual contents of the strip to span the project's Resolution while maintaining the original aspect ratio.
This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill:: Adjusts the strips Scale Transforms so the visual contents of the strip to fill the project's Resolution. Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.
This may mean that the original image becomes distorted to fit the content inside the rendered area.

Set View Transform

Automatically sets an appropriate View Transform based on the Color Space of the imported media. In most cases, the *Standard* should be used; using the wrong transform could result in inaccurate colors or degraded rendering performance.

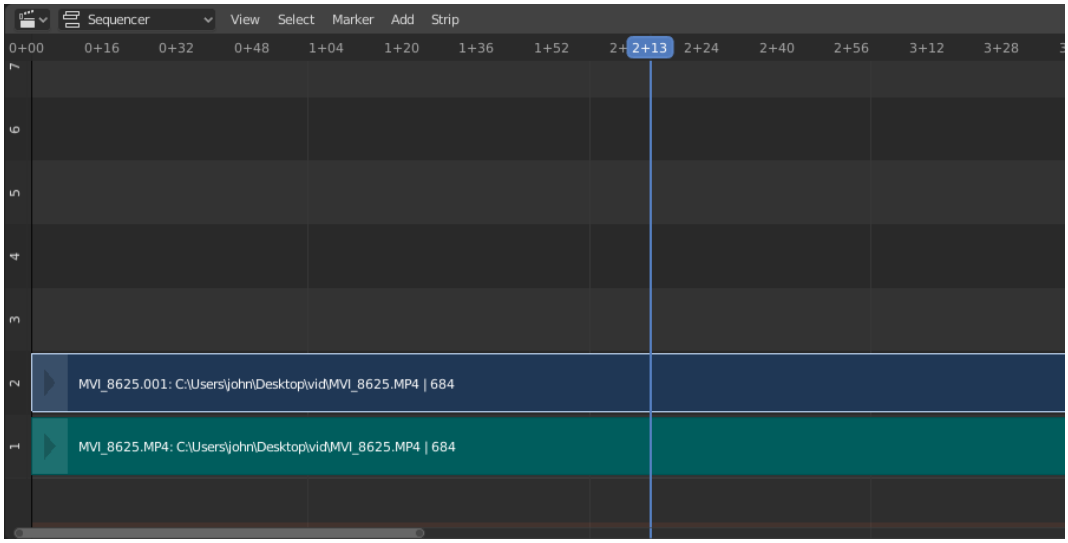
Adjust Playback Rate

Automatically adjusts the video's speed to playback at the original speed regardless of the scene's framerate.

Sound
: Add a Sound Strip that contains the movie's audio track.

Set Scene Frame Rate
: Sets the Scene Frame Rate to the frame rate encoded in the movie file.

Example

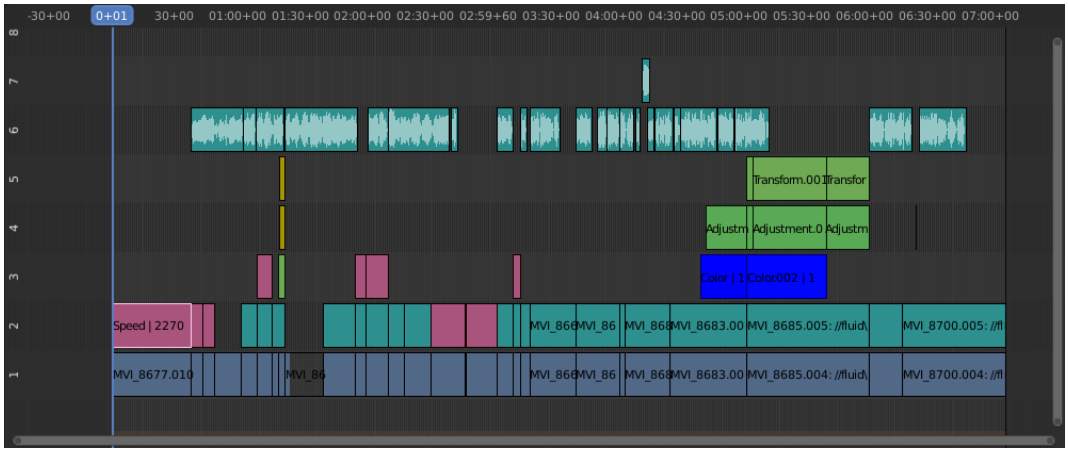


Imported Movie strip with audio track underneath.

In the strip itself, you can see strip name, path to source file, and strip length.

Sound Strip

As well as images and movies the Video Sequencer can also edit audio tracks. You can add Waveform Audio format `WAV`, `mp3` and other audio formats files from your drive, or from sound encoded within a movie, and mix them using an F-Curve as a volume control.



Example of sound editing.

Working with Audio Tracks

A Sound strip is just like any other strip in the Video Sequencer. You can select and move it, adjust its starting offset using `LMB` over the strip handles, and `K` cut it into pieces. A useful example is cutting out the “um’s” and dead voice time.

You can have as many Sound strips as you wish and the result will be the mixing of all of them. You can give each strip its own name and volume via the Sidebar region.

Overlapping strips are automatically mixed down during the rendering process. For example, you can have the announcer on channel 5, background music on channel 6, and Foley sound effects on channel 7.

See also

In the Playback Popover menu of the Timeline you will find some options concerning audio playback behavior.

Waveform

The waveform of the audio is shown depending on two options:

Overlay
: The Sequencer Overlay menu has options to show all strip wave-forms, none of them, or to use the per-strip option described below.

Strip
: Each strip has an option *Display Waveform*. It is only visible when the above overlay option is set to *Use Strip Option*.

Clipping audio, i.e. values over 100% amplitude, will be shown in red in the waveform.

More strip options are documented in Sound Sidebar Panel.

Animating Audio Track Properties

To animate Sound strips simply hit `I` over any of its values. Examples of animating an audio strip are to fade in/out background music or to adjust volume levels. Layered/crossed Sound strips are added together; the lower channel does not override and cut out higher channels (unlike image and video strips). This makes Blender an audio mixer. By adding audio tracks and using the curves to adjust each tracks sound level, you have an automated dynamic multi-track audio mixer!

See also

Sounds can be cross-faded by adding a Sound Crossfade effect.

Output

There are two ways to render out your audio. You can either have it encoded with a video file or in its own audio file. Read more on how to select a proper audio format and how to start rendering.

Add Sound Strip

Reference

Menu:: *Add ▸ Sound*

Relative Path
: Store the location of the image file relative to the blend-file.

Start Frame
: The Start Frame to place the left handle of the strip.

Channel
: The Channel to place the strip.

Replace Selection
: Replaces the currently selected strips with the new strip.

Cache
: Cache the sound in memory, enables Caching in the Source properties.

Mono
: Merge all sound channels into one channel, enables Mono in the Sound properties.

Image/Sequence Strip

Tip

Image strips can display thumbnails in the Sequencer overlaid on their strips by enabling the Thumbnails overlay.

Single Image

When you add a single still image (`*.jpg`, `*.png`, etc.), Blender creates a 25 frames long strip which will show this image along the strips range.

Image Sequence

In the case of (numbered) image sequences (e.g. `*-0001.jpg`, `*-0002.jpg`, `*-0003.jpg`, etc, of any image format), you have a choice:

Range
: Navigate into the directory and `LMB` click and drag over a range of names to highlight multiple files. You can page down and continue `Shift-LMB` click-dragging to add more to the selection.

Batch
: `Shift-LMB` click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (`jpg`, `png`, `exr`, etc.).

All
: Press `A` to select/deselect all files in the directory.

Tip

Dealing with Different Sizes

Dealing with different sized images and different sized outputs is tricky. If you have a mismatch between the size of the input image and the render output size, the Video Sequencer will try to auto-scale the image to fit it entirely in the output. This may result in clipping. If you do not want that, use *Crop* and/or *Offset* in the Input panel to move and select a region of the image within the output. When you use *Crop* or *Offset*, the auto-scaling will be disabled and you can manually re-scale by adding the Transform effect.

Add Image Strip

Reference

Menu:: *Add ▸ Image/Sequence*

Relative Path
: Store the location of the image file relative to the blend-file.

Start Frame
: The Start Frame to place the left handle of the strip.

End Frame
: The end frame to place the right handle of the strip.

> **Tip**
> Subtract the *Start Frame* from the *End Frame* to get the strip's duration.

Channel
: The Channel to place the strip.

Replace Selection
: Previously selected strips will be deselected. Only added strips will be selected.

Fit Method

Determines how images with an aspect ratio different than the scene's Resolution are scaled to fit inside the render area.

Scale to Fit:: Adjusts the strips Scale Transforms so the visual contents of the strip to fit exactly within the project's Resolution while maintaining the original aspect ratio.
This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill:: Adjusts the strips Scale Transforms so the visual contents of the strip to span the project's Resolution while maintaining the original aspect ratio.
This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill:: Adjusts the strips Scale Transforms so the visual contents of the strip to fill the project's Resolution. Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.
This may mean that the original image becomes distorted to fit the content inside the rendered area.

Set View Transform

Automatically sets an appropriate View Transform based on the Color Space of the imported media. In most cases, the *Standard* should be used; using the wrong transform could result in inaccurate colors or degraded rendering performance.

Use Placeholders

Image sequences can use placeholder files. This works by enabling *Use Placeholders* checkbox when adding an image strip. The option detects

the frame range of opened images using Blender's frame naming scheme (`filename + frame number + .extension`) and makes an image sequence with all files in between even if they are missing. This allows you to render an image sequence with a few frames missing and still the image strip will have the correct range to account for the missing frames displayed as black.

When the missing frames are rendered or placed in the same folder, you can refresh the Sequencer and get the missing frames in the strip. The option is also available when using the *Change Data/File* operator and allows you to add more images to the range.

Color Strip

This effect generates solid color frames. By default, when it is created, the Color strip is 25 frames long, but you can extend it by selecting and moving one of the ends. Use this strip crossed with your main movie to provide a fade-in or fade-out.

Options

Color
: Click on the color field in the Effect panel in the Sidebar region, to pick a different color.

Text Strip

The Text strip allows you to directly display text in the Sequence editor. The strip will display the text inserted in its text field on the final sequence.

Tip

All Text strips in a video sequence can be exported as a SubRip file. This is useful when using Text strips as subtitles.

Options

Text
: The actual text displayed.

Wrap Width
: Wraps the text by the percentage of the frame width, setting this to zero disables word wrapping.

Style

Font
: Data-Block Menu to choose which font-file is used to render the text.

  Bold
  : Use a bold font face with a strong/thick visual appearance.

  Italics
  : Use an italicized font face with a slanted visual appearance.

Size
: Size of the text.

Color
: The text color.

Shadow
: Creates a shadow of the specified color under the text.

Shadow Angle
: Defines the position of the shadow as an angle, 0° being to the right and 90° being below.

Shadow Offset
: Amount to shift the shadow compared to the normal text.

Shadow Blur
: Amount to blur the shadow.

Outline
: Creates a line with the defined color enclosing the shape of the text.

Outline Width
: The thickness of the outline.

Box
: Creates a background for the text to improve the readability and clarity of text in some situations. The color and opacity of the box can be adjusted using the color selector.

Box Margin
: The distance the box boundaries extends from the boundaries of the font glyphs. The distance is measured as a factor of the image's width.

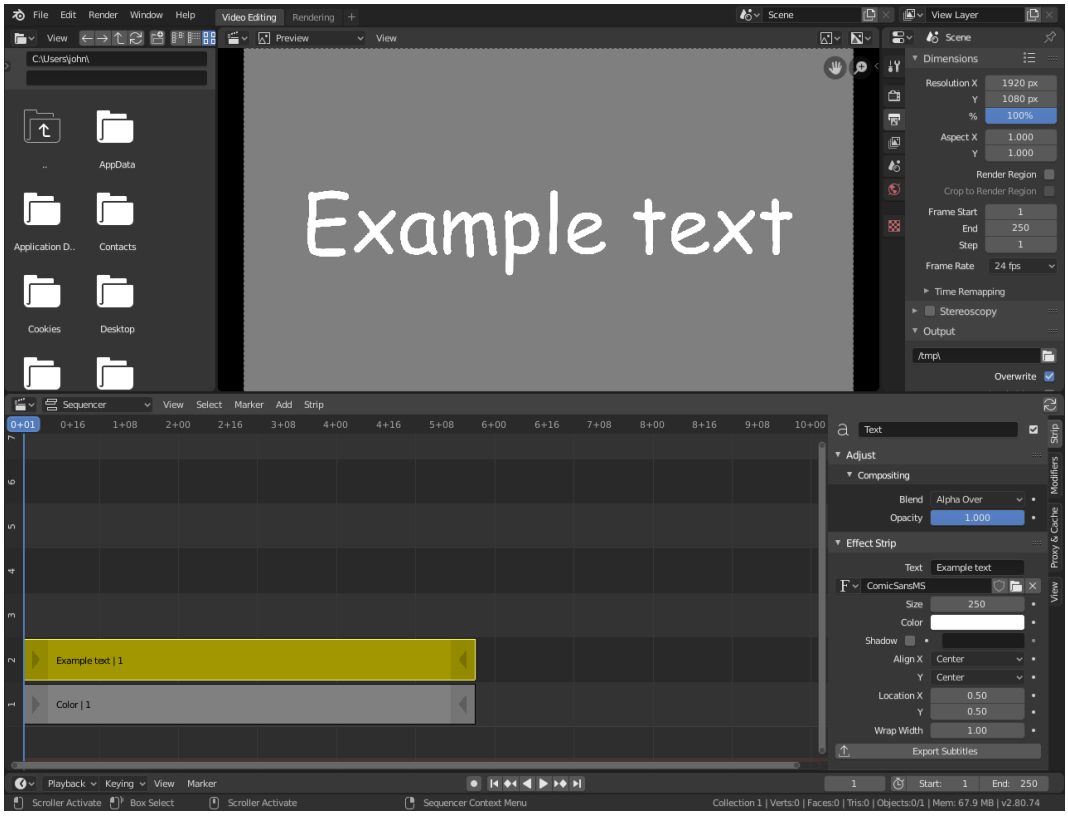
Layout

Location X, Y
: Positions the text on the X, Y axis.

Anchor X, Y

Horizontal (X) or vertical (Y) anchor point of the text relative to the location.

Example



Text effect.

Adjustment Layer Strip

The Adjustment Layer strip works like a regular input file strip except for the fact, that it considers all strips below it as its input.

Real-world use cases, you want to add some last finishing color correction on top of parts of your final sequence, timeline without messing with meta strips around. Just add an adjustment layer on top and activate the color balance.

Or you can stack a primary color correction and several secondary color corrections on top of each other (probably using the new mask input for area selection).

Options

This strip has no options.

Effect Strips

- Add Strip
- Subtract Strip
- Multiply Strip
- Alpha Over, Under & Over Drop Strips
- Color Mix Strip
- Multicam Selector Strip
- Transform Strip
- Speed Control Strip
- Glow Strip
- Gaussian Blur Strip

Add Strip

The Add effect strip adds the colors of two strips together. Use this effect with a base image strip, and a modifier strip. The modifier strip is either a solid color or a black-and-white mask, or another image entirely.

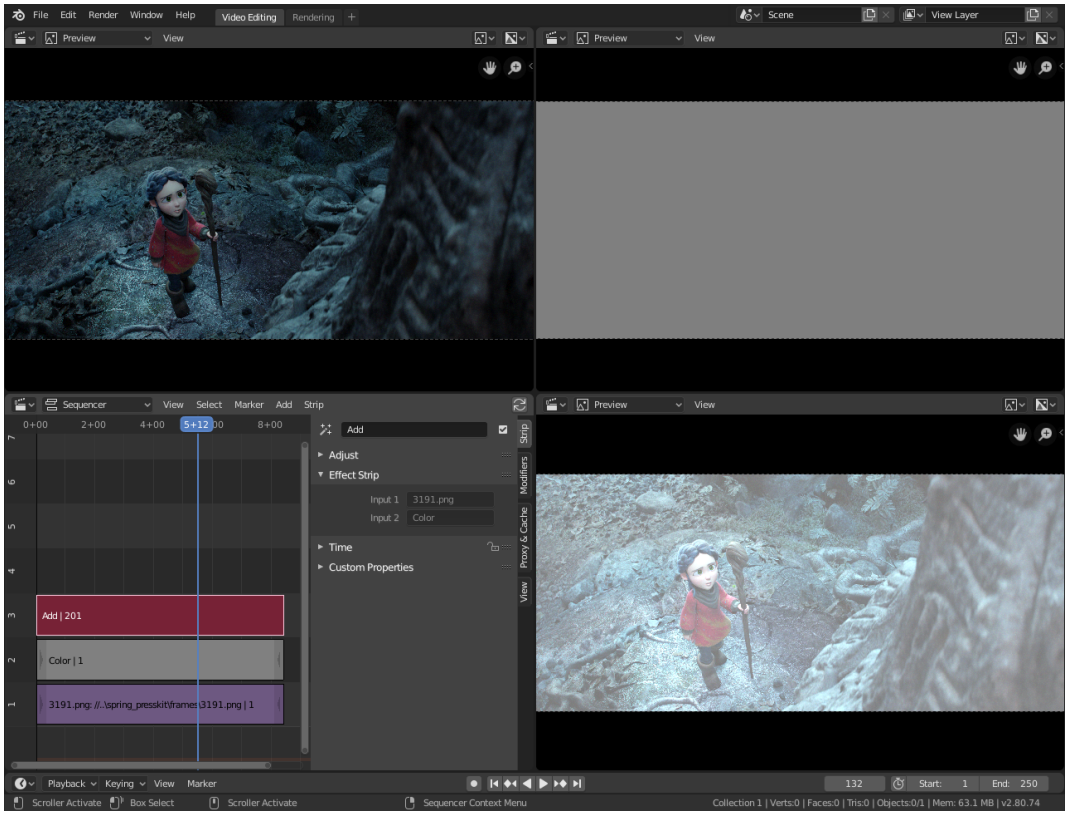
You can use this effect to increase the brightness of an image, or if you use a BW mask, selectively increase the brightness of certain areas of the image. The Mix node, in Add mode, does exactly the same thing as the Add SFX strip here, and is controlled the same way by feeding the Factor input.

The example shows what happens when you add gray to an image. The image gets bright because we are adding gray RGB(0.5, 0.5, 0.5) to say, a blue color RGB(0.1, 0.1, 0.5) resulting in RGB(0.6, 0.6, 1.0) which retains the original hue (relationship between the colors) but is much brighter (has a higher value). When applied to the whole image like this, it seems to flash.

Options

This strip has no options.

Example


File Edit Render Window Help
Video Editing
Rendering
Scene
View Layer
Preview
View
Sequencer
Select
Marker
Add
Strip
Add
Adjust
Effect Strip
Input 1
3191.png
Input 2
Color
Time
Custom Properties
Add | 201
Color | 1
3191.png: //..\spring_presskit\frames\3191.png | 1
Playback
Keying
Start: 1
End: 250
Scroller Activate
Box Select
Sequencer Context Menu

Add Effect.

Subtract Strip

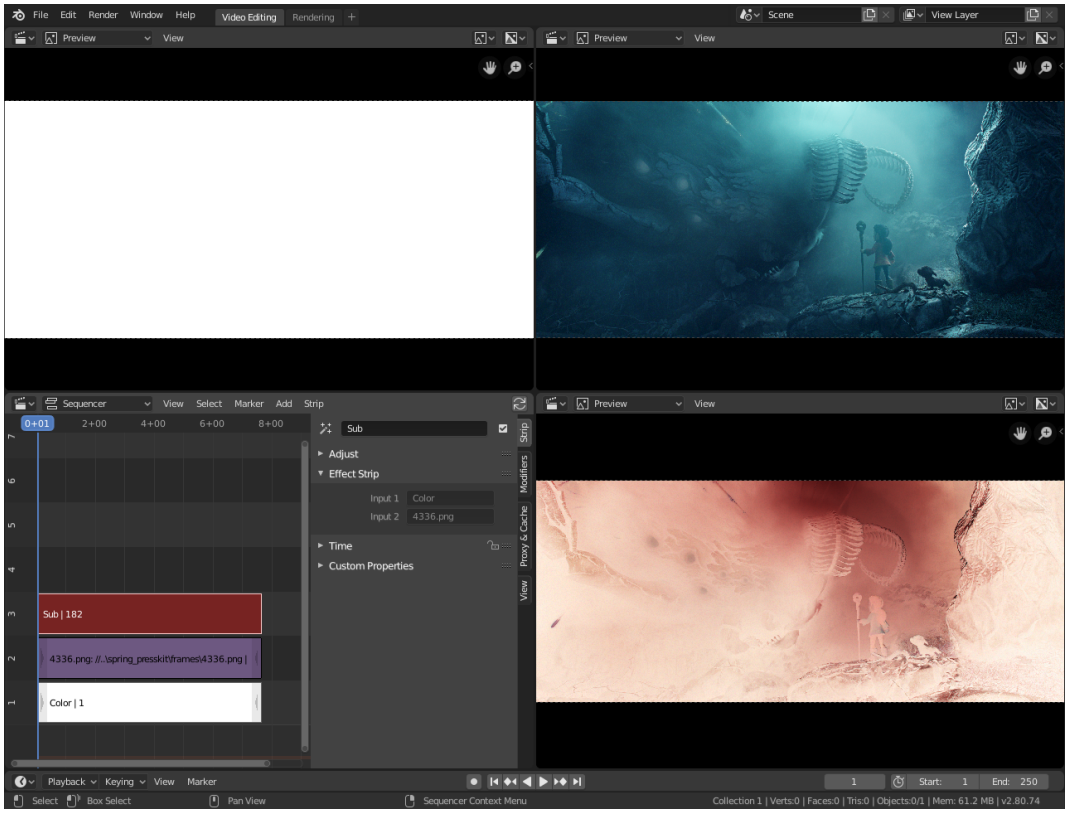
This effect takes away one strip's color from the second.

Make a negative of an image using this effect, or switch the order of the strips and just darken the strip. Subtracting a hue of blue from a white image will make it yellow, since red and green make yellow.

Options

This strip has no options.

Example



Subtract Effect.

Multiply Strip

The *Multiply* effect multiplies two colors. Blender uses values between (0.0 to 1.0) for the colors. This operation does not have to be normalized, the multiplication of two terms between (0.0 to 1.0) always gives a result between (0.0 to 1.0).

(With the "traditional" representation of three bytes, like RGB(124, 255, 56), the multiplications give far too high results, like RGB(7316, 46410, 1848), that have to be normalized (brought back) by dividing them by 256 to fit in the range of (0 to 255)…)

This effect has two main usages:

With a Mask

A mask is a black-and-white picture which, after multiplication with a "normal" image, only show this one in the white areas of the mask (everything else is black).

The opening title sequence to James Bond movies, where the camera is facing down the barrel of a gun at James, is a good example of this effect.

With Uniform Colors

Multiplying a color with a "normal" image allows you to soften some hues of this one (and so – symmetrically – to enhance the others).

For example, if you have a brown pixel RGB(0.50, 0.29, 0.05), and you multiply it with a cyan filter (uniform color RGB(0.0, 1.0, 1.0)), you will get a color RGB(0.0, 0.29, 0.05). Visually, the result is to zero the reds and bring up (by "symmetry" – the real values remain unchanged!) the blues and greens. Physically, it is the same effect as shining a cyan light onto a chocolate bar. Emotionally, vegetation becomes more lush, water becomes more Caribbean and inviting, skies become friendlier.

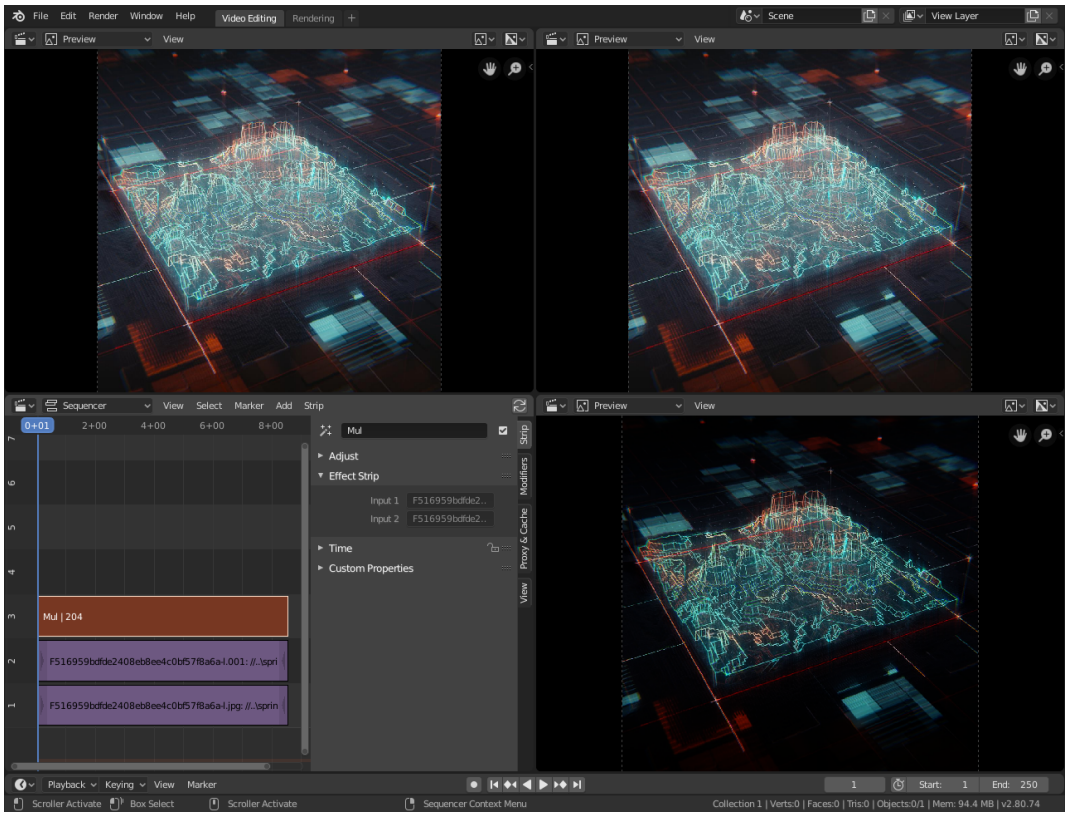
Note

This effect reduces the global luminosity of the picture (the result will always be smaller than the smallest operand). If one of the images is all white, the result is the other picture; if one of the images is all black, the result is all black!

Options

This strip has no options.

Example



Multiply Effect.

Alpha Over, Under & Over Drop Strips

Using the alpha (transparency channel), this effect composites a result based on transparent areas of the dominant image. If you use a Scene strip, the areas of the image where there is not anything solid are transparent; they have an alpha value of 0. If you use a Movie strip, that movie has an alpha value of 1 (completely opaque).

So, you can use the *Alpha Over* / *Alpha Under* effect to composite the CGI Scene on top of your movie. The result is your model doing whatever as if it was part of the movie. The *Adjust* ‣ *Compositing* ‣ *Opacity* controls how much the foreground is mixed over the background, fading in the foreground on top of the background. The colors of transparent foreground image areas are ignored and do not change the color of the background.

Alpha Over

With *Alpha Over*, the strips are layered up in the order selected; the first strip selected is the background, and the second one goes *over* the first one selected. The *Opacity* controls the transparency of the *foreground*, i.e. *Opacity* of 0.0; will only show the background, and an *Opacity* of 1.0 will completely override the background with the foreground (except in the transparent areas of this one, of course!)

Warning

By clicking the *Premultiply Alpha* button in the Sidebar of the foreground strip, the alpha values of the two strips are not multiplied or added together. Use this effect when adding a foreground strip that has a variable alpha channel (some opaque areas, some transparent, some in between) over a strip that has a flat opaque (alpha=1.0 or greater) channel. If you

notice a glow around your foreground objects, or strange transparent areas of your foreground object when using *Alpha Over*, enable *Premultiply*.

Example



Alpha Over Effect.

Alpha Under

With *Alpha Under*, this is the contrary: The first strip selected is the foreground, and the second one, the background. Moreover, the *Opacity* controls the transparency of the *background*, i.e. an *Opacity* of 0.0; will only show the foreground (the background is completely transparent), and an *Opacity* of 1.0 will give the same results as with *Alpha Over*.

Over Drop

Over Drop is between the two others: as with *Alpha Under*, the first selected strip will be the foreground, but as with *Alpha Over*, the *Opacity* controls the transparency of this foreground.

The *Over Drop* effect is much like the Cross, but puts preference to the top or second image, giving more of a gradual overlay effect than a blend like the Cross does. Of course, all of the Alpha effects respect the alpha (transparency) channel, whereas Cross does not.

The degree of Alpha applied, and thus color mixing, can be controlled by an F-Curve. Creating a Sine wave could have the effect of the foreground fading in and out.

Color Mix Strip

The *Color Mix* effect strip mixes two strips by working on the individual and corresponding pixels of the two input strips.

This effect can do the exact same operation as the Add, Subtract, or Multiply effect strips but also other color blending modes.

Options

Blend
: The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

  Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Opacity
: The amount of the blend of the second image gets composed onto the first.

Multicam Selector Strip

The Multicam Selector strip is used for multi-camera editing. Multi-camera editing is when a scene is recorded using multiple cameras from different angles and then edited together afterwards.

Workflow

The process of multi camera editing can be rather easy in the Video Sequencer if properly setup. The following guide shows the basic steps to setup a basic multi camera editing workflow.

1. First you are going to want to add in each of your video strips.

2. Next, you will want to sync all your cameras by either using Audio Waveforms or by the movement of objects.

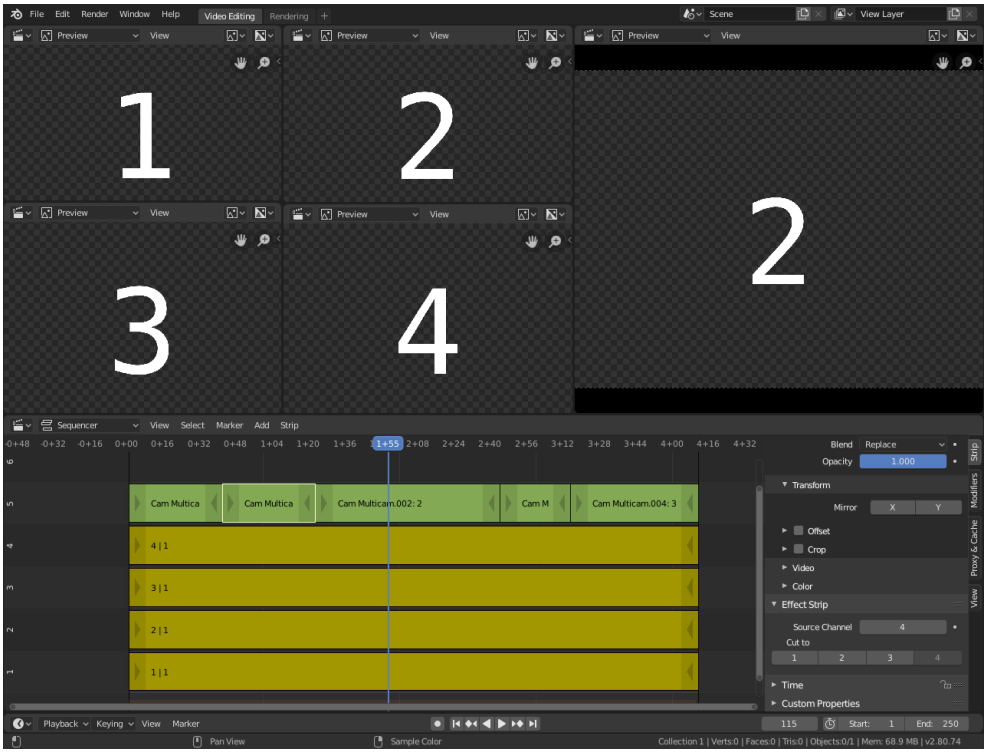
   Tip

   To make syncing strips easier you can group cameras, their audio, and their effects together using Meta Strips.

3. Split the editor into many Previews, one for each input track. Then change the Display Channel of each of the previews to the channel number of the input track.

4. Add a Multicam Selector strip *above* all the video channel tracks.

   After completing these steps you should get something similar to the following image:



Multi-camera editing setup.

5. Now select the Multicam strip, if you take a look at the strip options (in the Sidebar), you will notice, that Multicam is a rather simple effect strip: It just takes a selected channel as its input. That is all. The magic comes with the convenient keyboard layout.

6. When you select the Multicam strip, the keys `1` to `9` are mapped to the cut buttons. So, select the Multicam strip and start playback and press the keys for the correct input while watching the individual cameras.

7. You will end up with a small Multicam Selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what is coming, watch it again and do a rough cut using the number keys. Then fine-tune the placement by selecting the outer handles of two neighboring Multicam for A/B rolling.

Tip

To improve playback performance enable Proxies.

Options

Source Channel
: The channel which the Multicam Selector gets its input from.

Cut To
: Cuts the Multicam strip at the current frame and changes the *Source Channel* automatically to the selected channels.

Workflow

1. First you are going to want to add in each of your video strips.

2. Next, you will want to sync all your cameras by either using Audio Waveforms or by the movement of objects.

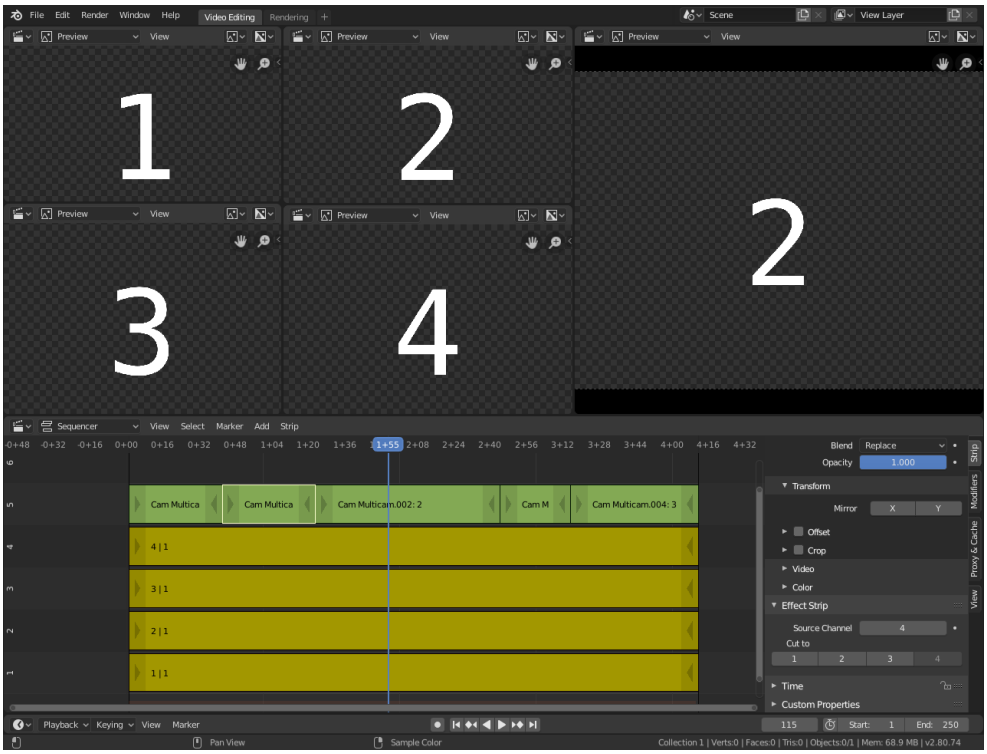
   Tip

   To make syncing strips easier you can group cameras, their audio, and their effects together using Meta Strips.

3. Split the editor into many Previews, one for each input track. Then change the Display Channel of each of the previews to the channel number of the input track.

4. Add a Multicam Selector strip *above* all the video channel tracks.

   After completing these steps you should get something similar to the following image:



Multi-camera editing setup.

5. Now select the Multicam strip, if you take a look at the strip options (in the Sidebar), you will notice, that Multicam is a rather simple effect strip: It just takes a selected channel as its input. That is all. The magic comes with the convenient keyboard layout.

6. When you select the Multicam strip, the keys `1` to `9` are mapped to the cut buttons. So, select the Multicam strip and start playback and press the keys for the correct input while watching the individual cameras.

7. You will end up with a small Multicam Selector strip for every cut.

In reality, it boils down to: watch a few seconds to see, what is coming, watch it again and do a rough cut using the number keys. Then fine-tune the placement by selecting the outer handles of two neighboring Multicam for A/B rolling.

Tip

To improve playback performance enable Proxies.

Transform Strip

Transform is a swiss-army knife of image manipulation. It moves, rotates, and scales the images within a strip.

Options

Interpolation

None::	No interpolation, uses nearest neighboring pixel.
Bilinear::	Simple interpolation between adjacent pixels.
Bicubic::	Highest quality interpolation.

Translation Unit

Control whether the input values are in *Percent* or *Pixels*.

Position

Moves the input along the X and Y axis.

Uniform Scale

Scale the input evenly along the X and Y axis.

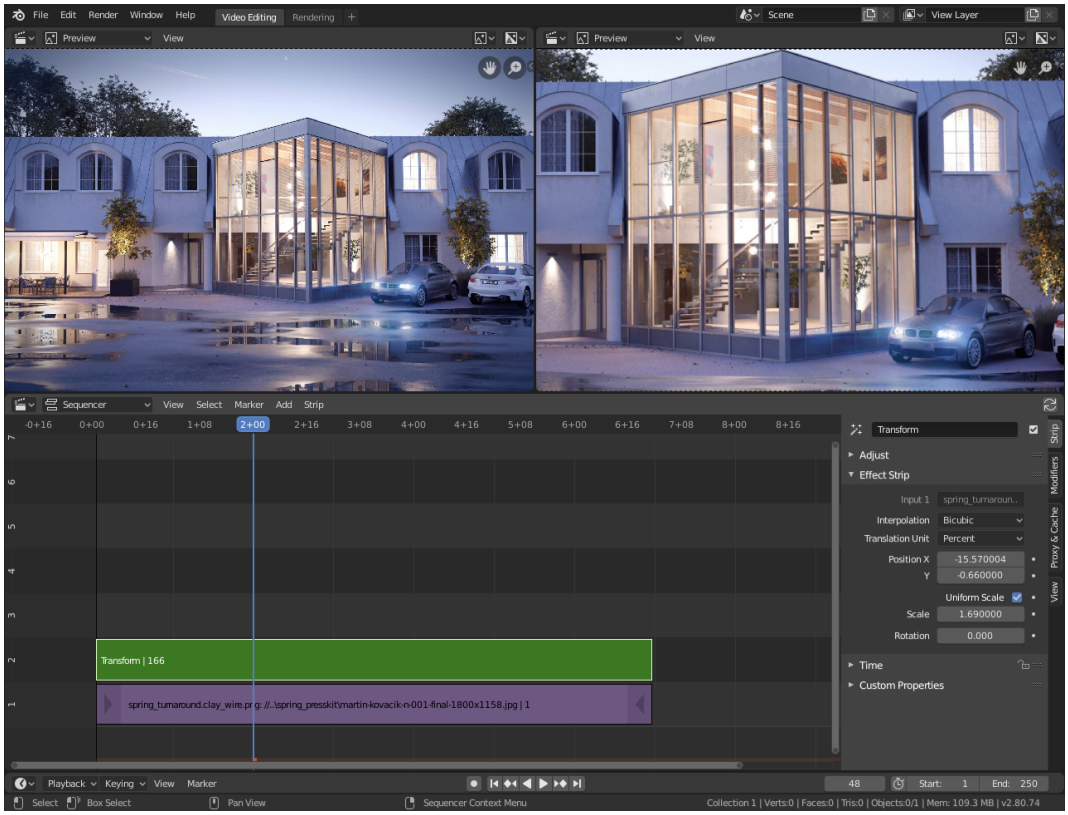
Scale

Scale the image on the X and Y axis.

Rotation

Rotates the input two-dimensionally along the Z axis.

Example



Transform Effect.

Speed Control Strip

Speed Control time-warps the strip, making it play faster or slower than it normally would. Playing faster means that some frames are skipped, and the strip will run out of frames before the end frame. When the strip runs out of frames to display, it will just keep repeating the last one action that will appear to be frozen. To avoid this, position the next strip under the original at a point where you want the motion to continue.

Options

Speed Control
: The method used to adjust the speed of the strip.

Stretch:: Automatically calculates the speed effect based on the length of the input strip. If you scale a strip to 1/2 the original size the sequence will play back at 2 times the speed.

Multiply:: Multiplies the current speed of the sequence by a *Multiply Factor*. Thus a value of 0.5 will make the sequence half as fast while 2 would make the sequence twice as fast. Negative values will reverse the input while also adjusting the speed, so a value of negative two will play in reverse and twice as fast as normal.

> **Note**
> You will have to manually re-adjust the length of the strip accordingly.

Frame Number:: Specifies a frame to remap the current frame to, for example, setting the *Frame Number* value to 50 displays the 50th frame. This can then be manually keyframed to recreate the animation.

Length:: Maps the frame range on a percentage scale. For example, using this and a value of 50% will select the frame halfway through the sequence.

Interpolation
: Crossfades between frames to reduce screen tearing when the speed is slower than the original frame rate.

Examples

Creating a Slow-Motion Effect

Suppose you want to slow your strip down. You need to affect the speed of the video clip without affecting the overall frame rate. Select the clip and *Add ▸ Effect ▸ Speed Control* effect strip.

Choose the *Multiply* option in the Effect Strip panel in the Sidebar. Set the Multiply Factor to be the factor by which you want to adjust the speed. To cut the displayed speed by 50%, enter 0.5. Now, a 275-frame clip will play at half speed, and thus display only the first 137 frames.

If you want the remaining frames to show in slow motion after the first set is displayed, double the Length of the source strip (since effects strip bounds are controlled by their source strips). If you are using a speed factor other than 0.5 then use the formula:

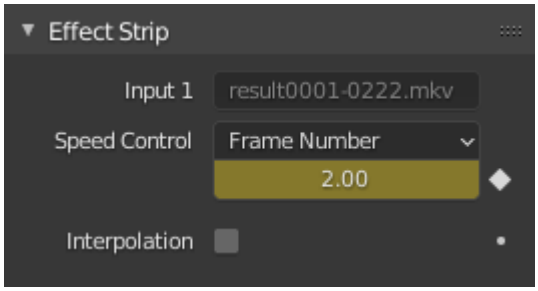
```
new_length = real_length / speed_factor
```

That is it, set your render to animate (in this example) all 550 frames.

Keyframing the Speed Control

To get even finer control over your clip timing, you can use curves! While it is possible to keyframe the Multiply factor, usually you want to keyframe the Frame number directly.

Choose the *Frame Number* option. You now have a Frame number field which you can keyframe. If you want the strip to animate **at all** you will have to insert some keyframes, otherwise it will look like a still. In most cases you will want to use the Graph editor view to set the curve interpolation to Linear since the default Bézier will rarely be what you want.



Keyframing the Frame number.

Tip

If you choose to keyframe the Speed factor instead, remember to Refresh All or the changes will not take effect.

Changing Video Frame Rates

You can use the speed control to change the frame rate in frames per second (fps) of a video. If you are rendering your video to a sequence set, you can effectively increase or decrease the number of individual image files created, by using a Multiply value less than or greater than one, respectively.

For example, if you captured a five-minute video at 30 fps and wanted to transfer that to film, which runs at 24 fps, you would enter a Multiply Factor of 30/24, or 1.25 (and enable *Interpolation* frame blending to create a film blur effect). Instead of producing `5 × 60 × 30 = 9000` frames, Blender would produce `9000 / 1.25 = 7200 = 5 × 60 × 24` frames. In this case, you set a *start* = 1 and *end* = 7200, set your Format output to `jpeg` 30fps, and image files `0001.jpg` through `7200.jpg` would be rendered out, but those images cover the entire 9000 frames. The image file `7200.jpg` is the same at frame 9000. When you read those images back into your film blend-file at 24 fps, the strip will last exactly 5 minutes.

Glow Strip

This effect makes parts of an image glow brighter by working on the luminance channel of an image. The *Glow* is the superposition of the base image and a modified version, where bright areas are blurred.

To “animate” the glow effect, mix it with the base image using the Gamma Cross effect, crossing from the base image to the glowing one.

Options

Threshold
: Areas brighter than the *Threshold* are blurred.

Clamp
: The maximum luminosity that is added.

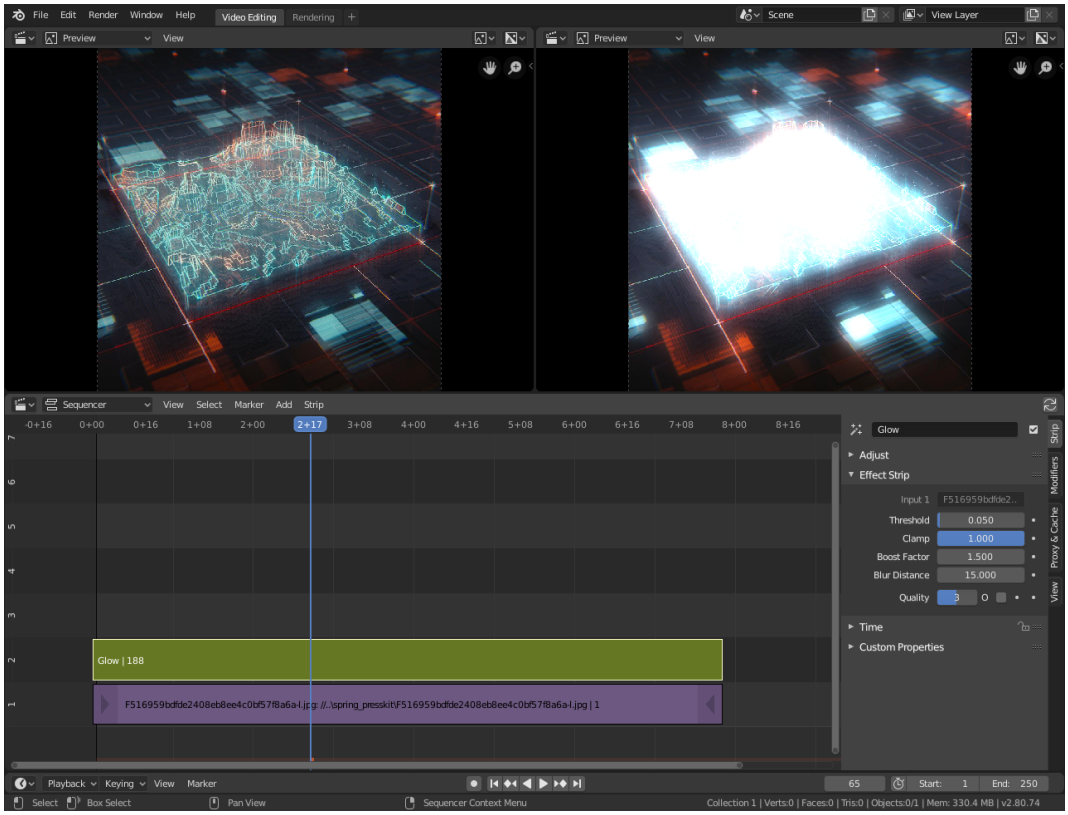
Boost Factor
: Multiplier of the brightness.

Blur Distance
: The size of the blur.

Quality
: Improves the quality of the glow by giving smoother results but will be slower.

Only Boost
: This checkbox allows you to only show/use the “modified” version of the image, without the base one.

Example


File Edit Render Window Help
Video Editing
Rendering
Scene
View Layer
Preview
View
Sequencer
Select
Marker
Add
Strip
Glow | 188
F516959bdfde2408eb8ee4c0bf57f8a6a-l.jpg | 1
Glow
Adjust
Effect Strip
Threshold 0.050
Clamp 1.000
Boost Factor 1.500
Blur Distance 15.000
Quality
Time
Custom Properties
Playback
Keying
Marker
Start: 1
End: 250
Select
Box Select
Pan View
Sequencer Context Menu

Glow effect.

Gaussian Blur Strip

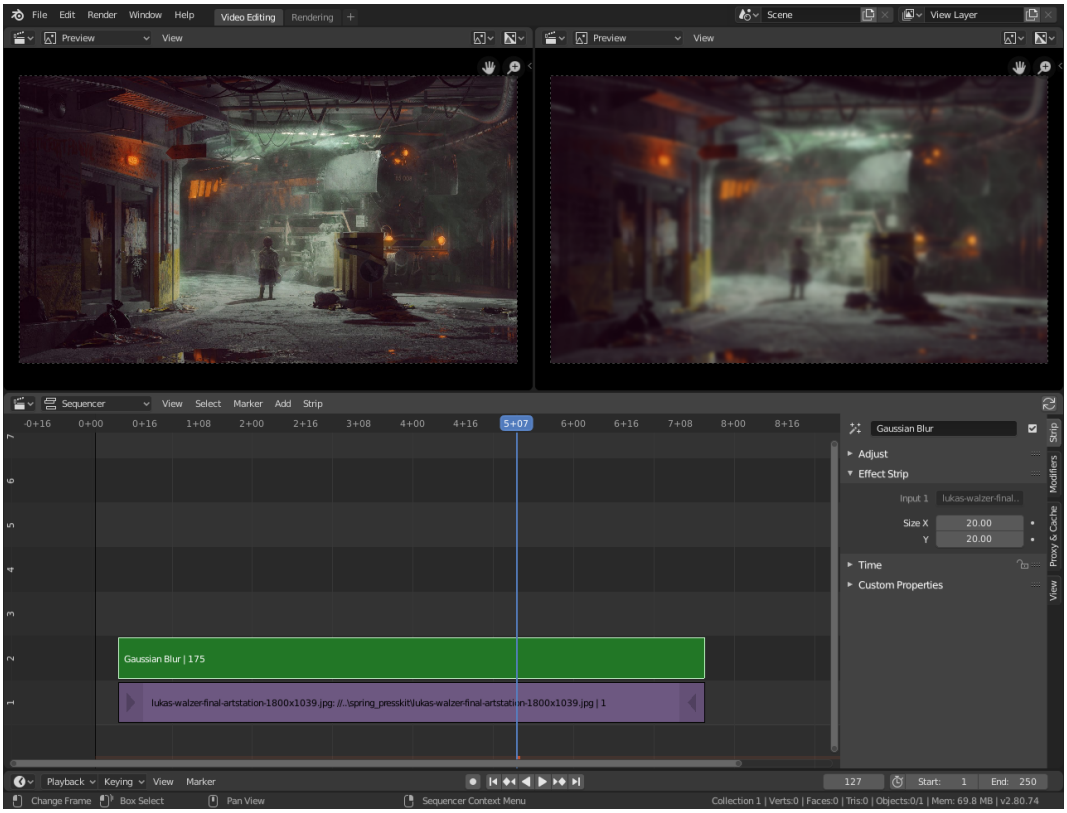
The Gaussian Blur strip is used to blur the input strip in a defined direction. This can be used to blur a background or to blur a transition strip.

Options

Size X
: Distance of the blur effect on the X axis.

Size Y
: Distance of the blur effect on the Y axis.

Example



Gaussian Blur Effect.

Transitions

- Sound Crossfade
- Cross Strip
- Gamma Cross Strip
- Wipe Strip

Sound Crossfade

The *Sound Crossfade* transition works by animating the *Volume* of two overlapping Sound strips to evenly fade between them. Because this simply animates a value it does not create a strip like other effects or transitions.

Cross Strip

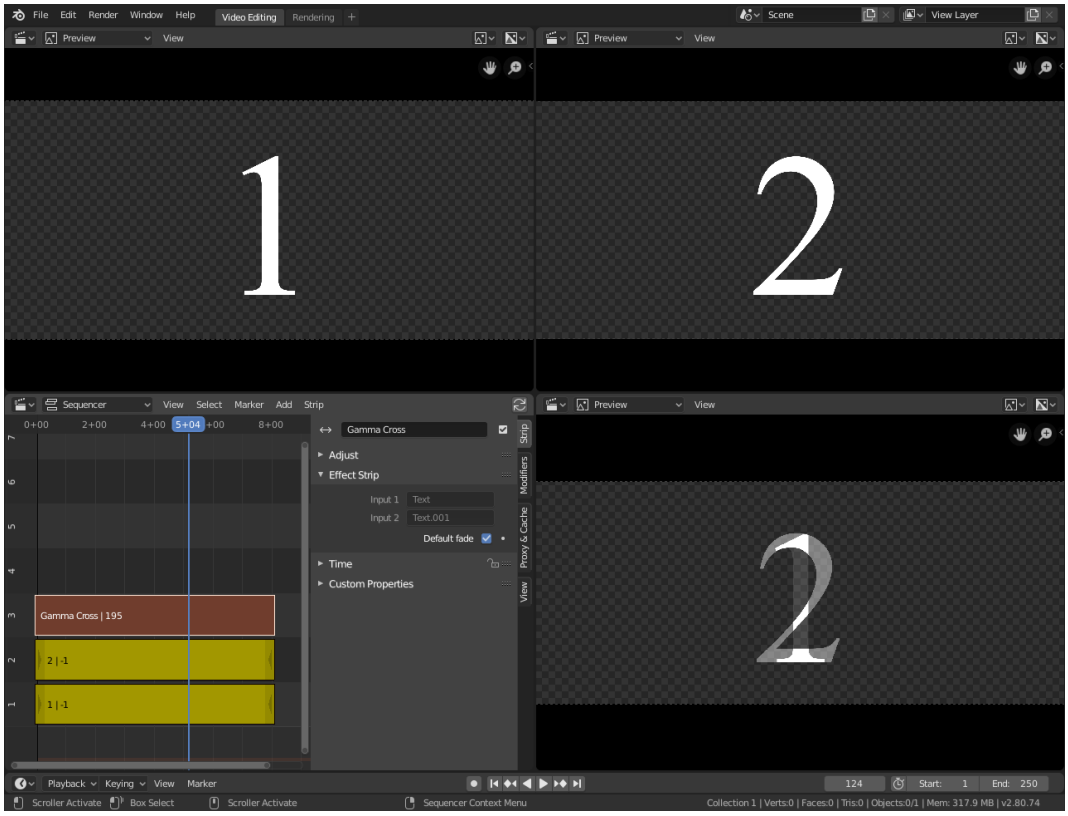
The *Cross* transition fades from one strip to another, also known as a crossfade. Strips can be overlapping or have a gap between them, however, when strips contain a gap the last and first frame of each strip is extend which can cause a pause if any of the strips are a sequence.

Options

Default Fade
: Automatically calculates a linear fade over the length of the strip.

Effect Fader
: Allows you to manually keyframe a custom fade. This can be used with different easings to fine-tune the fade in/out.

Example



Cross Effect.

Gamma Cross Strip

The Gamma Cross transition is similar to the Cross Strip transition, however, the Gamma Cross strip transition uses color correction while transitioning between the two strips, resulting in a smoother transition that is easier on the eyes.

Options

Default Fade
: Automatically calculates a linear fade over the length of the strip.

Effect Fader
: Allows you to manually keyframe a custom fade. This can be used with different easings to fine-tune the fade in/out.

Wipe Strip

The *Wipe* transition strip can be used to transition from one strip to the next. The wipe will have no effect if created from a single strip instead of two strips. The duration of the wipe is the intersection of the two source strips and cannot be adjusted. To adjust the start and end of the wipe you must adjust the temporal bounds of the source strips in a way that alters their intersection.

Options

Transition
: The type of transition used.

  Single
  : Reveals the next strip by uncovering it in a straight line moving across the image.

  Double
  : Similar to *Single*, but uses two lines either starting from the middle of the image or the outside. Like the blink of an eye.

  Iris
  : Reveals the next strip through an expanding (or contracting) circle. Like the aperture of a camera or pupil of an eye. You can blur the transition, so it looks like ink bleeding through a paper.

  Clock
  : Like the hands of an analog clock, it sweeps clockwise or (if Wipe In is enabled) counterclockwise from the 9:00 position. As it sweeps, it reveals the next strip.

Direction
: Controls whether to fade *In* or *Out*.

Blur Width
: The width of the blur used to blur the transition.

Angle
: Controls the angle of the line for *Single* and *Double* transition types.

Default Fade
: Automatically calculates a linear fade over the length of the strip.

  Effect Fader
  : Allows you to manually keyframe a custom fade. This can be used with different easings to fine-tune the fade in/out.

Example



Wipe Effect.

Selecting

The active sequence strip is displayed with a light outline. The *entire* strip could be selected by clicking `LMB` in the middle of the strip.

Select Menu

The Select menu lets you select strips in different ways.

All `A`
: Selects all the strips in the timeline.

None `Alt-A`
: Deselects all the strips in the timeline.

Invert `Ctrl-I`
: Inverts the current selection.

Box Select `B`
: See Box Select.

Box Select (Include Handles) `Ctrl-B`
: Works like *Box Select*, but also selects any strip handles inside the box. If a strip has only one handle selected, dragging it will change the strip's length. (If both handles are selected, the complete strip moves instead.)

Select Grouped `Shift-G`
: Select strips that are similar to the active strip. By default, unsimilar strips are deselected, but this can be changed in the Adjust Last Operation region.

  Type
  : Select strips that have the same specific type as the active strip. For example, if the active strip is a Movie strip, this selects all Movie strips.

Global Type
: Select strips that have the same general type (graphics or audio) as the active strip.

Effect Type
: If the active strip is an effect strip, selects all effect strips. Otherwise, selects all non-effect strips. (Despite the name, this operator does not check the effect type.)

Data
: Select strips that use the same source (file, scene, movie clip or mask) as the active strip.

Effect
: Find the effect types that are applied to the active strip, and select all strips that have any of the same effect types applied to them. For example, if the active strip has a Gaussian Blur effect on it, this will select all other strips that are also blurred.

Effect/Linked
: Select strips that are on a lower channel than a selected strip and overlap it in time; then, effect strips linked to the selected content strips; and finally, content strips linked to the selected effect strips.

Overlap
: Select strips that partially or completely overlap the active strip in time.

Select Linked

All `Ctrl-L` / Less `Ctrl-NumpadMinus` / More `Ctrl-NumpadPlus`
: Add/remove neighboring strips to/from the selection.

Side of Frame

Left/Right `[/]`
: Select the strips that lie completely to the left or right of the current frame.

Current

Select the strips that intersect the current frame.

Handle

Both, Left, Right

Select the left, right, or both handles of the selected strips.

Both/Left/Right Neighbor

Select the handle of the neighboring strip to the left, right, or on both sides of the selected strips.

Channel

Select all the strips that are in the same channels as the currently selected strips.

Editing

Transform

Options

Overlap Mode

Overlap Mode defines the result of transforming a strip so that it overlaps another strip.

Shuffle
: The overlapping strip will be moved to the nearest free space so that it does not overlap.

Overwrite
: The overlapped strip will be overwritten, trimmed or split by the overlapping strip.

Expand
: All strips on the right side of (each) transformed will be shifted forward to accommodate the overlapping strip.

Snapping

The icon toggles snapping; you can also do this temporarily by holding `Ctrl` after starting to drag a strip.

The drop-down arrow offers the following options:

Snap to
: Current Frame
  : Snaps the transformed selection to the Playhead.

Hold Offset
: Snaps the transformed selection to the Hold Offset.

Markers
: Snaps the transformed selection to Markers.

Ignore
: Muted Strips
: Muted Strips are not considered as snap targets.

Sound Strips
: Sound Strips are not considered as snap targets.

Current Frame
: Snap to Strips
: Snaps the Playhead to all strips.

Move

Reference

Menu:: *Strip ▸ Transform ▸ Move*
Shortcut:: `G`

Pressing `G` moves all the selected strip(s). Move your mouse horizontally (left/right) to change the strip's position in time. Move vertically (up/down) to change channels.

Holding down `Ctrl` while dragging enables or disables snapping.

You can also lock the direction to time with `X` or to change the strip's channel with `Y`.

It is possible to move strips using mouse by dragging them while holding `LMB`. Currently it is possible to move only one strip by dragging.

Start Frame Offset

The *Start Frame Offset* for that strip can be selected by clicking LMB on the left handle of the strip; holding it down (or pressing G and then moving the mouse left/right) changes the start frame within the strip by the number of frames you move it. The frame number label under the strip displays the start frame of the strip.

- If you have a 20-image sequence strip, and drag the left handle to the right by 10 frames, the strip will start at image 11 (images 1 to 10 will be skipped). Use this to clip off a roll-up or undesired lead-in.
- Dragging the left handle left will create a lead-in (copies) of the first frame for as many frames as you drag it. Use this when you want some frames for a transition at the start of the clip.

End Frame

The *End Frame* of the strip could be selected by clicking LMB on the right handle of the strip; holding it down (or pressing G) and then moving the mouse changes the ending frame within the strip. The frame number label over the strip displays the end frame of the strip.

- Dragging the right handle to the left shortens the clip; any original images at the tail are ignored. Use this to quickly clip off a roll-down.
- Dragging the right handle to the right extends the clip. For movies and images sequences, more of the animation is used until exhausted. Extending a clip beyond its length will render as a copy of the last image. Use this for transitions out of this clip.

Note

Multiple selection

You can select several (handles of) strips by Shift–LMB clicking: when you press G, everything that is selected will move with your mouse – this means that, for example, you can at the same time move a strip, shorten two others, and extend a forth one.

Move/Extend from Current Frame

Reference

Menu:: *Strip ▸ Transform ▸ Move/Extend from Current Frame*
Shortcut:: E

With a number of strips selected, pressing E lets you interactively extend the strips. This is similar to moving but is useful for extending (or shortening) time around the current frame.

All selected strip handles to the "mouse side" of the current frame indicator will transform together, so you can change the duration of strips at the current frame.

Hint

Extend is a convenient way to adjust the time of rough edits such as an "animatic" (sequential storyboards). Where it's possible to select everything and adjust the length of strips around the current frame. This can be especially useful when adding in audio or other elements that could cause the timing to need adjustment.

When performing this operation you may want to enable *Markers ▸ Sync Markers* so markers are updated too.

This simply a convenience operation, instead of manually selecting strips on one side of the current frame, as well as handles on one side of overlapping strips. Then selecting and transforming markers as well. This avoids the manual process, so re-timing can be accessed quickly.

Slip Strip Contents

Reference

Menu:: *Strip ▸ Transform ▸ Slip Strip Contents*
Shortcut:: `S`

The Slip tool allows you to change the position of the contents of a strip without moving the strip itself.

Snap Strips to the Current Frame

Reference

Menu:: *Strip ▸ Transform ▸ Snap Strips to the Current Frame*
Shortcut:: `Shift-S`

Moves the strip or control point to the current frame.

Clear Strips Offset

Reference

Menu:: *Strip ▸ Transform ▸ Clear Strips Offset*
Shortcut:: `Alt-O`

To reset the (soft) start/end frame handles.

Swap Strips

Reference

Menu:: *Strip ▸ Transform ▸ Swap Strips*

Left `Alt-Left`
: Swaps the active strip with the strip to the left.

Right `Alt-Right`

Swaps the active strip with the strip to the right.

Remove Gaps

Reference

Menu:: *Strip ▸ Transform ▸ Insert Gaps*
Shortcut:: `Backspace`

Remove blank frames between the current frame and the first strip to the left, independent of selection or locked state of strips.

All Gaps
Remove gaps to the right of the strip along with the left.

Insert Gaps

Reference

Menu:: *Strip ▸ Transform ▸ Insert Gaps*
Shortcut:: `Equals`

Insert blank frames between the current frame and the first strips to the right, independent of selection or locked state of strips.

Image Transform

Scale to Fit

Reference

Menu:: *Strip ▸ Image Transform ▸ Scale to Fit*

Adjusts the strips Scale Transforms so the visual contents of the strip to fit exactly within the project's Resolution while maintaining the original aspect

ratio.

This may mean that the transparent areas may be added along the content's border to fit the content in the rendered area.

Scale to Fill

Reference

Menu:: *Strip ▸ Image Transform ▸ Scale to Fill*

Adjusts the strips Scale Transforms so the visual contents of the strip to span the project's Resolution while maintaining the original aspect ratio.

This may mean that portions of the original image no longer fit the content inside the rendered area.

Stretch to Fill

Reference

Menu:: *Strip ▸ Image Transform ▸ Stretch to Fill*

Adjusts the strips Scale Transforms so the visual contents of the strip to fill the project's Resolution. Note, unlike the other two methods described above, *Stretch to Fill* does not maintaining the original aspect ratio.

This may mean that the original image becomes distorted to fit the content inside the rendered area.

Clear Position

Reference

Menu:: *Strip ▸ Image Transform ▸ Clear Position*

Resets the strips Position Transforms to a value of zero.

Clear Scale

Reference

Menu:: *Strip ▸ Image Transform ▸ Clear Scale*

Resets the strips Scale Transforms to a value of one.

Clear Rotation

Reference

Menu:: *Strip ▸ Image Transform ▸ Clear Rotation*

Resets the strips Rotation Transform to a value of zero.

Clear All

Reference

Menu:: *Strip ▸ Image Transform ▸ Clear All*

Resets the strips position, scale, and rotation Transforms to their default values.

Split

Reference

Menu:: *Strip ▸ Split*
Shortcut:: K

This splits the selected strip in two at the current frame. This will result in two strips which use the same source, fitting the original strip's timing and length.

Hint

This can be thought of as a quick way to duplicate the current strip, adjusting the start/end frames to form two non-overlapping strips showing the same content as before.

Hold Split

Reference

Menu:: *Strip ▸ Hold Split*
Shortcut:: `Shift-K`

Like *Split*, it splits a strip in two distinct strips; however you will not be able to drag the endpoints to show the frames past the split of each resulting strip.

Although you can adjust the Hold Offset number fields in the *Strip Info* panel.

Hint

This can be thought of as a way to simulate splitting the video file in two parts at the cut-point, replacing the current strip with each.

Duplicate Strips

Reference

Menu:: *Strip ▸ Duplicate Strips*
Shortcut:: `Shift-D`

Duplicate a strip to make an unlinked copy; drag it to a time and channel, and drop it by `LMB` click.

Delete

Reference

Menu:: *Strip ▸ Delete*
Shortcut:: `Delete`, `X`

Delete the selected strip(s).

Update Scene Frame Range

Reference

Menu:: *Strip ▸ Update Scene Frame Range*

For Scene strips only – Updates the strip's Time properties to match the referenced scene's frame range. This operator should be used when the referenced scene's length is extended or shortened.

Separate Images

Reference

Menu:: *Strip ▸ Separate Images*
Shortcut:: `Y`

For images sequence only – Converts the strip into multiple strips, one strip for each frame. Useful for slide shows and other cases where you want to

bring in a set on non-continuous images.

Length
: You have to specify the duration you want the resulting strips will be.

Movie Strip

Set Render Size

Reference

Menu:: *Strip ▸ Set Render Size*

Sets the render resolution and aspect to match the strip's resolution.

Deinterlace Movies

Reference

Menu:: *Strip ▸ Deinterlace Movies*

Converts interlaced video into progressive video.

Effect Strip

Change Effect Input

Reference

Menu:: *Strip ▸ Effect Strip ▸ Change Effect Type*

Swaps which strips are the input for the effect strip.

Change Effect Type

Reference

Menu:: *Strip ▸ Effect Strip ▸ Change Effect Type*

Switch the effects on a selected Effect strip.

Reassign Inputs

Reference

Menu:: *Strip ▸ Effect Strip ▸ Reassign Inputs*
Shortcut:: `R`

This tool can be used to assign (reconnect) effect strips in a different way. Select three arbitrary strips and press `R`. If you don't create a cycle, those will be connected to a new effect chain.

Swap Inputs

Reference

Menu:: *Strip ▸ Effect Strip ▸ Swap Inputs*
Shortcut:: `Alt-S`

Swaps the first two inputs for the effect strip.

Lock/Unlock

Lock Strips `Shift-L`
: Disables the strip from being transformed.

Unlock Strips `Shift-Alt-L`
: Enables disabled strips allowing them to be transformed.

Mute/Unmute

Mute/Unmute Strips `H`, `Alt-H`
: Mute or unmute the selected strips.

Mute/Unmute Deselected Strips `Shift-H`, `Ctrl-Alt-H`
: Mute or unmute all strips but the selected.

Inputs

Reload Strips `Alt-R`
: Reloads the strips from their external saved location.

Reload Strips and Adjust Length `Shift-Alt-R`
: Reloads the strips from their external saved location and re-adjusts the strip duration.

Change Path/Files
: Changes the source file contained in a selected strip.

Swap Data
: Swaps two sequence strips.

Context Menu

You can activate context menu by clicking `RMB` in the Sequencer's timeline. In this menu you can quickly access some commonly used tools.

Fades

Reference

Menu:: *Add ▸ Fades*

This submenu contains tools to add or remove fades to strips. In case of visual strips the tools will animate the opacity or volume in case of audio strips.

Clear Fades
: Removes fade animation from selected sequences.

Fade In and Out
: Fade selected strips in and out.

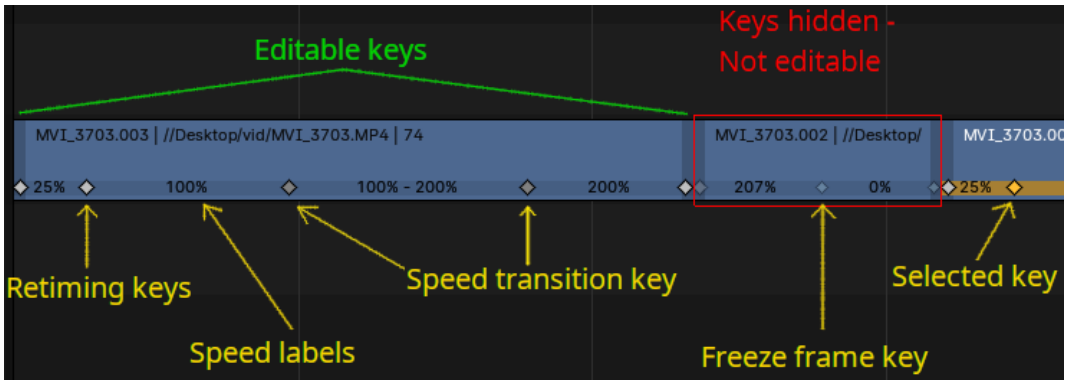
Fade In
: Fade in selected strips.

Fade Out
: Fade out selected strips.

From Current Frame
: Fade from the current frame to the end of overlapping sequences.

To Current Frame
: Fade from the start of sequences under the Playhead to the current frame.

Retiming



Strips can be sped up or, slowed down by adding and moving retiming keys. Retiming controls can be activated for individual strips, after which keys can be selected and moved.

Note

- Only strip content is retimed, existing animation is not handled by the tool.
- Effect strips can not be retimed.

Hint

To quickly change selected strip speed, press `R` and enter desired speed.

Selecting Retiming Keys

Retiming keys are always shown on strip as inactive by default. In order to select retiming keys, Show Retiming Keys must be enabled. This property can also be enabled using Toggle Retiming Keys.

Multiple keys can be selected at once with box selection. Box select will select keys, only if a key is already selected. Otherwise it will select strips only.

Use `Ctrl-LMB` to select all keys to the right of the selected key.

Moving Retiming Keys

Retiming key can be moved by dragging it with mouse, or by pressing `G`. The key is mapped to particular frame of strip content, so moving it effectively means moving a frame to a new position and therefore stretching, or contracting time flow.

When a key is moved, this does not affect position of other keys inside of the strip. If strip has more keys inside, multiple keys have to be selected, if only 1 segment has to be retimed. However if there are retiming keys outside of strip boundary, these will be moved along with first or last key in strip in order to preserve existing retiming, that is not visible.

Add Retiming Keys

Reference

Editor:: Video Sequencer
Menu:: *Strip ▸ Retiming ▸ Add Retiming Key*

Retiming key can be added to selected strips from retiming menu or by pressing `I` and choosing Add Retiming Key option. This adds a key to current frame. This operation will also create keys at strip start and end point, since these keys must be always present.

When keys are selected, strips are deselected, but it is still possible to add new keys. In this case keys will be added to strips where any key is selected.

Add Freeze Frame and Slide

Reference

Editor:: Video Sequencer
Menu:: *Strip ▸ Retiming ▸ Add Freeze Frame and Slide*

Freeze frame is used to stop strip playback at particular frame for any duration. Freeze frame can be added from strip retiming menu or context menu.

Note

It is not possible to make smooth transition into or from freeze frame.

Add Speed Transition and Slide

Reference

Editor:: Video Sequencer
Menu::

Strip ▸ Retiming ▸ Add Speed Transition and Slide

It is possible to create smooth transition from one speed to another speed. This can be done by selecting retiming key between 2 segments of different speeds, and choosing Add Speed Transition either from strip retiming menu or context menu. This will create 2 keys, that are linked and always move in opposite direction. If both keys are moved at once, this changes where transition starts and ends.

Delete Retiming Keys

Reference

Editor:: Video Sequencer
Menu:: *Strip ▸ Retiming ▸ Delete Retiming Keys*

Retiming key can be deleted by selecting and pressing `Delete` or `X`. When handle is deleted, strip size will not change, therefore speed will change to average between 2 retimed segments.

Note

When transition key is removed, it will re-create simple retiming key from which transition was created.

Reset Retiming

Reference

Editor:: Video Sequencer
Menu:: *Strip ▸ Retiming ▸ Reset Retiming*

Reverts all timing to the original strip.

Set Speed

Reference

Editor::	Video Sequencer
Menu::	*Strip ▸ Retiming ▸ Set Speed*

Sets the speed of a retimed segment.

Speed
: The rate compared to the original time.

Preserve Current retiming
: Keeps the speed of the other retiming segments unchanged by adjusting the Duration of the strip instead.

Toggle Retiming Keys

Reference

Editor::	Video Sequencer
Menu::	*Strip ▸ Retiming ▸ Toggle Retiming Keys*
Shortcut::	`Ctrl-R`

Enable's the Show Retiming Keys strip property. This allows retiming keys to be shown for the first time and enables interacting with them.

Meta Strips

A Meta Strip is a strip which contain multiple strips treated as if it was one strip. It allows you to reduce the vertical space used in the Sequencer. You can edit it the same way as any other strips.

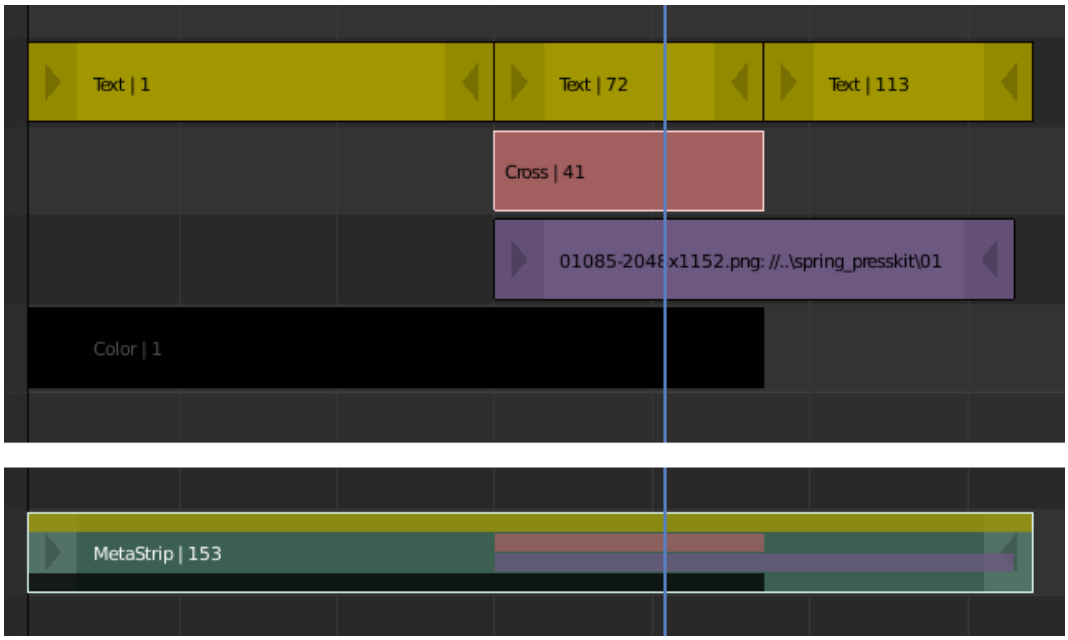
It is organization tool. For example, if you are using a lot of strips with complicated arrangement, you can group them together using Meta strips.

Make Meta Strip `Ctrl-G`

To create a Meta strip, select all the strips you want to group, and `Ctrl-G` to group them. The Meta strips will span from the beginning of the first strip to the end of the last one, and condenses all channels into a single strip.

UnMeta Strip `Ctrl-Alt-G`

Separating (ungrouping) the Meta strip restores the strips to their relative positions and channels. This can be used if you choose to delete a Meta strip and want to keep the strips inside.



Example of Meta strips.

You can edit the content inside a Meta strip by pressing `Tab`. It will expand the strip to the whole view and hide any other strips. To exit the Meta strip press `Tab` again. Meta strips can also be nested, which make editing them a little confusing. To exit out one level of Meta Strip make sure you do not have a Meta strips selected when you press `Tab`.

Note

The default blend mode for a Meta strip is Replace. There are many cases where this alters the results of the animation so be sure to check the results and adjust the blend mode if necessary.

One convenient use for Meta strips is when you want to apply the same effect to multiple strips. For example: if you have a video that was recorded in different files and want to add an effect strip. It is much more convenient to apply a single set of effects to one Meta strip than applying it to each individual strip.

See also

It is also possible to do the similar task described above with an Adjustment Layer effect strip.

Assets, Files, & Data System

- Introduction
  - Outliner
- Blender File
  - Opening & Saving
  - Compatibility
  - Packed Data
  - Blend-Files Previews
  - Rename
- Data-Blocks
  - Data-Block Types
  - Life Time
  - Name & Rename
  - Sharing
  - Making Single User
  - Removing Data-Blocks
- Custom Properties
  - Editing Properties
- Linked Libraries
  - Link & Append
  - Library Overrides
- Asset Libraries
  - Introduction
  - Asset Catalogs
- Media Formats
  - Supported Graphics Formats
  - Supported Video & Audio Formats
- Importing & Exporting Files
  - Alembic
  - Collada (Legacy)
  - Universal Scene Description
  - Wavefront OBJ
  - Stanford PLY
  - STL

- Import/Export SVG as Grease Pencil
- Export Grease Pencil as (PDF)

Introduction

Each blend-file contains a database. This database contains all scenes, objects, meshes, textures, etc. that are in the file.

A file can contain multiple Scenes and each scene can contain multiple Objects. Objects can contain multiple materials which can contain many textures. It is also possible to create links between different objects, or share data between objects. A file can link data from other Blender files.

Outliner

You can easily inspect the contents of your file by using the *Outliner* editor, which displays all of the data in your blend-file.

The *Outliner* allows you to do simple operations on objects, such as selecting, renaming, deleting, linking and parenting.

Read more about the Outliner.

Blender File

- Opening & Saving
  - Opening Files
  - Open Recent
  - Recover
  - Saving Files
  - Save Incremental
  - Save As
  - Save Copy
  - Relative Paths
- Compatibility
  - Backward Compatibility
  - Forward Compatibility
- Packed Data
  - Pack Data
  - Unpack Data
  - Pack Linked Libraries
  - Unpack Linked Libraries
- Blend-Files Previews
  - Blend-File Preview
  - Data-Blocks Previews
- Rename
  - Rename Active Item
  - Batch Rename

Opening & Saving

Opening and saving blend-files is usually done using the File Browser.

Tip

Blend-files can also be opened by dragging and dropping blend-files into the Blender window. This method also allows to link/append the file.

Note

Unsaved Changes

By default, when exiting Blender or loading a new blend-file, if you have unsaved changes, a pop-up will ask you to either confirm discarding those changes, or save them.

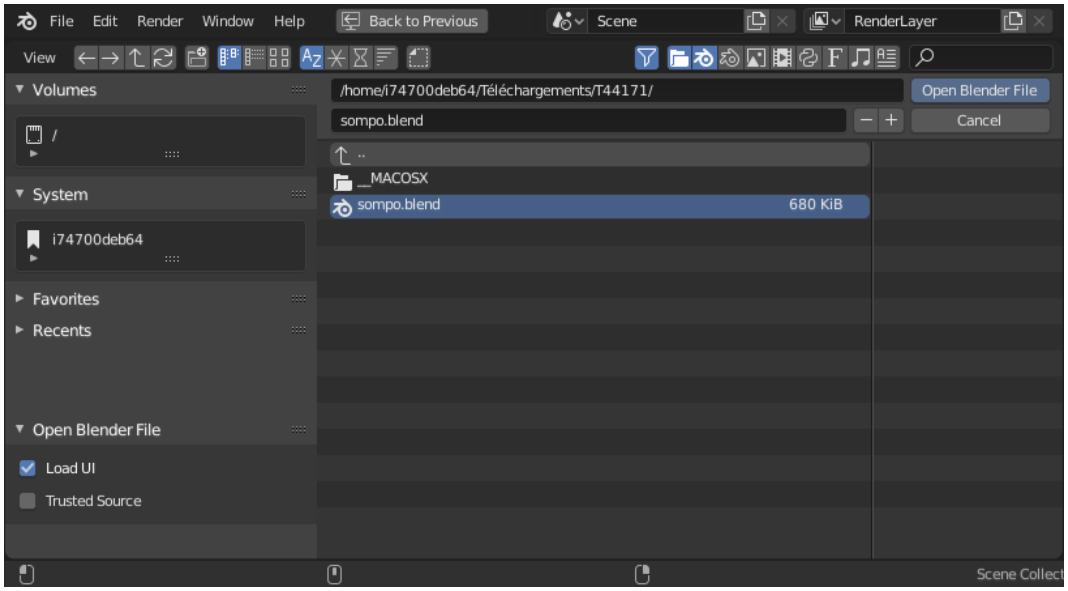
This behavior can be disabled with the *Save Prompt* option in the Save & Load section of the *Preferences*.

Opening Files

Reference

Menu:: *File* ▸ *Open…*
Shortcut:: `Ctrl-O`

The upper text field displays the current directory path, and the lower text field contains the selected filename.



The File Browser in open configuration.

Options

Load UI

When enabled, the screen layout saved inside each blend-file is used, replacing the current layout and Workspaces. Otherwise the file screen layout is ignored.

Tip

If you want to work on a blend-file using your own defaults, start a fresh Blender, open the File Browser and turn off the *Load UI* button, and open the desired file.

Trusted Source

When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See Python Security to configure default trust options.

Open Recent

Reference

Menu:: *File ▸ Open Recent*
Shortcut:: `Shift-Ctrl-O`

Displays a list of recently opened blend-files. Hovering over items will show a preview, and information about the blend-file. Select any of the file names in the list to open that blend-file. When `RMB` on a listed item, a context menu will appear; One of the available options is *Open File Location*, which will open that location in an OS file explorer or Finder window.

Clear Recent Files List

Removes items from the recent files list.

Remove
: Choose which type of items to remove.

All Items:: Removes all recent files.
Items Not Found:: Removes files from the list that cannot be found i.e. that have been moved or deleted.

Recover

Last Session

Reference

Menu:: *File ▸ Recover ▸ Last Session*

This will load the `quit.blend` file Blender automatically saved just before exiting. This option enables you to recover your last work session if, for

example, you closed Blender by accident.

Auto Save

Reference

Menu:: *File ▸ Recover ▸ Auto Save*

This will allow you to open an automatically saved file to recover it.

See also

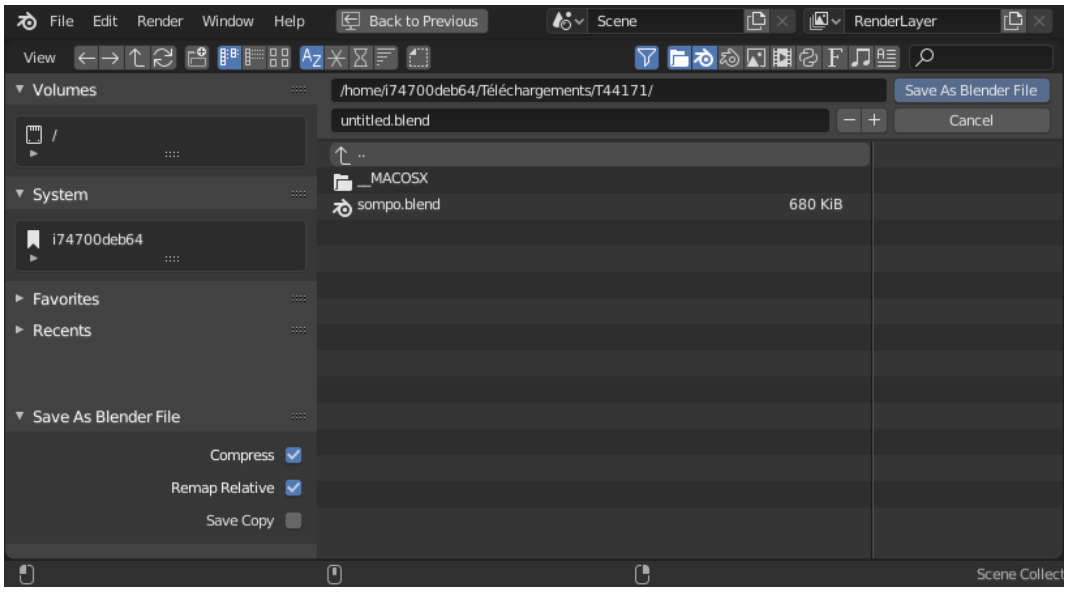
Auto Save

Saving Files

Reference

Menu:: *File ▸ Save*
Shortcut:: `Ctrl-S`

Save current blend-file over itself (if it was not saved yet, this will automatically switch to *Save As…*).



The File Browser in save configuration.

Save Incremental

Reference

Menu:: *File ▸ Save Incremental*
Shortcut:: `Ctrl-Alt-S`

Save the current Blender file with a numerically incremented name that does not overwrite any existing files.

Save As

Reference

Menu:: *File ▸ Save As…*
Shortcut:: `Shift-Ctrl-S`

Choose a file path to save the blend-file to.

Warning

If a file with the same given name already exists, the text field will turn red as a warning that the file will be overwritten.

Tip

Use the *plus* or *minus* buttons to the right of the file name, or `NumpadPlus`, `NumpadMinus` to increase/decrease a number at the end of the file name (e.g. changing `file_01.blend` to `file_02.blend`).

Options

Compress
: Reduces the file size of the resulting blend-file but takes longer to save and load. This option is useful for distributing files online and saving drive space for large projects. But it can cause slowdowns when quitting Blender, or under normal operation when auto-saving backup files. See Compression & Memory Use for more information.

  Hint

  The used compression algorithm is Zstandard. It is not unique to Blender so files can be compressed/decompressed with external tools.

  Changed in version 3.0: Prior to this version, the compression algorithm used was Gzip. This means to open newer blend-files in versions prior to 3.0, blend-files must first be saved without compression in a newer version of Blender or decompressed using an external Gzip tool.

Remap Relative
: This option remaps Relative Paths (such as linked libraries and images) when saving a file in a new location.

Save Copy

Saves a copy of the actual working state but does not make the saved file active.

Save Copy

Reference

Menu:: *File* ▸ *Save Copy…*

Choose a file path to save the blend-file to, but return to editing the original file upon completion. This can be used to save backups of the current working state without modifying the original file.

For options see Save As.

Relative Paths

Many blend-files reference external images or other linked blend-files. A path tells Blender where to look for these files. If the external files are moved, the blend-file that references them will not look right.

When you specify one of these external files, the default option is to make the path relative. Blender stores a partial path evaluated relative to the directory location of the referencing blend-file. This choice helps when you need to reorganize folders or move your files.

With a relative path, you can move the blend-file to a new location provided the externally linked files are moved along with it. For example, you could send someone a folder that contains a blend-file and a subfolder of external images that it references.

When relative paths are supported, the File Browser provides a *Relative Path* checkbox, when entering the path into a text field, use a double slash prefix (`//`) to make it so.

Relative paths are the default but this can be changed in the File tab of the *Preferences*.

Note

You cannot use relative paths into a new *untitled* blend-file. Save it before linking to external files.

Hint

If it is necessary to relocate a blend-file relative to its linked resources, use Blender's File Save As function which has an option to *Remap Relative* file links.

Compatibility

Blender can open blend-files saved with both older versions of the software (backward compatibility), and newer ones (forward compatibility). This comes with some limitations though.

Tip

When having issues with opening much older (or newer) blend-files, it can help to use a few intermediary Blender releases to perform conversions by smaller steps.

Note

Here is a more exhaustive documentation about compatibility handling, in the developer's documentation.

Backward Compatibility

Opening older files and converting them for the current version of Blender is usually straight-forward. It is expected to give very good and usable results.

There can be major feature changes, for which the backward compatibility will only be ensured for a limited amount of time. For example the changes to the animation system that happened during the Blender 2.5x project. This will never be less than a full major release cycle (i.e. two years at least).

Forward Compatibility

Loss of Data

Forward compatibility is inherently harder to ensure, and loss of feature should always be expected when opening a blend-file saved with a more recent version of Blender.

A warning is shown in the UI when editing a more recent blend-file. Trying to overwrite it (with a simple ‘Save’ operation) will also show a confirmation popup, as this could make that loss of data permanent.

Complete Incompatibility

When Blender switches to a new major version release (e.g. from 3.x to 4.0), there can also be major changes that will make the blend-file fully incompatible with older versions of Blender.

In such cases, older Blender will fail opening (or appending/linking from) the newer blend-file, with a message stating which minimal version is needed to open it.

In such cases, the last LTS release of the previous release cycle will be kept compatible with the newer file format version, and will be usable as converter between both versions.

For example, Blender 3.6LTS can open files from Blender 4.x, and will perform the necessary conversion such that when re-saved from 3.6, the files become compatible with all 3.x Blender versions.

Packed Data

Blender has the ability to encapsulate (incorporate) various kinds of data within the blend-file that is normally saved outside of the blend-file. For example, an image texture that is an external image file can be put "inside" the blend-file. This allows sharing a full project as a single file, instead of e.g. an archive containing the blend-file and all its dependencies.

You know that a data is packed when you see a little "gift box" icon displayed next to its path.

Warning

Not all external files can be packed

Some typically heavy external files, like videos from the Sequence Editor or Movie Clips, cannot be packed in a blend-file.

Pack Data

Pack Resources

Reference

Panel:: *File ▸ External Data ▸ Pack Resources*

Mark all eligible external resource files used by the blend-file as packed. Actual packing will happen on the next save of the blend-file.

Automatically Pack Resources

Reference

Panel:: *File ▸ External Data ▸ Automatically Pack Resources*

When enabled, this option will ensure that all eligible external resource files, existing or added later, are systematically marked as packed. As with *Pack Resources*, the blend-file must be saved to the drive for this to have an effect.

Disabling that option won't unpack anything, but future external files won't be automatically marked as packed anymore.

Selective Packing

A single file can be packed by clicking on the little "gift box" icon to the left of its file-path UI widget.

Unpack Data

Unpack Resources

Reference

Panel:: *File ▸ External Data ▸ Unpack Resources*

Unpack all external resource files stored into a blend-file.

Options

Use files in current directory (create when necessary)
: Unpacks all files in the same directory `//` as the blend-file, grouping them in proper folders (like ''textures'' for instance). However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to current directory (overwrite existing files)
: As with previous option, but if the final file exists already, it will overwrite it.

Use files in original location (create when necessary)
: Unpacks all files in their original location. However, if the final file exists already, it will use that file, instead of unpacking it.

Write files to original location (overwrite existing files)
: As with previous option, but if the final file exists already, it will overwrite it.

Disable AutoPack, keep all packed files
: Only deactivates the *Automatically Pack Resources* option.

Selective Unpacking

A single file can be unpacked by clicking on the little "gift box" icon to the left of its file-path UI widget.

Options

Remove Pack
: Just mark the file as unpacked, without actually writing it or reloading it from the drive.

Create <local file path>
: Unpack the file at the proposed path, which is local to the current blend-file.

Use <original file path> (differs)|(identical)
: If the original file path still exists, mark it as unpacked. Note that it won't be automatically reloaded from the drive. *(differs)* or *(identical)* show difference status between the packed version and the one on-drive.

Overwrite <original file path>
: If the original file path still exists but differs from the packed version, mark it as unpacked and overwrite the on-drive file with the packed version.

Create <original file path>

If the original file path does not exist, mark it as unpacked and write it to drive.

Pack Linked Libraries

Reference

Panel:: *File ▸ External Data ▸ Pack Linked Libraries*

Mark all linked library files in the current blend-file as packed. Actual packing will happen on the next save of the blend-file.

Unpack Linked Libraries

Reference

Panel:: *File ▸ External Data ▸ Unpack Linked Libraries*

Unpack all used linked library files from this blend-file.

Blend-Files Previews

A blend-file can store previews, both for itself, and for some of its data-blocks. You can disable writing any previews when saving a blend-file using the *Save Preview Images* setting from the Save & Load section of the Preferences.

Blend-File Preview

Blender saves by default a small preview of current scene in the blend-file. This will show in the *Thumbnail* view of the File Browser.

During its installation, Blender also adds a small tool to your OS, that will allow your system file browser to show those previews as file thumbnails as well.

Data-Blocks Previews

Blender will automatically generate previews for some type of data, mainly the ones related to shading (like images, textures, materials, lights and world shaders).

It can also store previews for scenes, collections and objects, but those need to be generated manually.

These previews can then be used by the *Thumbnail* view of the File Browser, when linking or appending data-blocks.

Refresh Data-Block Previews

Reference

Menu:: *File ▸ Data Previews ▸ Refresh Data-blocks Previews*

Refresh all data-block previews that can be automatically generated by Blender (shading-related ones), in the current blend-file. You still need to save the file if you want to write them to the drive.

Batch Generate Previews

Reference

Menu:: *File ▸ Data Previews ▸ Batch Generate Previews*

Generate some data-block types' previews (you can choose which in its options), in one or more blend-files on your drive. You should not use this operator on the file currently opened in Blender.

This is currently the only way to generate and store in blend-files previews for scenes, collections and objects. Note that since this involves a lot of rendering, even of small sizes, the process may take some time to complete.

Scenes
: Generate previews of scenes and their collections.

Collections
: Generate previews of collections of objects.

Objects
: Generate previews of objects.

Materials & Textures
: Generates previews for materials, textures, images, and other internal data.

Trusted Blend Files
: When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file

yourself, or you trust that the person who gave it to you did not include any malicious code with it. See Python Security to configure default trust options.

Save Backups
: Keep a backup version (`blend1-file`) of the files when saving with generated previews.

Clear Data-Block Previews

Reference

Menu:: *File ▸ Data Previews ▸ Clear Data-blocks Previews*

Clear all, a generic type of, or a specific data-block type of previews in the current blend-file. You still need to save the file if you want to clear them from the drive.

Batch Clear Previews

Reference

Menu:: *File ▸ Data Previews ▸ Batch Clear Previews*

Clear some data-block types' previews (you can choose which in its options), in one or more blend-files on your drive. You should not use this operator on the file currently opened in Blender.

Scenes
: Clear previews of scenes and their collections.

Collections
: Clear previews of collections of objects.

Objects
: Clear previews of objects.

Materials & Textures

Clear previews for materials, textures, images, and other internal data.

Trusted Blend Files
: When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See Python Security to configure default trust options.

Save Backups
: Keep a backup version (`blend1-file`) of the files when saving with cleared previews.

Rename

Rename Active Item

Reference

Menu:: *Edit* ▸ *Rename Active Item*
Shortcut:: `F2`

The *Rename Active Item* operator renames the active Bone, Node, Object and Sequence Strip.

When the operator is executed, a pop-up dialog appears. The text field shows the name of the current item and can be overwritten to rename the item. `Return` confirms the name while `Esc` cancels the operator.

Batch Rename

Reference

Menu:: *Edit* ▸ *Batch Rename*
Shortcut:: `Ctrl-F2`

The *Batch Rename* operator can rename many data-block names at once. This uses a pop-up dialog with operations and their options to change the name. These actions are applied in order, from first to last.

Data Source
: Where to look for the data-blocks that are intended to be renamed.

Selected:: Operates on the currently selected objects.
All:: Operates on all data in the blend file.

Data Type

The data-block type to perform the batch rename operations on.

Operations

The *Batch Rename* has several sub Operations to change the data names. The default operation is *Find/Replace* however, other operations can be added to change the data names further. Below all the operations gives a message in the status bar on how many data-blocks were renamed.

Find/Replace

Find/Replace searches for a particular text in the names and optionally replaces it with a new text. Regular Expressions can be used as a powerful way to tailor the *Find/Replace* texts and can be enabled using the icon to the right of the text fields.

Find
: The text to search for in names.

Replace
: The text to replace for in matching names found from the *Find* text.

Case Sensitive
: Search results must exactly match the case of the *Find* text.

Set Name

Set Name works the most similar to Rename Active Item by renaming the current data-block without having to do a find and replace operation.

Method

New:: Disregards the current name replacing it with the "new" name.

Prefix:: Adds text to the beginning of the current name. This is useful for tools that look for special text in the prefix of a data-block name.

Suffix::

Adds text to the end of the current name. This is useful for tools that look for special text in the suffix of a data-block name.

Name
: Defines the new name or the text to add as a prefix/suffix.

Strip Characters

Strip Characters cleans up names by removing certain character types from either the beginning or the end of the name.

Characters
: **Spaces::** Strips any space characters from the name, e.g. “Living Room “ becomes “Living Room”.
: **Digits::** Strips any numerical characters from the name, e.g. `cube.001` becomes `cube.`.
: **Punctuation::** Strips any punctuation characters (`,.?!:;` etc.) from the name, e.g. `cube?` becomes `cube`.

> **Tip**
> Multiple character types can be removed at once by `Shift-LMB` on the types.

Strip From
: **Start::** Strips any leading characters in the name.
: **End::** Strips any trailing characters in the name.

Change Case

Change Case modifies the case of names to be one of the following:

Convert To
: Upper Case
: Changes all text to be in upper case, e.g. `cube.001` becomes `CUBE.001`.

Lower Case
: Changes all text to be in lower case, e.g. `CUBE.001` becomes `cube.001`.

Title Caps
: Changes all text to be in title case, e.g. `living room` becomes `Living Room`.

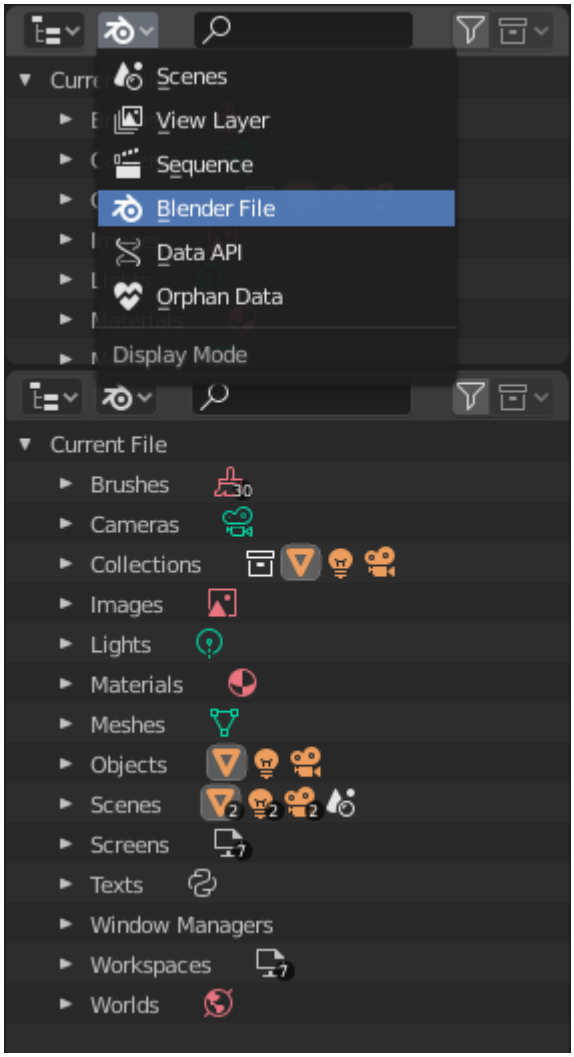
Data-Blocks

The base unit for any Blender project is the data-block. Examples of data-blocks include: meshes, objects, materials, textures, node trees, scenes, texts, brushes, and even Workspaces.

A data-block is a generic abstraction of very different kinds of data, which features a common set of basic features, properties and behaviors.

Some common characteristics:

- They are the primary contents of the blend-file.
- They can reference each other, for reuse and instancing. (Child/parent, object/object-data, materials/images, in modifiers or constraints too…)
- Their names are unique within a blend-file, for a given type.
- They can be added/removed/edited/duplicated.
- They can be linked between files (only enabled for a limited set of data-blocks).
- They can have their own animation data.
- They can have Custom Properties.



Blender File view of the Outliner.

User will typically interact with the higher level data types (objects, meshes, etc.). When doing more complex

projects, managing data-blocks becomes more important, especially when inter-linking blend-files. The main editor for that is the Outliner.

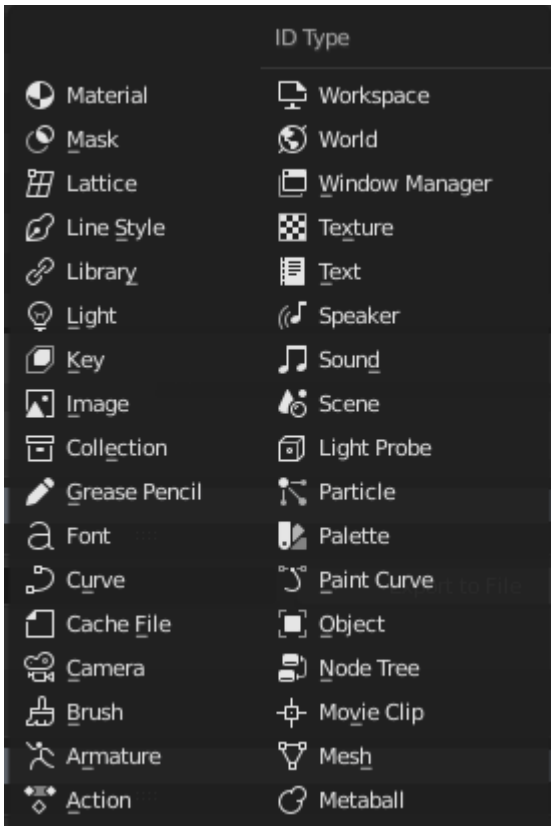
Not all data in Blender is a data-block, bones, sequence strips or vertex groups e.g. are not, they belong to armature, scene and mesh types respectively.

Data-Block Types

For reference, here is a table of data-blocks types stored in blend-files.

Link
: Library Linking, supports being linked into other blend-files.

Pack
: File Packing, supports file contents being packed into the blend-file (**not** applicable for most data-blocks which have no file reference).



Data-blocks types with their icon.

Type	Link	Pack	Description
Action	✓	—	Stores animation F-Curves. Used as data-block animation data, and the Nonlinear

Type	Link	Pack	Description
			Animation editor.
Armature	✓	—	Skeleton used to deform meshes. Used as data of armature objects, and by the Armature Modifier.
Brush	✓	—	Used as brush assets in sculpt and paint modes.
Camera	✓	—	Used as data by camera objects.
Cache File	✓	—	Used by Mesh Cache modifiers.
Curve	✓	—	Used as data by curve, font & surface objects.
Font	✓	✓	References font files. Used by curve object-data of text objects.
Grease Pencil	✓	—	2D/3D sketch data used by Grease Pencil objects. Used as overlay *helper* info, by the 3D Viewport, Image, Sequencer & Movie Clip editors.
Collection	✓	—	Group and organize objects in scenes. Used to instance objects, and in library linking.

Type	Link	Pack	Description
Image	✓	✓	Image files. Used by shader nodes and textures.
Keys (Shape Keys)	✗	—	Geometry shape storage, which can be animated. Used by mesh, curve, and lattice objects.
Light	✓	—	Used as object data by light objects.
Library	✗	✓	References to an external blend-file. Access from the Outliner's *Blender File* view.
Line Style	✓	—	Used by the Freestyle renderer.
Lattice	✓	—	Grid based lattice deformation. Used as data of lattice objects, and by the Lattice Modifier.
Mask	✓	—	2D animated mask curves. Used by compositing nodes & sequencer strip.
Material	✓	—	Set shading and texturing render properties. Used by objects, meshes & curves.
Metaball	✓	—	An isosurface in 3D space. Used as data of metaball objects.

Type	Link	Pack	Description
Mesh	✓	—	Geometry made of vertices/edges/faces. Used as data of mesh objects.
Movie Clip	✓	✗	Reference to an image sequence or video file. Used in the *Movie Clip* editor.
Node Tree	✓	—	Groups of re-usable nodes. Used in the node editors.
Object	✓	—	An entity in the scene with location, scale, rotation. Used by scenes & collections.
Paint Curve	✓	—	Stores a paint or sculpt stroke. Access from the paint tools.
Palette	✓	—	Store color presets. Access from the paint tools.
Particle	✓	—	Particle settings. Used by particle systems.
Light Probe	✓	—	Help achieve complex real-time lighting in EEVEE.
Scene	✓	—	Primary store of all data displayed and animated. Used as top-level storage for objects & animation.

Type	Link	Pack	Description
Sounds	✓	✓	Reference to sound files. Used as data of speaker objects.
Speaker	✓	—	Sound sources for a 3D scene. Used as data of speaker object.
Text	✓	✗	Text data. Used by Python scripts and OSL shaders.
Texture	✓	—	2D/3D textures. Used by brushes and modifiers.
Window Manager	✗	—	The overarching manager for all of Blender's user interface. Includes Workspaces, notification system, operators, and keymaps.
World	✓	—	Define global render environment settings.
Workspace	✗	—	UI layout. Used by each window, which has its own workspace.

Life Time

Every data-block has its usage counted (reference count), when there is more than one, you can see the number of current users of a data-block to the right of its name in the interface. Blender follows the general rule that unused data is eventually removed.

Since it is common to add and remove a lot of data while working, this has the advantage of not having to manually manage every single data-block. This works by skipping zero user data-blocks when writing blend-files.

Protected

Since zero user data-blocks are not saved, there are times when you want to force the data to be kept irrespective of its users.

If you are building a blend-file to serve as a library of assets that you intend to link to and from other files, you will need to make sure that they do not accidentally get deleted from the library file.

To protect a data-block, use the button with the shield icon next to its name. The data-block will then never be silently deleted by Blender, but you can still manually remove it if needed.

Name & Rename

Data-blocks names are unique within their namespace. A data-block namespace is defined by its type, and the blendfile it is stored in.

This means that there can be for example an Object and a Mesh named the same, but there cannot be two local objects named the same in a blendfile. However, it is possible to have one local and several linked Objects sharing the same name.

Data-block names have a fixed length of 63 bytes, i.e. 63 basic ASCII characters, or less when using diacritics or non-latin glyphs (the UTF8 encoding will then typically use more than a byte per character).

When Blender has to name a new data-block, or rename an existing one, it will check for name collisions. If a data-block with the same name already exists, the (re)named data-block will get a numeric extension added as a post-fix to its 'root name', like e.g. *.001*. The first available index is used (up to the *999* value, after that the postfix index values are simply incremented until no collision happen anymore).

In case adding the numeric suffix would make the data-block name too long, the root name part will be shortened as needed.

Blender will never rename another data-block when doing automatic naming. So e.g. when adding a new *Cube* object and there are already *Cube* and *Cube.001* local objects, the new one will be named *Cube.002*.

Local data-blocks can be renamed by the user in several places in the UI (like the ID selection widget, or the Outliner view). When renamed from the UI, the behavior in case of name collision is as follow:

- If the original root name is different than in the new requested name, the renamed data-block gets the first available numerical suffix.
  - E.g. assuming that there are three objects named *Sphere*, *Cube* and *Cube.001*, renaming *Sphere* to *Cube* will rename the data-block to *Cube.002*.
- If the original root name is the same as in the new requested name, the renamed data-block gets the requested name, and the conflicting of data-block is renamed accordingly.
  - E.g. assuming that there are three objects named *Sphere*, *Cube* and *Cube.001*, renaming *Cube.001* to *Cube* will rename the data-block to *Cube*, and the other data-block to *Cube.001*.

Sharing

Data-blocks can be shared among other data-blocks.

Examples where sharing data is common:

- Sharing textures among materials.
- Sharing meshes between objects (instances).
- Sharing animated actions between objects, for example to make all the lights dim together.

You can also share data-blocks between files, see linked libraries.

Making Single User

When a data-block is shared between several users, you can make a copy of it for a given user. To do so, click on the user count button to the right of its name. This will duplicate that data-block and assign the newly created copy to that usage only.

Note

Objects have a set of more advanced actions to become single-user, see their documentation.

Removing Data-Blocks

As covered in Life Time, data-blocks are typically removed when they are no longer used. They can also be manually *unlinked* or *deleted*.

Unlinking a data-block means that its user won't use it anymore. This can be achieved by clicking on the "X" icon next to a data-block's name. If you unlink a data-block from all of its users, it will eventually be deleted by Blender as described above (unless it is a protected one).

Deleting a data-block directly erases it from the blend-file, automatically unlinking it from all of its users. This can be achieved by `Shift-LMB` on the "X" icon next to its name.

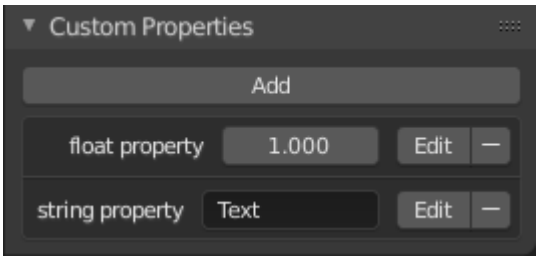
Warning

Deleting some data-blocks can lead to deletion of some of its users, which would become invalid without them. The main example is that object-data deletion (like mesh, curve, camera…) will also delete all objects using it.

Those two operations are also available in the context menu when `RMB`-clicking on a data-block in the *Outliner*.

Custom Properties

Custom properties are a way to store your own data in Blender's data-blocks. It can be used for rigging (where bones and objects can have custom properties driving other properties), and Python scripts, where it's common to define new settings not available in Blender. It is also possible to access custom properties from materials via the Attribute Node.



Custom Properties panel.

Only certain data supports custom properties:

- All data-blocks types.
- Bones and pose bones.
- Sequence strips.

To add a custom property, search for the *Custom Properties* panel, found at the bottom of most Properties or Sidebar region, and click *New*. Properties can be removed from the same location with the delete icon. Once properties are added they can be configured via the edit icon to work for a particular use case; see Editing Properties for more information.

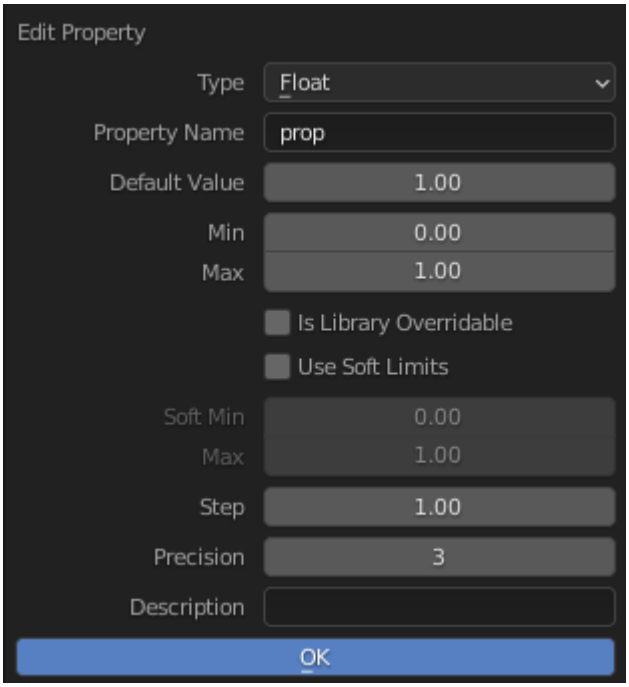
Editing Properties

User Interface

Custom properties can be edited using the panel available for data types that support it. Editing the properties allows you to configure things such as default values, ranges, and even add a custom tooltip.

Type

The data type of the property; different data types have can only have specific data properties.



Custom Properties edit pop-up.

Float:: A numeric value with decimals e.g. 3.141, 5.0, or 6.125.

Float Array:: A collection of multiple float data types e.g. `[3.141, 5.0, 6.125]` . This data type can also be used for data that can be represented as a float array such as colors. These special float arrays can be set in the *Subtype* selector.

Integer:: A numeric value without any decimals e.g. 1, 2, 3, or 4.

Integer Array:: A collection of multiple integer data types e.g. `[1, 2, 3, 4]` .

Boolean:: A data type that has two possible values e.g. `True` or `False`.

Boolean Array:: A collection of boolean values e.g. `[True, False, True]`

String:: A sequence of characters such as “Some Text”.

Data-Block:: A reference to a Blender object, see Data-Blocks.

Python::

Edit a Python data type directly, used for unsupported data types.

Array Length
: The number of elements in the array. Note that if the array length is greater than 7 you cannot directly edit its elements, you must press *Edit Value* to edit the elements of the array.

Property Name
: The text that is displayed to the left of the value. This name is also used to access the property via Python.

Default Value
: This sets the default value of the property used by the Reset to Default Value operator.

  Warning

  Default values are used as the basis of NLA blending, and a nonsensical default (e.g. 0 for a property used for scaling) on a property intended for being keyframed is likely to cause issues.

Min, Max
: The minimum/maximum value the custom property can take.

Library Overridable
: Allow the property to be overridden when the data-block is linked.

Soft Limits
: Enables limits that the *Property Value* slider can be adjusted to without having to input the value numerically.

  Soft Min, Max
  : The minimum/maximum value for the soft limit.

Step
: A multiplier to control how much the data type is incremented at a time. The internal step size for floats is 0.01, so a *Step* value of 5 will

increment at a rate of 0.05 and a *Step* value of 100 will increment by 1.0. For integers the internal step size is 1.

Precision

The number of digits after the decimal to display in the user interface for float data types.

Subtype

Specifies the type of data the property contains, which affects how it appears in the user interface. This option is only available for float properties and has different options for regular floats and float arrays. Note, the unit often depends on the Scene Units.

For regular floats:

Plain Data::	Data values do not have any special behavior.
Pixel::	A measure digital image resolution.
Percentage::	The displayed value is a percentage, typically you will want the Min and Max values to be 0 and 100.
Factor::	A percentage between an upper and lower bound which typical have a numerical significance.
Angle::	A measure between intersecting lines.
Time::	Time specified in seconds.
Distance::	Measure of space between items.
Power::	Work as a factor of time, measured in watts. This is used in Blender to measure light intensity.
Temperature::	Intensity of heat present.
Wavelength::	The distance between cycles of a wave measured in millimeters (mm), micrometers (μm), nanometers (nm), or picometers (pm).

For float arrays:

Plain Data::	Data values do not have any special behavior.
Linear Color::	Color in linear color space.
Gamma-Corrected Color::	Color in gamma corrected color space.

Euler Angles:: Euler Rotation angles.

Quaternion Angles:: Quaternion Rotation angles.

Note

For either of the color subtypes to work as expected the *Property Value* must be a vector with three or four values depending on the availability of an Alpha Channel.

ID Type *Data-Block*

The ID-block type. For example: Key, Image, Object, Material. See Data-Block Types for a full list.

Description

Allows you to write a custom Tooltip for your property.

Python Access

Custom properties can be accessed in a similar way to dictionaries, with the constraints that keys can only be strings, and values can only be strings, numbers, arrays of such, or nested properties.

See the API documentation for details.

Linked Libraries

- Link & Append
  - Link
  - Append
  - Library Reload & Relocate
  - Make Local
  - Known Limitations
- Library Overrides
  - Override Hierarchies
  - Animation & Overrides
  - Resyncing Overrides
  - Non-Editable Overrides
  - Make an Override
  - Reset an Override
  - Clear an Override
  - Edit an Override
  - Troubleshoot an Override Hierarchy

Link & Append

These functions help you reuse materials, objects and other data-blocks loaded from another blend-file. You can build libraries of common content and share them across multiple referencing files.

Newly added collections types are available in *Add ‣ Collection Instance* in the 3D Viewport.

Look in the *Outliner*, with display mode set to *Blender File*, to see all your linked and appended data-blocks.

Note

It is not possible to Append or Link data from much newer blend-files

Link

Reference

Editor::	Topbar
Mode::	All Modes
Menu::	*File ‣ Link*

Link creates a reference to the data in the source file such that changes made there will be reflected in the referencing file the next time it is reloaded. But linked data is usually not editable. Library Overrides can be created from linked data to allow some level of local editing, animation, etc.

In the File Browser, navigate to the external source blend-file and select the data-block you want to reuse.

When you link an object, it will be placed in your scene at the 3D cursor position. Many other data types, cameras, curves, and materials for example, must be linked to an object before they become visible.

Options

Relative Path
: See Relative Paths.

Select
: Makes the object *Active* after it is loaded.

Active Collection
: The object will be added to the active collection of the active view layer. Otherwise, it will be added to a new collection in the active view layer.

Instance Collections
: This option instantiates the linked collection as an object, adding it to the active scene. Otherwise, the linked collection is directly added to the active view layer.

Instance Object Data
: Create instances for object data which are not referenced by any objects.

Append

Reference

Editor::	Topbar
Mode::	All Modes
Menu::	*File* ‣ *Link*

Append makes a full copy of the data into your blend-file, without keeping any reference to the original one. You can make further edits to your local copy of the data, but changes in the external source file will not be reflected in the referencing file.

In the File Browser, navigate to the external source blend-file and select the data-block you want to reuse.

Tip

Blend-files can also be linked/appended by dragging and dropping blend-files into the Blender window.

Note

Appending data you already have linked will add objects/collections to the scene, but will keep them linked (and un-editable).

This is done so existing relationships with linked data remain intact.

Options

Select
: Makes the object *Active* after it is loaded.

Active Collection
: The object will be added to the active collection of the active view layer. Otherwise, it will be added to a new collection in the active view layer.

Instance Collections
: This option instantiates the linked collection as an object, adding it to the active scene. Otherwise, the linked collection is directly added to the active view layer.

Instance Object Data
: Create instances for object data which are not referenced by any objects.

Fake User
: Defines the appended data-block as Protected.

Localize All
: Appends also all indirectly linked data, instead of linking them.

Library Reload & Relocate

Reloading is useful if you changed something in the library blend-file and want to see those changes in your current blend-file without having to re-open it. You can reload and relocate a whole library from the context menu of the library items in the *Outliner*'s *Blender File* view,

Relocating allows you to reload the library from a new file path. This can be used to either fix a broken linked library (e.g. because the library file was moved or renamed after linking from it), or to switch between different variations of a same set of data, in different library files.

Broken Library

While loading a blend-file, if Blender cannot find a library, it will create placeholder data-blocks to replace missing linked ones. That way, references to the missing data is not lost, and by relocating the missing library, the lost data can be automatically restored.

Make Local

Reference

Editor:: 3D Viewport
Mode:: Object Mode
Menu:: *Object ▸ Relations ▸ Make Local…*

Reference

Editor:: Outliner
Menu:: *Context menu ▸ ID Data ▸ Make Local*

Makes the selected or all external objects local to the current blend-file. Links to the original library file will be lost, but it will make those data-blocks fully editable, just like the ones directly created in that blend-file.

Options

The operation available from the *Outliner*'s context menu has no options, and only affects the selected data-block.

The operation available from the *3D Viewport* only directly affects selected objects, but it can also make local the objects' dependencies:

Type
: Optionally unlinks the object's Object Data and Material Data.

  Selected Objects, + Object Data, + Materials, All (i.e. including all scenes)

Known Limitations

For the most part linking data will work as expected, however, there are some corner cases which are not supported.

Circular Dependencies

In general, dependencies should not go in both directions. Attempting to link or append data which links back to the current file will likely result in missing links.

Object Rigid Body Constraints

When linking objects *directly* into a blend-file, the *Rigid Body* settings **will not** be linked in since they are associated with their scene's world. As an alternative, you can link in the entire scene and set it as a Background Set.

Compression & Memory Use

Linking to blend-files with compression enabled may significantly increase memory usage while loading files.

Reading data on demand isn't supported with compression *(this only impacts load time, once loaded there is no difference in memory use)*.

Library Overrides

Library Overrides is a system designed to allow editing linked data, while keeping it in sync with the original library data. Most types of linked data-blocks can be overridden, and the properties of these overrides can then be edited. When the library data changes, unmodified properties of the overridden one will be updated accordingly.

Note

The old proxy system has been deprecated in Blender 3.0, and fully removed in Blender 3.2. Automatic conversion from proxies to library overrides happens when loading a blend-file, but results on complex characters are not guaranteed and may need manual fixes.

Library overrides supports:

- Multiple independent overrides of a same linked data (e.g. having the same character multiple times in the same scene).
- Adding new modifiers and constraints, anywhere in the stack.
- Recursively chaining overrides (i.e. link and override overrides from another library file, etc.).

Note

There are known issues that have to be addressed. See the main task of the project, for more details.

Warning

While in most cases library overrides data is preserved across a loss of reference linked data (if e.g. the library file becomes unavailable or is relocated), there are some exceptions.

The main one is probably posed (but not animated) armature objects, when their Armature obdata itself is not overridden. The Pose bones of an armature object are fully linked to the bones of its Armature obdata, if the later goes missing, the pose bones are definitively lost.

Note

Proper Collections Layout Matters

For library overrides to work well, it is much better if **all** the collections needed by the character are children of the root (linked and instantiated) one, such that there is a clear hierarchy. Otherwise, some data may not be properly automatically overridden, and other operations may be less reliable.

Override Hierarchies

Hierarchy is a very important concept to understand when working with library overrides. In Blender, a real-life asset (a character, a prop, a set, etc.) is almost never made of a single data-block, but is rather a group of data-blocks with dependency relationships to each-other. E.g. a character will typically have an armature object, several geometry objects, rig-controllers objects, the object data for all of these objects, materials, textures, etc.

These relationships can be represented as a tree, with a root data-block 'linking-in' all its dependencies, recursively. With library overrides, typically, the root of the hierarchy is also the data-block that is directly linked when importing the asset (usually a collection).

This concept of hierarchy can also be seen as some sort of super meta-data-block. It is critical when there are several overrides of the same linked data, since it allows to clearly identify a given data-block to one override, leaving no ambiguity to processes that affect the whole hierarchy (e.g. resyncing overrides with their linked data). It also allows to share relationships

between data-blocks of different hierarchies, like a parenting relationships between two different overrides of a same character.

Animation & Overrides

Due to current design of animation data in Blender, what is editable in overrides' animations can change greatly depending on whether animation data was already defined in the linked reference data-block. Animation data is created for a datablock if it gets animated by keyframes, or through drivers.

In general, an overrides can do much more with its animation data if no animation data exists in its linked reference data-block.

Keyframes (a.k.a. F-Curves)

Keyframed animation belongs to another data-block (an Action one). So it is possible to assign a purely local Action data-block replacing the one linked from the library. This will completely replace the keyframed animation from the linked data though, and not override it in any way.

Overridden Action data-blocks only support a very limited amount of editing. For example, an existing F-Curve can be muted, but its keyframes cannot be edited, and no new F-Curve can be added.

Drivers

If the linked reference data has animation data, then its overrides only have limited possibilities to edit the existing drivers. For example, it will be possible to change the exisitng target of a driver, but it won't be possible to add new drivers, or new targets to an existing driver.

If the linked reference data has no animation data, then its overrides will create a new one when they get some drivers defined. Drivers can then be fully edited, added or removed, just as with purely local data-blocks.

NLA

The NLA editor data also belongs to the animation data of a data-block. However, this data does support some greater level of edition in overrides, including moving or resizing existing strips from the linked data, and adding new local strips.

Resyncing Overrides

The relationships between linked data-blocks can change, resulting in outdated overrides. When this happens, overrides need to be resynced to match the new structure of their hierarchy. Overrides are automatically resynced if needed on blend-files opening. However, it may be needed to resynced them manually sometimes, see Troubleshoot an Override Hierarchy.

Tip

Blender is also able to resync library overrides from external libraries, that are then linked into a working file. However, this is a costly process that needs to be fully redone every time the working file is loaded, since Blender cannot edit/modify the external library directly.

So users linking overrides (or creating recursive overrides) should ensure that their library files are regularly updated, to avoid this overhead on file load (typically, opening and saving those library files should be enough to update them).

Tip

Auto resyncing can be disabled in the Experimental Preferences.

Non-Editable Overrides

For technical reasons (how relationships between data-blocks are stored), Blender needs to create overrides of a lot of data-blocks, even when only

one or two of them actually needs to be edited by the user. To reduce the amount of information and risk of potential unwanted editing, most of these data-blocks are now marked as non-editable by default. This can be changed once the override has been created.

Make an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	*3D Viewport* ▸ *Header* ▸ *Object* ▸ *Library Override* ▸ *Make Outliner* ▸ *Context Menu* ▸ *Library Override* ▸ *Make ID Widget* ▸ *Context Menu* ▸ *Library Override* ▸ *Make*
Shortcut::	`Shift-LMB` on the 'linked'/'overridden' button of an ID Widget.

Create overrides from the selected data-blocks.

Blender automatically create overrides for all required data-blocks to ensure that valid override hierarchies are created.

Only overrides created from selected items will be user-editable.

Warning

The support for the creation of library overrides from the ID Widget (mainly from within the *Properties* editor) is limited. While the most common usages should be supported, especially with Objects, meshes, etc., much remains to be implemented.

Selected Items

Depending on where from the override is created, there are several ways to ‘select’ items to be overridden and user-editable.

Note

This also applies to the other common operations (*Reset* and *Clear*).

The *Troubleshoot* advanced operations only available from the Outliner always apply to a whole override hierarchy.

3DView

The selected objects will be considered as selected.

When a selected object is a local *Empty* instantiating a linked collection, the following will happen: * The *Empty* object will be removed. * Its linked collection will be overridden, and that override will be instanced in the same collection in the current *View Layer*. * If the collection contains *Armature* objects, they will be user-editable. Otherwise, no created override will be defined as user-editable.

Outliner

The operation can be applied on either the selected items only, their content only, or both.

Tip

Using *Selected & Content* is an easy way to get all newly created overrides immediately user-editable.

ID Widget

Only the linked data-block in the ID Widget is considered as selected, and set as editable once overridden.

Make Editable

That same operation can also be used to make existing overrides user-editable, after they have been created, or cleared

Reset an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	*3D Viewport* ▸ *Header* ▸ *Object* ▸ *Library Override* ▸ *Reset Outliner* ▸ *Context Menu* ▸ *Library Override* ▸ *Reset ID Widget* ▸ *Context Menu* ▸ *Library Override* ▸ *Reset*

Reset the selected overrides to their original values (from the linked reference data). Unlike with the *Clear* operation, the overrides remain fully editable, and are never deleted.

Clear an Override

Reference

Editor::	3D Viewport, Outliner, Properties
Mode::	Object Mode
Menu::	*3D Viewport* ▸ *Header* ▸ *Object* ▸ *Library Override* ▸ *Clear Outliner* ▸ *Context Menu* ▸ *Library Override* ▸ *Clear ID Widget* ▸ *Context Menu* ▸ *Library Override* ▸ *Clear*
Shortcut::	`Shift-LMB` on the ‘overridden’ button of an ID Widget.

Reset the selected overrides to their original values, and if possible without breaking the existing hierarchy, delete them and replace them by their linked reference data. Otherwise, keep the overrides but mark them as non-editable.

Edit an Override

Essentially, an override is edited the same way as a regular local data-block. You can use operators on them, edit their properties from various editors, etc. There are some limitations however, most notably Edit Mode is not allowed for overrides. In most cases, as soon as you edit a property, you can see that it's overridden by its teal blue outline/background.

You can also animate overrides, animated properties just replace/supersede overrides then. Note that you cannot override/edit an existing animation, you'll have to create a new action. You can manually define or remove an override from the context menu of the relevant property. If an override is not editable, you have to make it editable first.

Define Overrides

Reference

Editor::	Any
Mode::	Object Mode
Property::	*Context Menu ▸ Define Overrides Context Menu ▸ Define Override*

Mark a property to be overridden in the local blend file. For array properties all elements will be overridden.

Define Single Override

Reference

Editor::	Any

Mode::	Object Mode
Property::	*Context Menu ▸ Define Single Override*

Mark a property to be overridden in the local blend file. For array properties only the selected element will be overridden.

Remove Overrides

Reference

Editor::	Any
Mode::	Object Mode
Property::	*Context Menu ▸ Remove Overrides Context Menu ▸ Remove Override*

Remove the property from the overrides. The value of the linked in data-block will be used. For array properties all elements will be removed from the override.

Remove Single Override

Reference

Editor::	Any
Mode::	Object Mode
Property::	*Context Menu ▸ Remove Single Override*

Remove the property from the overrides. The value of the linked in data-block will be used. For array properties only the selected elements will be removed from the override.

Troubleshoot an Override Hierarchy

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	*Context Menu ▸ Library override ▸ Troubleshoot*

These operations are only available from the Outliner contextual menu. They can help fixing a broken override hierarchy.

Resync

Reference

Editor::	Outliner
Mode::	Object Mode
Outliner::	*Context Menu ▸ Library override ▸ Troubleshoot ▸ Resync*

The hierarchy of the linked data (the relationships between linked data-blocks) can change. Overrides need to be resynced to match the new hierarchy. This operator will resync the override to match the new hierarchy in the library.

Warning

While resyncing a library override it is possible that edited overrides get deleted if they are changed in the original library. If this is the case, a warning message will be displayed stating how many overrides were deleted, if the deletion is undesirable the resync can be undone before saving the blend-file.

Note

This Process is Automatic

Usually, this operation happens automatically when blender detects it is needed, on file load, unless it is disabled in the Experimental Preferences.

Resync Enforce

Reference

Editor:: Outliner
Mode:: Object Mode
Outliner:: *Context Menu ▸ Library override ▸ Troubleshoot ▸ Resync Enforce*

In some cases, especially with older blend-files that were saved with 'broken' (non-hierarchy-matching) overrides, a regular resync itself cannot rebuild properly the override as expected (e.g. some objects might go missing). To solve this issue, this operator rebuilds the local override from its linked reference, as well as its hierarchy of dependencies, enforcing that hierarchy to match the linked data (i.e. ignoring existing overrides on data-blocks properties). This is similar to a regular resync, but is more forceful, aggressive, at the cost of a potential loss of some overrides on ID pointers properties.

Delete

Reference

Editor:: Outliner
Mode:: Object Mode
Outliner:: *Context Menu ▸ Library override ▸ Troubleshoot ▸ Delete*

Remove the whole library override hierarchy, and replace all of these override data-blocks by their original linked data-blocks. This fully reverts the *Make* operation.

Asset Libraries

- Introduction
  - What is an Asset?
  - What is an Asset Library?
  - Asset Types
  - The Current File Asset Library
  - Life Cycle of an Asset
  - Bundled Assets
  - Asset System Files (*.asset.blend* Extention)
  - Design Limitations
  - Future Development
- Asset Catalogs
  - The Home Location of Assets
  - Creating Catalogs
  - Assigning an Asset
  - Saving Catalogs
  - Components of a Catalog
  - Catalog Definition Files

Introduction

This section describes Blender's asset library system. It was introduced in Blender 3.0, and will be improved and expanded over multiple upcoming releases.

See also

Asset Browser
The main interface for organizing and using assets.

Asset Catalogs
For organizing assets.

Pose Library
Built on top of the Asset Browser.

What is an Asset?

An asset is a data-block with meaning.

A blend-file is a database with multiple Data-Blocks: objects, textures, materials, etc. When planning to re-use or share these, the data needs a meaning. What is this? What is this for? **Assets are curated data-blocks that are meant for easy reuse**.

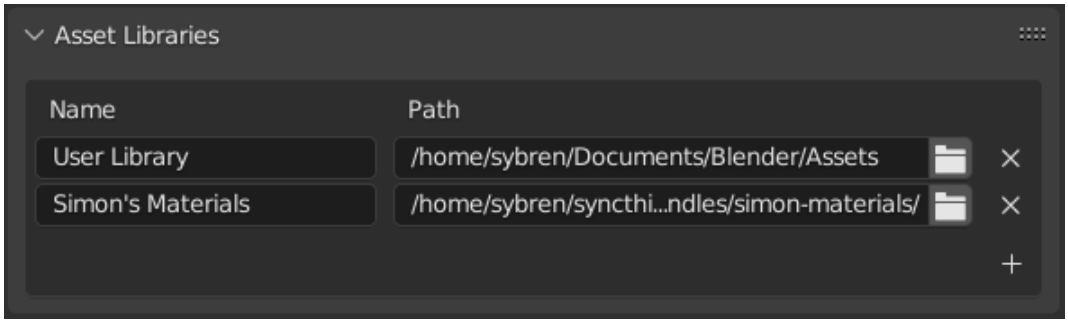
Note

The general term "asset" often also refers to other file types, such as images, sounds, video files, etc. These are currently not supported as asset in Blender.

For more info, see Future Development.

What is an Asset Library?

An asset library is a directory on your drive that is registered in the Preferences as an asset library. Registering it means that you give the library a name (like “Sprite Fright”) and the location on drive (like `/home/sybren/projects/sprite-fright/assets`).



Name and Location of asset libraries in the Preferences.

Once registered, you can select the asset library in the Asset Browser. All the blend-files in the asset library will be scanned for assets, and all those assets will be shown in the Asset Browser.

Note

Loading an asset library for the first time may take a while, but the next time it is loaded should be significantly faster. Blender generates an index of all assets contained inside an asset library, and keeps it up-to-date as files are modified within it. The indices are stored in the Local Cache Directory.

The blend-files can be directly in the top-level directory of the asset library, or in any subdirectory. The on-drive organization of asset libraries is all up to you. Regardless of which blend-file contains the assets, each asset can be

assigned a **catalog**. For more info about how to organize your assets this way, see Asset Catalogs.

Asset Types

Assets can be broadly divided into two types: **primitive** and **preset** assets. Which is which depends on the Data-Blocks type.

Primitive assets are data-blocks that are either **linked or appended** to the current file. Examples are objects, materials, and worlds. These can be dragged from the Asset Browser into the scene (objects and worlds), or onto existing objects (materials).

Preset assets are data-blocks that are loaded and then **applied** to something or **activated**. An example is a pose asset. When applying the pose, the data-block is loaded from its blend-file, and then the pose is applied to the active armature. Brush assets are an example of an asset type that is activated. They get loaded into the current file and activated for painting or sculpting, but don't get saved in the file.

In the future, the asset type definition will be expanded; see Future Development for more info.

The Current File Asset Library

To help with the management of assets in the current blend-file, you can set the Asset Browser to show the **Current File asset library**. This always shows the assets in the current file, even when the current file is not saved in an asset library. This also makes it possible to create assets and use them in the same file, for small single-file projects.

When the current blend-file is part of an asset library, you can also see its assets in that library, of course. The assets that are in the current file are marked with an icon; only those are editable.

Life Cycle of an Asset

This section describes how to create, edit, share, and use assets.

Creating an Asset

To create an asset, first create the thing you want to turn into an asset. That is, create the object, material, world, or pose your character. The next step depends on the type of asset (see Asset Types above).

For primitive assets, use the **Mark as Asset** operator. It can be found in the data-block selector, in the Outliner, and for objects in the 3D Viewport Object menu. When using *Mark as Asset*, an automatic preview is generated. If you want, you can also change or replace this with an image of your own choosing; use the folder button next to the preview image in the Asset Details region of the Asset Browser.

For preset assets, there will be a dedicated button for the different asset types. For example for poses there is a **Create Pose Asset** button in the Action editor. Brush assets are created by using **Duplicate Asset** from existing brush assets.

After creating the asset, make sure the current blend-file is saved in your asset library. Blender does not copy the asset into the asset library for you.

Editing Assets

Since assets are regular data-blocks, with just a little bit of metadata attached, they can be edited like any other Blender data. Just open the file and edit the object, material, world, etc.

For poses assets, this is also possible. With the pose library file open, just click the **Assign Action** button to assign the pose action to the currently selected armature. Then you can use all of the animation tooling to edit the pose, remove or add keys, etc.

Editing asset metadata can be done via the Asset Browser.

Sharing Assets

Because assets are simply stored in blend-files, they can be shared by sharing their blend-file. Be sure to include the Asset Catalog Definition File as well.

There is currently no functionality to extract selected assets and save them (together with their catalog definitions) into a different blend-file. This could be implemented as an add-on.

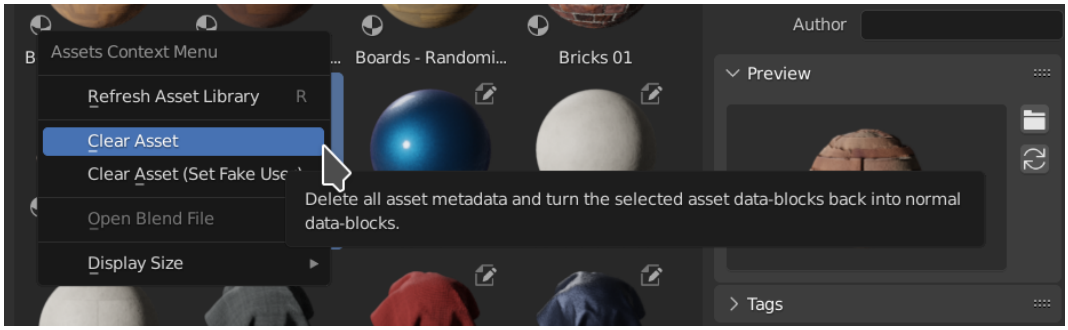
Using Assets

Assets can be used from the Asset Browser.

The pose library extends this, and adds an Asset View to the 3D Viewport. See Use from 3D Viewport.

Removing Assets

Asset metadata can be erased by the *Clear Asset* operator. This operator is available in data-block selectors, the Asset Browser, and for objects in the 3D Viewport menu.



Clear Asset in the Asset Browser.

Clear Asset
: Removes the asset metadata (catalog, description, author, tags), effectively turning an asset into a regular data-block. As such, the same removal rules apply as with other data-blocks. For example, if a mesh object is still placed in the scene, *Clear Asset* will not remove it from

the scene. See Life Time. The preview will be kept inside the data-block and not be removed.

Clear Asset (Set Fake User)
: Performs the same operation as *Clear Asset*, and then marks the data-block as protected. This makes it possible to no longer have the data-block marked as asset, and still be sure it is not lost when saving the blend-file.

Bundled Assets

Blender includes many assets out of the box, these are contained in the "Essentials" library.

Included in this library are:

- Hair node groups
- Smooth By Angle Node Group
- Brushes:
  - Mesh Sculpt
  - Curve Sculpt
  - Texture Paint
  - Vertex Paint
  - Weight Paint

Asset System Files (*.asset.blend* Extention)

Some types of assets can be edited without having to open a blend-file inside of an asset library. Blender saves these assets to libraries in special files using the *.asset.blend* extension. They are entirely managed by Blender's asset system, and only contain a single asset and its dependencies.

It is still possible to save a normal file with the *.asset.blend* extension. This will then not be treated as an asset system file, Blender knows the difference.

Asset system files have one more special characteristic: You can open, but not save them. The Save As operator can still be used to create a new file from them, which will then be just a normal blend-file. Thus contained assets cannot be edited without opening the file itself. Blender shows some clear warnings to communicate that asset system files cannot be changed and saved the normal way.

The reason these files are special is that the asset system might need to regenerate them. Any additional changes done by the user might be lost then. To prevent this data-loss, these files are protected from user modifications.

Currently, only brush assets support this feature.

Design Limitations

Blender is **not allowed to write to other blend-files** than the one you have currently open, or the special *.asset.blend* files explained above. This means that to edit an asset, you have to open its blend-file. Fortunately this is only a single click away, both in the Source List region of the Asset Browser and in the asset context menu.

Future Development

This section describes interesting avenues for further development. Even though it is not an exhaustive list, it might help to better understand the current functionality of Blender's Asset Browser.

Non-Data-Block Assets

Non-Blender assets, such as image or audio files, will likely be supported in a future version. For such files, asset metadata is then stored in XMP sidecar files, similar to what other software is also doing. Importers (USD, glTF, FBX, …) could add support for their file types as assets this way too. Furthermore, it should become possible to enrich an asset with a Python script, which can then provide code to be run when the asset is used.

Cross Blend-File Editing

As described above, Blender itself is not allowed to write to other blend-files than the currently open one. This rule helps to limit complexities; for example, it is hard to reliably implement an undo system when manipulating other files. The rule does get in the way of mass-updating assets when they are stored in various blend-files.

Since there is already tooling that can manipulate blend-files outside of Blender itself (see Blender Asset Tracer), it's possible to also create an external tool for doing such edits across blend-files. Such a tool might even be implemented via Blender's application templates system, or as an add-on; the rule above applies to Blender itself, not to its add-ons.

Asset Pushing

> **Note**
>
> The introduction of Brush assets in Blender 4.3 includes support for an asset pushing concept as described here. This might be brought to more asset types in future.

Asset **pushing** is a way of getting assets into the asset library, where you are working on a file and want to copy the asset from that file into the library. This is a concept that appears deceptively simple. In certain cases it is actually simple, but often enough it gets quite complex. For example, when you want to push an object into an external asset library, should that also copy the materials? What about the texture images referenced by those materials? What about objects referenced by custom properties, constraints, or modifiers? And in which files would they have to go? Do they all go into one big `assets.blend`, individual blend-files, or into a directory per asset type? Blender should not be making such decisions for you.

For specific cases, these things are all solvable. For this reason the pose library has been created as an add-on which is enabled by default. Studios with specific needs can disable the add-on and implement their own

functionality; the building blocks are all in Blender's core, and thus do not need to be copied for this. Furthermore, add-ons can write to other blend-files, so they could make the decisions for users.

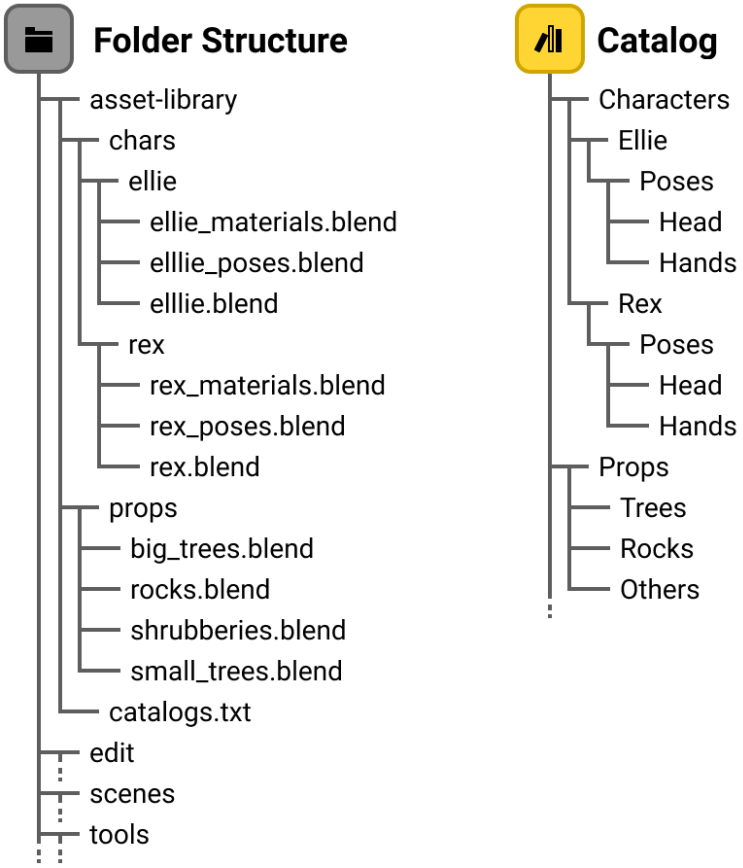
Asset pushing is desirable. Because of the questions above, it is unknown how to implement this well, in a way that still allows artists control over their assets.

Asset Catalogs

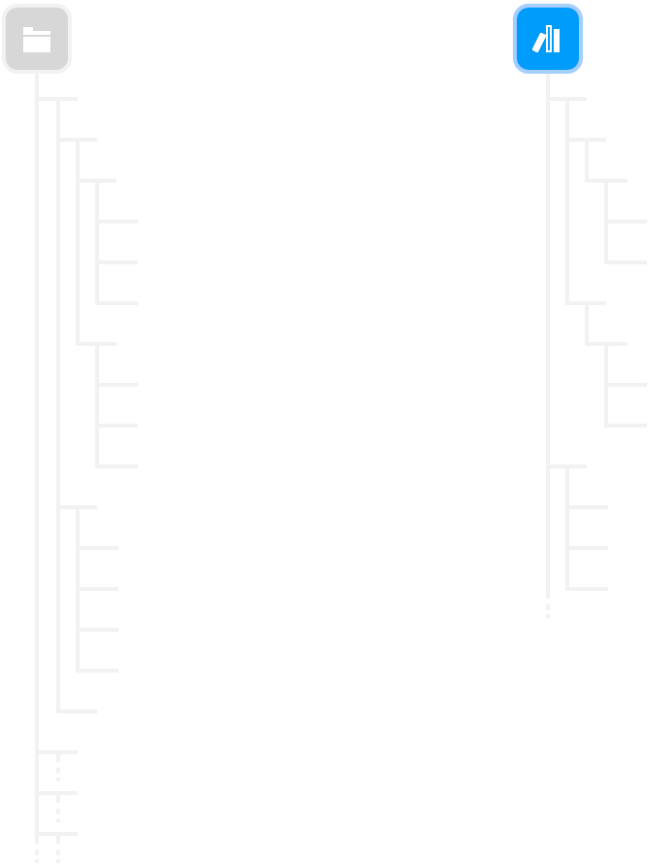
Asset Catalogs help you to organize your assets. They look a little bit like file directories, but they are completely independent of the location of your blend-files. Assign each asset in a blend-file to its own catalog, or have one big catalog with all the assets of all the blend-files combined. It's all up to you.

Similar to Collections, catalogs can be nested i.e. you can have a main catalog that contains several nested catalogs. For example, this allows you to have a catalog of assets for "Furniture" with sub-catalogs of "Tables", "Chairs", "Lamps", etc…

For more technical information, see Asset Catalogs on the Blender Developer Documentation.



Example file system and catalog structures.



Example file system and catalog structures.

The Home Location of Assets

There can be as many catalogs as you want, but **an asset can be assigned to a single catalog** at a time. This is similar to a file system, where a file is only in one directory (ignoring advanced things like symbolic links).

Catalogs themselves can be nested and moved by dragging and dropping. Moving a catalog will not modify the assets it contains; they will simply move along to the new location of the catalog.

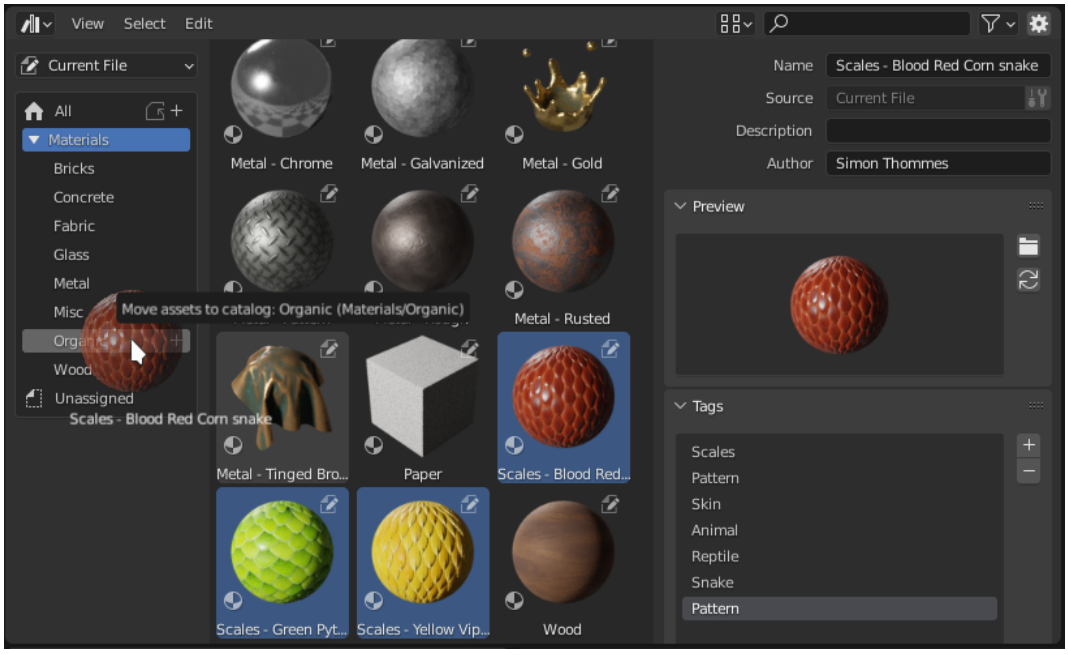
Selecting a catalog in the Asset Browser will show all assets in that catalog and in child catalogs. So, in the preceding example, selecting

`Characters/Ellie/Poses` will also show assets from `Characters/Ellie/Poses/Head` and `Characters/Ellie/Poses/Hands`.

Creating Catalogs

New catalogs can be created in the Asset Browser through *Header* ▸ *Catalog* ▸ *New Asset Catalog*. Once the catalog is created you can double `LMB` on it's name in the *Source List* region of the editor to give the catalog a more descriptive name. Catalogs can also be created in this region by clicking the plus icon found at the top of the tree view.

Assigning an Asset



Assigning a selection of "Scale material" assets to a catalog.

To assign assets to a catalog, just select and drag the assets on top of the catalog.

Tip

You can assign an asset to the “Unassigned” catalog, this will remove it from any existing catalogs.

Saving Catalogs

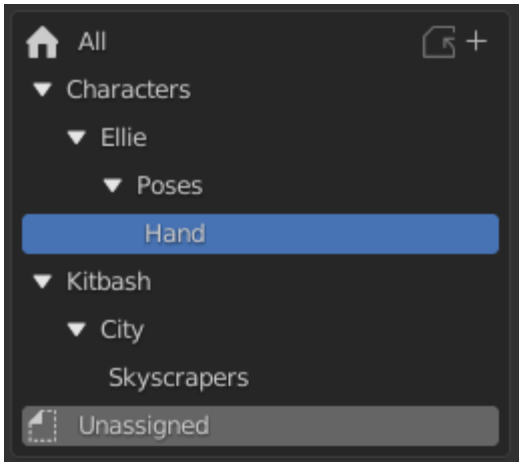
Saving catalogs makes any edits to any catalogs permanent by writing the current set up to the asset library. Catalogs can be saved in the Asset Browser through *Header* ▸ *Catalog* ▸ *Save Asset Catalog*. Once the catalog is created you can double `LMB` on it’s name Catalogs can also be saved in the *Source List* region of the editor by clicking the save icon found at the top of the tree view.

Components of a Catalog

Each catalog consists of a *catalog path*, a *UUID*, and a *simple name*. Normally you would only deal with the catalog path; the rest is for internal Blender use and/or for emergency situations.

Catalog Path

The path of a catalog determines where in the catalog hierarchy the catalog is shown. Examples are `Characters/Ellie/Poses/Hand` or `Kitbash/City/Skyscrapers`, which would result in the following catalog tree. The highlighted catalog has path `Characters/Ellie/Poses/Hand`.



Example tree of asset catalogs.

UUID

Each catalog has a UUID, which is normally hidden from the user interface (enable Developer Extras and the experimental Asset Debug Info option to see them). This is what is stored in the asset, and what determines the "identity" of the catalog. As a result, a catalog can be renamed or moved around (i.e. you can change its path), and all assets that are contained in it will move along with it. This only requires a change to the catalog itself, and not to any asset blend-file.

Simple Name

Each catalog has an optional *simple name*. This name is stored along with the UUID in each asset. The purpose is to make it possible for humans to recognize the catalog the asset was assigned to, even when the *catalog definition file* (see below) is lost.

Like the UUID, the simple name is normally hidden from the user interface. Enable Developer Extras in the interface preferences to make it visible in the Asset Browser.

Catalog Definition Files

Asset catalogs are stored in Catalog Definition Files (CDFs). Blender 3.0 supports a single CDF per asset library. It is stored in `blender_assets.cats.txt` in the root directory of the asset library. If the file does not exist, Blender will create it when the catalogs are saved. When catalogs are changed, Blender updates that file, but also creates a backup of the previous state to a file named `blender_assets.cats.txt~`.

Which File to Write To

Asset catalogs can be saved independently of the blend-file; the catalog editor has its own "Save" button.

Format

Catalog Definition Files (CDFs) are relatively simple text files, encoded in UTF-8. Each CDF consists of a version indicator, and a line of text per catalog. Each catalog line is colon-separated, of the form `{UUID}:{path}:{simple name}`.

Example

This is an example of a valid catalog definition file:

```
# This is an Asset Catalog Definition file for Blender.
#
# Empty lines and lines starting with `#` will be ignored.
# The first non-ignored line should be the version indicator.
# Subsequent lines are of the format
"CATALOG_UUID:catalog/path/for/assets:simple catalog name"

VERSION 1

313ea471-7c81-4de6-af81-
fb04c3535d0e:catalog/without/simple/name:
ee9c7b60-02f1-4058-bed6-
539b8d2a6d34:character/Ellie/poselib:character-Ellie-poselib
cd66bf52-58f4-45cb-a4e2-
dc0e0ee8f3fe:character/Ellie/poselib:character-Ellie
4eb44ec6-3424-405b-9782-
ca006953e799:character/Ellie/poselib/white space:character-
Ellie-poselib-white space
b63ed357-2511-4b96-8728-
1b5a7093824c:character/Ružena/poselib:Ružena pose library
dcdee4df-926e-4d72-b995-
33106983bb9a:character/Ružena/poselib/face:Ružena face
fb698f2e-9e2b-4146-a539-
3af292d44899:character/Ružena/poselib/hand:Ružena hands
```

Valid Catalog Paths

Catalog paths follow the following rules:

- All paths are absolute; there is no difference between `/a/b` and `a/b`.
- Only `/` as separator (no `\`; think less filesystem path and more URL).
- Not empty (it's required for a valid catalog).
- No empty components (so not `a//b`; `a/b` is fine).
- Invalid characters: `:`, `\`.
- Paths are always interpreted as UTF-8.

Media Formats

- Supported Graphics Formats
  - Image Formats
  - Opening Images
  - Saving Images
  - Format Details
- Supported Video & Audio Formats
  - FFmpeg Containers
  - FFmpeg Video Codecs
  - FFmpeg Audio Codecs
  - Known Limitations

Supported Graphics Formats

Image Formats

This is the list of image file formats supported internally by Blender:

Format	Channel Depth	Alpha	Metadata	DPI	Extensions
BMP	8bit	✗	✗	✓	`.bmp`
Iris	8, 16bit	✓	✗	✗	`.sgi` `.rgb` `.bw`
PNG	8, 16bit	✓	✓	✓	`.png`
JPEG	8bit	✗	✓	✓	`.jpg` `.jpeg`
JPEG 2000	8, 12, 16bit	✓	✗	✗	`.jp2` `.jp2` `.j2c`
Targa	8bit	✓	✗	✗	`.tga`
Cineon & DPX	8, 10, 12, 16bit	✓	✗	✗	`.cin` `.dpx`

Format	Channel Depth	Alpha	Metadata	DPI	Extensions
OpenEXR	float 16, 32bit	✓	✓	✓	`.exr`
Radiance HDR	float	✓	✗	✗	`.hdr`
TIFF	8, 16bit	✓	✗	✓	`.tif` `.tiff`
WebP	8bit	✓	✓	✓	`.webp`

Hint

If you are not interested in technical details, a good rule of thumb for selecting output formats for your project is:

Use OpenEXR
: if you intend to do compositing or color grading on these images.

Use PNG
: if you intend on-screen output or encoding into multiple video formats.

Use JPEG
: for on-screen output where file size is a concern and quality loss is acceptable.

All these formats support compression which can be important when rendering out animations.

Hint

Bit depths for image formats represent the following numbers of tonal levels per channel:

8::	256 levels
10::	1024 levels
12::	4096 levels
16::	65536 levels

Opening Images

Relative Path
: Sets the file path to be relative to the currently opened blend-file.

  See Relative Paths.

Detect Sequences
: Automatically looks for image sequences in the selected images (based on the file name). Disable this when you do want to get single images that are part of a sequence.

Detect UDIMs
: Automatically looks for UDIM tiles in the directory of the selected image; if matches are found they are loaded into Blender as UDIMs. This works by detecting if the filename has a `.xxxx` (four digit number) before the file extension.

Opening an Image Sequence

To load image sequence in any of the supported image file formats, the filename of the images must contain a digit to indicate the frame order (e.g. `*-0001.jpg`, `*-0002.jpg`, `*-0003.jpg`, etc, of any image format), indicating the frame.

The sequence could be opened by the selection of the images with any of the following methods by the confirmation with the *Open Image* button or `Return`.

Range
: Navigate into the directory and `LMB` click and drag over a range of names to highlight multiple files. You can page down and continue `Shift-LMB` click-dragging to add more to the selection.

Batch
: `Shift-LMB` click selected non-related stills for batch processing; each image will be one frame, in sort order, and can be a mix of file types (`jpg`, `png`, `exr`, etc.).

All
: Press `A` to select/deselect all files in the directory.

Saving Images

File Format
: Choose what format to save the image as.

Color Mode
: Choose the color format to save the image (or video) to. Note that *RGBA* is not available for all image formats, check the list above for details.

  BW, RGB, RGBA

Color Depth
: Some image file formats support a varying number of bits per pixel. This affects the color quality and file size. Commonly used depths:

  8-bit:: Most common for on-screen graphics and video.
  10, 12, 16-bit:: Used for some formats focusing on photography and digital films (such as DPX and JPEG 2000).
  16-bit Half Float:: Since full 32bit float is often more than enough precision, half float can save drive space while still providing a high dynamic range.
  32-bit Float:: Highest quality color depth.

Note

Internally Blender's image system supports either:

- 8 bits per channel (4 × 8 bits).
- 32 bits float per channel (4 × 32 bits) – *using 4 times as much memory.*

Images higher than 8 bits per channel will be converted into a float on loading into Blender.

Compression

Used to reduce the size of the image file. How this is done may vary depending on the file format and settings used.

Quality

Similar to *Compression* but is used for JPEG based file formats. The quality is a percentage, 0% being the maximum amount of compression and 100% is no compression.

Save As Render

Save image with render color management. For display image formats like PNG, apply view and display transform. For intermediate image formats like OpenEXR, use the default render output color space.

Copy

The Copy checkbox will define if the data-block will reference the newly created file or the reference will be unchanged, maintaining it with the original one.

Color Space

To specify the color space of the source file.

The list of color spaces depends on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

Note

Note, *Cineon*, *DPX*, *OpenEXR*, and *Radiance HDR* image types default to being saved in a linear color space.

Format Details

Cineon & DPX

Cineon is Kodak's standard for film scanning, 10 bits per channel and logarithmic. DPX has been derived from Cineon as the ANSI/SMPTE industry standard. DPX supports 16-bit colors/channels, linear as well as logarithmic. DPX is currently a widely adopted standard used in the film hardware/software industry.

DPX as well as Cineon only stores and converts the "visible" color range of values between 0.0 and 1.0 (as a result of rendering or composite).

OpenEXR

ILM's OpenEXR has become a software industry standard for HDR image files, especially because of its flexible and expandable structure.

An OpenEXR file can store multiple layers and passes. This means OpenEXR images can be loaded into a Compositor keeping render layers and passes intact.

Output Options

Available options for OpenEXR render output are:

Color Depth

Half saves images in a custom 16 bits per channel floating-point format. This reduces the actual "bit depth" to 10-bit, with a 5-bit power value and 1-bit sign.

Float (Half), Float (Full)

Codec

PXR24:: Lossy algorithm from Pixar, converting 32-bit floats to 24-bit floats.

ZIP:: Standard lossless compression using Zlib, operating on 16 scanlines at a time.

PIZ:: Lossless wavelet compression. Compresses images with grain well.

RLE:: Run-length encoded, lossless, works well when scanlines have same values.

ZIPS:: Standard lossless compression using Zlib, operating on a single scanline at a time.

DWAA:: JPEG-like lossy algorithm from DreamWorks; compresses blocks 32 scanlines together.

DWAB:: Same as `DWAA` but compresses blocks of 256 scanlines.

Preview

On rendering animations (or single frames via command line), Blender saves the same image also as a JPEG, for quick preview or download.

Radiance HDR

Radiance is a suite of tools for lighting simulation. Since Radiance had the first (and for a long time the only) HDR image format, this format is supported by many other software packages.

Radiance `.hdr` files store colors still in 8 bits per component, but with an additional (shared) 8-bit exponent value, making it 32 bits per pixel.

Supported Video & Audio Formats

Blender used FFmpeg to handle video encoding/decoding various video formats. These formats are primarily used for compressing rendered sequences into a playable movie. Video formats are composed of a container, a codec, and sometimes audio which is stored using its own codec. The roll of the container to encapsulate video and audio data that is compressed using a codec.

Codecs compress the channels of a video down to save space and enable continuous playback. *Lossy* codecs make smaller files at the expense of image quality, while *lossless* codecs compress as much as possible the video/audio, but without losing any existing data.

Some codecs, like H.264, are great for larger images. Codecs are used to encode and decode the movie, and so must be present on both the encoding machine (Blender) and the target machine. The results of the encoding are stored in a container file.

There are dozens, if not hundreds, of codecs, including Xvid, H.264, DivX, Microsoft, and so on. Each has advantages and disadvantages, and compatibility with different players on different operating systems.

Note

Most codecs can only compress the RGB or YUV colors, but some support the Alpha channel as well. Codecs that support RGBA include:

- FFmpeg video codec #1
- PNG
- QuickTime Animation
- WebM/VP9 (although Blender will not import the alpha channel due to a limitation of FFmpeg).

FFmpeg Containers

MPEG-1:: A standard for lossy compression of video and audio. It is designed to compress VHS-quality raw digital video and CD audio down to 1.5 Mbit/s. This container enforces the video codec, you can only define quality parameters, and the audio codec.
File Extensions: `.mpg`, `.mpeg`

MPEG-2:: A standard for "the generic coding of moving pictures and associated audio information". It describes a combination of lossy video compression and lossy audio data compression methods which permit storage and transmission of movies using currently available storage media (notably DVDs) and transmission bandwidth. This container enforces the video codec, you can only define quality parameters, and the audio codec.
File Extensions: `.dvd`, `.vob`, `.mpg`, `.mpeg`

MPEG-4:: While being a video codec, it is also a real container, in which you can store video and audio streams using various codecs. It is widely supported by many modern software and hardware players.
File Extensions: `.mp4`, `.mpg`, `.mpeg`

AVI:: A derivative of the Resource Interchange File Format (RIFF). One of the first and most widely used video container format.
File Extension: `.avi`

QuickTime:: A multi-tracks format. QuickTime and MP4 container formats can use the same codecs. They are mostly interchangeable in a QuickTime-only environment. MP4, being an international standard, has more support.
File Extension: `.mov`

DV::

An intra-frame video compression scheme, used by many digital camcorders back in the days. It uses the discrete cosine transform (DCT, similar algorithm to JPEG) to compress video on a frame-by-frame basis. Audio is stored uncompressed. This container enforces the video codec, you can only define quality parameters.
File extension: `.dv`

Ogg:: A free open-standard container format, that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.
File Extensions: `.ogg`, `.ogv`

Matroska:: A free open-standard container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file.
File Extension: `.mkv`

Flash:: A container file format used to deliver video over the Internet using Adobe Flash Player. This container enforces the video codec, you can only define quality parameters.
File Extension: `.flv`

WebM:: A free open-standard container format, designed to be used for internet streaming. Note that this container can only hold a VP9 video codec, and Vorbis or Opus audio codecs.
File Extension: `.webm`

FFmpeg Video Codecs

These options are not available with all Containers.

No Video:: For audio-only encoding.

DNxHD:: Intended to be usable as both an intermediate format suitable for use while editing, and as a presentation format. It can be either lossless or lossy.

DV:: See Containers.

FFmpeg video codec #1::
FFV1 is a lossless intra-frame video codec. It can use either variable length coding or arithmetic coding for entropy coding. The encoder and decoder are part of the free, open-source library libavcodec in FFmpeg. Supports an alpha channel.

Flash Video:: See Containers.

H.264:: A modern variation of the MPEG-4 family, this lossy codec is very commonly used. It offers a very good compression/quality ratio.

HuffYUV:: Lossless video codec created by Ben Rudiak-Gould which is meant to replace uncompressed YCbCr as a video capture format.

MPEG-1:: See Containers.

MPEG-2:: See Containers.

MPEG-4(DivX):: Inherits many of the features of MPEG-1, MPEG-2 and other related standards, but also adds new features.

PNG:: Lossless, this stores each frame as an independent image in the video stream. Compression will be poor, but as every frame is fully self-contained, scrubbing and editing can be simpler. Supports an alpha channel.

QuickTime Animation::
Original format of QuickTime videos. Supports an alpha channel.

Theora:: A free open-standard lossy codec designed together with the Ogg container.

WEBM / VP9:: A free open-standard lossy video compression format. One of the most recent codecs, it is widely used for internet streaming.

AV1:: A free open-standard lossy video compression format, designed as a successor to *VP9*. AV1 offers great compression rates and visual quality, *AV1* produces video files that are about 30% more space efficient than *VP9*

FFmpeg Audio Codecs

No Audio::	For video-only encoding.
AAC::	Advanced Audio Codec, a standardized, lossy compression and encoding scheme for digital audio. AAC generally achieves better sound quality than MP3 at similar bit rates.
AC3::	Audio Codec 3, an audio compression technology developed by Dolby Laboratories.
FLAC::	Free Lossless Audio Codec. Digital audio compressed by FLAC's algorithm can typically be reduced to 50-60% of its original size.
MP2::	A lossy audio compression format.
MP3::	A lossy audio compression format, widely used as final audio format.
Opus::	A lossy audio compression format, designed to encode speech or general audio and is intended to replace the *Vorbis* codec.
PCM::	Pulse Code Modulation, a method used to digitally represent sampled analog signals. It is the standard form for digital audio in computers and various Blu-ray, Compact Disc and DVD formats, as well as other uses such as digital telephone systems.
Vorbis::	An open-standard, highly-compressed format comparable to MP3 or AAC. Vorbis generally achieves better sound quality than MP3 at similar bit rates.

Known Limitations

Video Output Size

Some codecs impose limitations on output size, `H.264`, for example requires both the height and width to be divisible by 2.

Importing & Exporting Files

Reference

Menu:: *Topbar ▸ File ▸ Import/Export*

Sometimes you may want to utilize files that either came from other 2D or 3D software, or you may want to use the things you have made in Blender and edit them in other software. Luckily, Blender offers a wide range of file formats (e.g. ABC, USD, OBJ, FBX, PLY, STL, etc.) that can be used to import and export.

Popular formats are enabled by default, other formats are also supported and distributed with Blender, these can be enabled in the Preferences through the use of Add-ons.

- Alembic
- Collada (Legacy)
- Universal Scene Description
- Wavefront OBJ
- Stanford PLY
- STL
- Import/Export SVG as Grease Pencil
- Export Grease Pencil as (PDF)

See also

More information on the add-ons to import/export these file types can be found in the add-ons section.

Alembic

From the Alembic home page:

> Alembic is an open computer graphics interchange framework. Alembic distills complex, animated scenes into a non-procedural, application-independent set of baked geometric results. This 'distillation' of scenes into baked geometry is exactly analogous to the distillation of lighting and rendering scenes into rendered image data.
>
> Alembic is focused on efficiently storing the computed results of complex procedural geometric constructions. It is very specifically **not** concerned with storing the complex dependency graph of procedural tools used to create the computed results. For example, Alembic will efficiently store the animated vertex positions and animated transforms that result from an arbitrarily complex animation and simulation process which could involve enveloping, corrective shapes, volume-preserving simulations, cloth and flesh simulations, and so on. Alembic will not attempt to store a representation of the network of computations (rigs, basically) which are required to produce the final, animated vertex positions and animated transforms.

In brief, Alembic can be used to write an animated mesh to a drive, and read it back quickly and efficiently. This means that a mesh can be animated with a very CPU-intensive rig and then 'baked' to an Alembic file. Finally it can be load into the shot file for shading and lighting with only moderate CPU usage.

Due to the Open Source nature of the Alembic standard as well as the C++ library implementing that standard, **Blender can be used in a hybrid pipeline**. For example, other software, such as Houdini or Maya, can export files to Alembic, which can then be loaded, shaded, and rendered in Blender. It is also possible to animate characters (or other models) in Blender, export to Alembic, and load those files into other software for further processing.

Importing Alembic Files

When importing an Alembic file, Mesh Sequence Cache modifiers are automatically added to time-varying meshes. For time-varying object transforms (so animation of rotation, location, or scale) the Transform Cache Constraint is used.

General

Scale
: Value by which to enlarge or shrink the objects with respect to the world's origin

Options

Relative Path
: Select the file relative to the blend-file.

Set Frame Range
: If checked, update scene's start and end frame to match those of the Alembic archive.

Is Sequence
: Set to true if the cache is split into separate files.

Validate Meshes
: Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

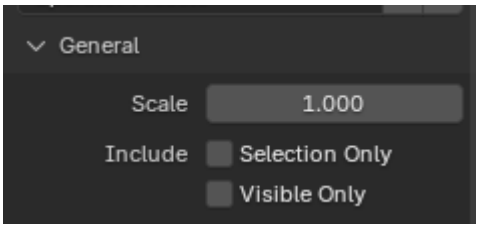
Always Add Cache Reader
: Add cache modifiers and constraints to imported objects even if they are not animated so that they can be updated when reloading the Alembic archive.

Exporting to Alembic Files

This section describes the effect of the different export options.

General

Scale
: This sets the global scale of the Alembic file. Keep it at the default value of 1.0 to use Blender's units.



General options.

Include
: Selected Objects
  : When enabled, exports only the selected objects. When disabled, all objects are exported.

  Visible Objects
  : Limits the export to scene collections that are currently visible.

Scene

Frame Start, End
: Sets the frame range to export to Alembic. This defaults to the current scene frame range.

Sub-frame Sampling
: These options control the sub-frame sampling of animations.

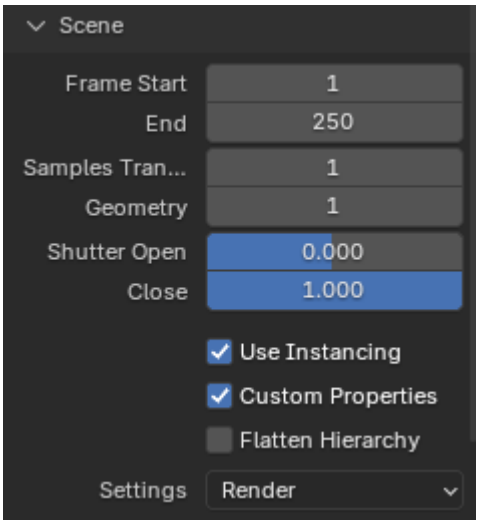
  Samples Transform
  : Transform Samples sets the number of times per frame at which animated transformations are sampled and written to Alembic.

  Geometry
  : Geometry Samples sets the same, but then for animated geometry.

  Shutter Open, Close

Shutter Open/Close define the interval [open, close] over which those samples are taken. The valid range is -1 to 1, where -1 indicates the previous frame, 0 indicates the current frame, and 1 indicates the next frame.

For example, if information for detailed mesh motion blur is desired, some subframes around the current frame can be written to Alembic by using a sample count of 5, Shutter Open at -0.25 and Shutter Close at 0.25. This mimics a "180 degree" shutter, opening 90 degrees before the frame and closing 90 degrees after the frame.



Scene options.

Use Instancing

Exports data of duplicated or instanced objects as Alembic instances; speeds up the export and can be disabled for compatibility with other software.

Custom Properties

When enabled (which it is by default), custom properties are exported to Alembic as well. The following custom property types are supported:

- Numbers (`int`, `float`) and strings. These are exported as arrays of a single element, so `47` will be exported as `[47]` to Alembic, and `"Agent"` to `["Agent"]`. This matches the behavior of many other DCCs.
- Lists of numbers and strings. These are exported as-is, so `[327, 47]` is exported as `[327, 47]`.
- Matrices and nested arrays of numbers. These are flattened into one long list, so a 3×2 matrix of numbers will become a list of 6 numbers. Similarly, nested lists `[[1, 2, 3], [4, 5], [6]]` will be exported as `[1, 2, 3, 4, 5, 6]`.
- Numbers can be animated as well.

Flatten Hierarchy

When disabled, parent/child relations between objects are exported too. Any parent object that is not exported itself, but with children that *are* exported, is replaced by an empty. When enabled, parent/child relations are not exported, and transformations are all written in world coordinates.

Settings

Determines visibility of objects, modifier settings, and other areas where there are different settings for viewport and rendering.

Render:: Use Render settings for object visibility, modifier settings, etc.

Viewport:: Use Viewport settings for object visibility, modifier settings, etc.

Geometry

UV Coordinates

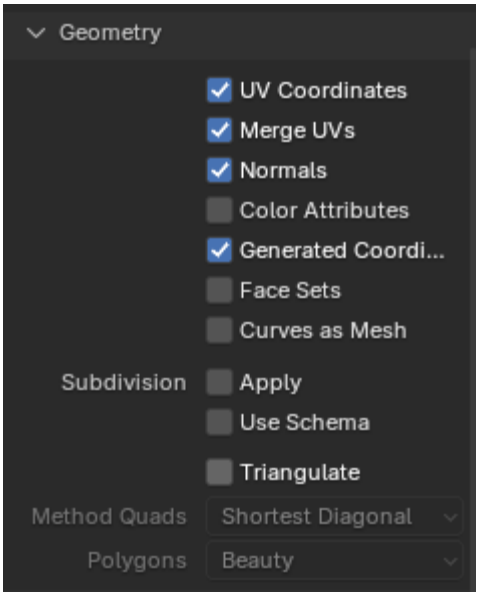
When enabled, UV maps are exported. Although the Alembic standard only supports a single UV map, Blender exports all UV maps in a way that should be readable by other software.

Merge UVs

When enabled, UVs sharing the same vertex and location will be merged into a single UV. The exported file can be slightly smaller.

Normals

When enabled, an object's Normals are exported. See Custom Split Normals of Meshes below for more information.



Geometry options.

Color Attributes

When enabled, exports Color Attributes.

Face Sets
: Exports the material names per face. The material data is not exported but only material names.

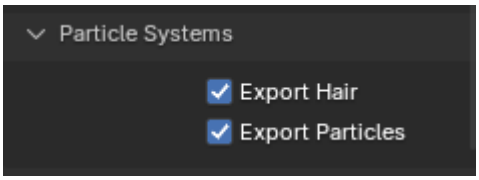
Subdivisions
: Apply
: Applies any Subdivision Surface modifiers before writing to Alembic.

: Use Schema
: Writes polygonal meshes using the “SubD” Alembic schema, rather than the “PolyMesh” schema. This sets an import option for the program, with which the file is opened, to apply its form of a non-destructive subdivision.

Triangulate
: Triangulates the mesh before writing to Alembic. For more detail on the specific option see the Triangulate modifier.

Particle Systems

Alembic has no support for Particle Systems, in the same way that it does not support armatures.



Particle Systems options.

Export Hair
: Hair is exported as animated zero-width curves.

Export Particles
: Particles are exported as animated points.

Custom Split Normals of Meshes

Blender supports the import and export of custom normals to Alembic files. As a basic rule of thumb, a completely smooth mesh will be exported without normals and thus produce the smallest Alembic file. This is reflected in the importer; an Alembic mesh without normals is loaded as a smooth mesh.

On export, for every mesh:

- If it has *Custom Loop Normals* then the loop normals are exported.
- If one or more polygons are marked flat then loop normals are also exported.
- Otherwise, no normals are exported.

On import, when the Alembic mesh contains:

- Loop normals (`kFacevaryingScope`) are used as custom loop normals.
- Vertex normals (`kVertexScope` or `kVaryingScope`) are convert to loop normals, and handle as above.
- If there are no normals then the mesh is marked as smooth.
- Unsupported normal types (`kConstantScope`, `kUniformScope`, `kUnknownScope`) are handled as *no normals*.

When an imported mesh does not contain normals, the final look can be controlled using the Normal's Shading.

Handling Time

Unlike Blender and many other applications and file formats, Alembic files don't have any concept of frames. Alembic works purely with time, and values that are sampled over time. For example, there is no way to distinguish 30 FPS with 2 samples per frame, and 60 FPS with 1 sample per frame. This has caused many developers to just hard-coded 24 FPS when reading Alembic files.

Blender uses the current scene frame rate to convert a frame number (in Blender) to a time in seconds (in Alembic). As a result, you can import an

Alembic file that was produced at 120 FPS into a Blender scene that is 30 FPS and still not see any time stretching.

Collada (Legacy)

Important

COLLADA I/O support is now considered as a legacy feature in Blender, and will be removed in a future release. Please see the official announcement for more insight on this topic.

The COLLADA™ module has been implemented as a flexible tool for exporting and importing `.dae` files. A design goal is to provide a set of parameters which should make it possible to export/import Collada files from/to a variety of tools. But please be aware that the Collada module is still a work in progress. So it may be possible that your particular usage scenario is not yet supported.

Collada Exporter

Reference

Menu:: *File ▸ Export Collada (.dae) (Legacy)*

Operator Presets

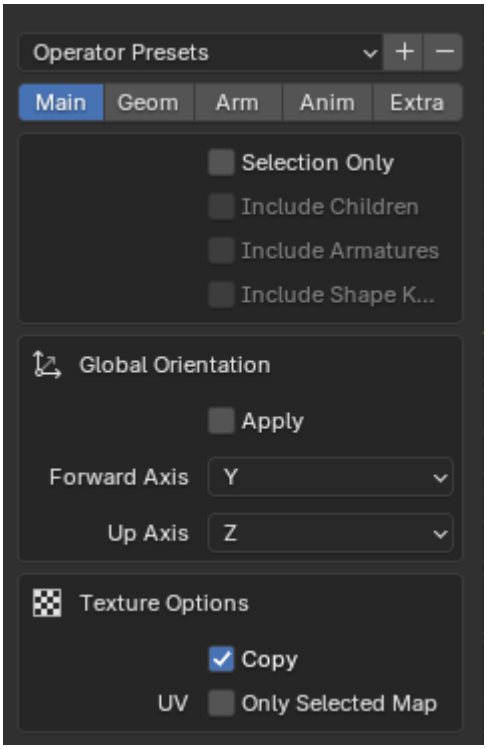
There are two operator presets (see top of the Sidebar) for Second Life (SL) users:

- *Second Life Static* – is good for exporting static meshes.
- *Second Life Rigged* – is good for exporting the SL default character.

Note

Special Notes for Second Life users:

- Please use the Operator presets. All other export settings will not work for Second Life.
- The character orientation needs to be such that the character looks towards positive X.
- Scale and Rotation must be applied before the export!



Main

Selection Only

When *Selection Only* is enabled, then only the selected objects will be exported. Otherwise the entire scene is exported with all visible and all invisible objects.

Include Children

When this option is enabled then all children of the selected objects will also be exported regardless of their selection state.

Hint

You can select only an armature, then using this option, all rigged meshes attached to the armature will also be exported.

Include Armatures

When this option is enabled, then all armatures related to the selected objects will also be exported regardless of their selection state.

Hint

You can select only the objects, then in the exporter enable this option to export the armature data also.

Include Shape Keys

Includes the application of shape keys by exporting meshes with the current shape key configuration baked in.

Global Orientation

Apply

Rotate all root objects to match the global orientation settings otherwise set the global orientation per Collada asset.

Forward / Up Axis

Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, it is common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Texture Options

Copy

When you export images either material based image textures, then the exporter creates absolute file references in the export file.

But if the *Copy* option is enabled, the exporter will create copies of the images instead and place the copies besides the export file. In that case the file references are made relative.

Only Selected UV Map

When your mesh contains multiple UV layers, then Blender exports all layers by default. This option allows you to only export the active (selected) UV layer.

Geometry

Export Data Options

Triangulate
: The mesh can be triangulated on-the-fly. The triangulation is based on the same function which is used in the *Triangulate Faces* tool for triangulating the current selection of faces. For full control over the triangulation you can do this manually before exporting. However, this option allows to apply the triangulation only on the exported data; the mesh itself is not affected.

Apply Modifiers
: Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

  Resolution
  : Controls whether to apply the 3D Viewport resolution or the render resolution for modifiers that provide a preview mode and a render mode.

Transform
: Collada supports two types of transformation matrix specifications. Either as `<Matrix>` or as a set of transformation decompositions (for move, rotate and scale). Note that the exporter will not strictly follow this option setting, but will rather take it as a hint to use the option if ever possible. This is so because some of the exported data types have specific rules about how the transformation matrix has to be exported. This is ongoing development and a less ambiguous method may be provided in the future.

Armature

Armature Options

Deform Bones Only

When this option is enabled, then the exporter strips all non-deforming bones from the exported armatures. This option is useful when your armatures contain control bones which are not actually part of the character skeleton. For example you can export the Avastar rig with this option enabled. The resulting exported rig is compatible with Second Life. But please note the restrictions further below.

Export to SL/OpenSim

When this option is enabled, some issues with bone orientation are calculated differently and is designed to be used to export to Second Life or OpenSim.

This is only relevant for rigged meshes, for static meshes it just does nothing at all.

Animation

Extra

Collada Options

Use Object Instances

In Blender you can reuse the same mesh for multiple objects. This is named “object instantiation”. When you enable this option, then Blender will propagate object instantiation to the Collada file.

Use Blender Profile

Collada can be extended with tool specific data (profiles). Blender has its own (unofficial) profile that allows to export rig information into the Collada file. Later It can be used to reconstruct the rig when it should ever be necessary to import a dae file back into Blender.

Sort by Object Name

The export order of data is bound to internal object order and it can not be influenced in a reliable way. This option ensures that the Geometry nodes and the Object nodes are both exported in alphabetical order.

Keep Bind Info

When a rig is imported to Blender, the rig's bind pose will be used as Blender's rest pose. So all Matrix information of the original rest pose is lost. But in some cases you may want to preserve the original rig information. This option checks each bone for having two arrays:

- `rest_mat` – an array of 16 floats which represent the bone's original rest-pose matrix.
- `bind_mat` – an array of 16 floats which represent the bone's original bind-pose matrix.

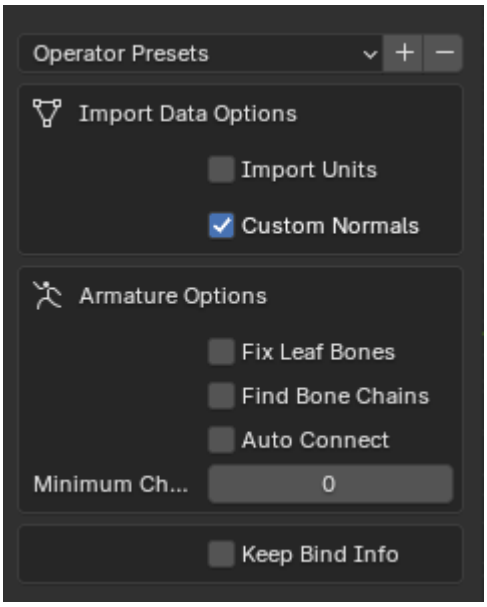
If the arrays are present, then those arrays will be used instead of the current rest pose/bind pose. Those two arrays are either created by a previous Collada import (see Collada Importer below), or they can be created manually, or by an add-on (script based).

Collada Importer

Reference

Menu:: *File ▸ Export Collada (.dae) (Legacy)*

The Collada importer is mostly driven by the imported data. There is one option for controlling the import units:



Import Data Options

Custom Normals
: Use the mesh normals defined in the collada file, if they are defined, otherwise Blender will recompute them during the import process.

Import Units
: If not enabled the imported data will be rescaled according to the currently used unit system. If this option is enabled, then Blender will adjust itself to the unit system as provided by the Collada file.

Armature Options

Fix Leaf Bones
: Collada only records “joints” which is mostly similar to Blender’s bone heads. But when you import a Collada file then the bone head/tail are not defined. This does not matter for connected bones where the bone parent only has one child. In that case the parent bone’s end location is adjusted to the child’s joint position. But especially for unconnected bones and for bones with more than one child a problem arises.

  When the *Fix Leaf Bones* option is enabled then Blender tries to guess where the bone head/tail of unconnected bones would best be placed. If

the option is disabled, then the bone head/tail are placed at an offset along the Y axis. That is why bones often point towards the Y axis.

Find Bone Chains
: When a bone has multiple children, then it is not defined which (if any) of the children should be connected to the bone. When the *Find Bone Chains* option is enabled, then Blender determines the longest bone chain (of children) for each bone. All bones along this chain will then be auto connected.

  If the option is disabled, then children will only be connected to parents, if the parent has only one child. But see the *Auto Connect* option below.

Auto Connect
: When this option is enabled, then children will automatically be connected to their parents, if the parent has only one child.

Keep Bind Info
: When this option is enabled, then the importer creates two custom properties for each bone:

  - `rest_mat` – an array of 16 floats which represent the bone's original rest-pose matrix.
  - `bind_mat` – an array of 16 floats which represent the bone's original bind-pose matrix.

  Those two arrays can later be used when you want to export the rig again and be sure the original rest pose/bind pose combination must be used.

Technical Details

Mesh

Import

Supported geometry types are:

- Tris (not tested)
- Polylist
- Polygons
- N-gons
- Tri-fans (not tested)
- Lines

Export

Mesh data is exported as `<polylist>`, `<lines>` and `<vertices>`.

Light

Import

Blender does a best effort on importing lights from a dae-file. If a Blender profile is detected for lights, all values from these will be used instead. This ensures full re-import from a Blender exported dae-file. `<extra>` support has been added in Blender 2.57.

Export

A Blender profile for lights has been added through the `<extra>` tag. The entire Light struct from Blender will be exported through this profile, with the exception of light curve falloff.

Animation

Export & Import

- Support for object (mesh, camera, light) transform animations. Only Euler rotations, which is the default option for Objects, can be exported. For armature bone animations, Euler and quaternion rotation types are supported.

- Import and export of animations for the following parameters are supported:
  - Light
  - Camera
  - Material effects
- Non-skin controlling armature bone animation.
- Animations of armatures with skin deforming bones.
- Animations of armatures in Object Mode.
- Fully rigified armature animations (referring to the Rigify add-on). For export of rigified armature animations:
  - Run the Bake Action operator.
  - If you have only the deform bones selected check *Only Selected*. This will give smaller dae. Otherwise uncheck *Only Selected*.
  - Check *Clear Constraints*.
  - Bake Action.
  - Select the mesh and the deform bones. Then export to Collada while checking only selected option. (Selecting only the Mesh and bones is not strictly necessary. Selecting and export only selected will give smaller dae.)
  - Demonstration video

For bone nodes which are leaf nodes in the armature tree, or if a bone has more than one child, a Blender profile for tip with an `<extra>` tag, is added for those joint nodes. To correctly derive the bone-to-tail location on re-import.

Note

Important Things to Remember

- Object and data-block names are constrained to 21 characters (bytes).
- UV layer names are constrained to 32 characters (bytes).
- Only armature animation on mesh, single skin controller.
- No support for modifiers yet.

When importing a dae-file that has `<instance_node>` on exporting this information is essentially lost and these nodes will be `<node>`s.

Universal Scene Description

Importing USD Files

USD files typically represent the scene as a hierarchy of primitives, or prims. Individual prims contain data to describe scene entities, such as geometry, lights, cameras and transform hierarchies. Blender's USD importer converts USD prims to a hierarchy of Blender objects. Like the USD exporter, the importer does not yet handle more advanced USD concepts, such as layers and references.

The following USD data types can be imported as Blender objects:

- Cameras
- Curves
- Lights
- Materials
- Meshes
- Primitive Shapes
- Volume

For more information on how the various data types are handled, see the following descriptions of the Import Options.

Note

When importing a USDZ archive, it is important to carefully consider the Import Textures option to determine whether and how to copy texture files from the zip archive.

Xform and Scope Primitives

USD provides an `Xform` prim type, containing transform data, which can be used to represent transform hierarchies and to organize the scene. Such `Xform` prims are imported as Blender empty objects.

USD also supports `Scope` primitives, which are entities that do not contain transform data, but which serve to group other element of the scene. Blender doesn't have an exact counterpart to the concept of a scope, so such primitives are imported as Blender empties located at the origin. This is an imperfect representation, because empty objects have a transform and `Scopes` do not, but this approach nonetheless helps preserve the structure of the scene hierarchy.

PointInstancer Primitives

USD provides a `UsdGeomPointInstancer` prim type, containing instances that are scattered on a primitive's points.

These are imported into Blender as Point Clouds using a Geometry Nodes Modifier and the Instance on Points Node.

Animations

The importer supports two types of animation:

- **Animating transforms**: If a USD primitive has time-varying transform data, a Transform Cache constraint will be added to the imported Blender object.
- **Animating geometry**: Animating mesh and curve geometry is supported by adding a Mesh Sequence Cache modifier to the imported data. Geometry attribute (USD Primvar) animation is currently supported only for Color Attributes and UVs. Note that USD file sequences (i.e. a unique file per frame) are not yet supported.

Materials

If a USD mesh or geometry subset has a bound material, the importer will assign to the Blender object a material with the same name as the USD

material. If a Blender material with the same name already exists in the scene, the existing material may be used, depending on the Material Name Collision option. Otherwise, a new material will be created.

If the USD material has a USD Preview Surface shader source, the Viewport Display color, metallic, and roughness are set to the corresponding USD Preview Surface input values.

There is also an *Import USD Preview* option to convert USD Preview Surface shaders to Blender Principled BSDF shader nodes. This option can be lossy, as it does not yet handle converting all shader settings and types, but it can generate approximate visualizations of the materials.

Coordinate System Orientation

If the imported USD is Y up, a rotation will be automatically applied to root objects to convert to Blender's Z up orientation.

Import Options

The following options are available when importing from USD:

General

Path Mask
: Import only the subset of the USD scene rooted at the given primitive.

Include
: Visible Primitives Only
: Do not import invisible USD primitives. Only applies to primitives with a non-animated visibility attribute. Primitives with animated visibility will always be imported.

: Defined Primitives Only
: When disabled this allows importing USD primitives which are not defined, such as those with an override specifier.

Set Frame Range
: Update the scene's start and end frame to match those of the USD stage.

Create Collection
: Add all imported objects to a new collection.

Relative Path
: Select the file relative to the blend-file.

Scale
: Value by which to scale the imported objects in relation to the world's origin.

Light Intensity Scale
: Scale for the intensity of imported lights.

Custom Properties
: Behavior when importing USD attributes as Custom Properties.

None::	Does not import USD custom attributes.
User::	Imports USD attributes in the `userProperties` namespace as custom properties. The namespace will be stripped from the property names.
All Custom::	Imports all USD custom attributes as custom properties. Namespaces will be retained in the property names.

Object Types

Cameras
: Import cameras (perspective and orthographic).

Curves
: Import curve primitives, including USD basis and NURBS curves. (Note that support for Bézier basis is not yet fully implemented.)

Lights

Import lights. Does not currently include USD dome, cylinder or geometry lights.

Materials
: Import materials.

Meshes
: Import meshes.

Volumes
: Import USD OpenVDB field assets.

Point Clouds
: Imports USD `UsdGeomPoints` as a Point Cloud object.

USD Shapes
: Imports USD primitive shapes (cubes, spheres, cones, ect) as Blender meshes.

Display Purpose
: Render
  : Include primitives with purpose `render`.

  Proxy
  : Include primitives with purpose `proxy`.

  Guide
  : Include primitives with purpose `guide`.

Material Purpose
: Attempt to import materials with the given purpose. If no material with this purpose is bound to the primitive, fall back on loading any other bound material.

  All Purpose:: Attempt to import `allPurpose` materials.
  Preview:: Attempt to import `preview` materials. Load `allPurpose` materials as a fallback.
  Full::

Attempt to import `full` materials. Load `allPurpose` or `preview` materials, in that order, as a fallback".

Geometry

UV Coordinates
: Read mesh UV coordinates.

Color Attributes
: Convert the USD mesh `displayColor` values to Blender's Color Attributes.

Mesh Attributes
: Read USD `Primvars` as mesh attributes.

Subdivision
: Create Subdivision Surface modifiers based on the USD `SubdivisionScheme` attribute.

Validate Meshes
: Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Rigging

Shape Keys
: Imports USD blend shapes as Blender's Shape Keys.

Armatures
: Imports USD skeletons as Blender's Armatures.

Materials

Import All Materials

Also import materials that are not used by any geometry. Note, when this option is false, materials referenced by geometry will still be imported.

Import USD Preview
: Convert USD Preview Surface shaders to Principled BSDF shader networks.

Create World Material
: Converts the first discovered USD dome light to a world background shader.

Set Material Blend
: If the *Import USD Preview* option is enabled, the material blend method will automatically be set based on the `opacity` and `opacityThreshold` shader inputs, allowing for visualization of transparent objects.

Material Name Collision
: Behavior when the name of an imported material conflicts with an existing material.

  Make Unique:: Import each USD material as a unique Blender material.

  Reference Existing:: If a material with the same name already exists, reference that instead of importing.

Textures

When importing a USDZ package, the following options specify whether and how texture asset dependencies of the USD should be copied from the zip archive so they can be loaded into Blender.

Import Textures
: Behavior when importing textures from a USDZ archive.

None:: Don't import textures. Note that, with this option, material textures may fail to be resolved in Blender.

Packed:: Import textures as packed data in the Blender file.

Copy:: Copy files to the directory specified in the **Textures Directory** option.

Textures Directory

Path to the directory where imported textures will be copied, when the **Import Textures** mode is **Copy**.

Note that the default textures directory is the relative path `//textures`, which requires the Blender file to have been saved before importing, so the relative path can be resolved.

File Name Collision

Behavior when the name of an imported texture file conflicts with an existing file.

Use Existing:: If a file with the same name already exists, use that instead of copying.

Overwrite:: Overwrite existing files.

Particles and Instancing

Scene Instancing

Import USD scene graph instances as collection instances, otherwise they are imported as copies.

Exporting to USD Files

Universal Scene Description (USD) files can contain complex layering, overriding, and references to other files. Blender's USD Exporter takes a much simpler approach. When exporting, all visible, supported objects in the scene are exported, optionally limited by their selection state. Blender does not (yet) support exporting invisible objects, USD layers, variants, etc.

The following objects can be exported to USD:

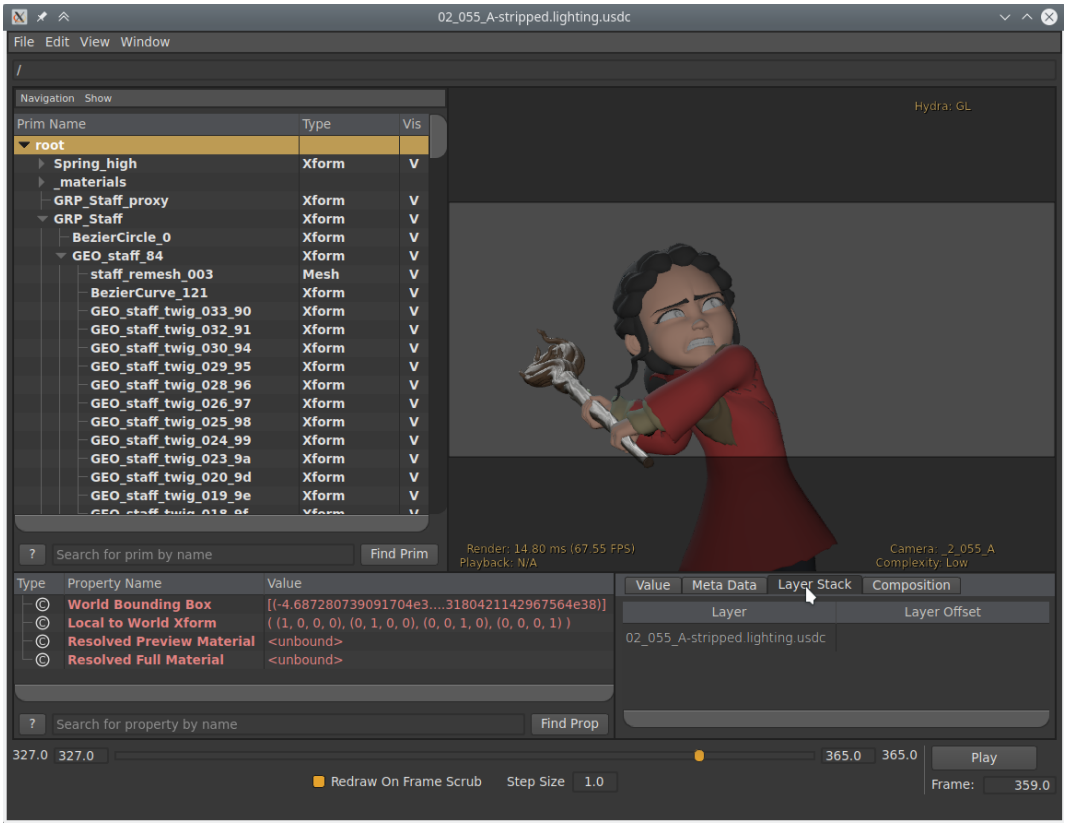
- Meshes (of different kinds, see below).
- Cameras (perspective cameras only at the moment, not orthogonal ones).
- Light (all types except area lights).
- Hair (exported as curves, and limited to parent strands).
- Volume (both static and animated volumes).
- Armatures

When exporting an animation, the final, evaluated mesh is written to USD. This means that the following meshes can be exported:

- Static meshes.
- Deforming meshes; here the topology of the mesh does not change, but the locations of the vertices change over time. Examples are animated characters or bouncing (but not cracking) objects.
- Arbitrarily animated meshes; here the topology does change. An example is the result of a fluid simulation, where splashes of fluid can break off the main body.
- Metaballs are exported as animated meshes.

Note

To export the Blender scene as a USDZ archive, set the file extension of the output file to `.usdz`. The exported USDZ package will be a zip archive containing the USD and its texture file dependencies.



Shot from Spring exported to USD and opened in USDView.

Export Options

The following options are available when exporting to USD:

General

Root Prim

If set, add a transform primitive with the given path to the stage as the parent of all exported data.

Include

Selection Only

When checked, only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered ‘selected’ when their instancer is selected.

Visible Only
: Only exports objects that are not hidden. Invisible parents of exported objects are exported as empty transforms.

Animation
: When checked, the entire scene frame range is exported. When unchecked, only the current scene frame is exported.

Blender Data
: Custom Properties
  : Exports Custom Properties as USD attributes. The *Namespace* property is used to determine the namespace that the attributes are written to.

  Namespace
  : If set, add the given namespace as a prefix to exported custom property names. This only applies to property names that do not already have a prefix (e.g., it would apply to name `bar` but not `foo:bar`) and does not apply to Blender object and data names which are always exported in the `userProperties:blender` namespace.

    By default, `userProperties` namespace is used.

  Blender Names
  : Author USD custom attributes containing the original Blender object and object data names.

Allow Unicode
: Preserves UTF-8 encoded characters when writing USD prim and property names (requires software utilizing USD 24.03 or greater when opening the resulting files).

File References
: Relative Paths
  : Use relative paths to reference external files (i.e. textures, volumes) in the exported USD file, otherwise use absolute paths.

Convert Orientation

Convert orientation axis to a different convention to match other applications. Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Forward / Up Axis

By mapping these to different axes you can convert rotations between applications default up and forward axes.

Xform Ops

The type of transform operators to use to transform prims.

Translate, Rotate, Scale::
Export with translate, rotate, and scale `Xform` operators.

Translate, Orient, Scale::
Export with translate, orient quaternion, and scale `Xform` operators.

Matrix:: Export matrix operator.

Use Settings for

Determines the whether to use *Viewport* or *Render* visibility of collection, modifiers, or any other property that can be set for both the *Viewport* and *Render*.

Object Types

Meshes

Exports Mesh Objects

Lights

Exports Light Objects USD does not directly support spot lights, so those are not exported.

Cameras

Exports Camera Objects Only perspective cameras are exported.

Curves
: Exports Curve Objects

Point Clouds
: Exports Point Cloud Objects as `UsdGeomPoints` primitives.

Volumes
: Exports Volume Objects

Hair
: Exports parent hair strands are exported as a curve system. Hair strand colors are not exported.

Geometry

UV Maps
: When checked, includes UV coordinates for exported meshes. The name of the UV map in USD is the same as the name in Blender.

Rename UV Maps
: Exports UV maps using the USD default name (`st`) as opposed to Blender's default name (`UVMap`).

Normals
: When checked, includes normals for exported meshes. This includes custom loop normals.

Triangulate
: Triangulates the mesh before writing. For more detail on the specific option see the Triangulate modifier.

Rigging

Shape Keys
: Export shape keys as USD blend shapes.

  Absolute shape keys are not supported.

Armatures

Export Armatures and meshes with Armature Modifiers as USD skeletons and skinned meshes.

Limitations:

- Modifiers in addition to Armature modifiers will not be applied.
- Bendy bones are not supported.

Only Deform Bones

Only export deform bones and their parents.

Materials

Exports material information of the object. By default the exporter approximates the Principled BSDF node tree by converting it to USD's Preview Surface format.

When a mesh has multiple materials assigned, a geometry subset is created for each material. The first material (if any) is always applied to the mesh itself as well (regardless of the existence of geometry subsets), because the Hydra viewport does not support materials on subsets. See USD issue #542 for more information.

Note

If *USD Preview Surface Network* and *MaterialX Network* are disabled, the material is set to the viewport materials of meshes.

USD Preview Surface Network

Approximates a Principled BSDF node tree to by converting it to USD's Preview Surface format.

Warning

Not all nodes are supported; currently only Diffuse, Principle, Image Textures, and UVMap nodes are support.

MaterialX Network

Generates material shading graphs using the MaterialX standard. This standard is designed to support a high amount of interoperability among DCCs <Digital Content Creation>. In Blender, MaterialX supports most of the shader nodes and their functionality but has a few caveats (see below).

Implementation Caveats

When using the Principled BSDF, the resulting graph is very usable. However, when using some of the other BSDFs, some of the generated shading graphs are difficult for other DCC's to understand.

Convert World Material

Convert the world material to a USD dome light. Currently works for simple materials, consisting of an environment texture connected to a background shader, with an optional vector multiply of the texture color.

Export Textures

Method for exporting textures.

Keep::	Use original location of textures.
Preserve::	Preserve file paths of textures from already imported USD files. Export remaining textures to a 'textures' folder next to the USD file.
New Path::	Export textures to a 'textures' folder next to the USD file.

Overwrite Textures

Allow overwriting existing texture files when exporting textures.

USDZ Texture Downsampling

Choose a maximum size for all exported textures.

Keep::	Keep all current texture sizes.
256::	Resize to a maximum of 256 pixels.
512::	Resize to a maximum of 512 pixels.
1024::	Resize to a maximum of 1024 pixels.
2048::	Resize to a maximum of 2048 pixels.
4096::	Resize to a maximum of 4096 pixels.
Custom::	Specify a custom size. USDZ Custom Downscale Size The size in pixels of the *Custom* downsampling.

Experimental

Instancing
: As this is an experimental option. When unchecked, duplicated objects are exported as real objects, so a particle system with 100 particles that is displayed with 100 meshes will have 100 individual meshes in the exported file. When checked, duplicated objects are exported as a reference to the original object. If the original object is not part of the export, the first duplicate is exported as real object and used as reference.

Exporter Limitations

Single-sided and Double-sided Meshes
: USD seems to support neither per-material nor per-face-group double-sidedness, so Blender uses the flag from the first material to mark the entire mesh as single/double-sided. If there is no material it defaults to double-sided.

Mesh Normals
: The mesh subdivision scheme in USD is ‘Catmull-Clark’ by default, but Blender uses ‘None’ instead, indicating that a polygonal mesh is

exported. This is necessary for USD to understand the custom normals; otherwise the mesh is always rendered smooth.

Vertex Velocities

Currently only fluid simulations (not meshes in general) have explicit vertex velocities. This is the most important case for exporting velocities, though, as the baked mesh changes topology all the time, and thus computing the velocities at import time in a post-processing step is hard.

Materials

When there are multiple materials, the mesh faces are stored as geometry subset and each material is assigned to the appropriate subset. If there is only one material this is skipped. Note that the geometry subsets are not time-sampled, so it may break when an animated mesh changes topology.

Hair

Only the parent strands are exported, and only with a constant color. No UV coordinates, and no information about the normals.

Camera

Only perspective cameras are exported.

Lights

USD does not directly support spot lights, so those are not exported.

Particles

Particles are only written when they are alive, which means that they are always visible. There is currently no code that deals with marking them as invisible outside their lifespan.

Objects instanced by particle system are exported by suffixing the object name with the particle's persistent ID, giving each particle transform a unique name.

Instancing/Referencing

This is still an experimental feature that can be enabled when exporting to USD. When enabled, instanced object meshes are written to USD as references to the original mesh. The first copy of the mesh is written for real, and the following copies are referencing the first. Which mesh is considered ‘the first’ is chosen more or less arbitrarily.

USDZ

Due to a current limitation in the USD library, UDIM textures cannot be include in the USDZ archive. This limitation will likely be addressed in a future version of USD. (See USD pull request #2133.)

Wavefront OBJ

Reference

Menu:: *File ▸ Import/Export ▸ Wavefront (.obj)*

OBJ format is a popular plain text format, however, it has only basic geometry and material support.

Note

There is no support for armatures, lights, cameras, empty objects, parenting, or transformations. See Compatibility for more information.

Importing

Import geometry and curves to the OBJ format.

If there is a matching `.MTL` for the OBJ then its materials will be imported too.

General

Scale
: Value by which to scale the imported objects in relation to the world's origin.

Clamp Bounding Box
: OBJ-files often vary greatly in scale, this setting clamps the imported file to a fixed size.

Forward Axis, Up Axis

Since many applications use a different axis for ‘Up’, these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it’s common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Options

Split By Object
: Import each OBJ “object name” group (`o`) as a separate object.

Split By Group
: Import each OBJ “object name” group (`g`) as a separate object.

Vertex Groups
: Import OBJ groups as vertex groups.

Validate Meshes
: Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Path Separator
: Character used to separate an object’s name into a hierarchical structure using Collections.

Exporting

Export geometry and curves to the OBJ format.

General

Include: Selected Only

Only export the selected objects. Otherwise export all objects in the scene.

Scale

Global scale to use on export.

Forward Axis, Up Axis

Since many applications use a different axis for 'Up', there are axis conversion settings, Forward and Up axis – By mapping these to different axis you can convert rotations between applications default up and forward axis.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry Properties

UV Coordinates

Write out the active UV layers coordinates from Blender.

Normals

Write out Blender's face and vertex normals (depending on the faces smooth setting).

Mostly this isn't needed since most applications will calculate their own normals but to match Blender's normal map textures you will need to write these too.

Colors

Write out the active vertex colors attribute layer, if present. Colors are exported in "xyzrgb" OBJ extension format.

Curves as NURBS

Write out NURBS curves as OBJ NURBS rather than converting to geometry.

Triangulated Mesh
: Write out quads as two triangles. Some programs only have very basic OBJ support and only support triangles.

Apply Modifiers
: Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

Properties
: For properties that have different settings for the viewport/final render pick which is used for output. One example where this is important is the Subdivision Surface Modifier.

Viewport:: Use viewport properties.
Render:: Use final render properties.

Grouping

Object Groups
: Write out each Blender object as an OBJ object.

> **Note**
>
> Note that as far as Blender is concerned there is no difference between OBJ Groups and Objects, this option is only included for applications that treat them differently.

Material Groups
: Generate an OBJ group for each part of a geometry using a different material.

Vertex Groups
: Export the name of the vertex group of a face. It is approximated by choosing the vertex group with the most members among the vertices of a face.

Smooth Groups

Write Blender's sharp edges as smooth groups.

Smooth Group Bitflags
: Generate Bitflags for smooth Groups.

Materials

Write out the MTL-file along with the OBJ. Most importers that support OBJ will also read the MTL-file.

PBR Extensions
: Export MTL library using PBR extensions (roughness, metallic, sheen, clearcoat, anisotropy, transmission).

Path Mode
: When referencing paths in exported files you may want some control as to the method used since absolute paths may only be correct on your own system. Relative paths, on the other hand, are more portable but mean that you have to keep your files grouped when moving about on your local file system. In some cases, the path doesn't matter since the target application will search a set of predefined paths anyway so you have the option to strip the path too.

Auto::	Uses relative paths for files which are in a subdirectory of the exported location, absolute for any directories outside that.
Absolute::	Uses full paths.
Relative::	Uses relative paths in every case (except when on a different drive on Windows).
Match::	Uses relative / absolute paths based on the paths used in Blender.
Strip Path::	Only write the filename and omit the path component.
Copy::	Copy the file on exporting and reference it with a relative path.

Animation

Exports a numbered OBJ for each frame from the start to the end frame. Please be aware that this can take quite a long time.

Frame Start, End
: The first and last frame to export, used to determine the range of exported frames.

Compatibility

NURBS surfaces, text3D and metaballs are converted to meshes at export time.

Stanford PLY

Reference

Category::	Import-Export
Menu::	*File ▸ Import/Export ▸ Stanford (.ply)*

Use the operator to import ASCII or binary PLY-files, you can select multiple files at once. For exporting, you can choose to enable or disable the modifiers during the export and you can choose which data you want to export (UV textures, Color Attributes, …).

Import

General

Scale
: Value by which to scale the imported objects in relation to the world's origin.

Scene Unit
: Apply current scene's unit (as defined by unit scale) to imported data.

Forward Axis, Up Axis
: Since many applications use a different axis for 'Up', these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

  Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Options

Merge Vertices
: Attempts to combine co-located vertices where possible.

Import Vertex Colors
: The color space that the color data in the ply-file was saved in.

None::	Does not import vertex color data.
sRGB::	Vertex colors in the file are in sRGB Color Space
Linear::	Vertex colors in the file are in Linear Color Space

Export

General

Format: ASCII
: Formats the file using the simple a ASCII format. This option might be helpful if the program that will later import the file does not support the binary file format.

Include: Selected Only
: Only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered ‘selected’ when their instancer is selected.

Scale
: Value by which to scale the exported objects in relation to the world’s origin.

Forward Axis, Up Axis
: Since many applications use a different axis for ‘Up’, these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry

UV Coordinates

Write out the active UV layers coordinates from Blender.

Vertex Normals

Write out Blender's face and vertex normals (depending on the faces smooth setting).

Mostly this isn't needed since most applications will calculate their own normals but to match Blender's normal map textures you will need to write these too.

Vertex Colors

The color space that the color data in the ply-file was saved in.

None::	Does not import vertex color data.
sRGB::	Vertex colors in the file are in sRGB Color Space
Linear::	Vertex colors in the file are in Linear Color Space

Triangulated Mesh

All N-gons with four or more vertices will be triangulated. Meshes in the scene will not be affected. Behaves like Triangulate Modifier with the following settings:

- N-gon-method: "Beauty"
- Quad-method: "Shortest Diagonal"
- Min vertices: 4

Apply Modifiers

Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

STL

Reference

Category::	Import-Export
Menu::	*File ▸ Import/Export ▸ Stl (.stl)*

The STL-file format is useful if you intend to import/export the files for CAD software. It is also commonly used for loading into 3D printing software.

Importing

General

Scale
: Value by which to scale the imported objects in relation to the world's origin.

Scene Unit
: Apply current scene's unit (as defined by unit scale) to imported data.

Forward / Up Axis
: Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

  Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, it is common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Options

Facet Normals
: Use (import) facet normals (note that this will still give flat shading).

Validate Mesh
: Check the imported mesh for corrupt data and fix it if necessary. When disabled, erroneous data may cause crashes displaying or editing the meshes. This option will make the importing slower but is recommended, as data errors are not always obvious.

Exporting

General

Format: ASCII
: Exports the stl-file in ASCII format rather than as a binary format

Batch
: Export each object as a separate STL file.

Include: Selection Only
: When checked, only selected objects are exported. Instanced objects, for example collections that are instanced in the scene, are considered 'selected' when their instancer is selected.

Scale
: Value by which to scale the exported objects in relation to the world's origin.

Scene Unit
: Apply current scene's unit (as defined by unit scale) to exported data.

Forward, Up
: Since many applications use a different axis for 'Up', these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, it's common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Geometry

Apply Modifiers
: Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

Import/Export SVG as Grease Pencil

The Scalable Vector Graphics (SVG) format is use for interchanging vector based illustrations between applications and is supported by vector graphics editors such as Inkscape, and modern browsers among others.

Warning

The exporter only works in Object Mode.

Import

Reference

menu:: *File ▸ Import ▸ SVG as Grease Pencil*

Resolution
: Resolution for generated strokes.

Scale
: Generated strokes scale.

Export

Reference

menu:: *File ▸ Export ▸ Grease Pencil as SVG*

Scene Options

Object
: Determine which objects include in the export.

Active::	Export only the active Grease Pencil object.
Selected::	Export all selected Grease Pencil objects.
Visible::	Export all visible Grease Pencil object in the scene.

Export Options

Sampling
: Precision for the stroke sampling. Low values mean a more accurate result.

Fill
: When enabled, Export the Grease Pencil strokes fill.

Uniform Width
: When enabled, Export strokes with constant thickness.

Clip Camera
: When enabled, and camera view is active export only the strokes clipped from camera view.

Export Grease Pencil as (PDF)

The Portable Document Format (PDF) is use for interchanging PDFs between applications, it support the export of Grease Pencil animation creating one page in the PDF document for each keyframe selected.

Warning

The exporter only works in Object Mode.

Scene Options

Object
: Determine which objects will be included in the export.

Active:: Export only the active Grease Pencil object.
Selected:: Export all selected Grease Pencil objects.
Visible:: Export all visible Grease Pencil object in the scene.

Export Options

Frame
: Determine which frames will be included in the export.

Active:: Export only the active keyframe.
Selected:: Export all selected keyframes as different PDF pages.
Scene:: Export all frames as different PDF pages.

Note

To enable multi-keyframe selection you must enable Multiframe Edition. See Multiframe Edition for more information.

Sampling
: Precision for the stroke sampling. Low values mean a more accurate result.

Fill
: When enabled, Export the Grease Pencil strokes fill.

Uniform Width
: When enabled, Export strokes with constant thickness.

Note

The export of the Grease Pencil strokes is always from camera view.

Add-ons

Important

This is work in progress.

Documentation might be outdated and on some pages images, videos, and links aren't added yet.

Add-ons Category Listings

- 3D View
- Animation
- Import-Export
- Node
- Rigging
- System

3D View

These add-ons relate to drawing or manipulating the 3D Viewport.

- VR Scene Inspection

VR Scene Inspection

The VR Scene Inspection add-on exposes and extends the native virtual reality features of Blender in the user interface. The feature set is limited to scene inspection use cases. More advanced use cases may be enabled through further development inside of Blender.

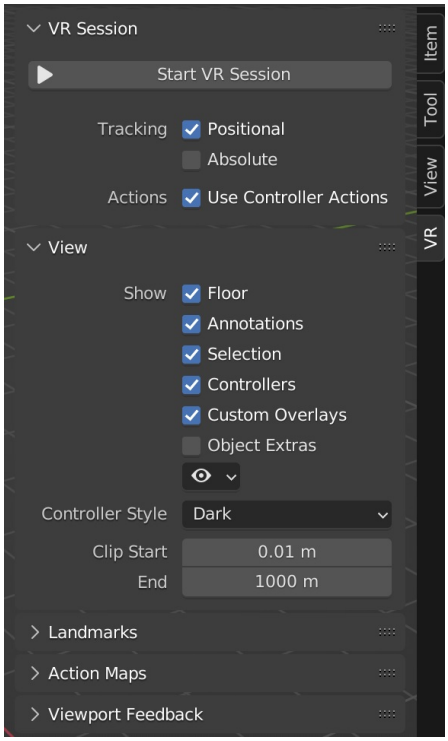
VR support in Blender is based on the OpenXR specification and requires some set up steps. These are explained in the Head-Mounted Displays (HMD) section.

Enabling Add-on

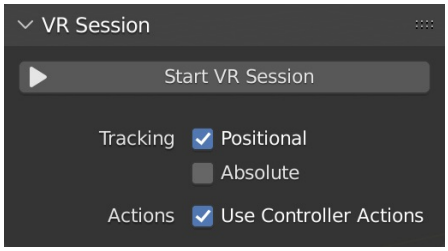
1. Open Blender and go to Add-ons section of the Preferences.
2. Search "VR Scene Inspection" and check the *Enable Extension* checkbox.

Interface

Located in the *3D Viewport* ▸ *Sidebar* ▸ *VR tab*.



VR Session



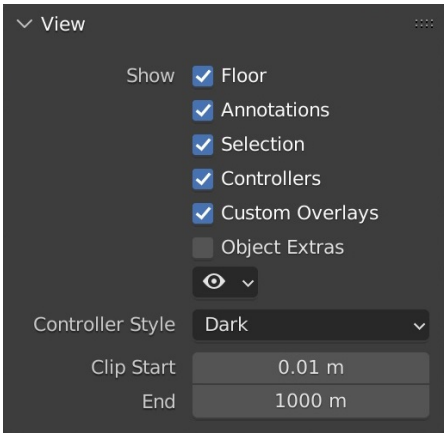
Start VR Session
: Try to set up a connection to the OpenXR platform to share the viewport with an HMD.

Tracking
: Positional
: Only track rotational changes of the head, do not allow the HMD to affect the location of the viewer in virtual space.

: Absolute
: Skip eye offsets that are normally added for placing the viewer exactly at landmarks. This allows the tracking origin to be defined independently of the HMD position.

Use Controller Actions
: Enable default controller actions for viewport navigation, controller tracking, and haptics.

View



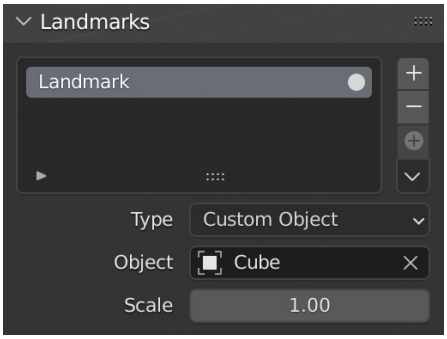
Show
- Floor
  - Set visibility of the ground plane in the VR view.
- Annotations
  - Set visibility of annotation strokes in the VR view.
- Selection
  - Set visibility of selection outlines in the VR view.
- Controllers
  - Set visibility of VR motion controllers. Requires enabling the Use Controller Actions option.
- Custom Overlays
  - Set visibility of custom operator drawing (e.g. default teleport beam).
- Object Extras
  - Set visibility of object extras, including empties, lights, and cameras.
- Object Type Visibility
  - Set visibility of objects by type.

Controller Style
- Preferred visualization of VR motion controllers.

Clip Start/End
- Clipping values of the VR view, as in the 3D Viewport.

Landmarks

Landmarks are used to store reusable base poses (position and rotation) for the viewer in the virtual space. In addition, a base viewer reference scale can be set for landmarks of types Custom Object and Custom Pose.



Landmark
A list view.

Selected Landmark
Defines which landmark's settings are shown below the list. Changing the selected landmark does not have an influence on the VR view.

Activate ○
Activates a landmark, making it change the base pose of the VR view.

Add +
Create a landmark.

Remove −
Delete the selected landmark.

Add from Session ⊕
Create a landmark from the viewer pose of the running VR session.

Landmark Controls ∨
Add Camera and VR Landmark from Session
Create a new camera and landmark from the viewer pose of the running VR session.

Add Landmark from Camera
Add a new landmark from the active camera object.

Update Custom Landmark
Update the selected landmark from the current VR viewer pose.

Cursor to Landmark

Move the 3D Cursor to the selected landmark.

Scene Camera to Landmark
: Position the scene camera at the selected landmark.

Camera from Landmark
: Create a new camera from the selected landmark.

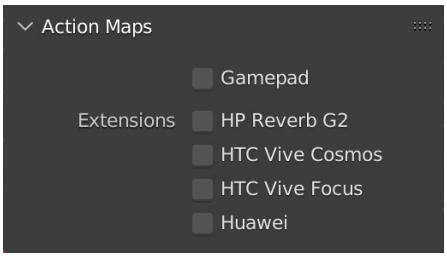
Type

Scene Camera
: Follow the scene's active camera to define the base pose of the viewer.

Custom Object
: Set an arbitrary object to define the base pose of the viewer.

Custom Pose
: Manually define a position and rotation to use as the base pose of the viewer.

Action Maps



Gamepad
: Use input from a gamepad (Microsoft Xbox Controller) instead of motion controllers for VR actions such as viewport navigation.

Extensions
: Enable additional controller bindings to ensure correct input-to-action mappings. Note that a given extension may not be supported by all VR platforms.

HP Reverb G2
: Enable bindings for the HP Reverb G2 controllers.

HTC Vive Cosmos
: Enable bindings for the HTC Vive Cosmos controllers.

HTC Vive Focus
: Enable bindings for the HTC Vive Focus 3 controllers.

Huawei
: Enable bindings for the Huawei controllers.

Viewport Feedback



Show VR Camera
: Draw an indicator of the current VR viewer pose (location and rotation in the virtual space) in the current 3D Viewport.

Show VR Controllers
: Draw indicators of tracked VR motion controllers in the current 3D viewport. Requires enabling the Use Controller Actions option.

Show Landmarks
: Draw landmark indicators in the current 3D Viewport.

Mirror VR Session
: Make the current 3D Viewport follow the perspective of the VR view.

Reference

Category::	3D View
Description::	View the viewport with virtual reality glasses (head-mounted displays).
Location::	*3D Viewport ▸ Sidebar ▸ VR tab*
File::	viewport_vr_preview folder
Author::	Julian Eisel, Sebastian Koenig, Peter Kim
Maintainer::	Julian Eisel, Peter Kim
License::	GPL
Support Level::	Official
Note::	This add-on is bundled with Blender.

Animation

These add-ons relate to helper tools for the animation process and animation.

- Copy Global Transform

Copy Global Transform

Copy and paste object and bone transforms with ease.

When copying, the global (World Space) transform is placed on the clipboard. This can then be pasted onto any object or bone, at the current frame or at another one.

Since the transform is placed on the clipboard as text, you can even copy-paste it into an instant messenger and send it to someone else.

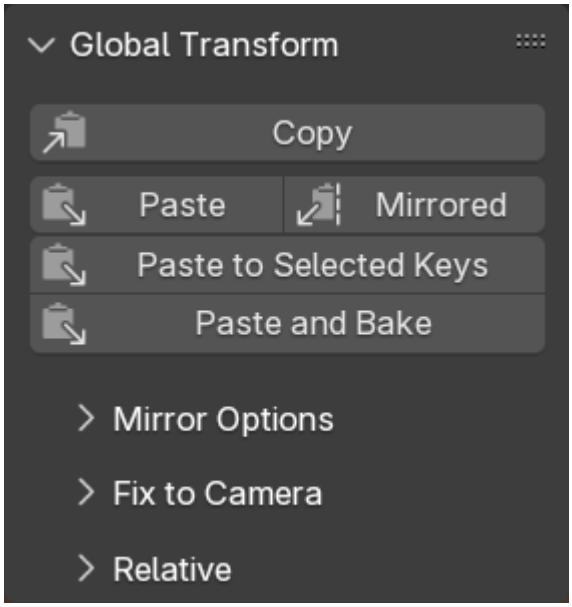
Enabling Add-on

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "Copy Global Transform" and check the *Enable Extension* checkbox.

Interface

Located in *3D Viewport* ▸ *N-panel* ▸ *Animation tab*.

The figure on the right shows the main functionality of the Copy Global Transform panel. The collapsed panels are described each in their own section below.



Description

Copy
: Inspects the active Object (in Object mode) or Bone (in Pose mode) and places its current global transform onto the clipboard as a matrix.

Paste
: Takes the copied global transform and applies it to the active Object or Bone. This is done by **adjusting its location, rotation, and scale properties**.

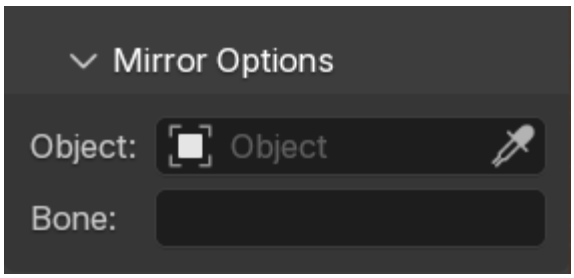
Mirrored
: Same as 'Paste' above, but then mirrored relative to some other object or bone. This can be useful, for example, to copy the foot position of one foot to the other. See Mirror Options below.

Paste to Selected Keys
: Paste as described above and additionally use auto-keying to update one or more frames. The key selection is used to tell Blender *which frames* this should happen on; it does not influence which parts of the transform are keyed. *What* is keyed is determined by the active keying set.

Paste and Bake
: Almost the same as *Paste to Selected Keys*. Instead of only pasting on the selected keys, *Paste and Bake* will paste & auto-key on every frame between the first and last selected keys.

Mirror Options

The copied transform can be mirrored relative to an object or a Bone. This requires choosing that object or bone first.



Armature + Bone
: Choosing an Armature object as mirror object will show the bone

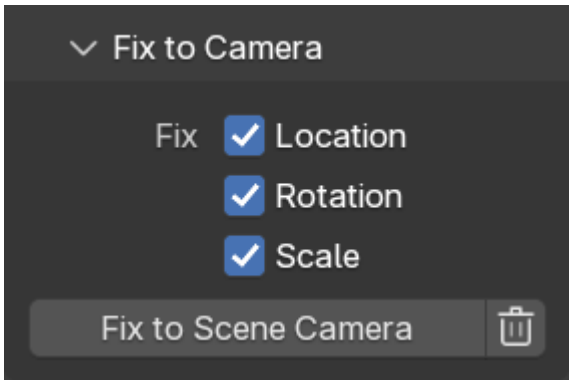
selector. You can use that to pick the bone to use as mirror. This will always use the named bone on that specific armature object.

Bone Only
: When you choose *no mirror object* at all, you can still choose a *bone name*. This is used for mirroring against a bone in the *active armature*. This can be useful to mirror bone transforms relative to the 'chest' bone of the active character.

Object Only
: This will just mirror relative to the chosen object.

After pasting with 'Paste Mirrored', the mirror axes can be chosen in the redo panel.

Fix to Camera



Also known as "bake to camera", this operator will ensure selected objects/bones remain static (relative to the camera) on unkeyed frames.

This is done by generating new keys. These keys will be of type 'Generated' so that it remains clear which keys were manually created, and which were generated. This way the tool can be re-run to re-generate the keys.

1. Ensure your animation is keyed using constant interpolation. If this is not the case yet, bake your animation (at least the transform channels). This tool does _not_ work with the "Stepped" F-Curve modifier
2. Choose which of the Location/Rotation/Scale channels you want to fix to the camera. When unsure, make sure they are all checked.
3. Press the "Fix to Camera" button.

To undo the effect of the "Fix to Camera" operator, click on the trash bin button. That will remove all the generated keys in either the scene range or the frame range.

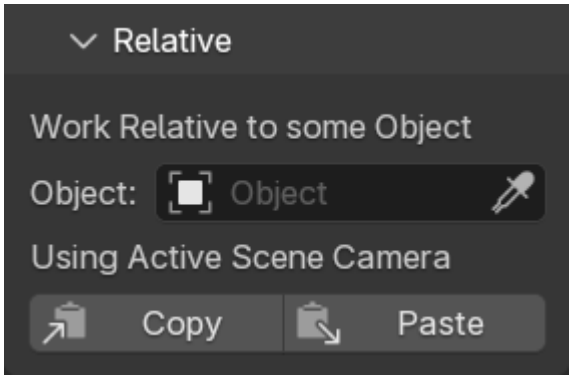
The tool operates on the scene frame range, or on the preview range if that is active. Keys outside that range are ignored, both when fixing to the camera and when removing generated keys.

Warning

This tool assumes that *all* keys with type 'Generated' are equal. It will overwrite them (or remove them, depending on which button you press).

Relative Copy-Paste

The "Relative" panel has copy/paste buttons that work relative to a chosen object. When copying, the world-space transform is determined, and then adjusted to become relative to the world-space transform of the chosen object. When pasting, this happens in reverse.



If no object is chosen, the copy/paste will happen relative to the active scene camera. What is the active scene camera is determined for every action, so when you paste it can be different from when you copied. This can help to keep an object visually in the same place when switching cameras, or when switching between scenes.

Limitations

Pasting a transform adjusts the Object/Bone's location, rotation, and scale. This means that when copying a skewed transform, this skew is lost.

If there are constraints on the Object/Bone, the resulting visual transformation may not be the same as the pasted one. To give a concrete example: if you have a constraint that adds a rotation, it will always add that rotation on top of the pasted transform.

See also

Pose Library for a way to manage and share entire poses.

Reference

Category::	Animation
Description::	Simple add-on for copying world-space transforms.
Location::	*3D Viewport ▸ N-panel ▸ Animation tab*
File::	copy_global_transform.py
Author::	Sybren A. Stüvel
Maintainer::	Sybren A. Stüvel
License::	GPL 2+
Support Level::	Official
Note::	This add-on is bundled with Blender.

Import-Export

- BioVision Motion Capture (BVH)
- FBX
- Scalable Vector Graphics (SVG)
- UV Layout
- glTF 2.0

BioVision Motion Capture (BVH)

Reference

Category::	Import-Export
Menu::	*File ▸ Import/Export ▸ Motion Capture (.bvh)*

Imports or exports bvh-files or files with BioVision Hierarchical data or data of a skeleton (rig) including its animation. Useful for importing data from motion capture devices.

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "BioVision Motion Capture (BVH) format" and check the *Enable Extension* checkbox.

Properties

Import

Target
: The motion capture data type.

Armature::	The bvh-file contains an animated rigged skeleton such as a walking motion capture.
Object::	The bvh-file contains a static (not animated) mesh object such as a character model.

Transform

Scale
: Factor to increase the physical size of the BVH.

Rotation
: Rotation order of the BVH.

Forward / Up
: Since many applications use a different axis for pointing upwards, these are axis conversion for these settings, Forward and up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

  Blender uses Y forward, Z up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z forward, Y up is needed.

Animation

Start Frame
: The start frame, in Blender, to start playback of the BVH animation.

Scale FPS
: Scales the frame rate from the BVH file to the scene frame rate set in Blender, otherwise each BVH frame maps directly to a frame in Blender.

Loop
: Cycles the animation playback.

Update Scene FPS
: Set the scene's frame rate to match the frame rate of the BVH file.

Update Scene Duration
: Extend the scene's duration to match the BVH's duration.

Export

Transform

Scale
: Factor to increase the physical size of the BVH.

Rotation
: Rotation order of the BVH.

Root Translation Only
: Only write the translation animation channels for the root bone.

Animation

Start / End
: Sets the range of animation to export to the BVH file.

FBX

Reference

Category::	Import-Export
Menu::	*File ▸ Import/Export ▸ FBX (.fbx)*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "FBX" and check the *Enable Extension* checkbox.

Usage

This format is mainly use for interchanging character animations between applications and is supported by applications such as Cinema4D, Maya, Autodesk 3ds Max, Wings3D and engines such as Unity3D, Unreal Engine 3/UDK and Unreal Engine 4.

The exporter can bake mesh modifiers and animation into the FBX so the final result looks the same as in Blender.

Note

- Bones would need to get a correction to their orientation (FBX bones seems to be -X aligned, Blender's are Y aligned), this does not affect skinning or animation, but imported bones in other applications will look wrong.
- Animations (FBX AnimStacks, Blender actions) **are not linked** to their object, because there is no real way to know which stack to use as 'active' action for a given object, mesh or bone. This may be

enhanced to be smarter in the future, but it's not really considered urgent, so for now you'll have to link actions to objects manually.
- Armature instances **are not supported**.

Note

- Bones' orientation importing is complex, you may have to play a bit with related settings until you get the expected results.
- Animation support is minimal currently, we read all curves as if they were 'baked' ones (i.e. a set of close keyframes with linear interpolation).
- Imported actions are linked to their related object, bone or shape key, on a 'first one wins' basis. If you export a set of them for a single object, you'll have to reassign them yourself.

Note

Saving Just Animations

The FBX file format supports files that only contain takes. It is up to you to keep track of which animation belongs to which model. The animation that will be exported is the currently selected action within the Action editor. To reduce the file size, turn off the exporting of any parts you do not want and disable *All Actions*. For armature animations typically you just leave the armature enabled which is necessary for that type of animation. Reducing what is output makes the export and future import much faster. Normally each action will have its own name but the current or only take can be forced to be named "Default Take". Typically, this option can remain off.

Note

Blender now only supports complex node-based shading. FBX having a fixed pipeline-like support of materials, this add-on converts between them.

Properties

Import

Include

Import Normals
: TODO.

Import Subdivision Surface
: Todo.

Import User Properties
: TODO.

Import Enums as Strings
: TODO.

Image Search
: TODO.

Transform

Scale
: Todo.

Decal Offset
: TODO.

Manual Orientation
: TODO.

Forward / Up Axis

Since many applications use a different axis for ‘Up’, these are axis conversion for these settings, Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Apply Transform

TODO.

Use Pre/Post Rotation

TODO.

Animation

TODO.

Animation Offset

TODO.

Armature

Ignore Leaf Bones

TODO.

Force Connect Children

TODO.

Automatic Bone Orientation

TODO.

Primary/Secondary Bone Axis

TODO.

Export

Path Mode

When referencing paths in exported files you may want some control as to the method used since absolute paths may only be correct on your own system. Relative paths, on the other hand, are more portable but mean that you have to keep your files grouped when moving about on your local file system. In some cases, the path doesn't matter since the target application will search a set of predefined paths anyway so you have the option to strip the path too.

Auto::	Uses relative paths for files which are in a subdirectory of the exported location, absolute for any directories outside that.
Absolute::	Uses full paths.
Relative::	Uses relative paths in every case (except when on a different drive on Windows).
Match::	Uses relative / absolute paths based on the paths used in Blender.
Strip Path::	Only write the filename and omit the path component.
Copy::	Copy the file on exporting and reference it with a relative path.

Embed Textures

TODO.

Batch Mode

When enabled, export each group or scene to a file.

Group/Scene

Choose whether to batch export groups or scenes to files. Note, when Group/Scene is enabled, you cannot use the animation option *Current Action* since it uses scene data and groups are not attached to any scenes. Also note, when Group/Scene is enabled you must include the armature objects in the group for animated actions to work.

Batch Own Directory
: When enabled, each file is exported into its own directory, this is useful when using the *Copy Images* option. So each directory contains one model with all the images it uses. Note, this requires a full Python installation. If you do not have a full Python installation, this button will not be shown.

Include

Selected Objects
: Only export the selected objects. Otherwise export all objects in the scene. Note, this does not apply when batch exporting.

Active Collection
: Todo.

Object Types
: Enable/Disable exporting of respective object types.

Custom Properties
: TODO.

Transform

Scale
: Scale the exported data by this value. 10 is the default because this fits best with the scale most applications import FBX to.

Apply Scaling
: TODO.

Forward / Up
: Since many applications use a different axis for ‘Up’, these are axis conversions for Forward and Up axes – By mapping these to different axes you can convert rotations between applications default up and forward axes.

Blender uses Y Forward, Z Up (since the front view looks along the +Y direction). For example, its common for applications to use Y as the up axis, in that case -Z Forward, Y Up is needed.

Apply Unit
: TODO.

Apply Transform
: TODO.

Geometry

Smoothing
: TODO.

Export Subdivision Surface
: Todo.

Apply Modifiers
: Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

Loose Edges
: TODO.

Tangent Space
: TODO.

Armatures

Primary/Secondary Bone Axis
: TODO.

Armature FBXNode Type
: TODO.

Only Deform Bones
: TODO.

Add Leaf Bones
: TODO.

Bake Animation

TODO.

Key All Bones
: TODO.

NLA Strips
: TODO.

All Actions
: Export all actions compatible with the selected armatures start/end times which are derived from the keyframe range of each action. When disabled only the currently assigned action is exported.

Force Start/End Keying
: TODO.

Sampling Rate
: TODO.

Simplify
: TODO.

Compatibility

Import

Note that the importer is a new addition and lacks many features the exporter supports.

- binary FBX files only.
- Version 7.1 or newer.

Missing

- Mesh: shape keys.

Export

NURBS surfaces, text3D and metaballs are converted to meshes at export time.

Missing

Some of the following features are missing because they are not supported by the FBX format, others may be added later.

- Object instancing – exported objects do not share data, instanced objects will each be written with their own data.
- Material textures
- Vertex shape keys – FBX supports them but this exporter does not write them yet.
- Animated fluid simulation – FBX does not support this kind of animation. You can however use the OBJ exporter to write a sequence of files.
- Constraints – The result of using constraints is exported as a keyframe animation however the constraints themselves are not saved in the FBX.
- Instanced objects – At the moment instanced objects are only written in static scenes (when animation is disabled).

Scalable Vector Graphics (SVG)

Reference

Category:: Import-Export
Menu:: *File ▸ Import ▸ Scalable Vector Graphics (.svg)*

Note

Currently the script allows only importing and is limited to path geometry only.

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to Add-ons section of the Preferences.
2. Search “Scalable Vector Graphics (SVG)” and check the *Enable Extension* checkbox.

Properties

This add-on does not have any properties.

Usage

Todo.

UV Layout

Reference

Category::	Import-Export
Menu::	*UV Editor* ▸ *UV* ▸ *Export UV Layout*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "UV Layout" and check the *Enable Extension* checkbox.

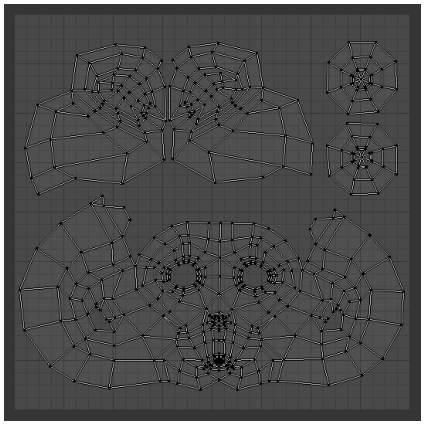
Usage

Using your favorite image painting program, you could use an exported UV layout to create a texture. Then save your changes, and back in Blender, use the *Image* ▸ *Open* to load it as your UV image for the mesh in Edit Mode for the desired (and active) UV map.

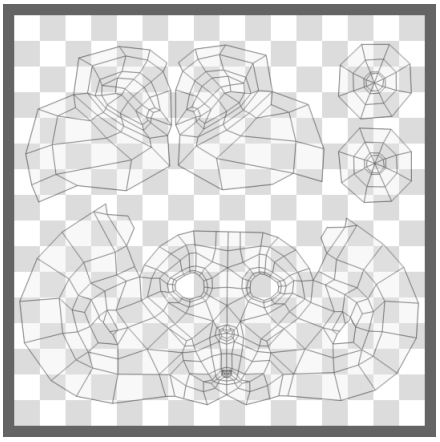
As a way of communicating to an artist who is painting your UV Texture for you, Blender has a tool called *UV Layout* (*UV Editor* ▸ *UV* ▸ *Export UV Layout*) that saves an image as a `Targa` (`.tga`), `EPS`, or `SVG` format for the object you have selected.

The image will be lines defining the UV edges that are within the image area of the UV mapping area. Edges outside the boundary, even if selected, will not be shown in the saved graphic. The artist will use this as a transparent layer in their paint program as a guide when painting your texture. The example below shows Blender in the background, and the Gimp working on the texture, using the saved layout as a guide. Note that

`targa` format supports the Alpha channel, so you can paint transparent areas of the mesh. For using images as textures, see the page on Image Textures.

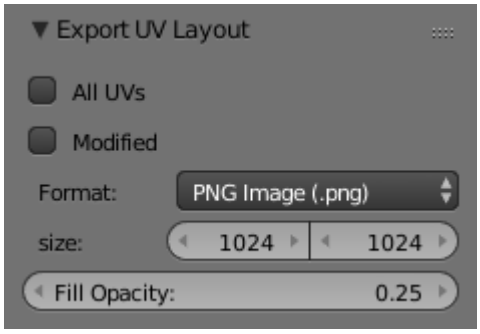


A UV layout in the UV Editor.



A UV layout in a paint program.

Properties



Export options.

All UVs
: if disabled, then only the UV faces selected will be outlined.

Modified
: Export UVs from the modified mesh.

Format

Select the type of image file to save (`.png`, `.eps`, `.svg`).

Size
: Select the size of the image in pixels.

Fill Opacity
: Set the opacity of the fill.

glTF 2.0

Reference

Category::	Import-Export
Menu::	*File ▸ Import/Export ▸ glTF 2.0 (.glb, .gltf)*

Enabling Add-on

This add-on is enabled by default, in case it is not:

1. Open Blender and go to Add-ons section of the Preferences.
2. Search “glTF 2.0” and check the *Enable Extension* checkbox.

Usage

glTF™ (GL Transmission Format) is used for transmission and loading of 3D models in web and native applications. glTF reduces the size of 3D models and the runtime processing needed to unpack and render those models. This format is commonly used on the web, and has support in various 3D engines such as Unity3D, Unreal Engine 4, and Godot.

This importer/exporter supports the following glTF 2.0 features:

- Meshes
- Materials (Principled BSDF) and Shadeless (Unlit)
- Textures
- Cameras
- Punctual lights (point, spot, and directional)
- Extensions (listed below)
- Extras (custom properties)
- Animation (keyframe, shape key, and skinning)

Meshes

glTF's internal structure mimics the memory buffers commonly used by graphics chips when rendering in real-time, such that assets can be delivered to desktop, web, or mobile clients and be promptly displayed with minimal processing. As a result, quads and n-gons are automatically converted to triangles when exporting to glTF. Discontinuous UVs and flat-shaded edges may result in moderately higher vertex counts in glTF compared to Blender, as such vertices are separated for export. Likewise, curves and other non-mesh data are not preserved, and must be converted to meshes prior to export.

GPU Instances

When the option is enable in Exporter, instances are exported using the `EXT_mesh_gpu_instancing` extension. There are some limitations, at export:

- Instances must be meshes, and don't have any children themselves
- Instances must all be children of the same object.
- This extension doesn't manage material variation. That means that the generated file may include all instances with same materials.
- Instances detected are objects sharing the same mesh data.

At import, instances are created by creating objects sharing the same mesh data.

Materials

The core material system in glTF supports a metal/rough PBR workflow with the following channels of information:

- Base Color
- Metallic
- Roughness
- Baked Ambient Occlusion

- Normal Map (tangent space, +Y up)
- Emissive

Some additional material properties or types of materials can be expressed using glTF extensions. The complete list can be found in Extensions part of this documentation.



An example of the various image maps available in the glTF 2.0 core format. This is the water bottle sample model shown alongside slices of its various image maps.

Imported Materials

The glTF material system is different from Blender's own materials. When a glTF file is imported, the add-on will construct a set of Blender nodes to replicate each glTF material as closely as possible.

The importer supports Metal/Rough PBR (core glTF), Spec/Gloss PBR (`KHR_materials_pbrSpecularGlossiness`) and some extension materials. The complete list can be found in Extensions part of this documentation.

Tip

Examining the result of the material import process is a good way to see examples of the types of material nodes and settings that can be exported to glTF.

Exported Materials

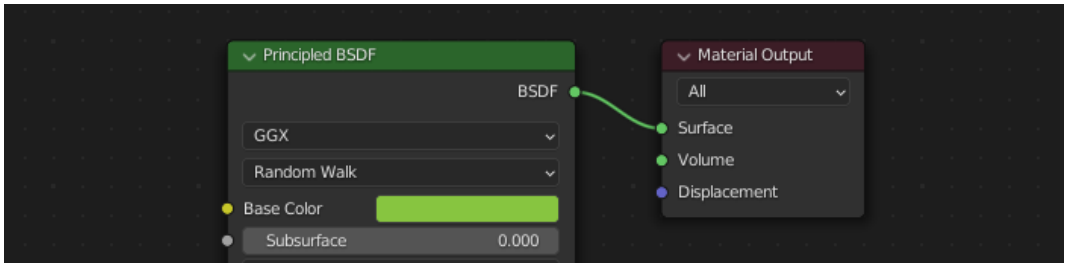
The exporter supports Metal/Rough PBR (core glTF) and Shadeless (`KHR_materials_unlit`) materials. It will construct a glTF material based on the nodes it recognizes in the Blender material. The material export process handles the settings described below.

Note

When image textures are used by materials, glTF requires that images be in PNG or JPEG format. The add-on will automatically convert images from other formats, increasing export time.

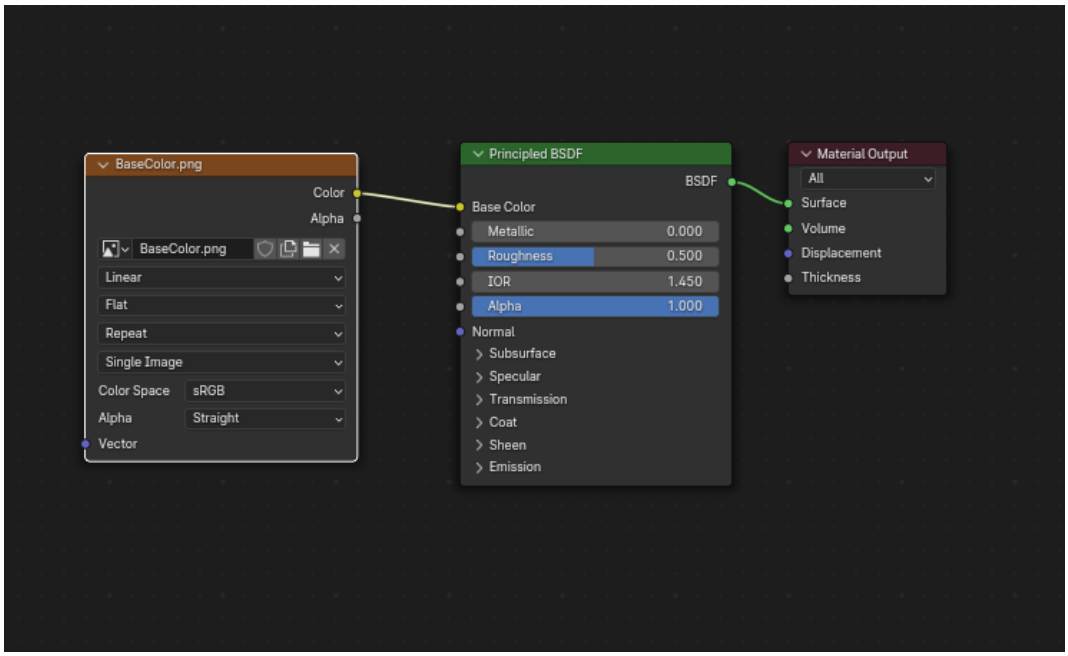
Base Color

The glTF base color is determined by looking for a Base Color input on a Principled BSDF node. If the input is unconnected, the input's default color (the color field next to the unconnected socket) is used as the Base Color for the glTF material.



A solid base color can be specified directly on the node.

If an Image Texture node is found to be connected to the Base Color input, that image will be used as the glTF base color.



An image is used as the glTF base color.

Metallic and Roughness

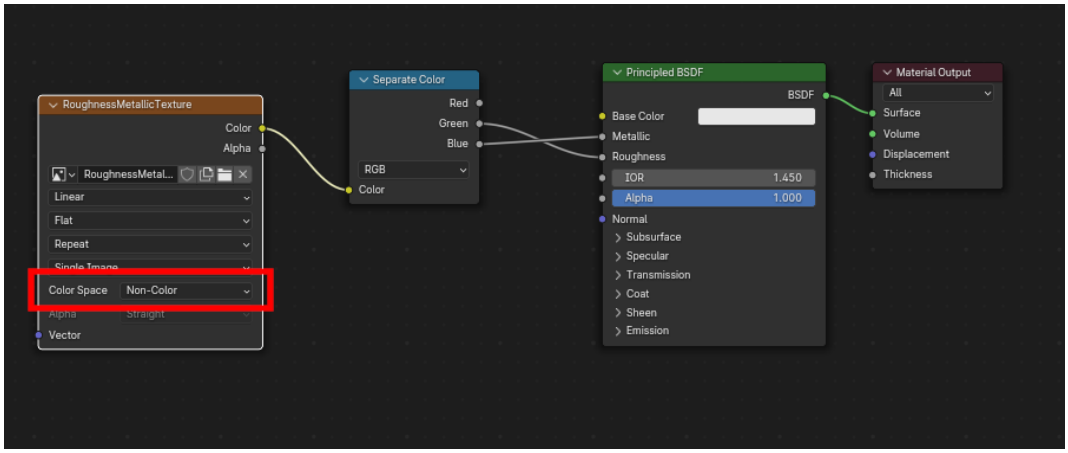
These values are read from the Principled BSDF node. If both of these inputs are unconnected, the node will display sliders to control their respective values between 0.0 and 1.0, and these values will be copied into the glTF.

When using an image, glTF expects the metallic values to be encoded in the blue (`B`) channel, and roughness to be encoded in the green (`G`) channel of the same image. If images are connected to the Blender node in a manner that does not follow this convention, the add-on may attempt to adapt the image to the correct form during exporting (with an increased export time).

In the Blender node tree, it is recommended to use a Separate RGB node to separate the channels from an Image Texture node, and connect the green (`G`) channel to Roughness, and blue (`B`) to Metallic. The glTF exporter will

recognize this arrangement as matching the glTF standard, and that will allow it to simply copy the image texture into the glTF file during export.

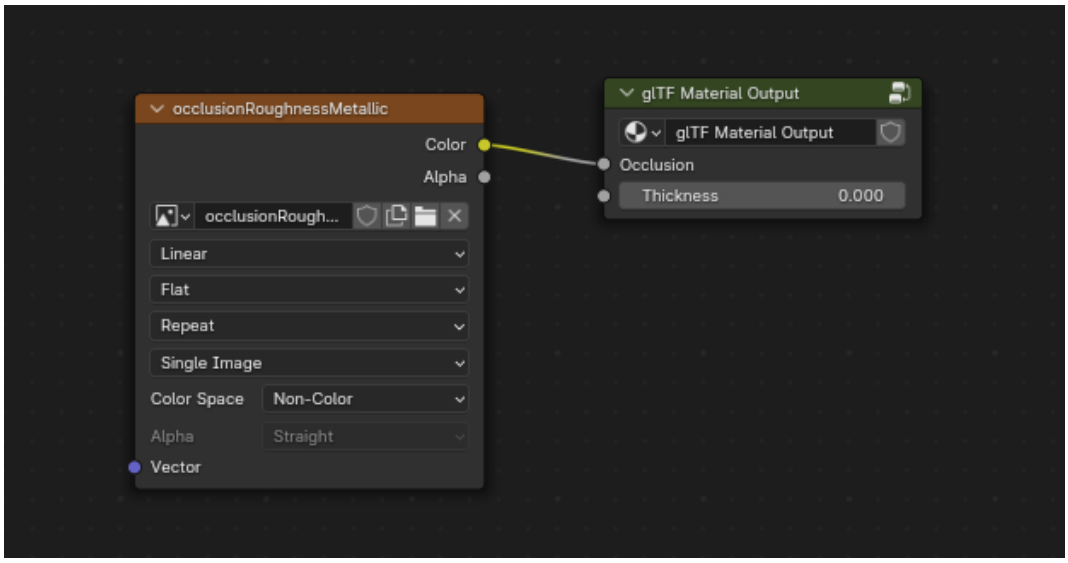
The Image Texture node for this should have its *Color Space* set to Non-Color.



A metallic/roughness image connected in a manner consistent with the glTF standard, allowing it to be used verbatim inside an exported glTF file.

Baked Ambient Occlusion

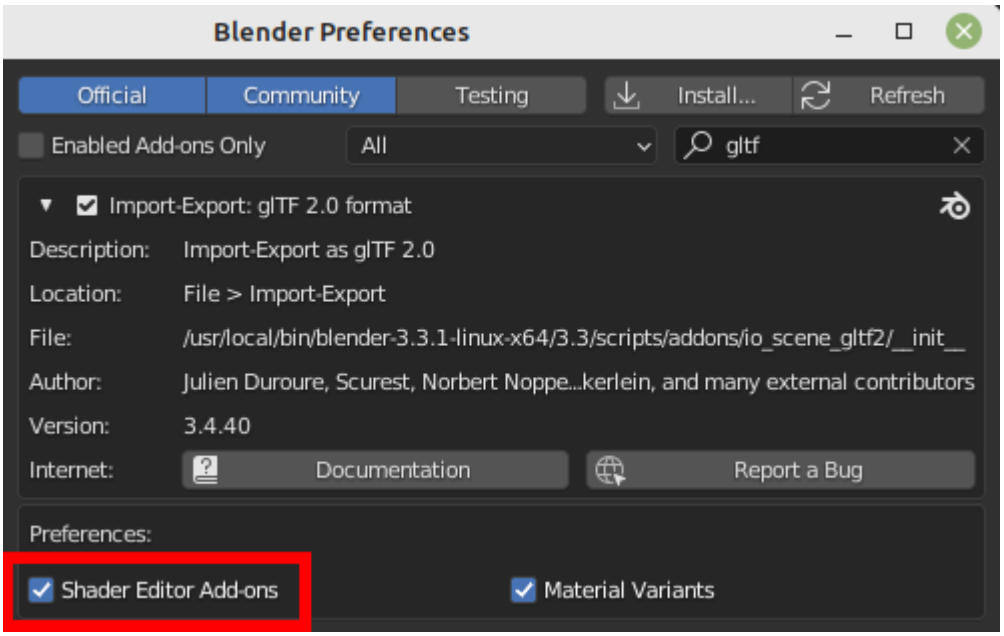
glTF is capable of storing a baked ambient occlusion map. Currently there is no arrangement of nodes that causes Blender to use such a map in exactly the same way as intended in glTF. However, if the exporter finds a custom node group by the name of `glTF Material Output`, and finds an input named `Occlusion` on that node group, it will look for an Image Texture attached there to use as the occlusion map in glTF. The effect need not be shown in Blender, as Blender has other ways of showing ambient occlusion, but this method will allow the exporter to write an occlusion image to the glTF. This can be useful to real-time glTF viewers, particularly on platforms where there may not be spare power for computing such things at render time.



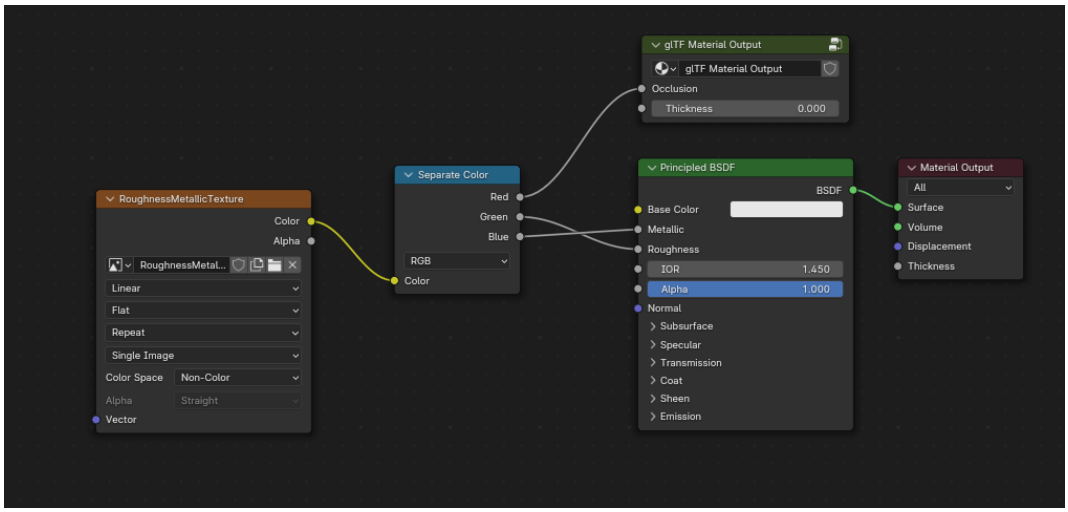
A pre-baked ambient occlusion map, connected to a node that doesn't render but will export to glTF.

Tip

If you enable Shader Editor Add-ons in preferences, you will be able to add this custom node group from Menu: Add > Output > glTF Material Output



glTF stores occlusion in the red (`R`) channel, allowing it to optionally share the same image with the roughness and metallic channels.



This combination of nodes mimics the way glTF packs occlusion, roughness, and metallic values into a single image.

Tip

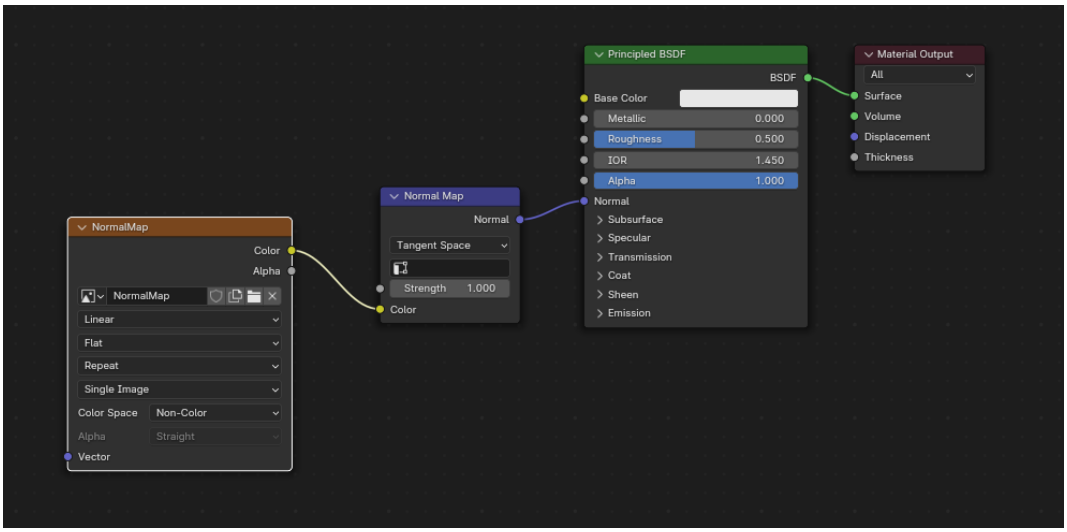
The Cycles render engine has a Bake panel that can be used to bake ambient occlusion maps. The resulting image can be saved and connected directly to the `glTF Material Output` node.

Normal Map

To use a normal map in glTF, connect an Image Texture node's color output to a Normal Map node's color input, and then connect the Normal Map normal output to the Principled BSDF node's normal input. The Image Texture node for this should have its *Color Space* property set to Non-Color.

The Normal Map node must remain on its default property of Tangent Space as this is the only type of normal map currently supported by glTF. The strength of the normal map can be adjusted on this node. The exporter

is not exporting these nodes directly, but will use them to locate the correct image and will copy the strength setting into the glTF.



A normal map image connected such that the exporter will find it and copy it to the glTF file.

Tip

The Cycles render engine has a Bake panel that can be used to bake tangent-space normal maps from almost any other arrangement of normal vector nodes. Switch the Bake type to Normal. Keep the default space settings (space: Tangent, R: +X, G: +Y, B: +Z) when using this bake panel for glTF. The resulting baked image can be saved and plugged into to a new material using the Normal Map node as described above, allowing it to export correctly.

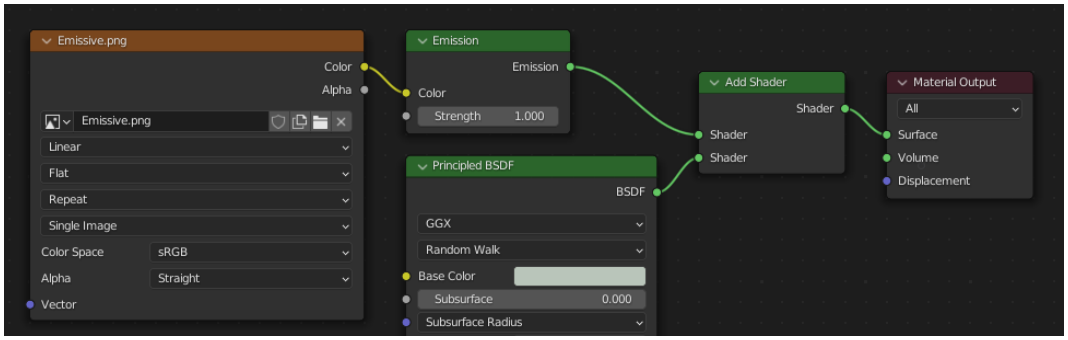
See: Cycles Render Baking

Emissive

An Image Texture node can be connected to the Emission input on the Principled BSDF node to include an emissive map with the glTF material. Alternatively, the Image Texture node can be connected to an Emission

shader node, and optionally combined with properties from a Principled BSDF node by way of an Add Shader node.

If the emissive map is alone in the material, it is best to set the Base Color default to black, and the Roughness default to 1.0. This minimizes the influence of the other channels if they are not needed.



> This arrangement is supported for backwards compatibility. It is simpler to use the Principled BSDF node directly.

If any component of emissiveFactor is > 1.0, `KHR_materials_emissive_strength` extension will be used.

Clearcoat

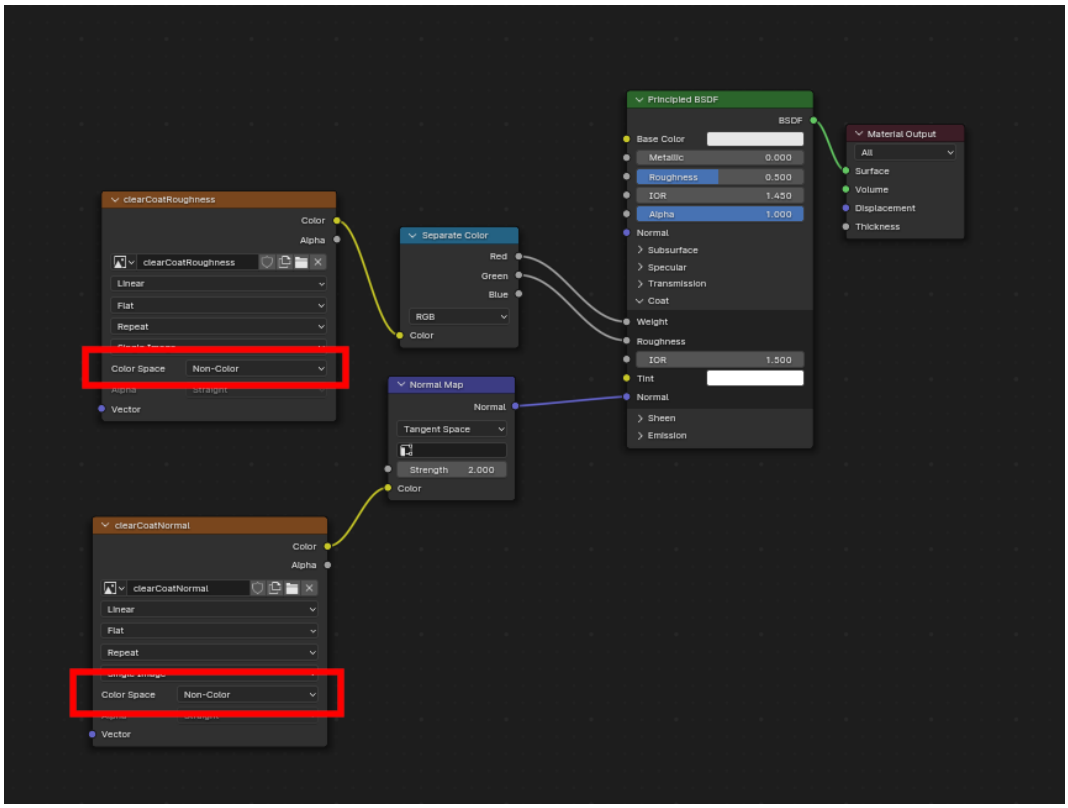
When the *Clearcoat* input on the Principled BSDF node has a nonzero default value or Image Texture node connected, the `KHR_materials_clearcoat` glTF extension will be included in the export. This extension will also include a value or Image Texture from the *Clearcoat Roughness* input if available.

If Image Textures are used, glTF requires that the clearcoat values be written to the red (`R`) channel, and *Clearcoat Roughness* to the green (`G`) channel. If monochrome images are connected, the exporter will remap them to these color channels.

The *Clearcoat Normal* input accepts the same kinds of inputs as the base Normal input, specifically a tangent-space normal map with +Y up, and a user-defined strength. This input can reuse the same normal map that the

base material is using, or can be assigned its own normal map, or can be left disconnected for a smooth coating.

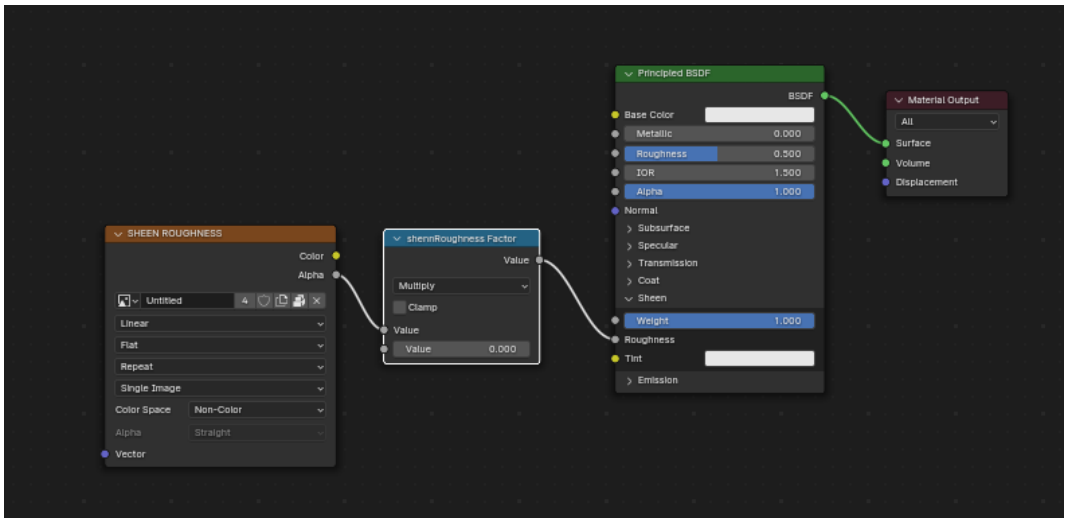
All Image Texture nodes used for clearcoat shading should have their *Color Space* set to Non-Color.



An example of a complex clearcoat application that will export correctly to glTF. A much simpler, smooth coating can be applied from just the Principled BSDF node alone.

Sheen

If a Sheen Roughness Texture is used, glTF requires the values be written to the alpha (`A`) channel.



Tip

Sheen BSDF node is only available on Cycles render engine. You may have to temporary switch to Cycles to add this node, and get back to EEVEE.

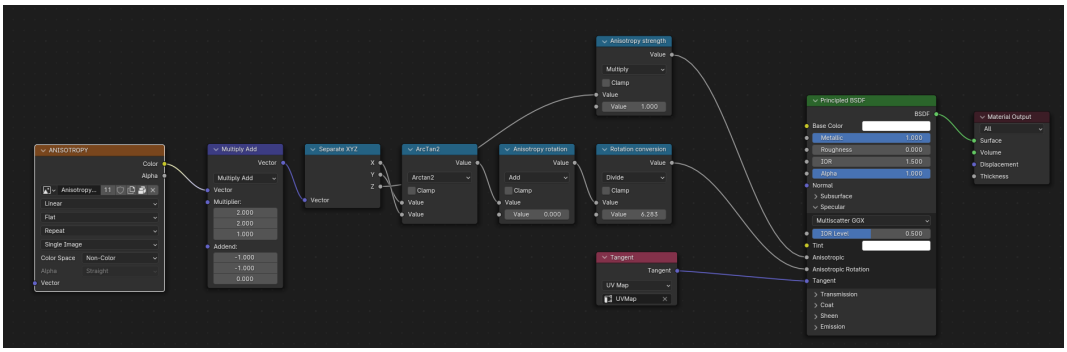
Specular

When the *Specular IOR Level* or *Specular Tint* input of Principled BSDF node have a non default value or Image Texture node connected, the `KHR_materials_specular` glTF extension will be included in the export.

Anisotropy

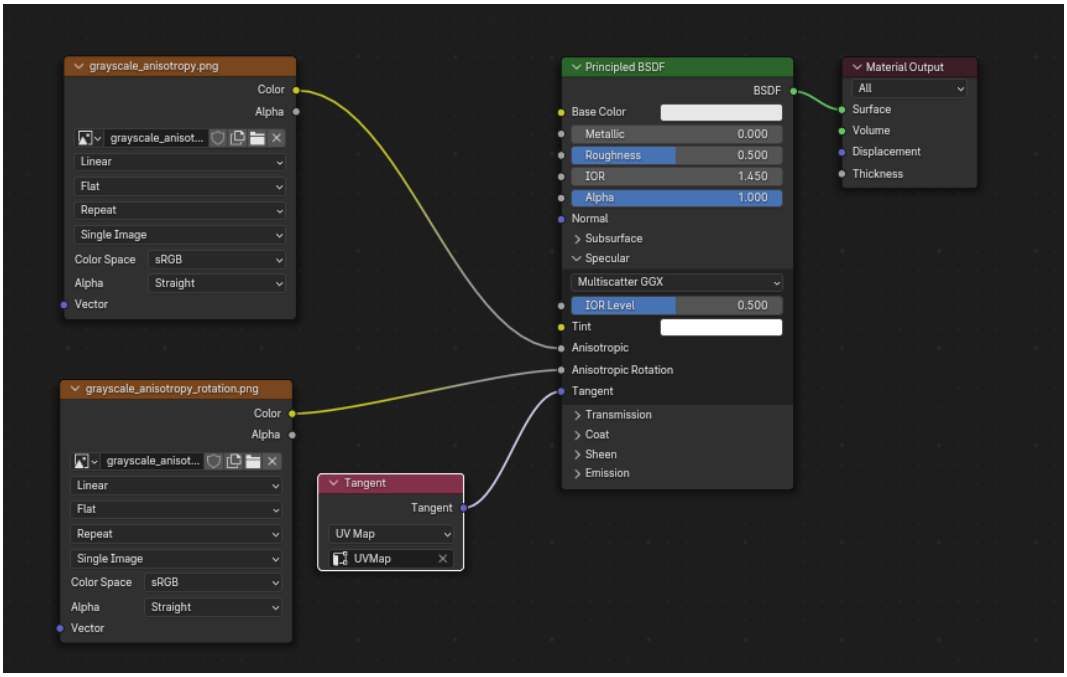
Anisotropic textures and data need to be converted at export, and at import.

At import, some nodes are created to manage this conversion



At export, this exact same nodes are detected, and used to export data.

At export, you can also plug some grayscale textures for *Anisotropic* and *Anisotropic Rotation* sockets. Then, exporter will convert these texture into a glTF compatible texture.



Note that the *tangent* socket must be linked to a *tangent* node, with UVMap. The choosen UVMap must be the UVMap of the Normal Map.

Transmission

When the Transmission input on the Principled BSDF node has a nonzero default value or Image Texture node connected, the

`KHR_materials_transmission` glTF extension will be included in the export. When a texture is used, glTF stores the values in the red (`R`) channel. The *Color Space* should be set to Non-Color.

Transmission is different from alpha blending, because transmission allows full-strength specular reflections. In glTF, alpha blending is intended to represent physical materials that are partially missing from the specified geometry, such as medical gauze wrap. Transmission is intended to represent physical materials that are solid but allow non-specularly-reflected light to transmit through the material, like glass.

Tip

The material's base roughness can be used to blur the transmission, like frosted glass.

Tip

Typically the alpha blend mode of a transmissive material should remain "Opaque", the default setting, unless the material only partially covers the specified geometry.

Note

In real-time engines where transmission is supported, various technical limitations in the engine may determine which parts of the scene are visible through the transmissive surface. In particular, transmissive materials may not be visible behind other transmissive materials. These limitations affect physically-based transmission, but not alpha-blended non-transmissive materials.

Note

If you want to enable refraction on your model, `KHR_materials_transmission` must also be used in addition with

KHR_materials_volume. See the dedicated *Volume* part of the documentation.

Warning

Transmission is complex for real-time rendering engines to implement, and support for the KHR_materials_transmission glTF extension is not yet widespread.

IOR

At import, there are two different situation:

- if KHR_materials_ior is not set, IOR value of Principled BSDF node is set to 1.5, that is the glTF default value of IOR.
- If set, the KHR_materials_ior is used to set the IOR value of Principled BSDF.

At export, IOR is included in the export only if one of these extensions are also used:

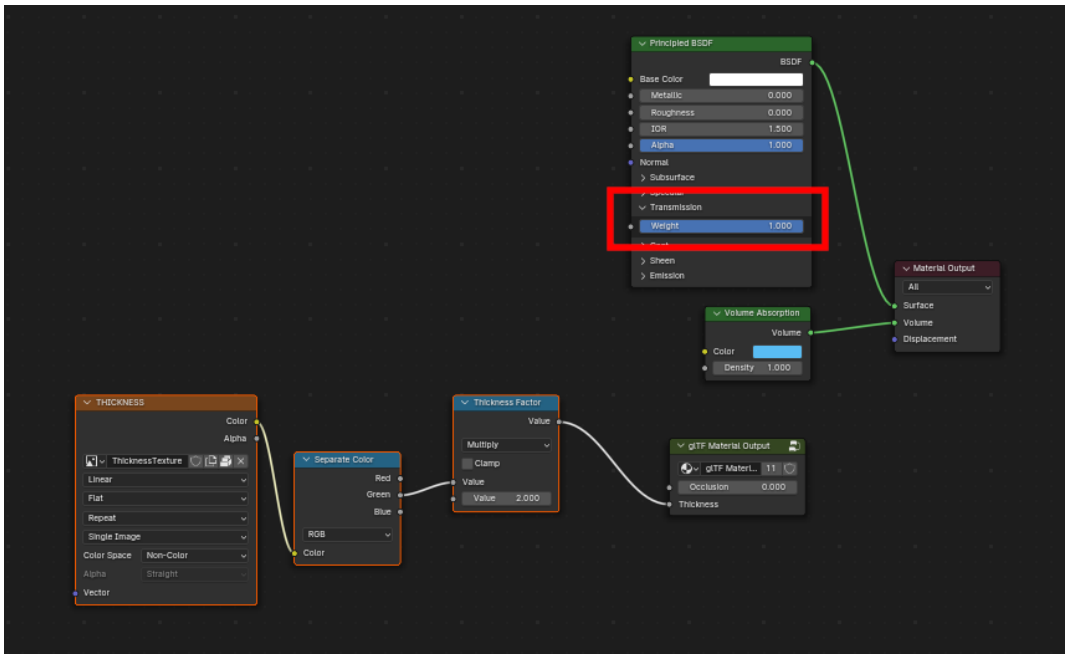
- KHR_materials_transmission
- KHR_materials_volume
- KHR_materials_specular

IOR of 1.5 are not included in the export, because this is the default glTF IOR value.

Volume

Volume can be exported using a Volume Absorption node, linked to Volume socket of Output node. Data will be exported using the KHR_materials_volume extension.

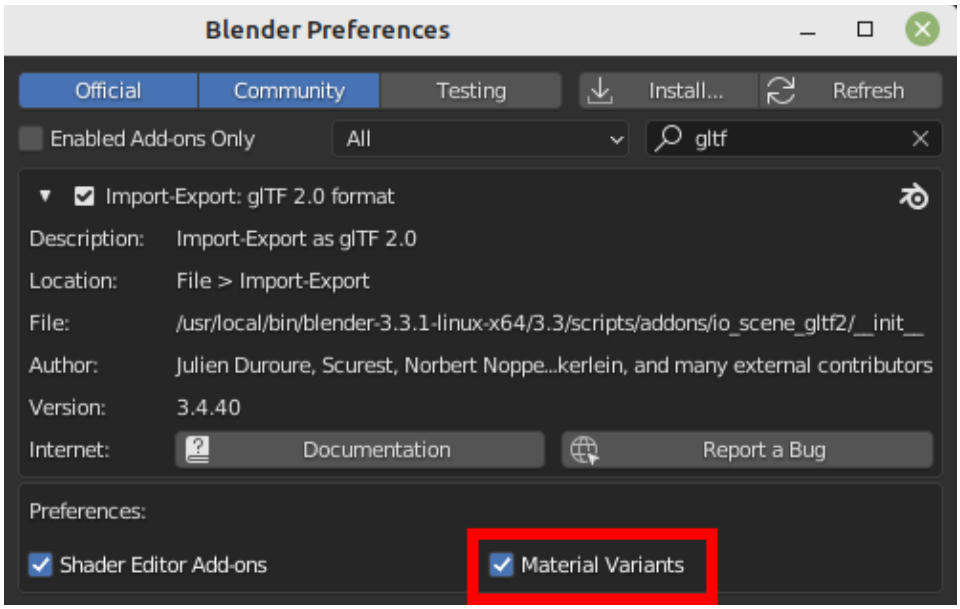
- For volume to be exported, some *transmission* must be set on Principled BSDF node.
- Color of Volume Absorption node is used as glTF attenuation color. No texture is allowed for this property.
- Density of Volume Absorption node is used as inverse of glTF attenuation distance.
- Thickness can be plugged into the Thickness socket of custom group node `glTF Material Output`.
- If a texture is used for thickness, it must be plugged on (`G`) Green channel of the image.



glTF Variants

Note

For a full Variants experience, you have to enable UI in Add-on preferences



There are two location to manage glTF Variants in Blender

- In 3D View, on `glTF Variants` tab
- For advanced settings, in Mesh Material Properties (see Advanced glTF Variant checks)

The main concept to understand for using Variants, is that each material slot will be used as equivalent of a glTF primitive.

glTF Variants switching

After importing a glTF file including `KHR_materials_variants` extension, all variants can be displayed.



You can switch Variant, by *selecting* the variant you want to display, then clicking on *Display Variant*.

You can switch to default materials (when no Variant are used), by clicking on *Reset to default*.

glTF Variants creation

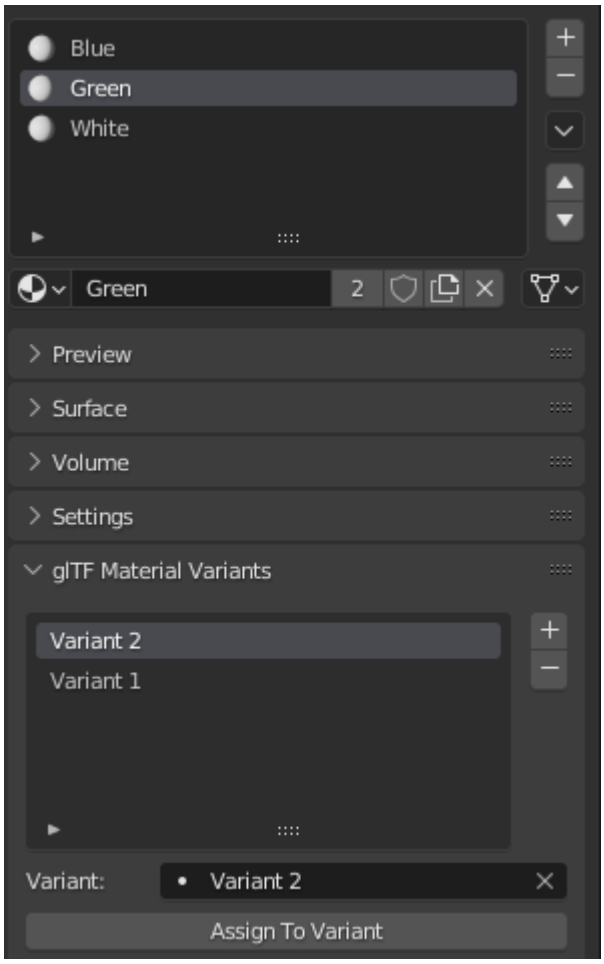
You can add a new Variant by clicking the + at right of the Variant list. Then you can change the name by double-clicking.

After changing Materials in Material Slots, you can assign current materials to the active Variant using *Assign to Variant*.

You can also set default materials using *Assign as Original*. These materials will be exported as default material in glTF. This are materials that will be displayed by any viewer that don't manage `KHR_materials_variants` extension.

Advanced glTF Variant checks

If you want to check primitive by primitive, what are Variants used, you can go to Mesh Material Properties.



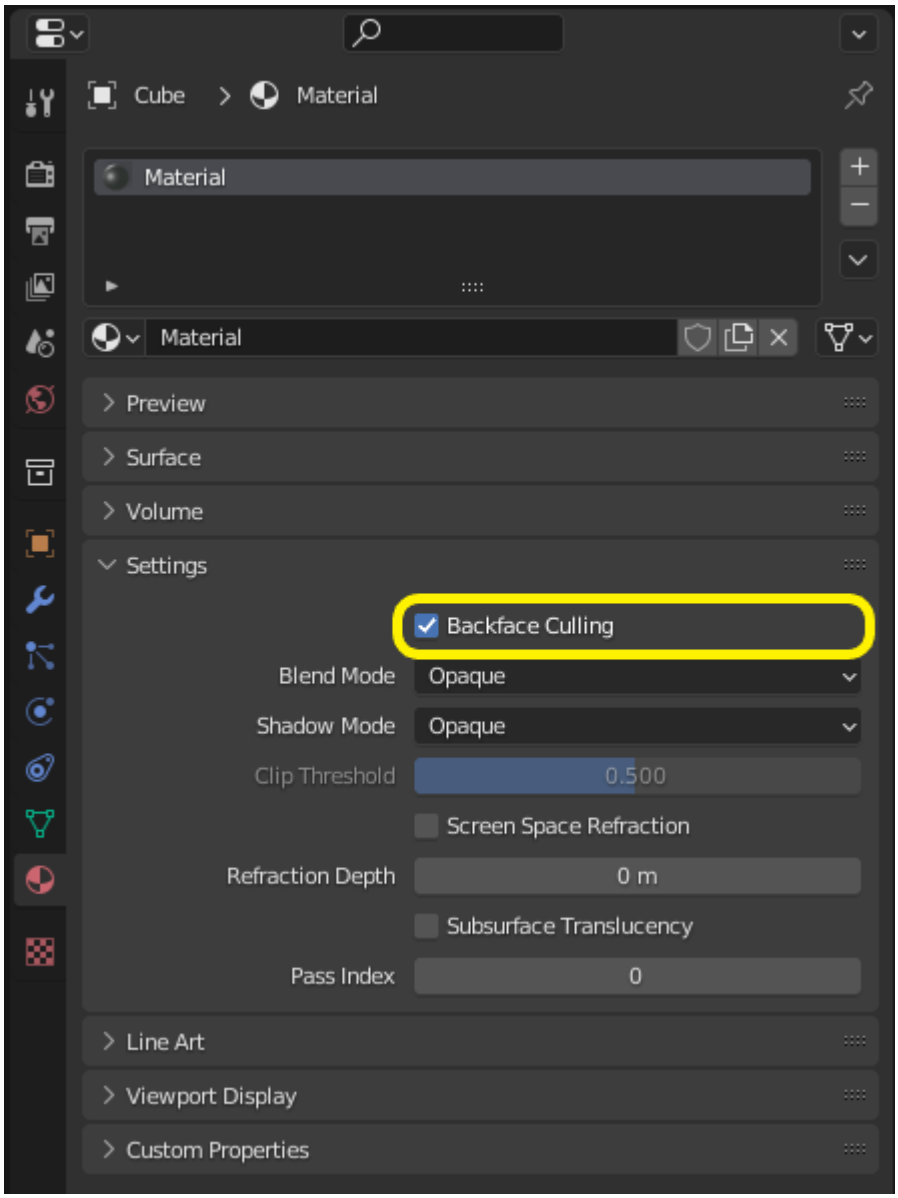
The *glTF Material Variants* tab refers to the active material Slot and Material used by this slot. You can see every Variants that are using this material for the given Slot/Primitive.

You can also assign material to Variants from this tab, but recommendation is to perform it from 3D View tab.

Double-Sided / Backface Culling

For materials where only the front faces will be visible, turn on *Backface Culling* in the *Settings* panel of an EEVEE material. When using other engines (Cycles, Workbench) you can temporarily switch to EEVEE to configure this setting, then switch back.

Leave this box unchecked for double-sided materials.



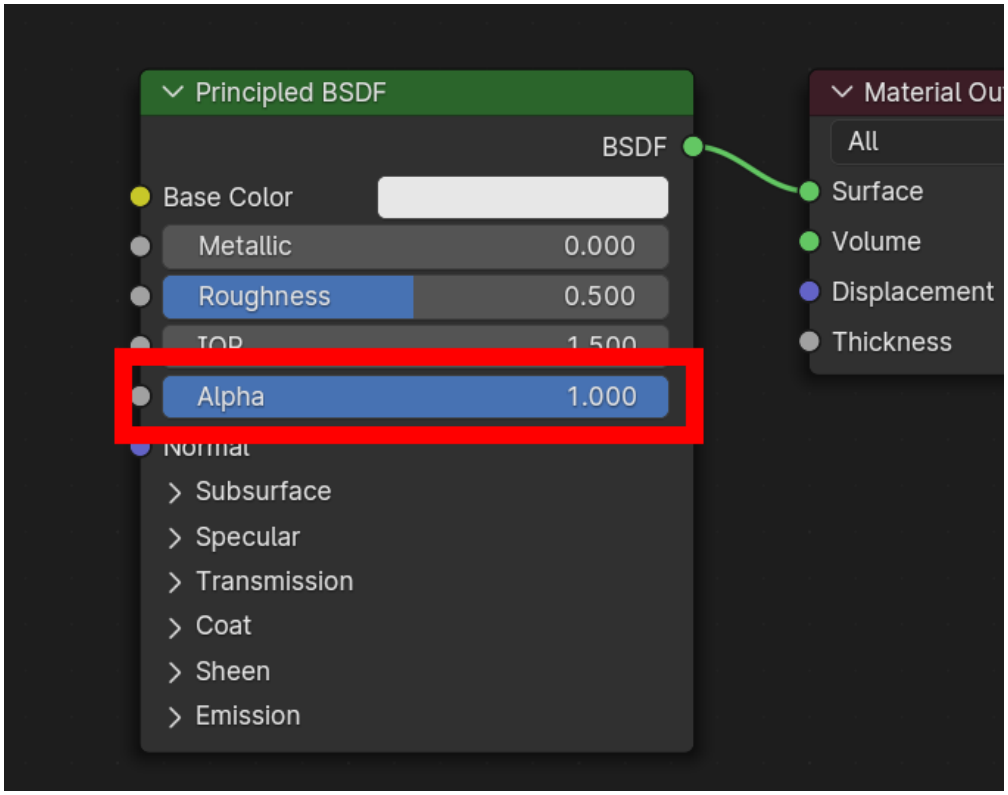
The inverse of this setting controls glTF's `DoubleSided` flag.

Alpha Modes

glTF has three alpha modes, depending on whether the alpha value is always 1, always 0 or 1, or can be between 0 and 1. The exporter determines the alpha mode automatically from the nodes connected to the Alpha socket.

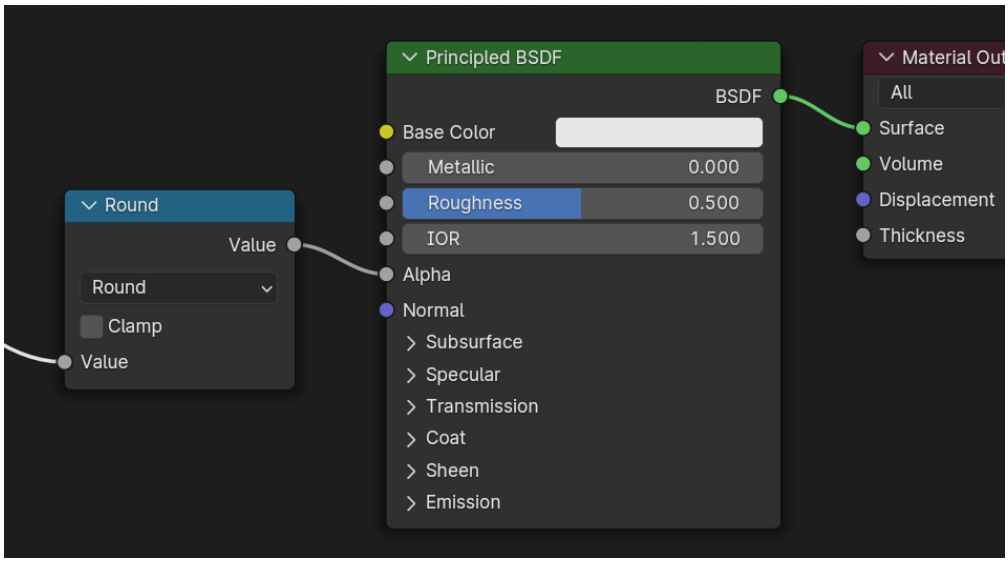
Opaque

In *opaque mode*, the material alpha is always 1.



Mask

In *mask mode*, the material alpha is always 0 or 1. This creates “cutout” transparency, where there is a hard edge between opaque and transparent regions, and can be used for things like leaves or cloth with holes. To enable this mode, use a Math node to round the alpha value to either 0 or 1.



Rounding snaps alpha values that are 0.5 or greater up to 1, and ones below 0.5 down to 1. It is also possible to use a cutoff value different than 0.5 by using Math nodes to do `1 - (alpha < cutoff)`.

Mask mode is essentially the same as EEVEE's "Alpha Clip" blend mode, but is done with shader nodes so it works in other render engines.

Blend

Materials that use neither of these will use *blend mode*. Blend mode allows partially transparent surfaces that tint the objects seen through them, like layers of colored film. However, partial transparency is complex to render, and glTF viewers may show visual artifacts in non-trivial scenes that use blend mode.

To avoid artifacts, it may be a good idea to separate out parts of a model that can use opaque or mask mode, and use blend mode only on the parts where it is necessary, or to use only a *single* layer of transparent polygons in front of opaque objects.

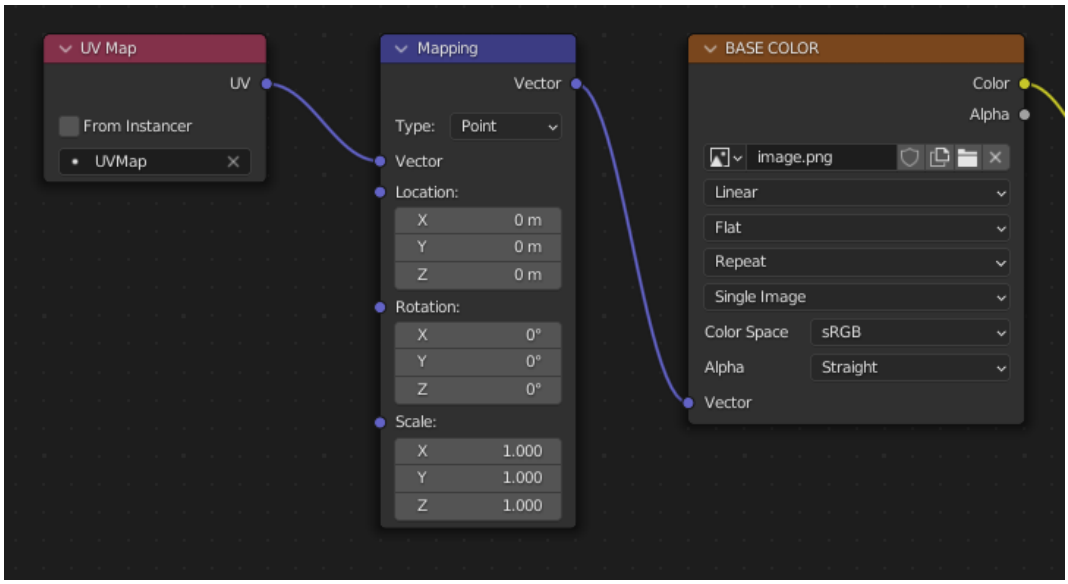
UV Mapping

Control over UV map selection and transformations is available by connecting a UV Map node and a Mapping node to any Image Texture node.

Settings from the Mapping node are exported using a glTF extension named `KHR_texture_transform`. There is a mapping type selector across the top. *Point* is the recommended type for export. *Texture* and *Vector* are also supported. The supported offsets are:

- *Location* - X and Y
- *Rotation* - Z only
- *Scale* - X and Y

For the *Texture* type, *Scale* X and Y must be equal (uniform scaling).



A deliberate choice of UV mapping.

Tip

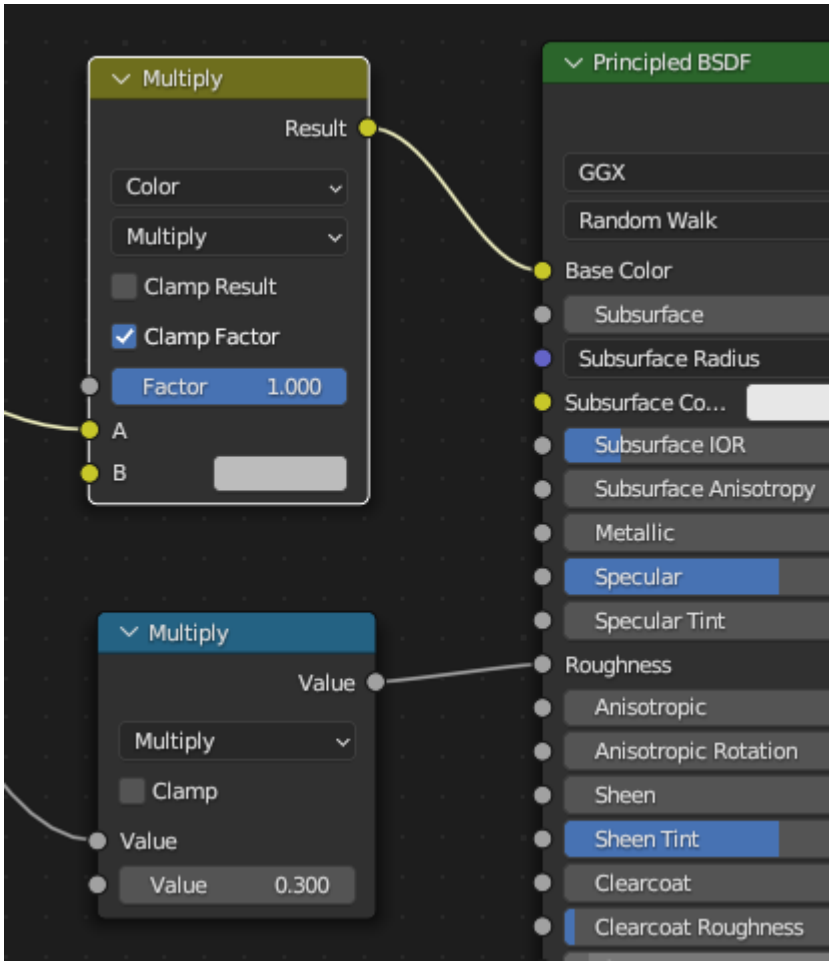
These nodes are optional. Not all glTF readers support multiple UV maps or texture transforms.

Factors

Any Image Texture nodes may optionally be multiplied with a constant color or scalar. These will be written as factors in the glTF file, which are

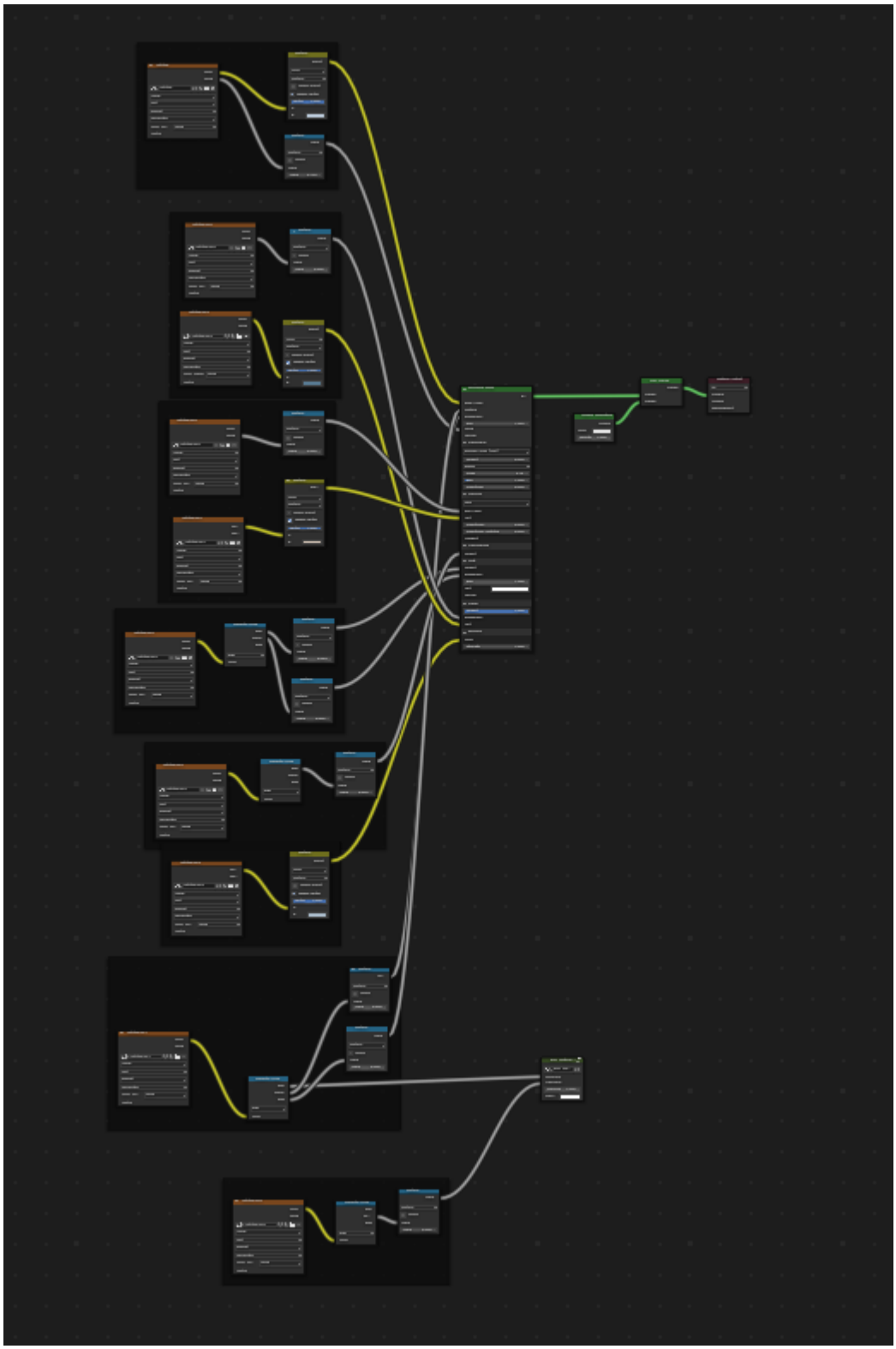
numbers that are multiplied with the specified image textures. These are not common.

- Use Math node (multiply) for scalar factors. Use second value as factor
- Use Mix node (color / multiply) for color factors. Set Factor to 1, and use Color2 (B) as factors



Example

A single material may use all of the above at the same time, if desired. This figure shows a typical node structure when several of the above options are applied at once:



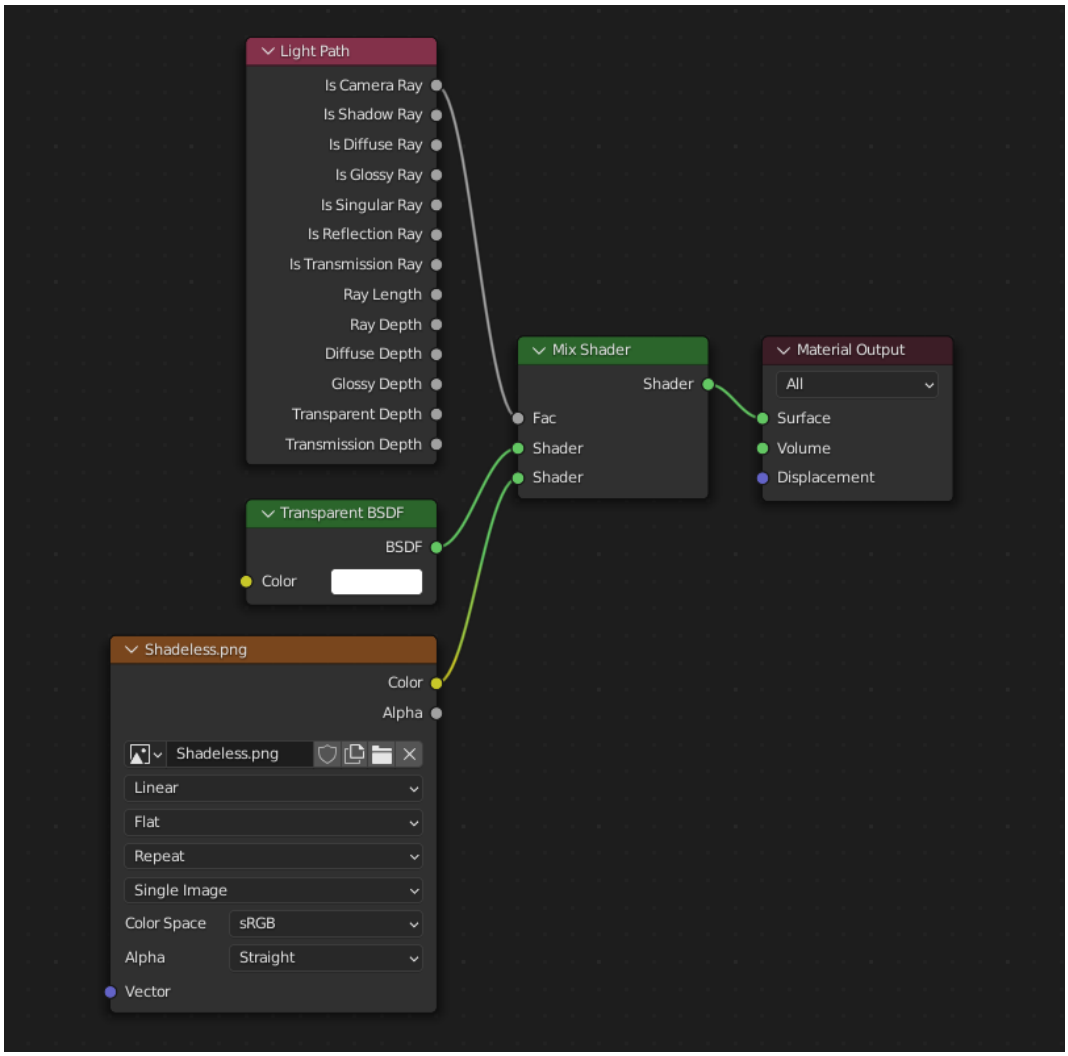
A Principled BSDF material with an emissive texture.

UDIM

UDIM is a way to store multiple textures in a single image file. The UDIM system is supported by Blender, but is not supported by glTF. When exporting a model that uses UDIM, the add-on will automatically split the image into multiple images, one for each tile, and will update the material nodes to use the new images. All UDIM texture must use the same UV map to be exported.

Exporting a Shadeless (Unlit) Material

To export an unlit material, mix in a camera ray, and avoid using the Principled BSDF node.



One of several similar node arrangements that will export `KHR_materials_unlit` and render shadeless in Blender.

Extensions

The core glTF 2.0 format can be extended with extra information, using glTF extensions. This allows the file format to hold details that were not considered universal at the time of first publication. Not all glTF readers support all extensions, but some are fairly common.

Certain Blender features can only be exported to glTF via these extensions. The following glTF 2.0 extensions are supported directly by this add-on:

Import

- `KHR_materials_pbrSpecularGlossiness`
- `KHR_materials_clearcoat`
- `KHR_materials_transmission`
- `KHR_materials_unlit`
- `KHR_materials_emissive_strength`
- `KHR_materials_volume`
- `KHR_materials_sheen`
- `KHR_materials_specular`
- `KHR_materials_anisotropy`
- `KHR_materials_ior`
- `KHR_materials_variants`
- `KHR_lights_punctual`
- `KHR_texture_transform`
- `KHR_mesh_quantization`
- `EXT_mesh_gpu_instancing`

Export

- `KHR_draco_mesh_compression`
- `KHR_lights_punctual`
- `KHR_materials_clearcoat`

- `KHR_materials_transmission`
- `KHR_materials_unlit`
- `KHR_materials_emissive_strength`
- `KHR_materials_volume`
- `KHR_materials_sheen`
- `KHR_materials_specular`
- `KHR_materials_anisotropy`
- `KHR_materials_ior`
- `KHR_materials_variants`
- `KHR_texture_transform`
- `EXT_mesh_gpu_instancing`

Third-party glTF Extensions

It is possible for Python developers to add Blender support for additional glTF extensions by writing their own third-party add-on, without modifying this glTF add-on. For more information, see the example on GitHub and if needed, register an extension prefix.

Custom Properties

Custom properties are always imported, and will be exported from most objects if the *Include ▸ Custom Properties* option is selected before export. These are stored in the `extras` field on the corresponding object in the glTF file.

Unlike glTF extensions, custom properties (extras) have no defined namespace, and may be used for any user-specific or application-specific purposes.

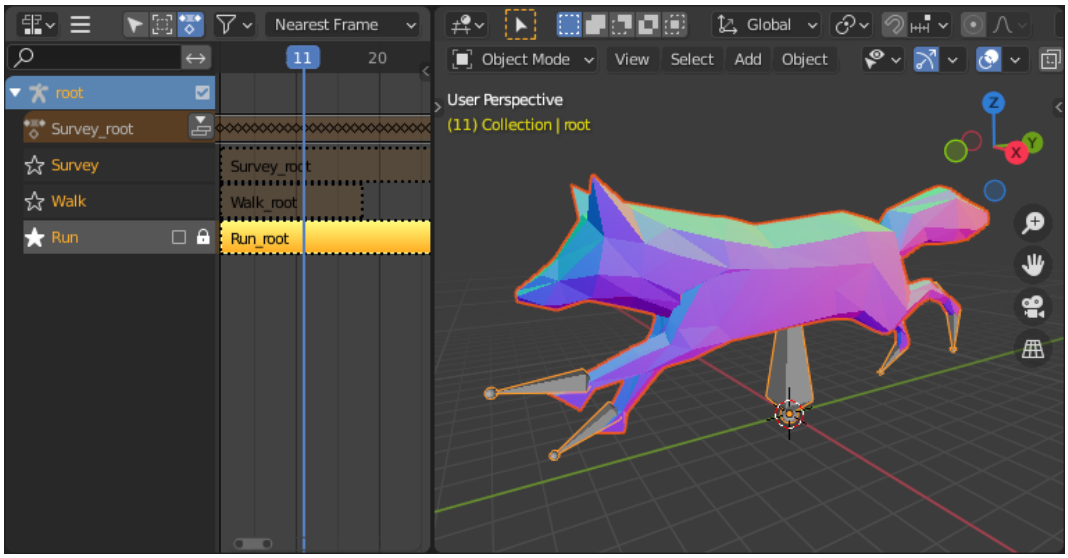
Animations

A glTF animation changes the transforms of objects or pose bones, or the values of shape keys. One animation can affect multiple objects, and there can be multiple animations in a glTF file.

Import

Imported models are set up so that the first animation in the file is playing automatically. Scrub the Timeline to see it play.

When the file contains multiple animations, the rest will be organized using the Nonlinear Animation editor. Each animation becomes an action stashed to an NLA track. The track name is the name of the glTF animation. To make the animation within that track visible, click Solo (star icon) next to the track you want to play.



This is the fox sample model showing its “Run” animation.

If an animation affects multiple objects, it will be broken up into multiple parts. The part of the animation that affects one object becomes an action stashed on that object. Use the track names to tell which actions are part of the same animation. To play the whole animation, you need to enable Solo (star icon) for all its tracks.

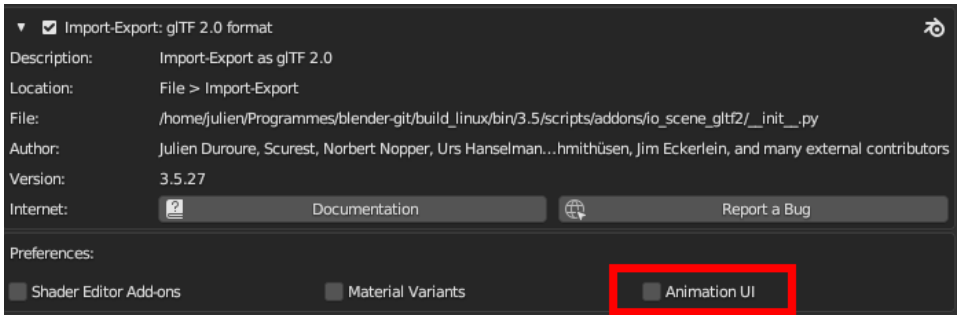
Note

There is currently no way to see the non-animated pose of a model that had animations.

You can also use the animation switcher that can be found in Dope Sheet editor.

Note

You have to enable UI in Add-on preferences for seeing the animation switcher



You can switch all animation imported. It automatically enables Solo (star icon) for all needed tracks. It also reset non animated object to Rest transformation.

Export

You can export animations using different ways. How glTF animations are made from actions / NLA is controlled by the *Animation ▸ Mode* export option.

Actions (default)

An action will be exported if it is the active action on an object, or it is stashed to an NLA track (e.g. with the *Stash* or *Push Down* buttons in the Action Editor). Actions which are **not** associated with an object in one of these ways are **not exported**. If you have multiple actions you want to export, make sure they are stashed!

A glTF animation can have a name, which is the action name by default. You can override it by renaming its NLA track from `NLATrack`/`[Action`

`Stash]` to the name you want to use. For example, the Fig. fox model will export with three animations, “Survey”, “Walk”, and “Run”. If you rename two tracks on two different objects to the same name, they will become part of the same glTF animation and will play together.

The importer organizes actions so they will be exported correctly with this mode.

This mode is useful if you are exporting for game engine, with an animation library of a character. Each action must be on its own NLA track.

Active Actions merged

In this mode, the NLA organization is not used, and only one animation is exported using the active actions on all objects.

NLA Tracks

In this mode, each NLA Track will be export as an independent glTF animation. This mode is useful if you are using Strip modifiers, or if you get multiple action on a same Track.

If you rename two tracks on two different objects to the same name, they will become part of the same glTF animation and will play together.

Scene

Using Scene option, animations will be exported as you can see them in viewport. You can choose to export a single glTF animation, or each object separately.

Note

Remember only certain types of animation are supported:

- Object transform (location, rotation, scale)
- Pose bones

- Shape key values

Animation of other properties, like physics, lights, or materials, will be ignored.

Note

In order to sample shape key animations controlled by drivers using bone transformations, they must be on a mesh object that is a direct child of the bones' armature.

Note

Only Actions (default) and Active Actions merged mode can handle not sampled animations.

File Format Variations

The glTF specification identifies different ways the data can be stored. The importer handles all of these ways. The exporter will ask the user to select one of the following forms:

glTF Binary (`.glb`)

This produces a single `.glb` file with all mesh data, image textures, and related information packed into a single binary file.

Tip

Using a single file makes it easy to share or copy the model to other systems and services.

glTF Separate (`.gltf` + `.bin` + textures)

This produces a JSON text-based `.gltf` file describing the overall structure, along with a `.bin` file containing mesh and vector data, and optionally a number of `.png` or `.jpg` files containing image textures referenced by the `.gltf` file.

Tip

Having an assortment of separate files makes it much easier for a user to go back and edit any JSON or images after the export has completed.

Note

Be aware that sharing this format requires sharing all of these separate files together as a group.

glTF Embedded (`.gltf`)

This produces a JSON text-based `.gltf` file, with all mesh data and image data encoded (using Base64) within the file. This form is useful if the asset must be shared over a plain-text-only connection.

Warning

This is the least efficient of the available forms, and should only be used when required. Available only when you activated it in addon preferences.

Properties

Import

Pack Images

Pack all images into the blend-file.

Merge Vertices
: The glTF format requires discontinuous normals, UVs, and other vertex attributes to be stored as separate vertices, as required for rendering on typical graphics hardware. This option attempts to combine co-located vertices where possible. Currently cannot combine verts with different normals.

Shading
: How normals are computed during import.

Guess Original Bind Pose
: Determines the pose for bones (and consequently, skinned meshes) in Edit Mode. When on, attempts to guess the pose that was used to compute the inverse bind matrices.

Bone Direction
: Changes the heuristic the importer uses to decide where to place bone tips. Note that the Fortune setting may cause inaccuracies in models that use non-uniform scaling. Otherwise this is purely aesthetic. The default value will not change axis, and is best for re-exporting from Blender. This default option will change display mode (adding shape and changing relationship line) to have a better view, even if original bones axis are not the most accurate (estheticaly speaking)

Lighting Mode
: Optional backwards compatibility for non-standard render engines. Applies to lights. Standard: Physically-based glTF lighting units (cd, lx, nt). Unitless: Non-physical, unitless lighting. Useful when exposure controls are not available Raw (Deprecated): Blender lighting strengths with no conversion

Import WebP textures
: If a texture exists in WebP format, loads the WebP texture instead of the fallback png/jpg one.

Export

Format
: See: File Format Variations.

Keep Original
: For glTF Separate file format only. Keep original textures files if possible. Warning: if you use more than one texture, where PBR standard requires only one, only one texture will be used. This can lead to unexpected results

Textures
: For glTF Separate file format only. Folder to place texture files in. Relative to the gltf-file.

Copyright
: Legal rights and conditions for the model.

Remember Export Settings
: Store export settings in the blend-file, so they will be recalled next time the file is opened.

Include

Selected Objects
: Export selected objects only.

Visible Objects
: Export visible objects only.

Renderable Objects
: Export renderable objects only.

Active Collection
: Export objects from active collection only.

Include Nested Collections
: Only when Active Collection is On. When On, export recursively objects on nested active collections.

Active Scene
: Export active scene only.

Custom Properties
: Export custom properties as glTF extras.

Cameras
: Export cameras.

Punctual Lights
: Export directional, point, and spot lights. Uses the `KHR_lights_punctual` glTF extension.

Transform

Y Up
: Export using glTF convention, +Y up.

Data - Scene Graph

Geometry Nodes Instances
: Export Geometry nodes instances. This feature is experimental.

GPU Instances
: Export using `EXT_mesh_gpu_instancing` extensions.

Flatten Object Hierarchy
: Useful in case of non-decomposable TRS matrix. Only skined meshes will stay children of armature.

Full Collection Hierarchy
: Export collections as empty, keeping full hierarchy. If an object is in multiple collections, it will be exported it only once, in the first collection it is found.

Data - Mesh

Apply Modifiers
: Export objects using the evaluated mesh, meaning the resulting mesh after all Modifiers have been calculated.

UVs
: Export UVs (texture coordinates) with meshes.

Normals
: Export vertex normals with meshes.

Tangents
: Export vertex tangents with meshes.

Attributes
: Export Attributes with meshes, when the name starts with underscore.

Loose Edges
: Export loose edges as lines, using the material from the first material slot.

Loose Points
: Export loose points as glTF points, using the material from the first material slot.

Shared Accessor
: For triangles, use shared accessor for indices. This is more efficient (smaller files when you have lots of materials).

Data - Mesh - Vertex Color

Use Vertex Color

Material::	Export vertex color when used in material node tree as Base Color multiplier. This is the default, and the most accurate regarding glTF specification.
Active::	Export active vertex colors, even if not used in material node tree. A fully compliant glTF viewer should display this VC as Base Color multiplier.

None:: Do not export vertex color.

Export all vertex colors

Export all vertex colors, additional VC will be COLOR_1, COLOR_2, etc.

Export active vertex color when no material

Export active vertex color when no material is assigned to the object.

Data - Material

Materials

Export full materials, only placeholders (all primitives but without materials), or does not export materials. (In that last case, primitives are merged, losing material slot information).

Images

Output format for images. PNG is lossless and generally preferred, but JPEG might be preferable for web applications due to the smaller file size. If WebP is chosen, all textures will be saved as WebP, without any png/jpg fallback. If None is chosen, materials are exported without textures.

Image Quality

When exporting jpeg or WebP files, the quality of the exported file.

Create WebP

Creates WebP textures for every textures, in addition to the existing texture. For already WebP textures, nothing happen.

WebP fallback

For all WebP textures, create a png fallback texture.

Unused images

Export images that are not used in any material.

Unused textures

Export texture info (sampler, image, texcoord) that are not used in any material.

Data - Shape Keys

Export shape keys (morph targets).

Shape Key Normals
Export vertex normals with shape keys (morph targets).

Shape Key Tangents
Export vertex tangents with shape keys (morph targets).

Data - Shape Keys - Optimize

Use Sparse Accessor if better
Sparse Accessor will be used if it save space (if the exported file is smaller)

Omitting Sparse Accessor if data is empty
If data is empty, omit to export SParce Accessor. Not all viewer managed it correctly, so this option is Off by default

Data - Armature

Use Rest Position Armature
Export Armatures using rest position as joint rest pose. When Off, the current frame pose is used as rest pose.

Export Deformation Bones only
Export Deformation bones only, not other bones. Animation for deformation bones are baked.

Remove Armature Object
Remove Armature Objects if possible. If some armature(s) have multiple root bones, we can't remove them.

Flatten Bone Hierarchy
Useful in case of non-decomposable TRS matrix.

Data - Skinning

Export skinning data

Bone influences
: How many joint verex influences will be exported. Models may appear incorrectly in many viewers with value different to 4 or 8.

Include All Bone Influences
: Export all joint vertex influences. Models may appear incorrectly in many viewers.

Data - Lighting

Lighting Mode
: Optional backwards compatibility for non-standard render engines. Applies to lights. Standard: Physically-based glTF lighting units (cd, lx, nt). Unitless: Non-physical, unitless lighting. Useful when exposure controls are not available Raw (Deprecated): Blender lighting strengths with no conversion

Data - Compression

Compress meshes using Google Draco.

Compression Level
: Higher compression results in slower encoding and decoding.

Quantization Position
: Higher values result in better compression rates.

Normal
: Higher values result in better compression rates.

Texture Coordinates
: Higher values result in better compression rates.

Color
: Higher values result in better compression rates.

Generic
: Higher values result in better compression rates.

Animation

Animation mode
: Animation mode used for export (See Animations)

Shape Keys Animations
: Export Shape Keys Animation. Need Shape Keys to be exported (See Data - Shape Keys)

Bake All Objects Animations
: Useful when some objects are constrained without being animated themselves.

Animation - Rest & Ranges

Use Current Frame as Object Rest Transformations
: Export the scene in the current animation frame. When off, frame 0 is used as rest transformation for objects.

Limit to Playback Range
: Clips animations to selected playback range.

Set all glTF Animation starting at 0
: Set all glTF Animation starting at 0. Can be useful for looping animation

Negative Frames
: When some frames are in negative range, slide or crop the animation.

Animation - Armature

Export all Armature Actions
: Export all actions, bound to a single armature. Warning: Option does not support exports including multiple armatures.

Reset pose bones between actions
: Reset pose bones between each action exported. This is needed when some bones are not keyed on some animations.

Animation - Sampling

Apply sampling to all animations. Do not sample animation can lead to wrong animation export.

Sampling Rate
: How often to evaluate animated values (in frames).

Animation - Optimize

Optimize Animation Size
: Reduce exported file size by removing duplicate keyframes when all identical.

Force keeping channel for armature / bones
: if all keyframes are identical in a rig, force keeping the minimal animation.

Force keeping channel for objects
: if all keyframes are identical for object transformations, force keeping the minimal animation.

Disable viewport for other objects
: When exporting animations, disable viewport for other objects, for performance reasons, when possible.

Animation - Filter

Restrict actions to be exported to the ones matching the filter.

Contributing

This importer/exporter is developed through the glTF-Blender-IO repository, where you can file bug reports, submit feature requests, or contribute code.

Discussion and development of the glTF 2.0 format itself takes place on the Khronos Group glTF GitHub repository, and feedback there is welcome.

Node

These add-ons relate to the node editors and related tools.

- Node Wrangler

Node Wrangler

Node Wrangler provides various tools that help you to work with nodes quickly and efficiently.

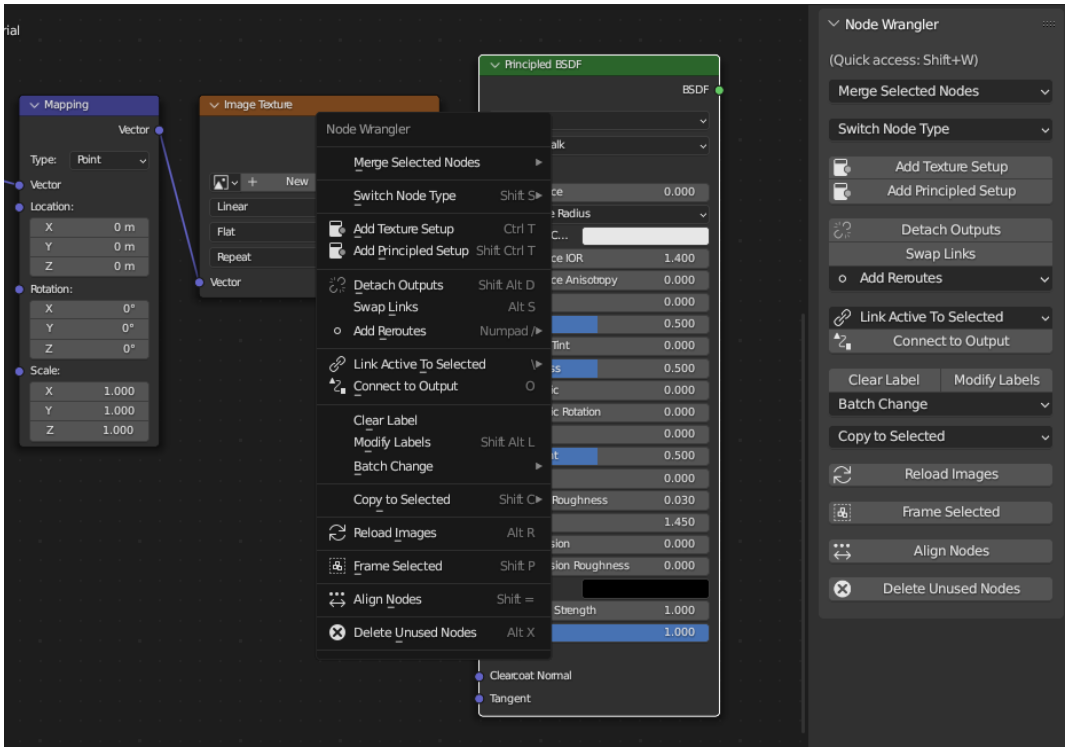
While many of this add-on's functions work in all supported node editors (Compositor, Shader, Geometry Nodes, and Texture Nodes) some functions only work in specific node editors, and some functions work differently per editor. Functions that only work in specific editors are marked with labels (*Compositor*, *Shader*, *Geometry Nodes*, *Texture Nodes*). Functions without labels should work for all node editors.

Enabling Add-on

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "Node Wrangler" and check the *Enable Extension* checkbox.

Usage

Use the panel in Sidebar of the node editor or press `Shift-W` to bring up the quick access menu. You can also look up the shortcut list in the add-on preferences panel.



You can access most functions from the sidebar panel or quick access menu.

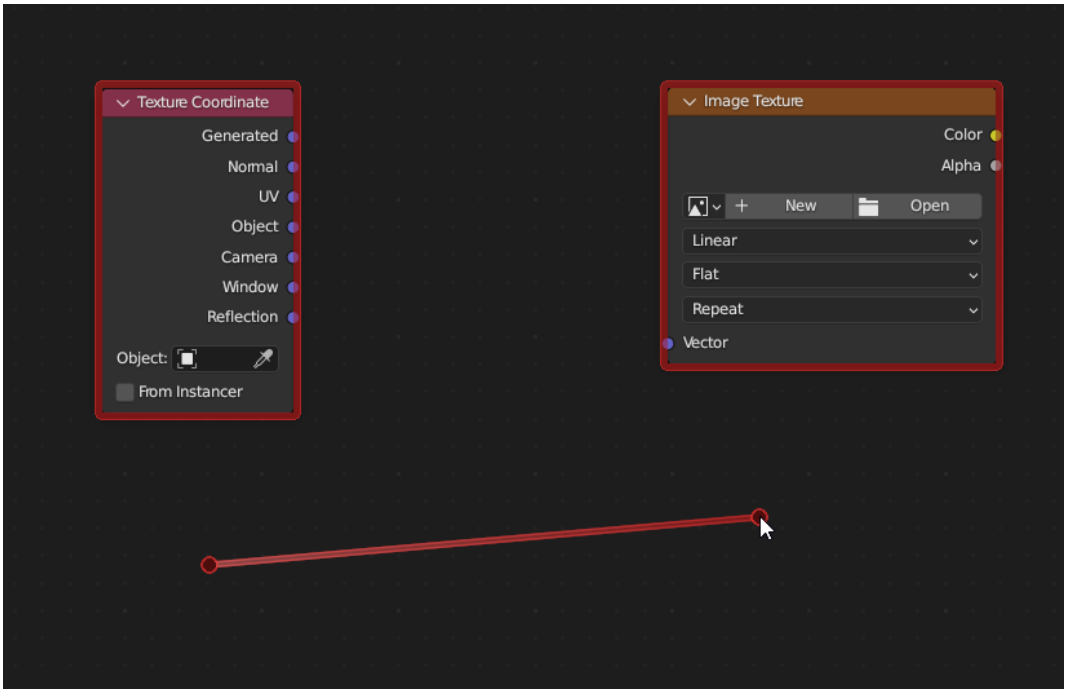
Description

Lazy Connect

Reference

Shortcut:: `Alt-RMB`-drag, `Shift-Alt-RMB`-drag

Connect two nodes without even clicking the sockets. Just drag the cursor from one node to another while holding `Alt-RMB`. It will select the nodes nearest the start and end points of the drag for connection, so you don't even have to click on the nodes.



Selection can be lazy.

It tries to connect the best-matched sockets possible, based on their names, types, and whether they are open or not.

For a more precise connection, you can alternatively use `Shift-Alt-RMB`. It brings up menus of available inputs and outputs before connection, so you can select the exact sockets to connect. It's especially useful when working with a large node tree since you can make connections without frequently zooming in and out.

Lazy Mix

Reference

Shortcut:: `Shift-Ctrl-RMB`-drag

Connect the outputs of two nodes into an appropriate "mix" type of node. This is the "lazy" way of selecting nodes and executing the *Mix* function from Merge with Automatic Type Detection.

Merge

Reference

Menu:: *Node Wrangler ▸ Merge Selected Nodes*

Connect outputs of the selected nodes into a “mix” type of node (Mix, Math, Z-Combine, Alpha Over, Mix Shader, Add Shader, Join Geometry).

Note

Merge currently does not support outputs of Integer, String, or Boolean types from Geometry Nodes.

There are automatic and manual ways of merging. The automatic ways let the add-on determine which “mix” node to use based on the types of outputs to merge. The manual ways let you decide and force connections even if the types of outputs and the “mix” node are not compatible.

Note

Generally, the modifier part of the shortcut signifies the type of “mix” node you want to use (`Ctrl` for automatic detection, `Ctrl-Alt` for the Mix node, and `Shift-Ctrl` for the Math node), the non-modifier part signifies the mode of “mix” node you want to set (`NumpadPlus` for add, `NumpadMinus` for subtract, `NumpadSlash` for divide, and `NumpadAsterisk` for multiply).

Merge with Automatic Type Detection

The automatic merge functions determine the type of “mix” node to use based on the types of outputs to merge. If it has a Color output, it will use the Mix node. It will use the Math node if both outputs are of Value type.

Add Shader, Mix Shader, and Join Geometry nodes will also be used for specific cases.

Modes

Add `Ctrl-=`, `Ctrl-NumpadPlus`
: Merge into Mix or Math nodes, then set blend mode or math operation as Add. If the outputs are Shaders, it will use Add Shader node instead.

Multiply `Ctrl-8`, `Ctrl-NumpadAsterisk`
: Merge into Mix or Math nodes, then set blend mode or math operation as Multiply.

Subtract `Ctrl-Minus`, `Ctrl-NumpadMinus`
: Merge into Mix or Math nodes, then set blend mode or math operation as Subtract.

Divide `Ctrl-Slash`, `Ctrl-NumpadSlash`
: Merge into Mix or Math nodes, then set blend mode or math operation as Divide.

Mix `Ctrl-0`, `Ctrl-Numpad0`
: Merge into Mix node, then set blend mode as Mix. If the outputs are Shaders, it will use Mix Shader node instead. If the outputs are Geometry, it will use Join Geometry node.

Merge Using Mix Node

Reference

Menu:: *Node Wrangler ▸ Merge Selected Nodes ▸ Use Mix Nodes*

Use the Mix nodes for merging, regardless of the selected nodes. You can choose the mode of the node via the menu. You can quickly set some operations by using corresponding shortcuts.

- Add: `Ctrl-Alt-=`, `Ctrl-Alt-=`
- Substract: `Ctrl-Alt-Minus`, `Ctrl-Alt-NumpadMinus`
- Multiply: `Ctrl-Alt-8`, `Ctrl-Alt-NumpadAsterisk`
- Divide: `Ctrl-Alt-Slash`, `Ctrl-Alt-NumpadSlash`

Merge Using Math Node

Reference

Menu::	*Node Wrangler ▸ Merge Selected Nodes ▸ Use Math Nodes*

Use the Math nodes for merging, regardless of the selected nodes. You can choose the mode of the node via the menu. You can quickly set some operations by using corresponding shortcuts.

- Add: `Shift-Ctrl-=`, `Shift-Ctrl-=`
- Substract: `Shift-Ctrl-Minus`, `Shift-Ctrl-NumpadMinus`
- Multiply: `Shift-Ctrl-8`, `Shift-Ctrl-NumpadAsterisk`
- Divide: `Shift-Ctrl-Slash`, `Shift-Ctrl-NumpadSlash`
- Greater than: `Ctrl-Comma`
- Less than: `Ctrl-Period`

Merge Using Z-Combine Node

Compositor

Reference

Menu::	*Node Wrangler ▸ Merge Selected Nodes ▸ Use Z-Combine Nodes*
Shortcut::	`Ctrl-NumpadPeriod`

Use the Z-Combine nodes for merging. If possible, Image and Z-Depth outputs will be linked. If the current node editor is not Compositor, this will execute the *Mix* function from the automatic merge.

Merge Using Alpha Over Node

Compositor

Reference

Menu::	*Node Wrangler ▸ Merge Selected Nodes ▸ Use Alpha Over Nodes*
Shortcut::	`Ctrl-Alt-0`

Use the Alpha Over nodes for merging. If the current node editor is not Compositor, this will execute the *Mix* function from the automatic merge.

Batch Change Blend Mode / Math Operation

Reference

Menu::	*Node Wrangler ▸ Batch Change*

Change the blend mode or math operation of the selected Mix and Math nodes at once. You can use `Alt-Up` or `Alt-Down` to cycle through previous or next blend modes or math operations. You can also quickly set some operations by using corresponding shortcuts.

- Add: `Alt-=`, `Alt-=`
- Substract: `Alt-Minus`, `Alt-NumpadMinus`
- Multiply: `Alt-8`, `Alt-NumpadAsterisk`
- Divide: `Alt-Slash`, `Alt-NumpadSlash`
- Greater than: `Alt-Comma`
- Less than: `Alt-Period`

Change Mix Factor

Reference

Shortcut:: `Alt-Left`, `Shift-Alt-Left`, `Alt-Right`, `Shift-Alt-Right`, `Shift-Ctrl-Alt-Left`, `Shift-Ctrl-Alt-0`, `Shift-Ctrl-Alt-Right`, `Shift-Ctrl-Alt-1`

Change the Factor value of the selected Mix and Mix Shader nodes with shortcuts.

- Increase Factor by 0.1: `Alt-Right`
- Decrease Factor by 0.1: `Alt-Left`
- Increase Factor by 0.01: `Shift-Alt-Right`
- Decrease Factor by 0.01: `Shift-Alt-Left`
- Set Factor to 0.0: `Shift-Ctrl-Alt-Left`, `Shift-Ctrl-Alt-0`
- Set Factor to 1.0: `Shift-Ctrl-Alt-Right`, `Shift-Ctrl-Alt-1`

Delete Unused Nodes

Reference

Menu:: *Node Wrangler ▸ Delete Unused Nodes*
Shortcut:: `Alt-X`

Clean up your node tree. Delete all nodes that don't contribute to the final result.

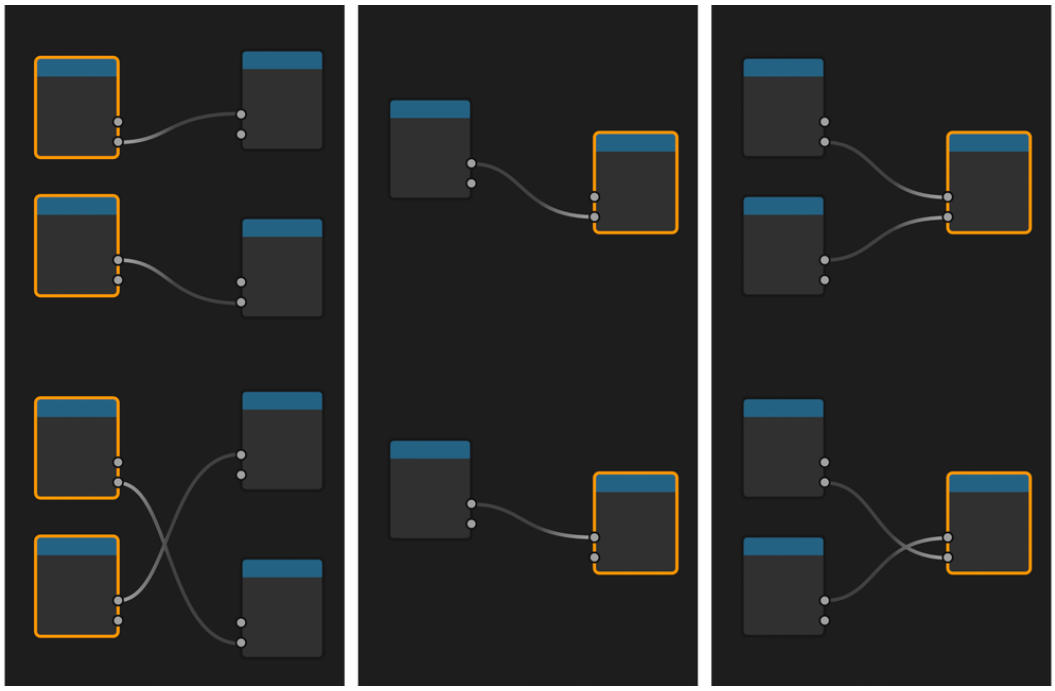
Swap Links

Reference

Menu:: *Node Wrangler ▸ Swap Links*
Shortcut:: `Alt-S`

When two nodes are selected, this swaps each other's output link. Note that some output connections can be lost if the two nodes have a different number of connected outputs.

With one node selected, if the node has one linked input, it cycles the link through the available input sockets. If the node has two linked inputs, it swaps those two links. If there are more than two inputs linked, it swaps the two inputs with matching types (the Mix node's two Color inputs, for example).



Swap works differently depending on the selected nodes and their links.

Reset Backdrop

Compositor

Reference

Menu:: *Node Wrangler ▸ Reset Backdrop*
Shortcut:: Z

Reset the position and scale of the backdrop.

Add Attribute Node

Shader

Reference

Menu:: *Header* ▸ *Add* ▸ *Input* ▸ *Attributes*

Add an Attribute node with the selected attribute.

Preview Node Output

Shader Geometry Nodes

Reference

Shortcut:: `Shift-Ctrl-LMB` for *Shader*, `Shift-Alt-LMB` for *Geometry Nodes*

Connect an output of the selected node to the final output of the node tree (the Material Output or World Output for Shader, the final Group Output for Geometry Nodes) to preview its output in the viewport. You can cycle through the available outputs by clicking it again while holding the modifier keys.

See also

While in Shader, any output can be connected to the final output, in Geometry Nodes, only Geometry outputs can be connected to the final output. To preview other types of outputs in Geometry Nodes, use its own Viewer Node.

See also

Also check out *Connect to Output*. It is a similar function but has different behaviors. It also works in all node editors.

Frame Selected

Reference

Menu::	*Node Wrangler ▸ Frame Selected*
Shortcut::	`Shift-P`

Insert the selected nodes into a Frame node.

Reload Images

Compositor Shader Texture Nodes

Reference

Menu::	*Node Wrangler ▸ Reload Images*
Shortcut::	`Alt-R`

Reload all of the images used in the node tree. This lets you reload the images without using the Image Editor.

Copy Settings

Reference

Menu::	*Node Wrangler ▸ Copy to Selected ▸ Settings from Active*
Shortcut::	`Shift-C`

Copy the settings of the active node to all selected nodes of the same type.

Reset Nodes

Reference

Shortcut:: `Backspace`

Revert the settings of the selected nodes to default while maintaining connections.

Copy Label

Reference

Menu:: *Node Wrangler ▸ Copy to Selected ▸ Copy Label*
Shortcut:: `Shift-V`, `Shift-C`

Copy custom labels to all of the selected nodes. You can copy them from the active node (`Shift-V`), from the nodes that are linked to the selected ones, or from the names of the sockets that the selected nodes are linked to. `Shift-C` will bring up a submenu with all available options.

Clear Label

Reference

Menu:: *Node Wrangler ▸ Clear Label*
Shortcut:: `Alt-L`

Clear the custom labels of selected nodes and revert them back to their default node names.

Modify Labels

Reference

Menu:: *Node Wrangler ▸ Modify Labels*
Shortcut:: `Shift-Alt-L`

Batch rename the custom labels of selected nodes. You can add text to the beginning and the end and replace parts of the text.

Add Texture Setup

Shader

Reference

Menu::	*Node Wrangler ▸ Add Texture Setup*
Shortcut::	`Ctrl-T`

Add a setup of a texture node, Texture Coordinate, and Mapping nodes to any shader node. If you select a texture node, it will only add the Texture Coordinate and Mapping nodes. For a background shader it will add an Environment Texture node.

Add Principled Texture Setup

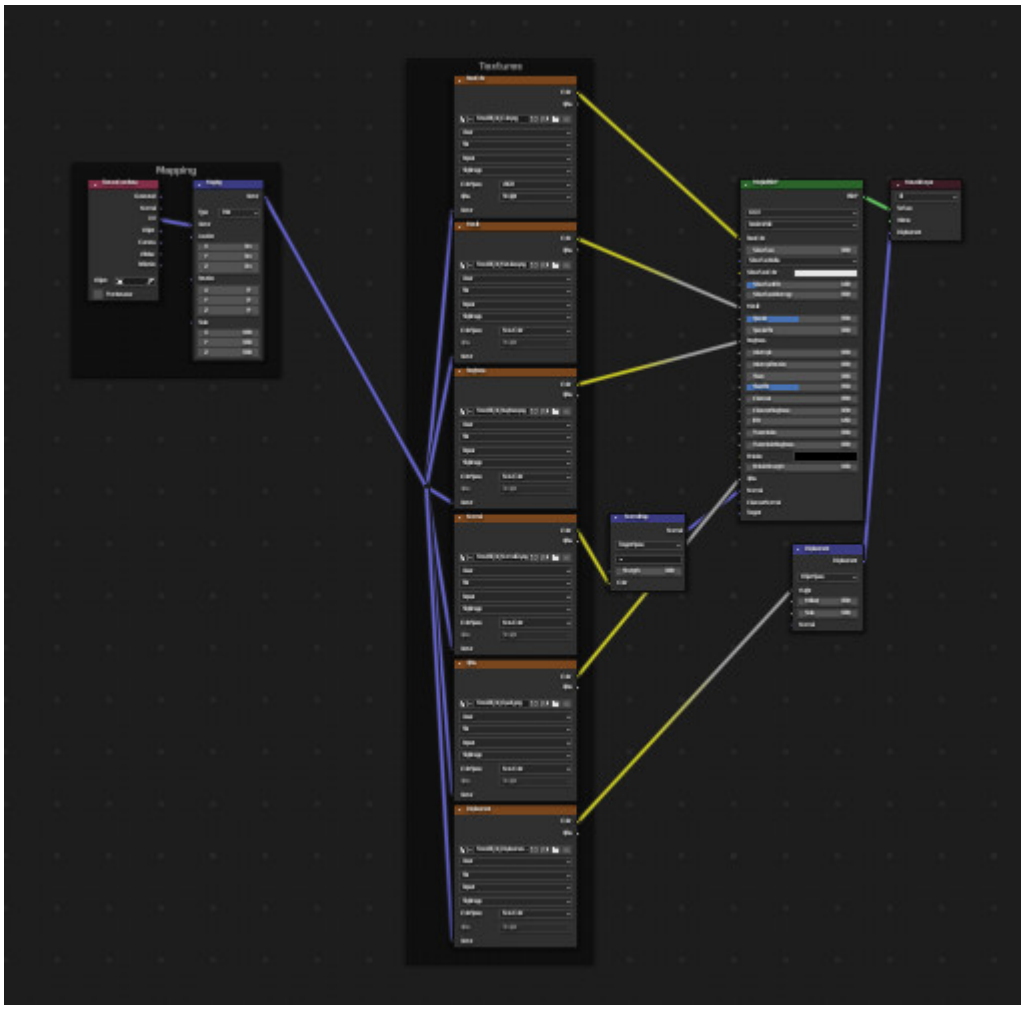
Shader

Reference

Menu::	*Node Wrangler ▸ Add Principled Setup*
Shortcut::	`Shift-Ctrl-T`

Add a principled texture setup from the selected texture files. Select a Principled BSDF node, select *Add Principled Setup* from the quick access menu (or press `Shift-Ctrl-T`), and select texture files. It automates the process of adding Image Texture nodes, loading images, selecting the appropriate Color Space, and connecting their outputs to the Principled BSDF node.

It detects the type of textures by looking at their file names. You can edit the tags used for this matching process in the add-on preferences.



Setting up these textures can take dozens of clicks, even with Node Wrangler's other tools. With Principled Texture Setup, you can reduce that to a few clicks.

Add Reroutes to Outputs

Reference

Menu:: *Node Wrangler* ▸ *Add Reroutes*
Shortcut:: `Slash`

Add reroute nodes to each output of the selected nodes.

Link Active to Selected

Reference

Menu::	*Node Wrangler ▸ Link Active to Selected*
Shortcut::	`Backslash`

Link the active node to the selected nodes based on various criteria.

To All Selected
: Link the active node to all selected nodes. (`K`) You can force it to replace existing links. (`Shift-K`)

Use Node Name/Label
: Link only to the selected nodes that have the same label as the active node. (`'`) You can force it to replace existing links. (`Shift-'`)

Use Outputs Names
: Link only when the name of the outputs matches the name or label of the selected nodes. (`;`) You can force it to replace existing links. (`Shift-;`) This is handy for replacing sources at the same time. (For example, connecting outputs from Render Layer to image (multi-layer EXR) in Compositor.)

Align Nodes

Reference

Menu::	*Node Wrangler ▸ Align Nodes*
Shortcut::	`Shift-=`

Align the selected nodes horizontally or vertically. The effect is similar to scaling nodes on an axis (`S X 0` or `S Y 0`), but it places the nodes at an even distance.

Select within Frame (Parent/Children)

- `]` – Select all direct child nodes of the selected frame.

- [– Select the direct parent frame node of the selected nodes.

Detach Outputs

Reference

Menu::	*Node Wrangler* ▸ *Detach Outputs*
Shortcut::	`Shift-Alt-D`

Detach the selected node's outputs while leaving linked inputs intact.

Connect to Output

Reference

Menu::	*Node Wrangler* ▸ *Connect to Output*
Shortcut::	`0`

Connect the output of the selected node to the final output of the node tree (Composite in Compositor, Material Output or World Output in Shader, the final Group Output in Geometry Nodes, Output in Texture Nodes), or, if the node is inside a group, to the Group Output.

Add Image Sequence

Compositor Shader

Reference

Menu::	*Add* ▸ *Input* for *Compositor*, or *Add* ▸ *Texture* for *Shader*

Add an Image Sequence by only selecting one image from a sequence of image files. It will automatically detect the length of the sequence and set the node appropriately.

Add Multiple Images

Compositor Shader

Reference

Menu::	*Add* ▸ *Input* for *Compositor*, or *Add* ▸ *Texture* for *Shader*

Select multiple images and add a node for each image. (Useful for importing multiple render passes or renders for image stacking.)

See also

Please see the old Wiki for the archived original docs.

Reference

Category::	Node
Description::	Various tools to enhance and speed up node-based workflow.
Location::	*Node editor* ▸ *Sidebar* or see the shortcuts of individual tools.
File::	node_wrangler.py
Author::	Bartek Skorupa, Greg Zaal, Sebastian Koenig, Christian Brinkmann, Florian Meyer
License::	GPL
Note::	This add-on is bundled with Blender.

Rigging

These add-ons relate to rigging and armatures.

- Rigify

Rigify

Basics

- Introduction
  - Main Features
  - Enabling Add-on
- Basic Usage
  - Basic Rig Generation
  - Advanced Rig Generation
  - Library Linking
- Bone Positioning Guide
  - Face Bones
  - Torso Bones
  - Limbs Bones
  - Fingers Bones
- Generated Rig Features
  - Common Features
  - Limbs
  - Fingers & Tentacles
  - Spine, Head & Tail
  - Face

Customization

- Creating Meta-rigs
- Rig Types

Extensions

- Feature Sets

Development

Developer documentation is available on the Blender Developer Documentation.

Reference

Category::	Rigging
Description::	Automatic rigging from building-block components.
Location::	*Properties* ▸ *Armature*, *Bone*, *3D Viewport* ▸ *Tools panel*, *3D Viewport* ▸ *Add menu* ▸ *Armature*
File::	rigify folder
Author::	Nathan Vegdahl, Lucio Rossi, Ivan Cappiello, Alexander Gavrilov
License::	GPL
Note::	This add-on is bundled with Blender.

Introduction

Rigify helps automate the creation of character rigs. It is based around a building-block approach, where you build complete rigs out of smaller rig parts (e.g. arms, legs, spines, fingers…). The rig parts are currently few in number, but as more rig parts are added to Rigify it should become more and more capable of rigging a large variety of characters and creatures.

Rigify also operates on the principle that once a rig is created, that rig should no longer need Rigify. This means you can always distribute rigs created with Rigify to people who do not have it and the rigs will still function completely.

It is important to note that Rigify only automates the creation of the rig controls and bones. It does not attach the rig to a mesh, so you still have to do skinning etc. yourself.

Main Features

Modular rigging
: Rigify build blocks can be mixed together to rig any character you want. If you need to build a character with five arms and one leg, Rigify can handle it for you creating all the required complex controls system (FK, IK, and all the relative snapping tools and the UI) in few seconds.

Nondisruptive re-rig
: If the generated rig doesn't fit all the features you need or, for example, you decide to add something more to your character (like a sixth arm or a tail), you can re-generate your rig without losing your previously generated features and your animation data.

Advanced and flexible feature set for character animation
: The included rig samples (limbs, spines, tails, fingers, faces…) adds to all the stretchy FK/IK features a direct deformation secondary layer that

lets you flex, bend and deform the character as you like through interactive Bendy Bones controls.

Shareable animation through all Rigify rigs
: Since the control system is generated by Rigify, if you share a meta-rig through different characters you will be able to share data between them even if they have different proportions.

Extendable feature set
: You can save and encode your meta-rigs to a button to have them available at any time without recreating it by hand or share your meta-rigs with other people. Through Python scripting you can also extend Rigify with new Rigify-types or new rig samples by implementing your own feature set package.

Ready to go
: Once you generate your rig you won't need Rigify or any other add-on to use it.

Enabling Add-on

1. Open Blender and go to Add-ons section of the Preferences.
2. Search "Rigify" and check the *Enable Extension* checkbox.

Basic Usage

Basic Rig Generation

1. Add a meta-rig structure from the *Add ▸ Armature* menu.
2. Edit the bone positions to match the character geometry.
3. In the armature properties click on the *Generate Rig* button to generate the rig.

Add a Predefined Meta-Rig

Reference

Mode::	Object Mode
Menu::	*Add ▸ Armature*
Shortcut::	`Shift-A`

Rigify stores all the information required to generate complex rig controls and mechanism in more simple armatures called “meta-rigs”.

The predefined meta-rigs can be found in the *Add* menu. Currently available meta-rig types are:

- Basic Human (doesn’t include face and fingers)
- Basic Quadruped
- Human
- Cat
- Wolf
- Horse
- Shark

Edit Bone Positions

To correctly match your character, meta-rig bones must be moved to correct positions. This can be achieved in two different ways: Pose Mode or Edit Mode.

Note

Rigify assumes that 1 unit corresponds to 1 meter. So a human is about 2 units tall. If your character is in a different scale and you are more familiar with modeling rather than rigging, it is suggested to scale it to Rigify dimensions before positioning the meta-rig bones. If you want to scale the character's geometry, we suggest you to first scale up the character in Object Mode, then apply the geometry scale with the *Apply Scale* tool.

Rigify Human Alignment Tips

- Limbs: Keep the legs as straight as possible in the front view (Rigify human works better in predictable cases). Give the knee and the elbow a slight bend angle (Rigify needs to know where your knee/elbow is pointing).

- Torso: Keep the spine as straight as possible in the front view (Rigify human works better in predictable cases). The last bone of the spine is the head. By default the next two bones (top to bottom) are considered the neck bones. It is suggested to keep the neck bones as aligned as possible while editing.

- Face: Positioning face bones can be tricky if you are not an expert in bone editing and they are almost useless if you plan to make facial animation through shape keys. Consider removing face features from your character if they aren't really needed. If you don't need the face all the face bones can be deleted. All the face bones are in the *Face* armature bone collection by default. You can select them by displaying only that collection, selecting all of its content and then deleting the bones in Edit Mode to correctly remove the face.

If you want to scale all the face bones at once, consider scaling the face master bone in Pose Mode (see Pose Mode matching method). The face master bone is placed in the same position of the head bone. To select it easily, hide all other bone collections.

For more tips, see the Positioning Guide.

Pose Mode Matching (Basic)

Enter the meta-rig Pose Mode. Rotate, scale, and translate the bones in the correct position. When bones are in correct positions (always staying in Pose Mode) use *Apply ▸ Apply Pose As Rest Pose*.

Note

Connected bones cannot be translated in Pose Mode. You can scale the parent bones to match the general length and then refine child bones scale. For more detailed information on armature modes please refer to the armatures section.

Edit Mode Matching (Advanced)

Some basic armature display setup is suggested before entering bone Edit Mode.

With the meta-rig selected, go in the Properties and click on the Object tab. Scroll down to the display panel and enable X-ray and under *Maximum Draw Type* selector select *Wire*. This way the bones will always be drawn in wireframe on top of your geometry.

Then, always in the Properties click on the Armatures tab and under display check the *Axis* checkbox. This way you the bones rotation axes will be displayed during the edit process.

For more detailed information on armature display modes please refer to the Display panel page.

Generating the Rig

With the bones in the correct positions, jump back in Object Mode, go to the Armature tab, scroll down to the bottom and click on the *Generate Rig* button to finalize the rig creation. The generation process will take from few seconds to one minute depending on rig complexity and hardware specifications of your machine. If the generated rig needs tweaking, you can modify the meta-rig accordingly and then click again on the generate button. If the rig already exists, Rigify will simply overwrite it retaining all your modifiers and constraints and – where possible – all the previously generated features.

For information about additional generation options, see the Advanced Rig Generation section.

Tip

If the metarig uses the legacy face rig, you can use the *Upgrade Face Rig* button that appears above *Generate Rig* to automatically upgrade to the new modular face system.

The upgrade will preserve compatibility with existing skinning, but existing poses and animations will likely not be compatible due to subtle changes in control behavior.

Note

To make the rig overwriting work as expected, you need to have **both** the rig and the meta-rig visible before generating again. Rigify will try to unhide them in simple cases, but will abort generation if that fails.

Warning

As with all Python add-ons, Blender interface cannot be updated until the Python script execution is over. Wait until the rig appears to see the results.

Warning

Rigify is designed assuming a workflow where the meta-rig is kept available to allow re-generating the main rig whenever it is necessary to make changes to it. Removing the meta-rig after generating the main rig, or significantly modifying the generated rig is not advised: it will make it impossible to introduce features added in later versions of Rigify, or easily adapt it to breaking changes in later Blender versions. In general, automatic version update scripts will be provided for meta-rigs when necessary, but not generated rigs.

Binding the Geometry to the Rig

To bind the geometry to the rig you can use your preferred tools. Just few things you have to know:

- All the deforming bones are in the *DEF* bone collection.
- Eyes and Teeth bones of the legacy face are not deforming. You are supposed to bind the eyes and teeth geometry through Child Of constraints.
- Usually armature deform with automatic weights do a really good job out of the box if you correctly place your bones (and there is enough topology to work with!).

For more detailed information on bone collections, Armature modifier and weight painting refer to the Blender manual.

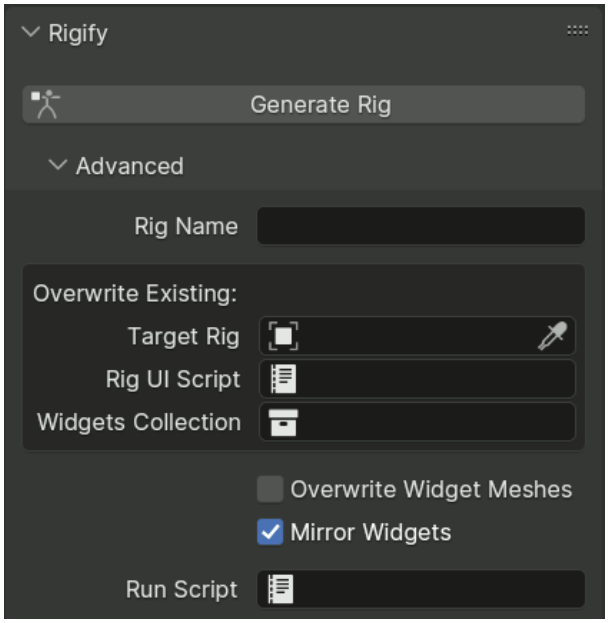
Advanced Rig Generation

Advanced Options Features

By using options in the Advanced sub-panel, it is possible to:

- Generate more than one rig per scene.
- Update/Override a specific rig.
- Force previously generated widget objects to be overwritten.
- Choose whether to use linked duplicates for left and right side widgets.
- Execute a script data-block after generation.

Advanced Options Sub-Panel



Advanced rig generation options are by default hidden in a sub-panel. Click on the *Advanced* line to open it.

Some of the options will be automatically set by Rigify if they have no value when a rig is generated, while others are fully controlled by the user.

Rig Name

When a brand new rig is generated, as opposed to overwriting an existing one, the value of this option is used to name it.

If this field is empty, the new object will be named based on the name of the metarig according to the following rules:

- If the name contains `META`, it is replaced with `RIG`.
- If the name contains `metarig`, it is replaced with `rig`.
- Otherwise, `RIG-` is prepended to the name.

When overwriting an existing rig object specified by the *Target Rig* option, its name is not changed, allowing it to be freely renamed without having to keep the value of this option in sync.

Target Rig *auto*

This option specifies the generated rig to overwrite when re-generating from this metarig.

If the option is not set, Rigify will generate a new rig object and store it in this option.

Note

When the option isn't set, Rigify will create a brand new rig object even if an object with a matching name already exists.

Rig UI Script *auto*

This option specifies the generated script datablock to overwrite when re-generating, and works in the same manner as *Target Rig*.

The script controls the UI in the 3D Viewport that allows conveniently switching visible bone collections, changing custom properties, converting between IK and FK and so on.

Widgets Collection *auto*

This reference specifies the collection containing generated widgets, and works in the same manner as *Target Rig*.

Overwrite Widget Meshes

If enabled, Rigify will generate new widgets every time the rig is re-generated. By default, it tries to reuse the already generated widget objects that exist in the widget collection, allowing them to be manually edited to fit the character better.

Mirror Widgets

When enabled, Rigify generates widgets for left and right side bones as linked duplicates, using negative X scale to flip the right side version. This enforces symmetry and reduces the number of meshes to adjust to fit the character.

When reusing an already generated widget, Rigify detects if it was originally generated mirrored by checking object scale to avoid flipping existing controls. Therefore switching to mirrored widgets for an existing character requires deleting the right side widgets, or *Force Widget Update*.

Run Script

It is possible to configure Rigify to execute a Python script contained in a text data-block after generation in order to apply user-defined customizations. The script is executed with the generated rig active and selected in Object Mode.

The simplest use of this may be adjusting properties of generated constraints when Rigify rig types don't have any relevant meta-rig settings. That can be done by using the *Copy Full Data Path* context menu option on the property, pasting it into the script and making an assignment, e.g.:

```
import bpy

bpy.data.objects["rig"].pose.bones["MCH-
spine.003"].constraints[0].influence = 0.6
```

Doing such changes via a script ensures they aren't lost if the rig is re-generated.

Users familiar with Rigify scripting can import Rigify utility modules, and access the generator instance through `rigify.get_generator()`. Yet note that, since generation is already finished, the only use of that is reading data created in the generation process.

Library Linking

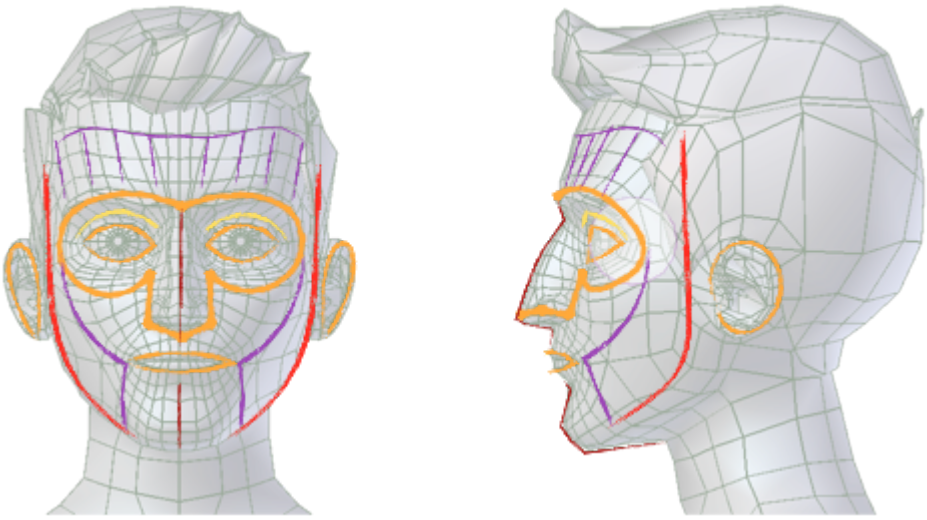
When linking a rig into another file, you generally want to create a collection that includes the generated rig and the character mesh. You do not need to include the meta-rig or the widget object collection. You then link in the collection and run Make Library Override.

The `rig_ui_template.py` text data-block responsible for the rig UI will be automatically linked along with the rig, you don't need to link it separately. However, the script will not run until you run it manually from the Text editor or save and restart Blender.

Bone Positioning Guide

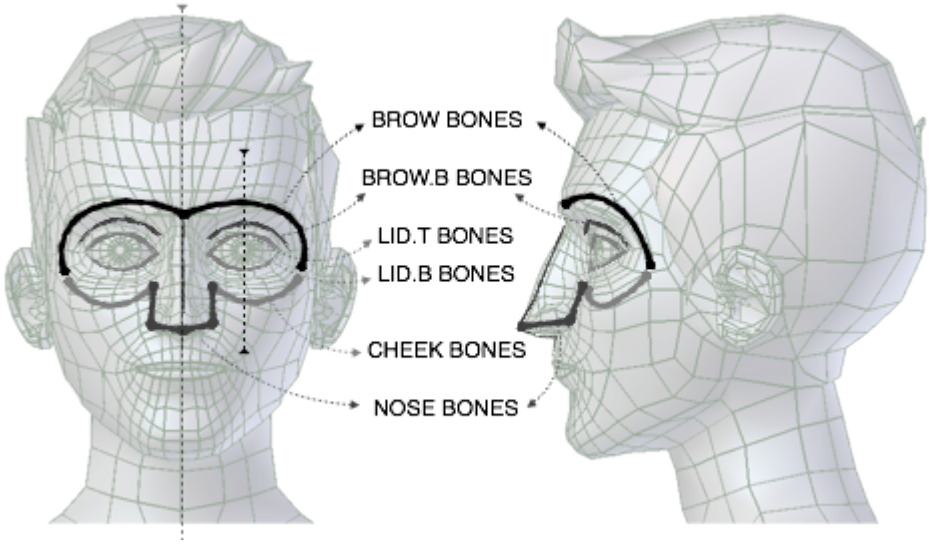
Face Bones

Start by identifying basic face landmarks to follow as guide for bones placement.



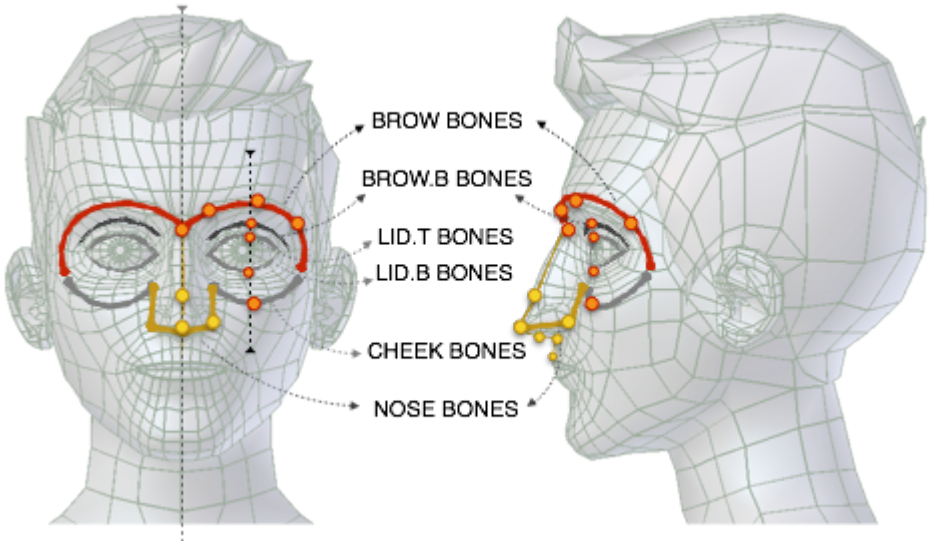
Basic Face Landmarks.

- Orange lines represent bones that should be placed in closed loops.
- Yellow lines represent bones whose position depends on surrounding bone loops.
- Red lines represent outer edge bones.
- Purple lines represent bridging bones used to cover deforming flesh.



Eyes-Nose Landmarks.

The eyes-nose loop area is split in different parts identified by bone names. Follow the image to place the bones.

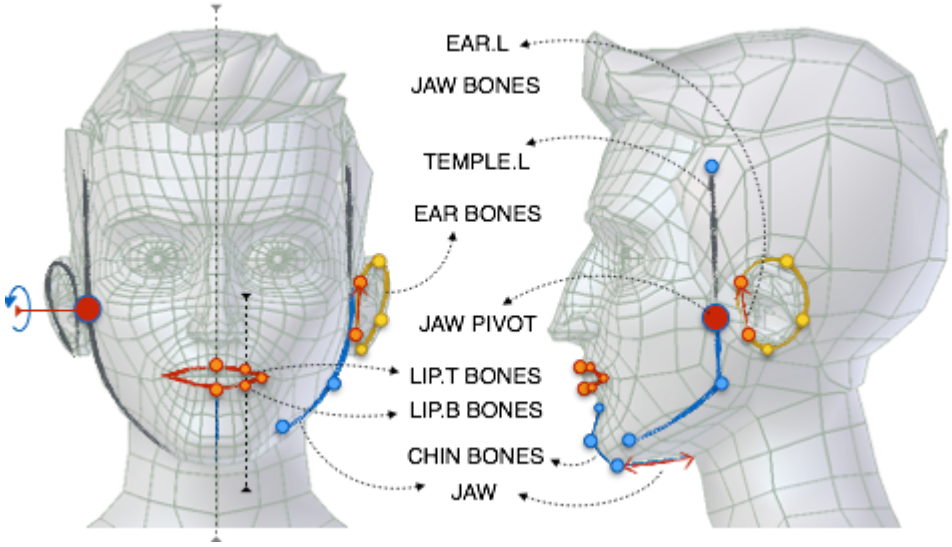


Eyes-Nose Bone Positions.

Tip

Brow Placement

Keeping aligned the mid bones in “brow”, “brow.b”, “lid.t”, “lid.t” and cheek will give better results after rig generation.



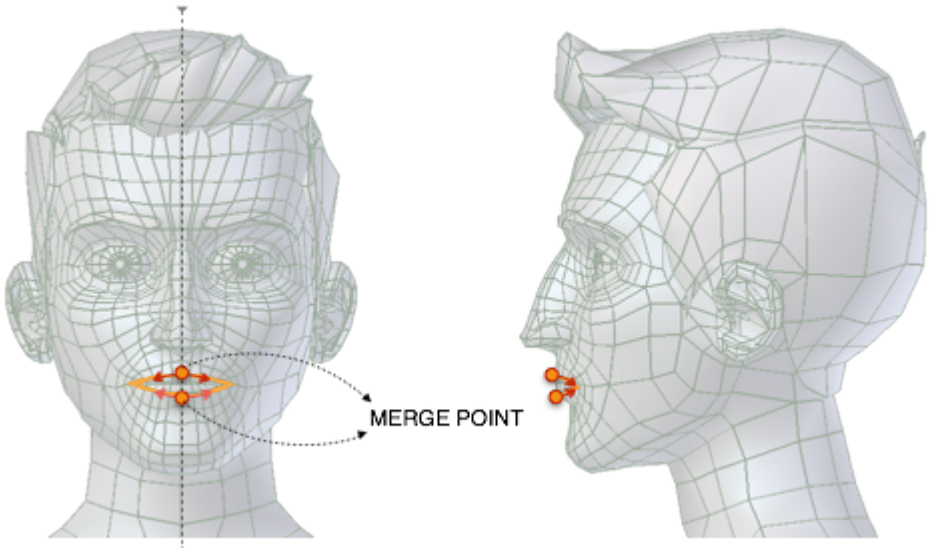
Jaw-Ear Bone Positions.

Also the jaw-ear area is split in different parts identified by bone names. Follow the image to place the bones.

Tip

Jaw Placement

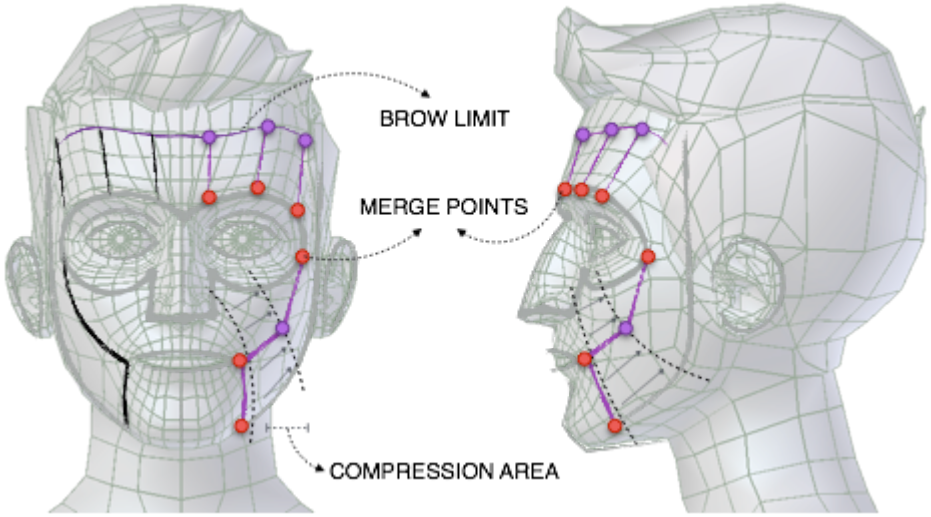
Try to place “ear.L” bone covering the part of the ear attached to the mandible (lower jaw). Do the same with temple bone trying to cover the part you don’t want to move with the jaw, this way you will also determine the jaw pivot position.



Lips Merge Point.

Warning

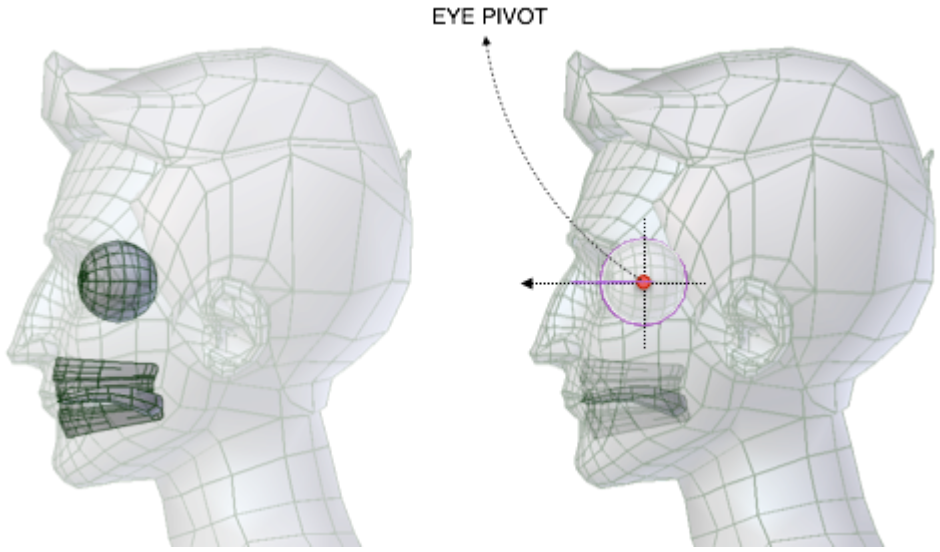
While moving the face bones it is necessary to preserve merge points, i.e. whenever heads or tails of two or more bones overlap at the same point, they should still do so after repositioning. Tearing a merge point apart may result in multiple controls being created instead of one, or even the generation of errors.



Face Stretcher Bones.

After the main face bones are placed use the cheek bone to connect the eye-nose area to the jaw mouth area. Then do the same with the brow area. This process will automatically define face muscles compression areas.

Position the eye bones in the eye pivot point facing right **toward** the face on the Y axis. The length of the eye bones should correspond to the radius of the eye.



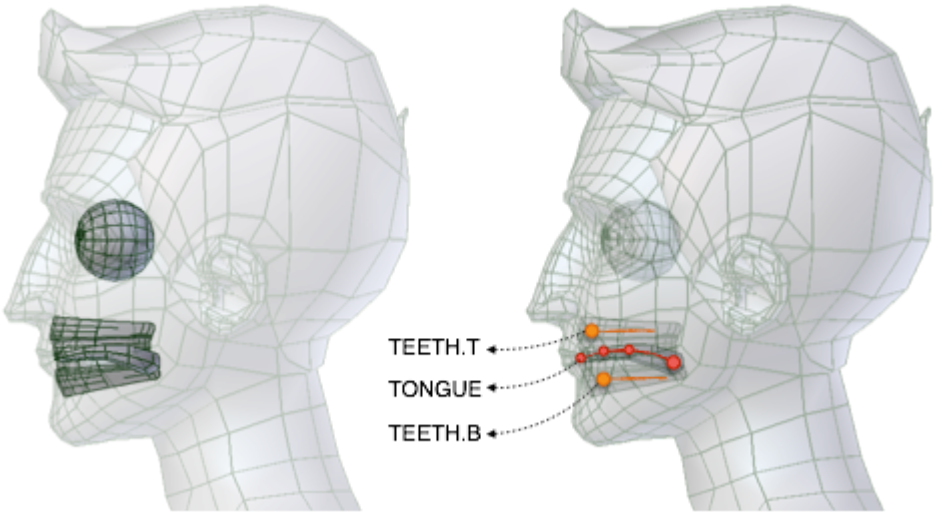
Eyes Pivot Position.

Tip

Eye Pivot

If your eye has a spherical shape you can define its pivot by entering Edit Mode. Select two opposite vertices on the center meridian – or the opposite poles – and snapping the cursor to selection by pressing *Snap ▸ Cursor To Selected*. If your eye is a complete sphere and its location it's not applied, then you can just use its center of mass.

Finally position the teeth bones on your teeth geometry and the tongue bone chain as described in the figure.



Mouth and Teeth Positions.

Tip

Tongue

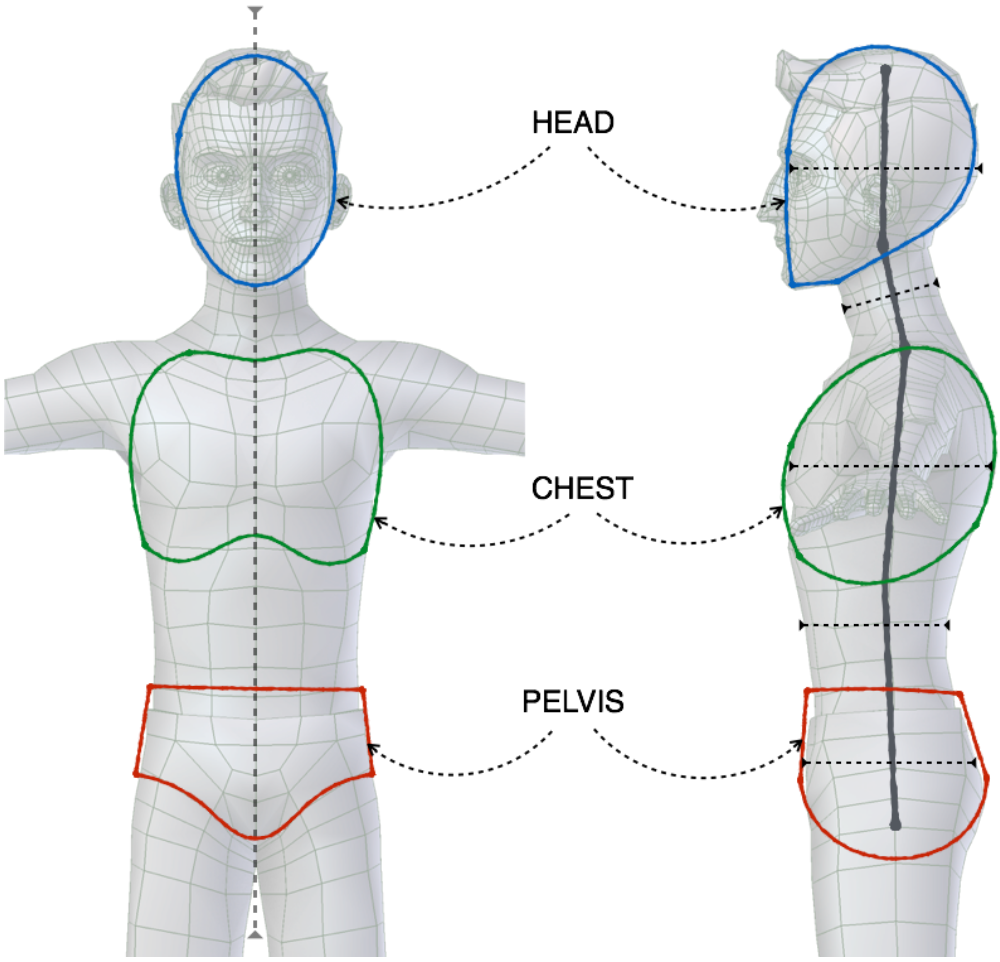
The tongue will work better if the bones are aligned at the symmetry line.

Before generating the rig ensure the face master bone is facing upward.

Torso Bones

Start by identifying on your character basic torso zones to follow as guide for bones placement.

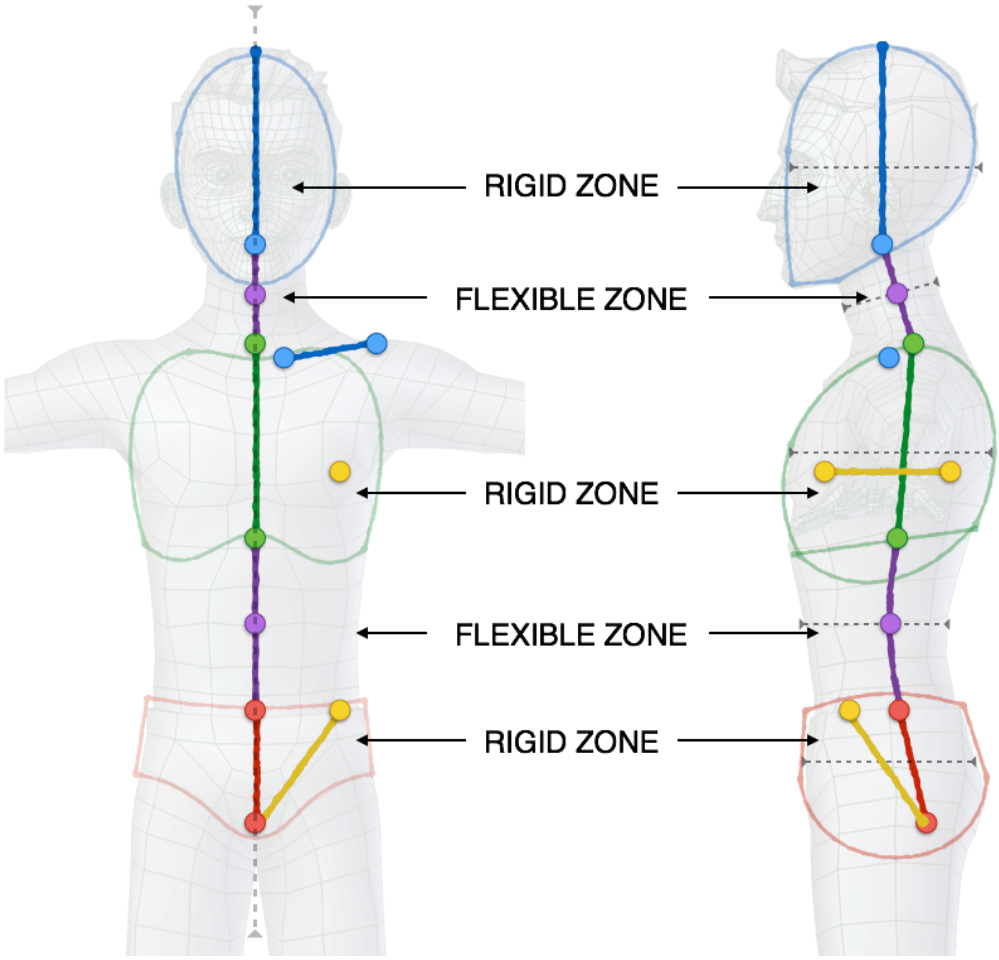
Head, chest and pelvis are rigid zones, so they require less bones. Having a good edge loop placement around zone boundaries on your model will help in having correct deformation after armature binding.



Torso Landmarks.

Starting from the side view, place the main spine bones trying to use one bone for the rigid areas and two for the flexible ones. In addition to the main spine, the torso is provided with additional pelvis bones (to oppose the leg bending), two breast controls and two shoulder bones.

Even if the pelvis bones will not appear in the final rig as controls, they will contribute to deformation.



Torso Bones Positioning.

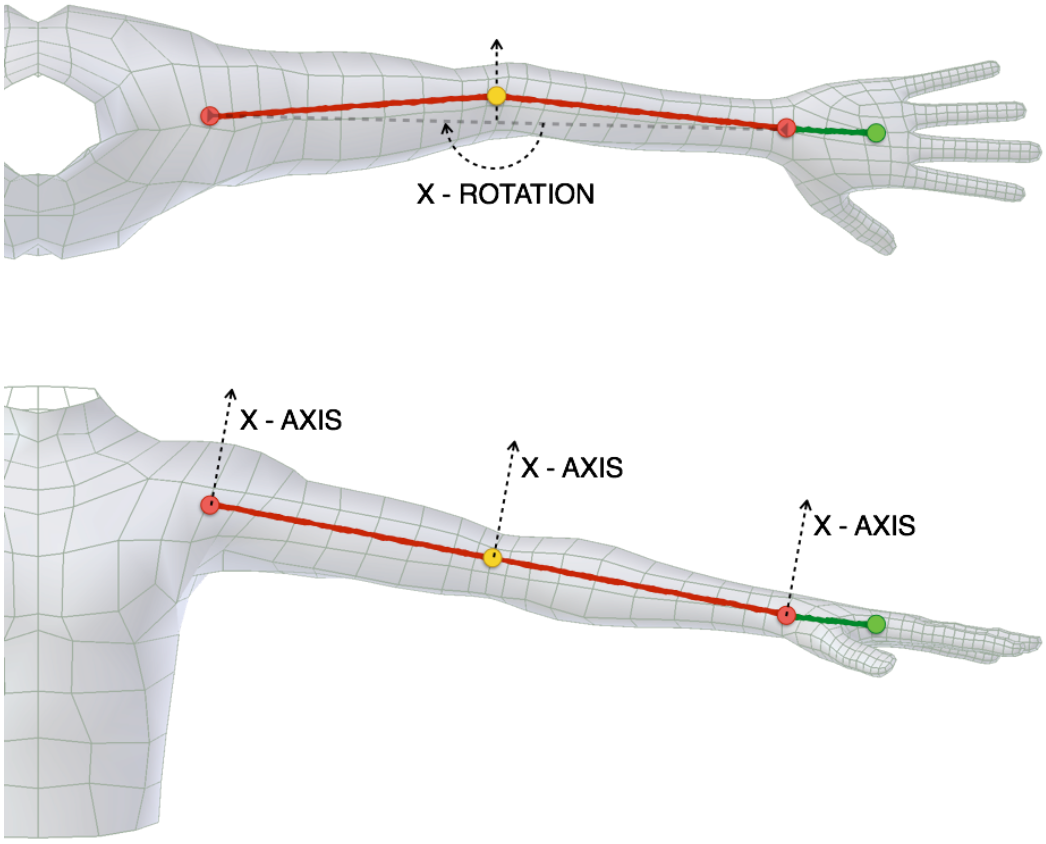
Tip

Bone Placement

Try to keep the spine as centered as possible inside the mesh bounding volume, just apply a slight offset toward the back. In a similar way, consider the shoulder bones as general deformers; placing it too forward – where the collar bone should be – could cause undesired deformations.

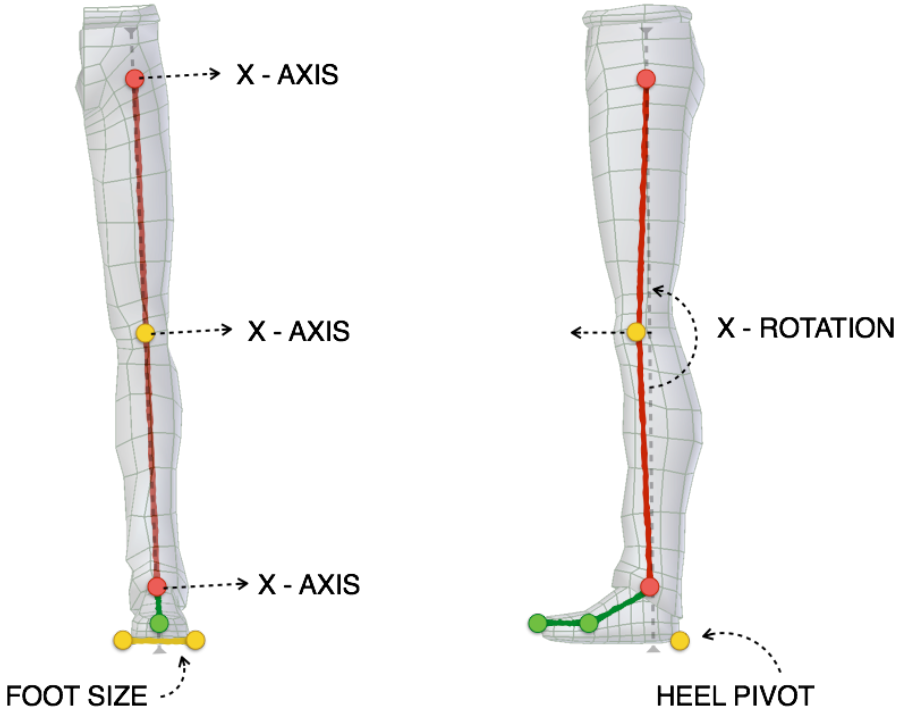
Limbs Bones

While placing the arm bones try to start having a straight line that goes from the shoulder to the hand in both front and top view. After this is done just add a slight bend to the elbow. This can be easily done by going in the top view, entering armature Edit Mode and sliding the bone junction between forearm and upper_arm slightly toward the world's Y axis.



Arm Bones Positioning.

For the leg you can follow a similar process. Start by aligning the leg bones creating a straight line from the hips to the ankle, then place the foot and the toe accordingly. Remember to add a slight bend to the knee. This can be easily done by going in the side view, entering armature Edit Mode and sliding the bone junction between thigh and shin slightly toward the world's Y axis.



Leg Bones Positioning.

Finally align the heel bone by going in the front view and placing the head and tail to fill the foot size from side to side. Then, in the side view, align the bone at the point where the heel just touches the ground floor.

Note

From version 0.5 and above there is no more need of manual bone rolls alignment. The generate function will take care of that for you by evaluating it from bend axis; just insert a slight bend in your limb and it's done! If you need more control on the orientation, follow the guidelines described in Advanced Usage.

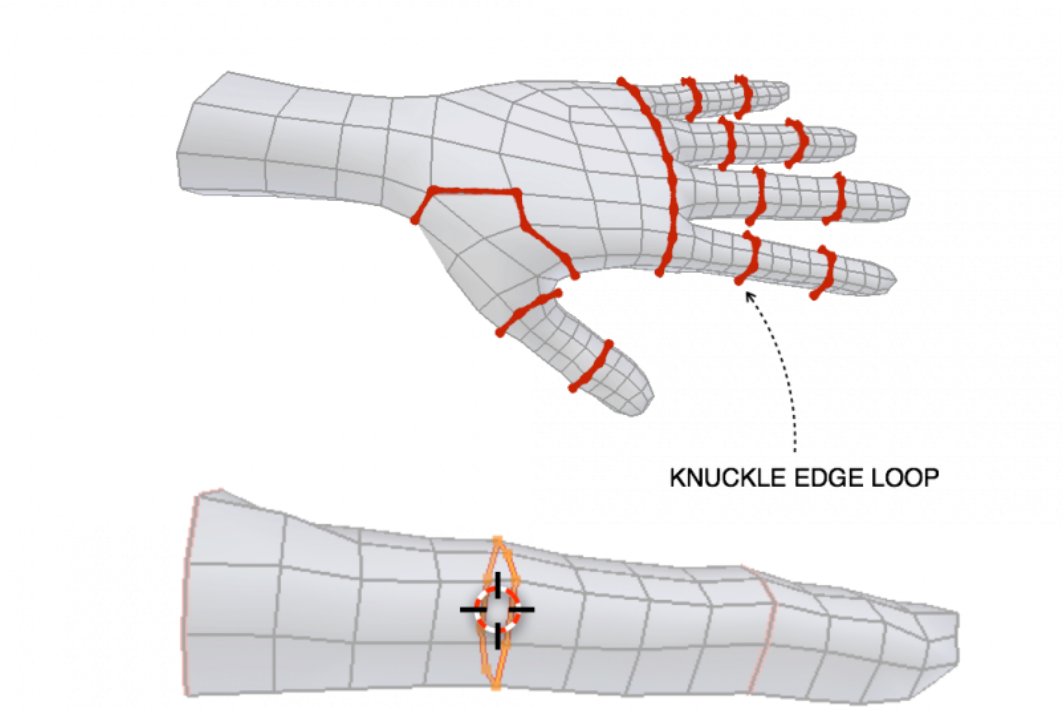
Fingers Bones

Start by placing, finger by finger, all the knuckles in place.

Tip

Fingers Placement

An easy and effective method to do this operation is to select on the mesh the corresponding edge loop in Edit Mode and use the *Cursor to Selection* snap. Then you can snap the bone to the corresponding loop using the *Selection to Cursor* snap.



Knuckles Edge Loops and Cursor Snapping.

Finalize the positioning by taking care of bone rolls (the X axis is set as bend axis).

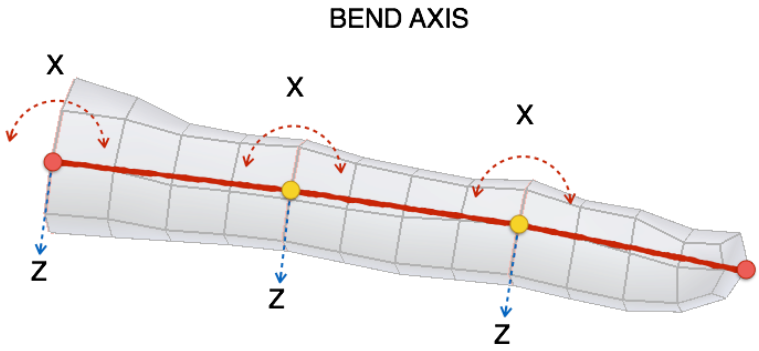
Tip

Bone Roll

Finger axis alignment can be easily be made consistent by selecting all the finger bones and recalculating the bone rolls *Recalculate Roll ▸ Global -Z Axis*.

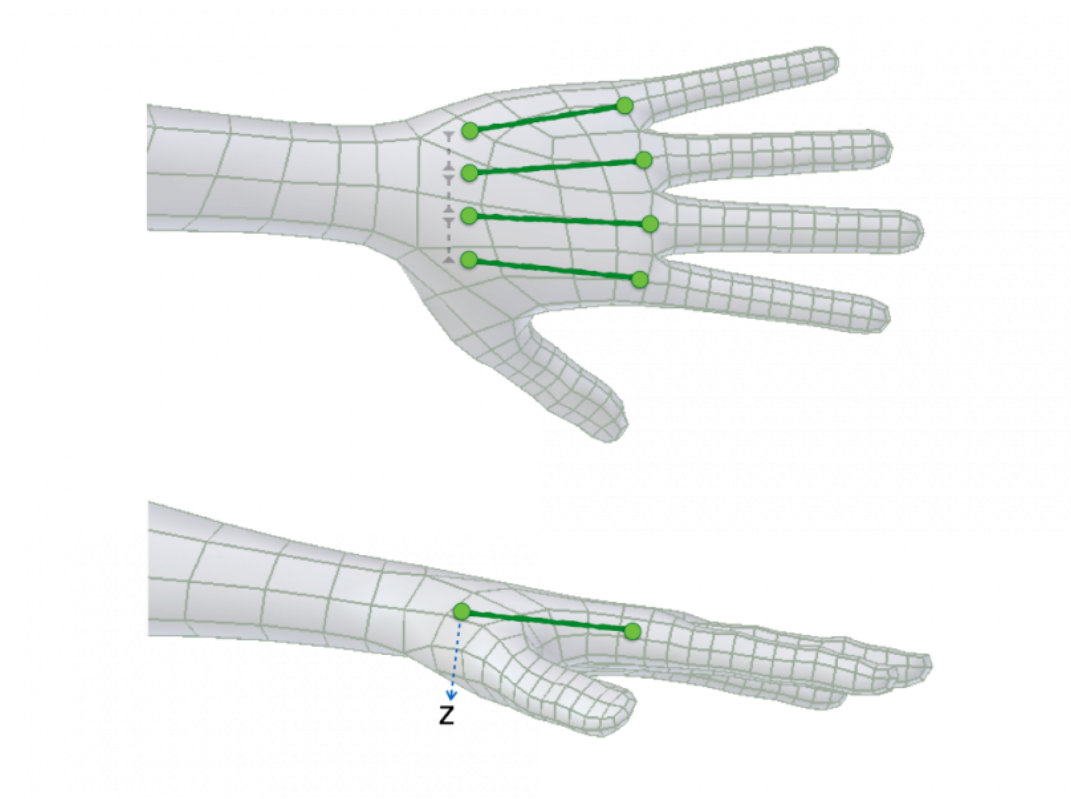
Thumb may require more tweaking depending on your character's mesh topology, usually *Recalculate Roll ▸ Global +Y Axis* is a good starting point.

Once your bone rolls are consistent, try generating the rig and scaling the finger master controls. This should cause the fingers to curl. If they are rotating on the wrong axis, change the Bend Rotation Axis parameter on the first finger's parameters under Rigify Type.



Fingers Bend Axis.

When the fingers are in place proceed placing the palm bones.



Palm Alignment.

Tip

Palm Placement

Try to keep palm bones' heads at a little distance between each other. This distance is required for Rigify to define the palm controls hierarchy. Palm axis alignment can be easily done by selecting all the palm bones and recalculating the bone rolls *Recalculate Roll ▸ Global -Z Axis*.

See also

For more detailed information on bones and rolls refer to the Bone Structure and Bone Roll.

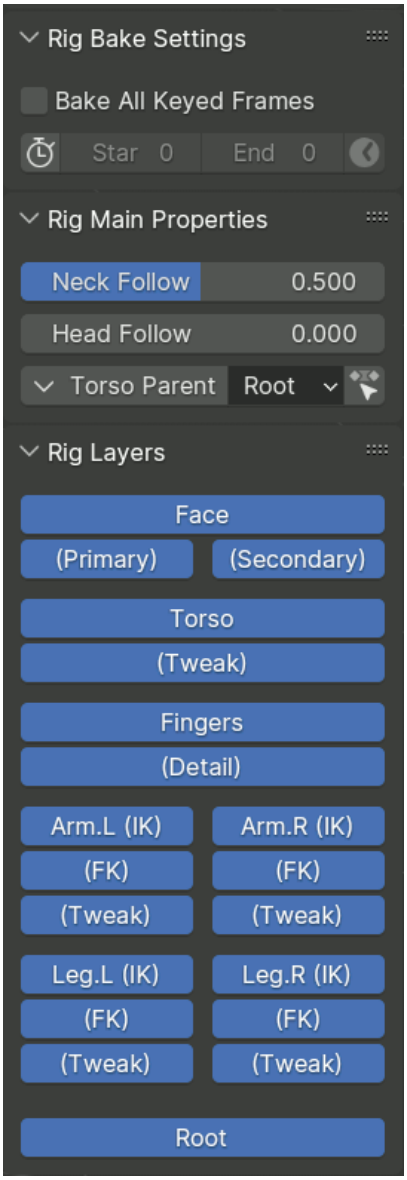
Generated Rig Features

After human rig generation a new armature named `rig` will be added to your scene. This is the character rig you have generated from the human meta-rig and will contain all the features.

Common Features

Rig UI Panels

The generated rig is accompanied by a script that implements a set of panels that appear in the Item tab of the 3D view sidebar when a bone belonging to the generated rig is active.



Rig Bake Settings

This panel is displayed if the armature has an active Action, and is used by operators that apply an operation to multiple keyframes.

Bake All Keyed Frames

When enabled, the operator computes and keyframes its result on every frame that has a key for any of the bones, as opposed to just relevant ones.

Limit Frame Range
: When enabled, the operator is limited to a certain frame range.

Start, End
: Specify the frame range to process.

Get Frame Range
: Sets the baking frame range from the scene frame range.

Rig Main Properties

This panel shows properties and operators that are relevant to the selected bones.

Rig Layers

This panel contains buttons for toggling visibility of bone collections.

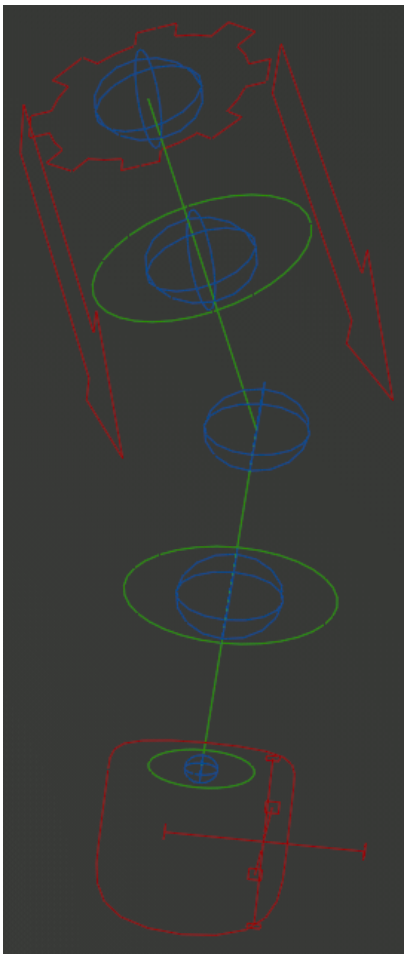
The layout and labels of the buttons are defined in the metarig Bone Collection UI panel.

Common Controls

Rigify rigs are built from standardized components called sub-rigs, which are linked together in a parent-child hierarchy. Although the precise behavior of each sub-rig is determined by its implementation, there are certain conventions that are followed by many of them.

Root Bone

Every Rigify rig has a bone called *root*, which serves as a parent for all bones of the rig. It is assigned to a bone collection called *Root*. Unless the metarig has a custom bone of that name, it is positioned at the origin of the rig object. Its widget looks like a circle with four arrow shaped protrusions.



Limb Master

Many limb-like sub-rigs have a gear-shaped bone at their base.

This bone can in some cases be used to transform the whole sub-rig as a rigid unit, and is also used as a container for its custom properties that are displayed in the *Rig Main Properties* panel. If you are looking in the Graph editor for the animated values of the properties, this is most likely the bone to look at.

As an exception, if multiple controls of the sub-rig need their own copy of conceptually the same property, it may be placed on those controls directly instead.

Tweak Controls

These controls look like blue spheres in the default color scheme, and are the final control layer above the deformation bones themselves.

Tweaks are subordinate to the general IK or FK limb position but can be moved apart, twisted and scaled freely, even reaching virtually impossible limb shapes.

Rubber Tweak



Some sub-rigs provide a slider in their *Rig Main Properties* when tweaks are selected, which controls the smoothness of the Bendy Bone joint at that position. When zero, the joint deforms with a sharp bend, while setting it to 1 makes the transition smooth for a more rubber hose cartoon like appearance.

Custom Pivots

Some bones that can be freely moved in space (like IK controls) can be optionally accompanied by a custom pivot control. These controls usually look like a plain axes empty with the axis lines capped with squares or crosses, like the one in the image above. The control can be freely moved to change the location of the pivot, and then rotated or scaled to transform the target bone around the pivot.

IK and FK Switching



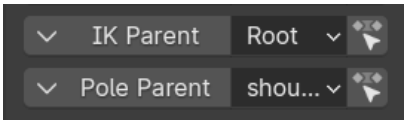
A number of rig types provides both IK and FK controls (red for IK and green for FK in the image above), with an ability to switch and snap between them.

Switching is controlled by a slider in *Rig Main Properties*, usually blending between full IK at 0 and full FK at 1.

Snapping one type of controls to the shape of the other is done via buttons, which form a group of three in their complete set:

- The main button will snap on the current frame, and auto-key the result if enabled.
- The *Action* button will bake the change on multiple keyframes, according to *Rig Bake Settings*.
- The *Clear* button will delete keyframes on the corresponding controls within the bake interval.

Parent Switching



Some freely movable controls, e.g. usually the IK controls, can have a mechanism to switch their parent bone between a set of choices, including the root bone, or none at all.

This mechanism is exposed in the *Rig Main Properties* panel through a row with three controls:

- A button that presents a dropdown menu, which allows switching the parent on the current frame while preserving the bone position and orientation in the world space.
- A dropdown input field that directly exposes the switch property for keyframing and direct manipulation. Changing the value can cause the bone position to jump.
- A button to apply the position preserving parent switch over the bake range of keyframes.

Note

When manually placing a Child Of constraint on the control bone, the built-in parent should be switched to none.

Limbs

Limbs have a master bone and tweaks. Depending on the user defined meta-rig options, multiple deform bone segments with tweaks will be created.

The IK control may have an optional custom pivot, as well as additional predefined pivots.

Rigify's limbs have the following controls in the Sidebar panel:



FK Limb Follow *Slider*

When set to 1 the FK limb will not rotate with the torso and will retain is rotation relative to the root bone instead.

IK-FK *Slider*

Controls whether the limb follows IK or FK controls, blending between full IK at 0 and full FK at 1.

IK<->FK Snapping *Buttons*

Snaps one type of controls to another.

IK Stretch *Slider*

Blends between the limb stretching freely at 1, or having its maximum length constrained at 0.

Toggle Pole *Switch*

When the toggle is Off, the IK limb will use the rotational pole vector (the arrow at the base of the limb). Rotating/translating/scaling the arrow will control the IK limb base.

When the toggle is On, the classic pole vector will be displayed and used to orient the IK limb. The arrow will continue to handle the scale and the location of the IK limb base.

Similar to *Parent Switching*, the row includes buttons to convert the current pose between types, or bake the whole action.

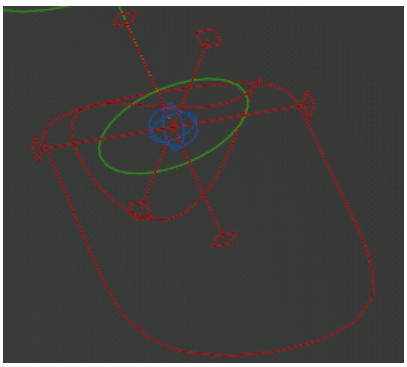
IK Parent *Switch*

Switches the effective parent of the main IK control.

Pole Parent *Switch*

Switches the effective parent of the classic IK Pole control.

Arms

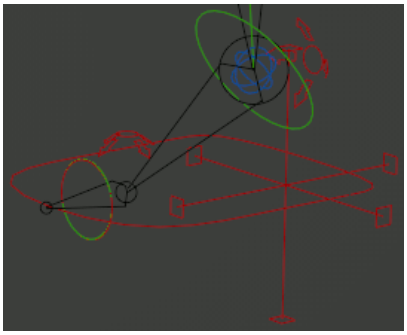


Arms have the simplest control structure: the IK controls consist of the main IK control, the optional custom pivot control, and the optional wrist control (the bent circle), which pivots around the tail rather than the head of the hand bone.

There are no additional controls in the *Rig Main Properties* panel.

Legs

Legs have a more complicated setup, which has:



IK & FK Toe *Optional*
: Two separate IK and FK controls for the toe (this is on by default in the bundled metarigs, and is recommended for stable IK<->FK snapping).

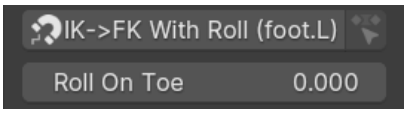
IK Heel
: A heel control which can be rotated to command forward or backward roll, sideways rock, or yaw of the heel.

Toe Pivot *Optional*
: An extra pivot control rotating around the base of the toe.

Custom Pivot *Optional*
: A custom pivot control.

The properties panel has two additional features:

IK->FK Snap With Roll *Buttons*



Standard IK to FK snapping resets the transformations of all IK controls other than the main one. This is not convenient to use in an animation that involves the use of the heel control, because roll and rock would be folded into the transformation of the main control.

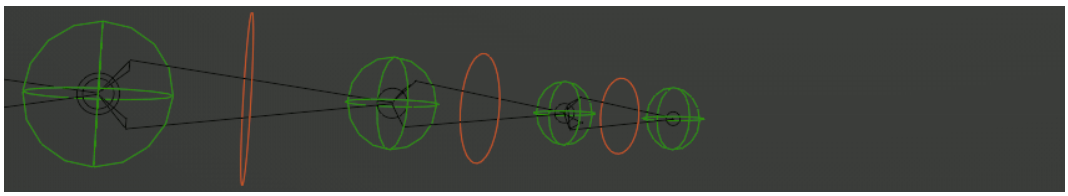
This alternative snapping operator tries to deduce the rotation of the heel control so as to keep the main IK control parallel to the ground plane inferred from the *current* orientation of the IK control. The operator has options to specify which rotational axes to use for the heel control rotation.

Roll On Toe *Slider Optional*

If enabled in the sub-rig settings, this slider can be used to control whether the heel rotation (excluding backward roll) is applied at the base or the tip of the toe.

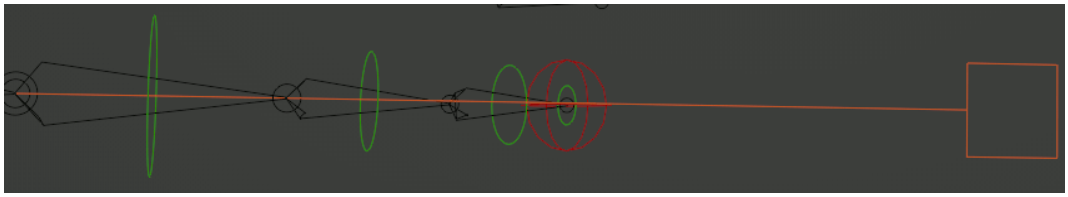
Fingers & Tentacles

Simple Tentacle



The simplest type of rig for a finger or appendage in general is the simple tentacle sub-rig. It has only basic FK controls and tweaks, with the only automation being the ability to copy certain axes of the local rotation of a FK control to the next one.

Advanced Finger



For fingers specifically, Rigify has a dedicated finger sub-rig type, which provides:

Master
: A master control (orange), which can be used to rotate the finger as a whole, as well as to bend it via Y scaling.

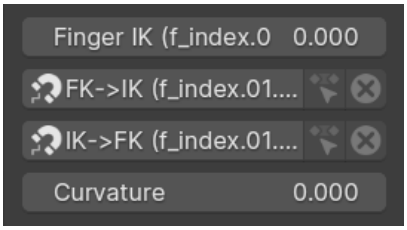
FK Chain
: FK control chain (green) that can also operate as semi-tweaks through allowing translation.

IK Control *Optional*
: IK control for the tip (red).

Note

IK in this sub-rig is rudimentary and operates as an adjustment for FK. The intended way of use is to pose the finger in FK, and then enable IK after using IK->FK snap if it is necessary to pin the tip of the finger in place.

The properties panel has the following features:



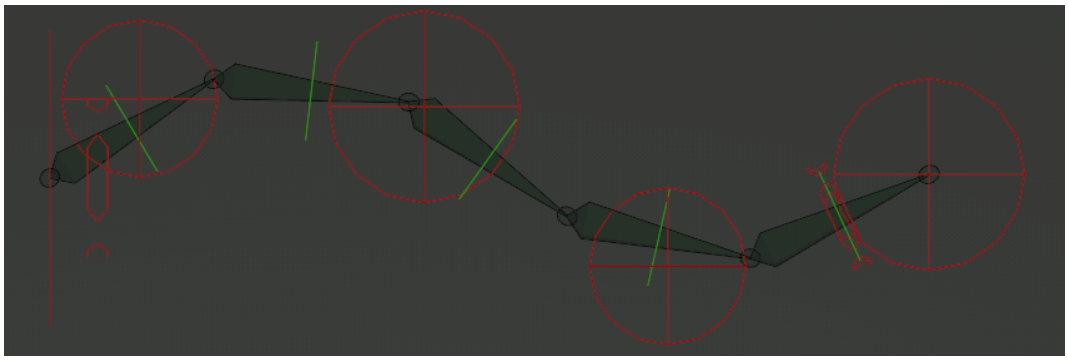
Finger IK *Slider Optional*
: Slider controlling the influence of the IK.

FK<->IK Snapping *Buttons Optional*
: Snaps the IK control to the end of the finger, or adjusts the FK controls to the result of the IK correction.

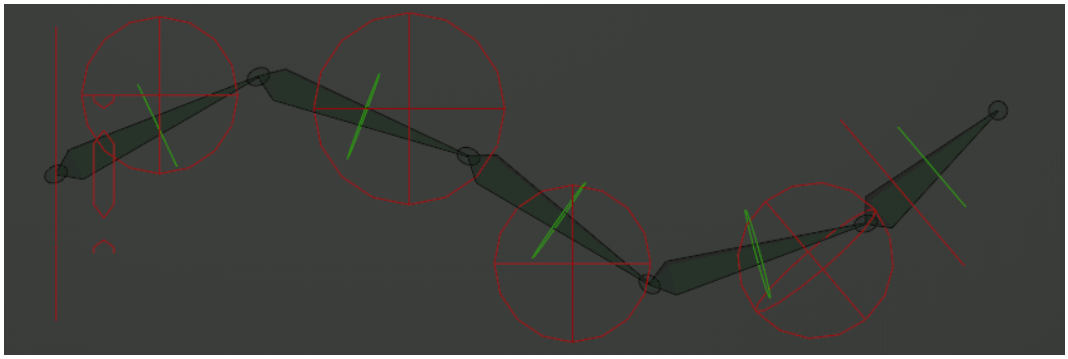
Curvature *Slider*

Has the same effect as *Rubber Tweak* on limbs, controlling the rubber hose cartoon effect.

Spline Tentacle



Spline Tentacle (Stretch To Fit, Manual Squash & Stretch)



Spline Tentacle (Direct Tip Control)

The spline tentacle is an advanced rig for a flexible appendage (tentacle) based on the Spline IK constraint. The IK control bones manage control points of a Bezier spline curve, which in turn is followed by the IK chain.

The tentacle can be generated in three major modes:

Stretch To Fit
: In this simplest mode all bones of the sub-rig deform chain follow the curve and squash & stretch to match its length.

Manual Squash & Stretch

This mode is almost the same, but the chain does not automatically scale to match the curve length. Instead, it tries to cover as much as possible of the curve given its manually scaled length. If the curve is too short, the chain will overhang it and straighten out, but this can result in jitter.

Direct Tip Control

This mode is more similar to the behavior of IK limbs: the final bone of the chain is directly controlled by the tip IK control, while the other bones of the chain stretch and follow the curve to bridge the gap.

The tentacle sub-rig has the following control bones:

Master

The tentacle has the same gear master control as other limbs (seen as a line in the images).

IK Start

The IK control at the base of the tentacle, which can be used to control the base twist and sideways scale, and is one of the potential switchable parents for other IK controls.

In the *Manual Squash & Stretch* mode it controls uniform scale of the tentacle in all directions.

IK Start (Extra) *Optional*

Extra start controls, optional and hidden by default. Switchable parents default to the *IK Start* control. The scale of the control may optionally affect the thickness of the chain via the radius of the curve point.

IK Middle

Controls for the middle of the curve. The switchable parents default to *Master*, but may be set to *IK Start* or *IK End* controls. The scale of the control may optionally affect the thickness of the chain via the radius of the curve point.

IK End (Extra) *Optional*

Extra end controls, optional and hidden by default. Switchable parents default to the *IK End* control. The scale of the control may optionally affect the thickness of the chain via the radius of the curve point.

The *Direct Tip Control* mode adds one more extra end control next to the middle ones that cannot be hidden.

IK End
: Controls the last control point of the curve, and is one of the potential parents for the other chain controls.

  In the *Direct Tip Control* mode also directly controls the last bone of the chain.

IK End Twist *Optional*
: This control is visually attached to the last bone of the chain, and must use Euler rotation.

- *Stretch To Fit*: it controls the twist of the tip of the tentacle, interpolated to nothing at the base.
- *Manual Squash & Stretch*: it also controls the scaling of the tip of the tentacle.
- *Direct Tip Control*: the control does not exist.

FK Chain *Optional*
: If enabled, the rig has an alternative fully FK control chain.

The properties panel has the following features:

Start/End Controls *Optional*
: If extra controls exist, this property controls how many of them are visible and active.

  When a control is disabled, it is snapped to a position extremely close to the corresponding end control point, thus effectively neutralizing its effect. Thus, changing the setting during an animation can cause jumps.



The plus and minus buttons can help with maintaining a continuous transition in an animation by keyframing the change in the property value with Constant interpolation, and also snapping and keyframing the control itself to its 'hidden' position.

End Twist Estimate *Optional*

In the *Direct Tip Control* mode the twist at the end of the tentacle is deduced from the free form orientation of the tip control, rather than using a separate twist control with constrained Euler rotation. However, for technical reasons, that can only give values within the 180 degrees range of neutral.

A long tentacle can accept more twist than 180 degrees, so a workaround is necessary. This property allows specifying an approximate estimate of the twist value (effectively shifting the neutral position), and the rig then applies the automatic correction within 180 degrees of this value.

IK-FK, IK<->FK Snapping *Optional*

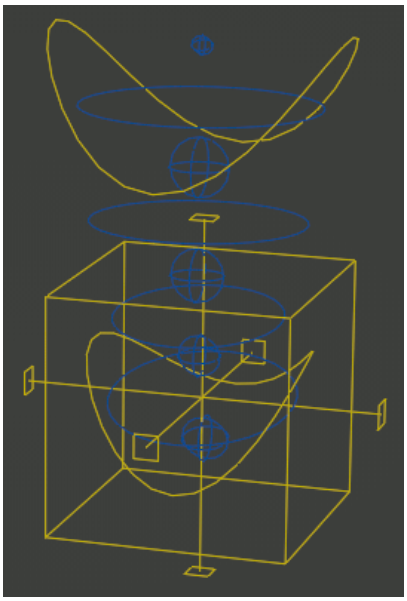
If the FK controls are enabled, these provide standard IK-FK switching and snapping.

However, unlike other limbs, for this rig automatic IK to FK snapping can only be approximate and requires manual tuning. For this reason, buttons for baking the snapping over a range of keyframes are not provided.

Parent Switch

Switches the parent of the selected IK control.

Spine, Head & Tail

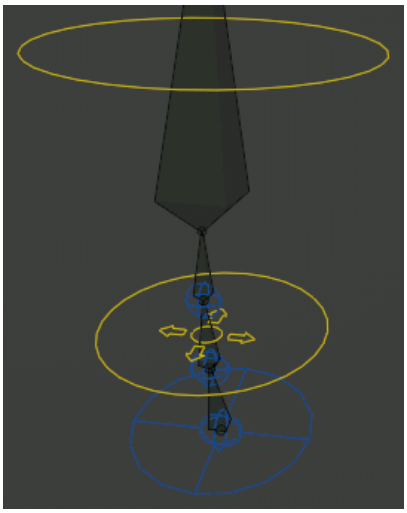


Spine

The spine sub-rig provides a cube shaped torso control with switchable parent, and bent circle shaped hip and chest controls subordinate to it. For low level deformation tweak controls are provided.

The torso control can optionally be accompanied with a custom pivot control. The rig can also optionally provide a full set of FK controls that are subordinate to the normal simplified ones, but above tweaks.

The rig properties panel for the spine controls usually includes options for the head and/or tail as well.



Head

The head sub-rig attaches to the end of the spine, and provides rotational controls for the head and neck, as well as tweaks for fine control of the neck.

If the neck is three or more bones long, an additonal tweak-like translational neck bend control is provided (the widget looks like a circle with arrows).

The properties panel contains the following options:

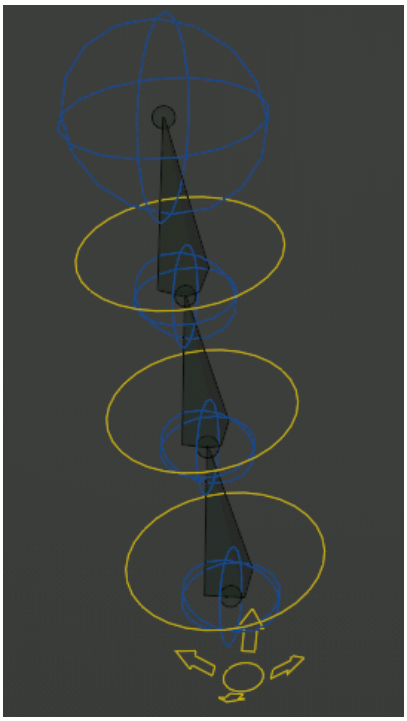
Neck Follow *Slider*



This slider controls the rotations isolation for the neck bones. The neck will follow the orientation of the Torso when set to 0, and the Chest when set to 1.

Head Follow *Slider*

This slider controls the rotations isolation for the head. The head will follow the orientation of the Torso when set to 0, and the Chest when set to 1.



Tail

The tail sub-rig attaches to the start of the spine, and provides FK controls for the tail, as well as a master control that replicates its local rotation around certain axes to all individual bones.

The properties panel contains the following options:

Tail Follow *Slider*
: This slider controls the rotations isolation for the tail. The tail will follow the orientation of the Torso when set to 0, and the Hips when set to 1.

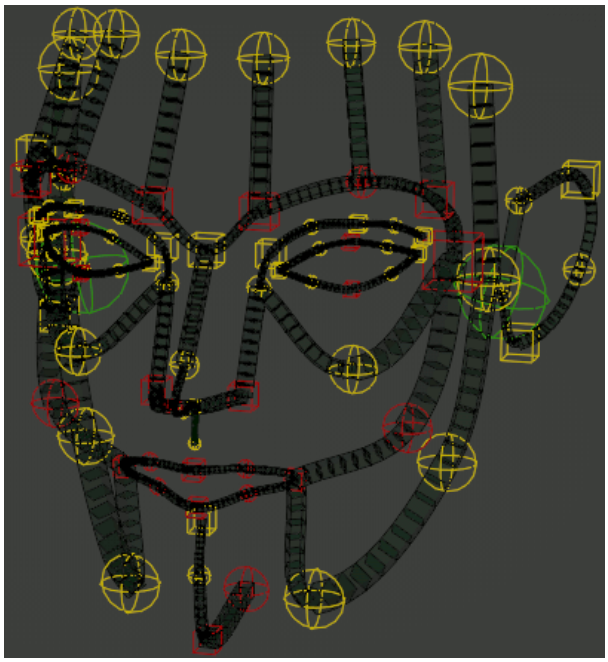
Face

Note

This describes the new-style modular face produced by the Upgrade Face operator button.

Basic Concepts

Skin Bone Chains



The foundation of the Rigify face is a network of Bendy Bone chains with controls placed at every bone end. These controls affect all bones that meet at that specific point.

When the controls are merely translated, the B-Bone chains retain the normal automatic bezier handle behavior. Local rotation and/or scaling of the controls are applied on top of that.

In case of certain chains, the transformation of the end and/or middle controls is interpolated to other controls located between them. In such cases the controls often have different colors and/or shapes.

Additionally, certain controls have arbitrary constraints that partially copy transformation from nearby control points.

Specialized Controllers

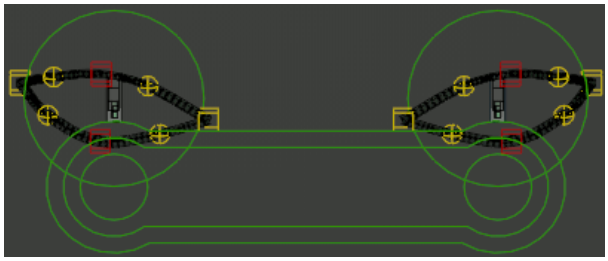
Certain areas of the face, like eyes or mouth, have additional specialized controllers that apply custom behavior on top of the chains and their controllers within the relevant area.

Eyes

The eyes have the following controls in addition to the eyelid chains:

Master

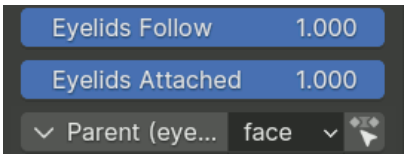
This large circular control can be used to transform the whole eye as one unit.



Common Target
: This large control enveloping all individual eye targets has a switchable parent and can be used to specify the point that the eyes should look at.

Eye Target
: These small circle controls within the common target control specify the point targeted by each individual eye. Their local scale can also be used to affect the iris or pupil of the eye, depending on how it was weight painted.

The rig properties panel contains the following options:



Eyelids Follow *Slider*
: Controls how much the rotation of the eyeball affects the eyelids. Depending on the sub-rig generation options, this slider can be split to separately control the horizontal and vertical directions.

Eyelids Attached *Slider Optional*
: If enabled in the sub-rig generation options, this slider can be used to disable the mechanism that forces the eyelids to conform to the sphere of the eye.

Parent *Parent Switch*
: Selects the parent for the common target control.

Mouth

The mouth has the following controls:

Jaw Master

Controls rotation of the jaw, directly affecting the main jaw deform bone, as well as chains fully belonging to the jaw. Chains forming the lip loop(s) are adjusted to open the mouth as the jaw rotates or moves.



Mouth Master

This control uniformly transforms the lips without moving the jaw.

The rig properties panel contains the following options:



Mouth Lock *Slider*

This slider can be changed from 0 to 1 in order to suppress opening of the mouth when the jaw rotates or moves.

Creating Meta-rigs

1. Add a single bone from the *Add ▸ Armature* menu.
2. Go in armature Edit Mode and build the meta rig by samples or Rigify-types.
3. Define the Rigify bone collection UI, color sets, and selection sets.
4. In the armature properties click on the *Generate* button to generate the rig.

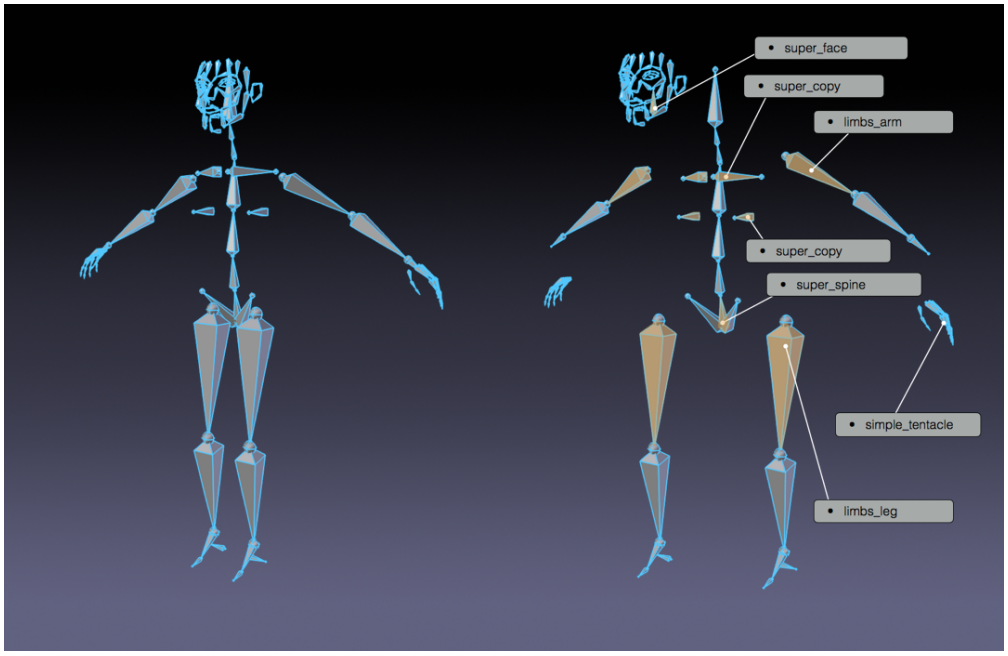
How Rigify Works

Rigify Meta-Rigs are split in multiple Sub-Rigs

A meta-rig is an assembly of bone chains. A bone chain is identified by the *Connected* attribute. Bone chains can be further connected together by parenting them without using the *Connected* attribute (i.e. using the *Keep Offset* option while parenting).

A custom attribute is set on the first bone of the sub-rig chain

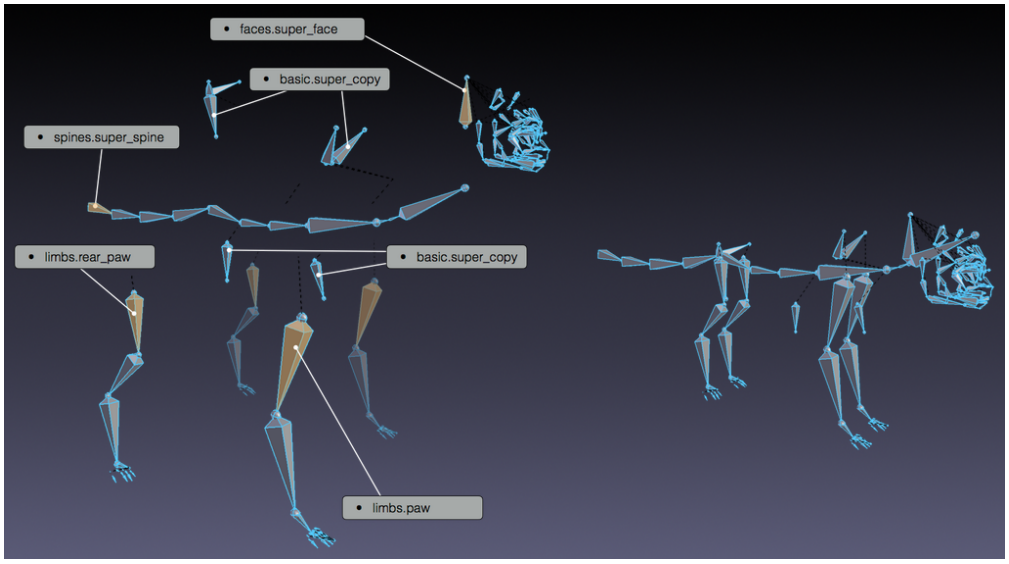
Each first bone of a bone chain has a custom attribute on it which is a Rigify custom property that identifies the sub-rig type. At rig generation time Rigify will determine which controls and deform bones will be created processing the meta-rig from the first bone to the last of each chain.



Human meta-rig split by samples.

New meta-rigs are created assembling sub-rigs samples

Since a meta-rig is just a collection of sub-rigs, new meta-rigs can be built assembling sub-rigs in different ways. This way an infinite number of meta-rigs can be built from the same rigging blocks.



Cat meta-rig built by samples.

All the mechanics, deformation bones and widget are created on a single click

The meta-rig contains more information than the visualized bones. In fact at generation time Rigify will identify each sub-rig type and depending on the selected options will create all the sophisticated controls, switches, and deforming bones with a single click.

Creating a new Meta-rig

Add a new Armature Object

Reference

Mode::	Object Mode
Menu::	*Add ▸ Armature ▸ Single Bone*
Shortcut::	`Shift-A`

Building your own meta-rig from scratch requires an armature object to work with. Just add a single bone from the *Add* menu.

Tip

At this stage naming the newly added armature `metarig` is a good idea. You can do it at any time (or not at all) but it's suggested to do it before going on so it will always be clear on which armature you have to work when editing the meta-rig structure.

Editing the Armature

Now that there is an armature object to work – with the armature selected – enter armature Edit Mode. Building a meta-rig from scratch in Edit Mode can be done in two ways:

1. Adding rig samples.
2. Creating bone chains.

Adding Samples (Basic)

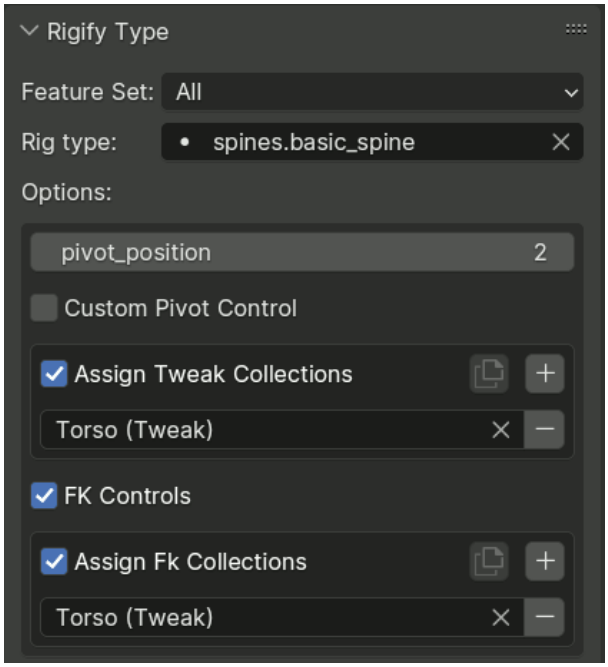
Adding pre-defined samples in Edit Mode is a good way to start building a meta-rig. This way you can become familiar with the available building blocks and how they are meant to be used. To add a rig sample:

1. Go in the armature tab.
2. Scroll down to Rigify panel.
3. Select a sample from the list.
4. Click on the *Add sample* button.
5. Edit the bone positions to match your character.

For the list of available samples, see the Rig Types page.

Using Rig Types (Advanced)

For full control, you can use the Rigify Type panel of bone properties in Pose Mode to assign any Rigify sub-rig type to any bone, as well as change its options.



For the list of available sub-rig types and their options, see the Rig Types page.

At the top of the panel you can find a field specifying the rig type for the active bone. The drop-down list can be optionally filtered by the Feature Set it belongs to.

Below that you can change options relevant to the selected rig type, if it has any.

Bone Collection References

Some rig types that generate many control bones have options that reference Bone Collections. These reference lists have a standard UI with

the following features:

- A checkbox controlling whether the reference should be used.
- A button to copy the reference list contents from the active to all selected bones.
- A plus button to add a new reference to the list.
- A list of references, each entry with a field to specify the target collection, and a button to remove the entry from the list.

Note

Each sub rig has a required number of bones as input. If you are unsure on how to use rig-types properties, add a rig sample to your armature to see how it is supposed to be used.

Preserved Bone Properties

Certain properties of the metarig bones are often copied to the generated rig control, deform and mechanism bones.

The exact set depends on the sub-rig and the specific generated bone, and the sub-rig may override some properties even when it preserves others from the same subset, but there are certain common patterns:

Parenting Settings
: This subset consists of the parent ORG bone, Use Connect, Use Inherit Rotation, Use Local Location, and Inherit Scale.

  It is usually copied to deform bones, FK controls, and in other cases where the sub-rig doesn't have a reason to completely override them.

Bendy Bone Settings (Edit Mode)
: Consist of the segment count, Vertex Mapping Mode, Ease In/Out, Roll In/Out, Curve In/Out and Scale In/Out.

The segment count is often overridden via a sub-rig option, but other settings are usually copied to deform bones as is.

Transformation Settings
: Consist of the rotation mode, pose mode rotation values, and channel locks.

  These settings are usually copied to FK controls.

Custom Properties
: Usually copied to one of the controls generated based on the metarig bone (mainly FK). Intra-armature drivers that access the property are retargeted to the copied instance.

Custom Widget
: Usually copied to one of the controls generated based on the metarig bone (mainly FK), and suppresses automatic generation of a widget for the bone if specified.

Custom Root Bone

If the meta-rig contains a bone called `root`, it is used as the root control bone instead of creating a new one. This allows changing the rest position of the root bone, assigning a custom widget, or adding custom properties to the bone.

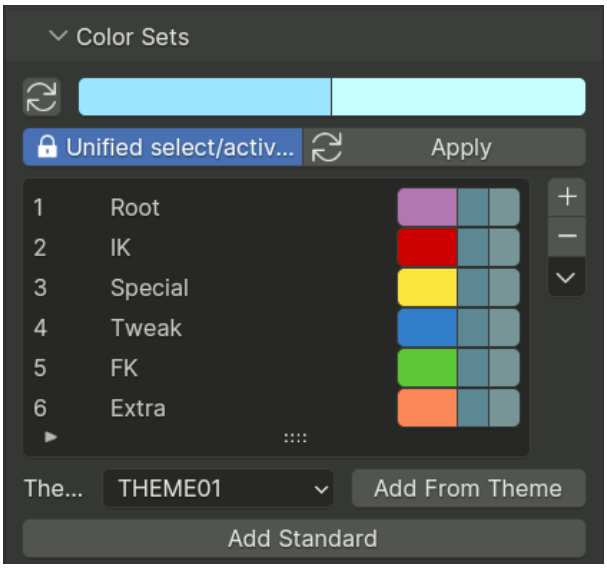
The custom root bone must have no parent, and use the basic.raw_copy sub-rig type or none.

Color Sets

The Color Sets panel is used to define the bone color scheme for the final rig. The colors from the list can be associated with bone collections from the relevant panel.

The top two rows of the Color Sets panel are used to define the general behavior of the bone colors. Usually color themes use a gradient of colors

to define the different bone states: default, selected and active. When multiple color themes are used in the same rig, identifying which bone is selected or active can be tricky since each color will have its corresponding state.



To override this behavior Rigify unifies the active and selected states using the same color. This is defined by two values:

Unified Selected/Active Colors
: When this option is active adding a bone group in the list will always keep the colors consistent. When a color scheme is added from a theme, the color scheme is loaded as is. Click on the *Apply* button to force the system to unify selected and active colors.

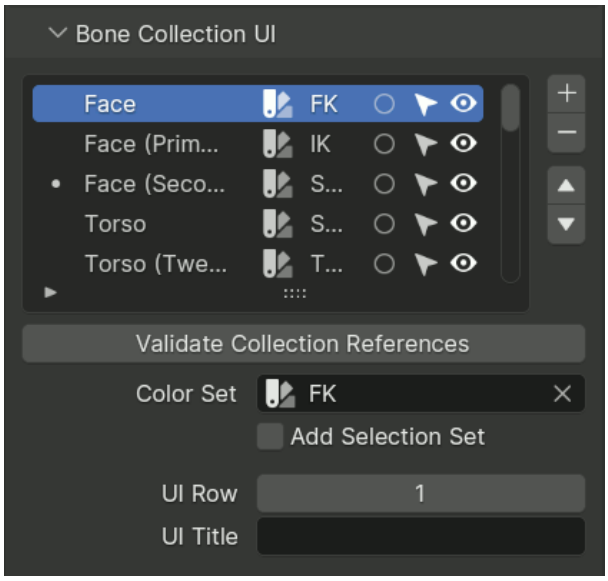
Selected/Active Colors
: This two color fields define respectively the *Selected* and *Active* colors. By default Rigify reads these colors from the theme defined by the user in the Blender preferences. This way the *Selected/Active* colors can always have a predictable and consistent behavior in the UI. The colors can be customized by clicking on the relevant color field. To reset them to the Blender current theme value just click on the button with the update icon.

Color Sets can be added and deleted by clicking on the + or – buttons. All color sets can be deleted at once by clicking on the Specials menu.

To add the colors from the predefined Rigify default color scheme (as shown in the image) to the list click the *Add Standard* button.

To add a specific theme with its own color scheme, select it from the list and click on the *Add From Theme* button.

Bone Collections UI



Bone Collections are used to group related bones together so that they can be hidden or revealed together.

Rigify can take advantage of collections to generate extra features and the user interface for the final rig. A panel named Rig Layers is generated with buttons for hiding the collections, arranged in an intuitive layout.

The Bone Collections UI panel allows configuring the layout of that generated panel, as well as specifying some other settings for bone collections, such as the color set to use.

The top of the panel is occupied by a list that duplicates the main bone collection list, but displays additional properties, such as the color set, whether the collection has a button, or whether it generates a selection set.

Validate Collection References

> Some sub-rig types have references to bone collections in their properties. Rigify uses a referencing scheme that is robust to collection renames, but deleting collections or joining armatures can still lead to broken references.
>
> This button runs a scan that validates and normalizes all collection references, reporting any errors, and reducing the chance of breakage being caused by subsequent user actions.
>
> This scan is also performed automatically every time the rig is generated.
>
> **Warning**

To avoid breakage this operation should be used both immediately before and after joining two metarig armatures. More specifically, it must be always done between the actions of renaming any collections and joining.

Color Set
: Specifies the color set to use for bones in this collection. If a bone belongs to multiple collections, in general the collection located earlier in the list has priority.

Add Selection Set
: Specifies whether a selection set should be generated for this collection.

UI Row
: If nonzero, specifies which row of the Rig Layers panel should contain the button controlling the visibility of this collection. When zero, no button is generated, and the collection is hidden.

UI Title
: This field can be used to override the title used on the UI button to be distinct from the true collection name. Unlike collection names, titles are not required to be unique, so this can be used to reduce clutter by relying on contextual cues within the panel.

UI Layout sub-panel

The UI Layout sub-panel provides a WYSIWYG editor for the layout of the generated UI panel (as defined by the UI Row and UI Title settings above).

Each row contains three buttons at the end:

Arrow
: Moves the active collection button to this row.

Plus
: Inserts a new row before the current one.

Minus
: Removes the current row and shifts all buttons up.

To the left of the editing control buttons, rows display buttons corresponding to the collections, same as the final UI, except that rather than hiding or unhiding, clicking these buttons selects the collection.

For the active collection the selection button is replaced with an input field for editing the UI Title, and an **X** button to unassign the collection from the UI.

For any collections not assigned to the UI, their select buttons are displayed in a separate section at the bottom of the sub-panel.

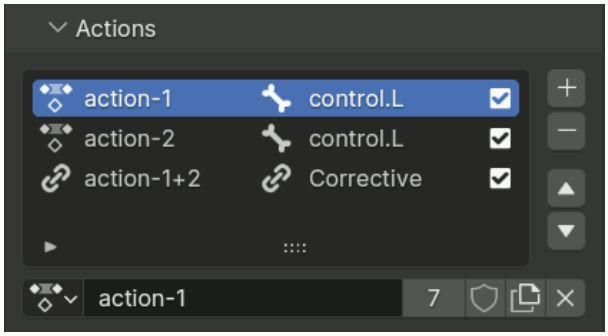
The `Root` collection will be added and/or assigned a UI button automatically if necessary when the rig is generated. If desired, it is possible to manually assign UI buttons to the internal `ORG`, `DEF` and `MCH` collections.



Tip

Blank rows appear much thinner in the final interface, since they don't have to contain editing buttons, and can be used as logical separators.

Actions



The Action constraint allows applying poses defined by an action to bones based on the transformation of another bone. This requires adding the constraint to every bone affected by the action, which is very tedious. For this reason, Rigify includes a system to do this automatically through the Actions panel.

The panel defines a list of actions to be applied to the generated rig bones. Each action must be listed only once.

The list entries show the name of the action, the trigger (a bone or a corrective action driven by two others), and a checkbox that can be used to temporarily disable applying this action to the rig. The icon at the start of the entry is changed from an action icon to a link icon to highlight corrective actions that depend on the active normal one, or normal actions used by the active corrective action.

Note

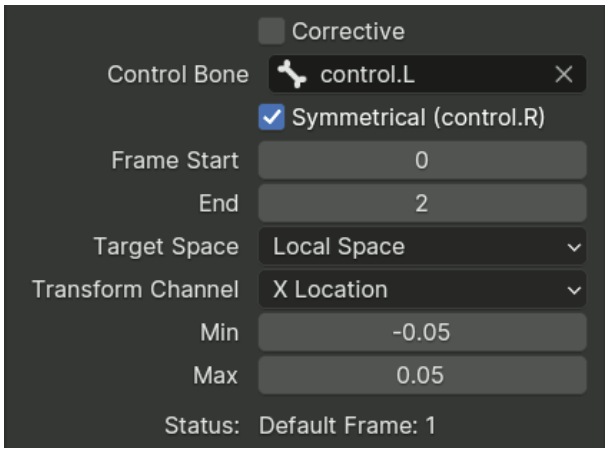
The Action constraints are added to the bones in such an order as to exactly reproduce the intended deformation, assuming the actions were created (posed and keyframed) in the order listed.

Normal Actions

Normal actions are applied based on the transformation of a specific control bone from the generated rig. They have the following properties:

Control Bone

Specifies the bone that drives the action.



Symmetrical

If the control bone has a suffix that specifies that it belongs to the left or right side, this option can be enabled to automatically apply symmetry.

When enabled, left-side bones keyframed in the action will be controlled by the left-side control, and right-side bones by the right side control. Bones that don't have a a side suffix are assumed to belong to the center of the character. They are rigged with two Action constraints with influence 0.5 that are controlled by each of the control bones.

Frame Start & End

Specifies the frame range of the action that will be used by the created constraints.

Target Space, Transform Channel

Specifies the coordinate space and transformation channel of the target bone that should be used.

Min, Max

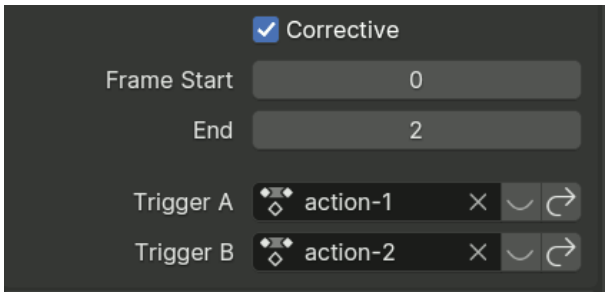
Specifies the range of the transformation channel values that is mapped to the specified action frame range.

Default Frame

Shows the frame within the action that maps to the neutral value (1 for scale and 0 otherwise) of the transformation channel, as computed from the specified range values.

Corrective Actions

Corrective actions are applied based on the progress of two other actions from the list, and are used to improve the pose when they are used together.



Frame Start & End

Specifies the frame range of the action that will be used by the created constraints.

Trigger A & B

Specifies the two actions that control the correction. The interface rows contain buttons to show the settings for that action, or jump to it in the list.

The progress of the corrective action from the start to the end frame is calculated as the product of the progress values of the two trigger actions. Thus, the start frame is applied when either of the triggers is at the start frame, and the end frame is used when both are at their end frame.

Corrective actions must be below their triggers in the list, which is enforced via an implicit reorder even if violated.

Tip

Corrective actions behave in the most intuitive way when both triggers have the Default Frame equal to Start Frame. To create a corrective action in such case:

- Create the two trigger actions, add them to the panel and generate the rig.
- Pose your controls so that both trigger actions are fully activated to the end frame.
- Pose and keyframe the necessary corrections in the end frame of the new action, while keying the start frame to the neutral values.
- Add the newly created action to the end of the list in the panel and configure its settings.

Rig Types

Rig types are components used by Rigify to process specific parts of the meta-rig when generating the armature. They represent common character features, like the spine, limbs, fingers etc.

Note

The list of available rig types appears in the Bone properties tab when the bone is selected in Pose Mode. Scroll down the Properties to find Rigify Type panel.

This documents rig types that are bundled with Rigify.

- Basic
  - basic.copy_chain
  - basic.pivot
  - basic.raw_copy
  - basic.super_copy
- Spines
  - spines.super_spine
  - spines.basic_spine
  - spines.basic_tail
  - spines.super_head
- Limbs
  - limbs.simple_tentacle
  - limbs.super_finger
  - limbs.super_limb
  - limbs.arm
  - limbs.leg
  - limbs.paw
  - limbs.front_paw
  - limbs.rear_paw
  - limbs.super_palm

    - limbs.spline_tentacle
- Faces
    - faces.super_face
- Skin
    - skin.basic_chain
    - skin.stretchy_chain
    - skin.anchor
    - skin.glue
    - skin.transform.basic
- Face
    - face.basic_tongue
    - face.skin_eye
    - face.skin_jaw

Basic

These rig types are used to generate simple single-bone features, and for custom rigging done directly in the meta-rig.

The single-bone rig types must be applied separately to every bone even within a connected chain, and can have connected children controlled by a different rig type. This is unlike chain-based rig types that usually consume the whole connected chain.

basic.copy_chain

Copies the bone chain keeping all the parent relations within the chain untouched. Useful as a utility rig type for custom rigs.

Requirement: A chain of at least two connected bones.

Control (Boolean)
: When enabled control bones and widgets will be created.

Deform (Boolean)
: When enabled deform bones will be created.

basic.pivot

Single-bone rig type that creates a ‘custom pivot’ control for rotating and scaling its child sub-rigs.

This type of control transforms its children when rotated or scaled, while moving it merely changes the pivot point used by rotation or scaling.

Master Control
: When enabled an extra parent control bone with a box widget is created to allow moving the rig. It is also required by all other options besides

Deform Bone.

Widget Type
: Allows selecting one of the predefined widgets to generate for the master control instead of the default cube.

Switchable Parent
: Generates a mechanism for switching the effective parent of the rig based on the value of a custom property.

Register Parent
: Registers the rig as a potential parent scope for its child sub-rigs' parent switches.

  Tags
  : Specifies additional comma-separated tag keywords for the registered parent scope. They can be used by other rigs to filter parent choices, or for selecting the default parent.

    Some of the existing tags that are useful here:

    `injected` (special)
    : The parent scope will be made available for all children of the *parent* sub-rig, rather than just this rig's children.

    `held_object`
    : A control for the object held in the character's hand. Preferred by finger IK.

      The `injected,held_object` combination is perfect for such a control.

Pivot Control
: Disabling this avoids generating the actual custom pivot control, effectively turning this rig type into a version of basic.super_copy with parent switching support and a different widget.

Deform Bone

When enabled a deform bone will be created.

basic.raw_copy

Single-bone rig type that copies the bone without the `ORG-` name prefix.

Normally all bones copied from the meta-rig are prefixed with `ORG-` and placed on an invisible layer. This precludes their use as controls or deforming bones, which makes it difficult to transfer complex fully custom rigging verbatim from the meta-rig.

This rig type does not add the automatic prefix, thus allowing an appropriate `ORG-`, `MCH-` or `DEF-` prefix to be manually included in the meta-rig bone name, or alternatively using no prefix to create a control bone.

Relink Constraints
: Allows retargeting constraints belonging to the bone to point at bones created in the process of generating the rig, thus allowing custom rigging to integrate with generated bones.

  To use this feature, add `@` and the intended target bone name to the constraint name, resulting in the `...@bone_name` syntax. After all bones of the rig are generated, the constraint target bone will be replaced. If the new bone name is just `CTRL`, `MCH` or `DEF`, this will just replace the `ORG` prefix in the existing target bone name. For the Armature constraint you can add a `@` suffix for each target, or just one `@CTRL`, `@MCH` or `@DEF` suffix to update all.

  Parent
  : If the field is not empty, applies the same name substitution logic to the parent of the bone.

  When this feature is enabled, the bone will not be automatically parented to the root bone even if it has no parent; enter `root` in the *Parent* field if that is necessary.

basic.super_copy

Single-bone rig type that simply copies the bone. Useful as utility rig type for adding custom features or specific deform bones to your rigs.

Control (Boolean)
: When enabled a control bone and widget will be created.

Widget (Boolean)
: When enabled a widget will be created in replacement to the standard.

Widget Type (String):
: Allows selecting one of the predefined widget types to generate instead of the default circle.

Deform (Boolean)
: When enabled a deform bone will be created.

Relink Constraints
: Works the same as in the basic.raw_copy rig. In addition, when enabled any constraints that have names prefixed with `CTRL:` are moved to the control, and with `DEF:` to the deform bone.

Spines

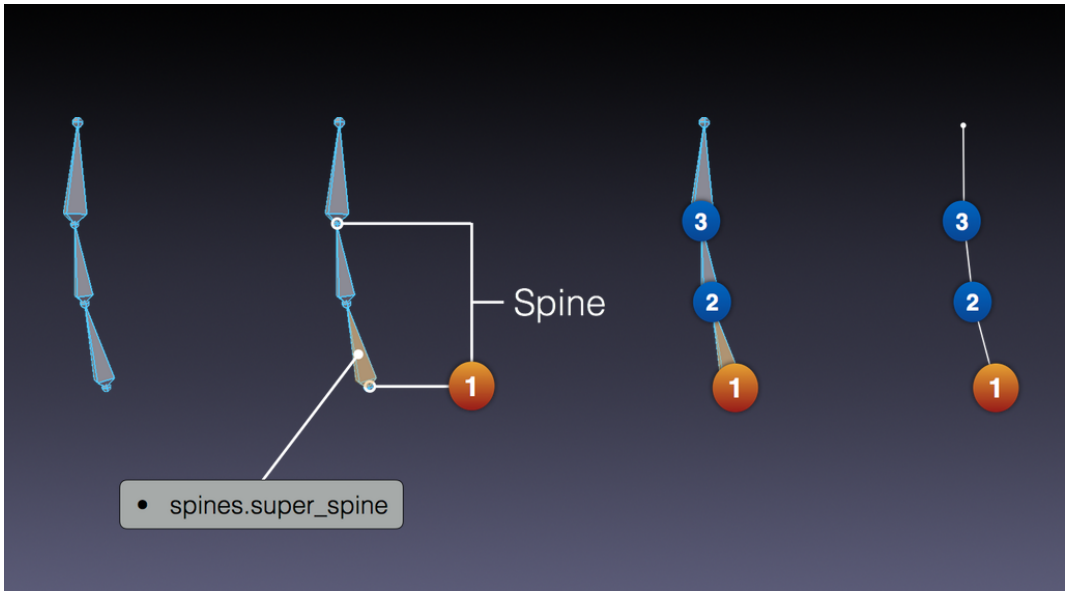
These rigs are used to generate spine structures, including the head and tail.

spines.super_spine

Will create a complete bendy and stretchy b-bones spine system based on bone numbers of your bone chain and user defined options.

This is a composite wrapper of spines.basic_spine, spines.super_head and spines.basic_tail. Note that for the tail, the direction of the bones is reversed compared to the separate rig.

Requirement: A chain of at least three connected bones (base system).



Spine required bones.

Pivot Position (integer)

Defines the pivot position for torso and hips.

Head (Boolean)

When checked neck and head systems will be added to your spine rig.

Neck Position (integer)

Defines the bone where the neck system starts. The last bone will always be the head system. If neck position is the last bone of the chain, then only the head system will be created ignoring the neck.

Tail (Boolean)

When checked tail system will be added to your spine rig.

Tail Position (integer)

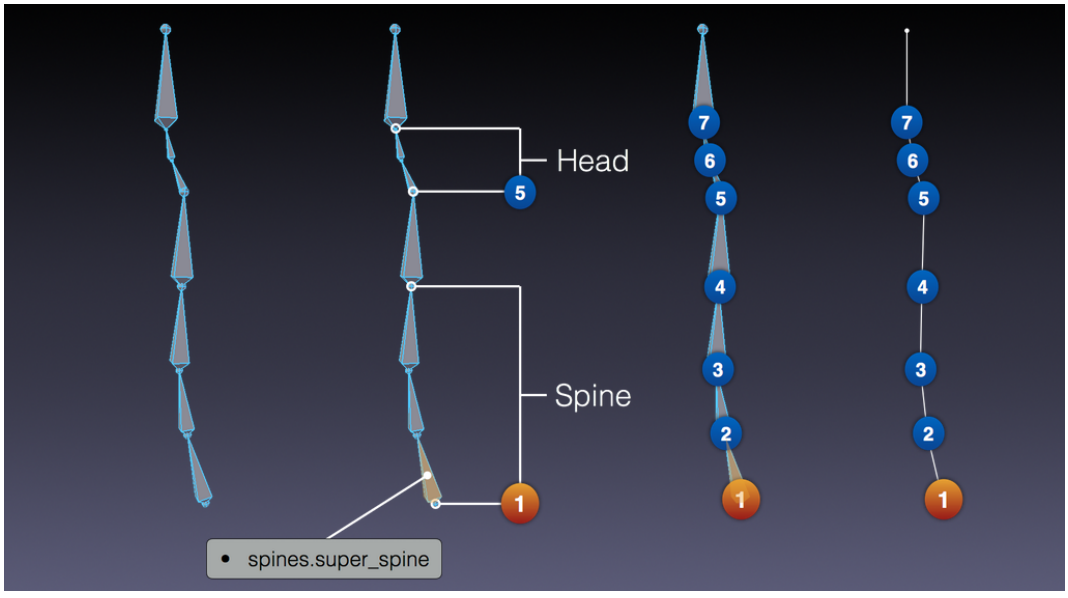
Defines the bone where the tail system starts. The next bone will always be the hips system.

X, Y, Z (Boolean)

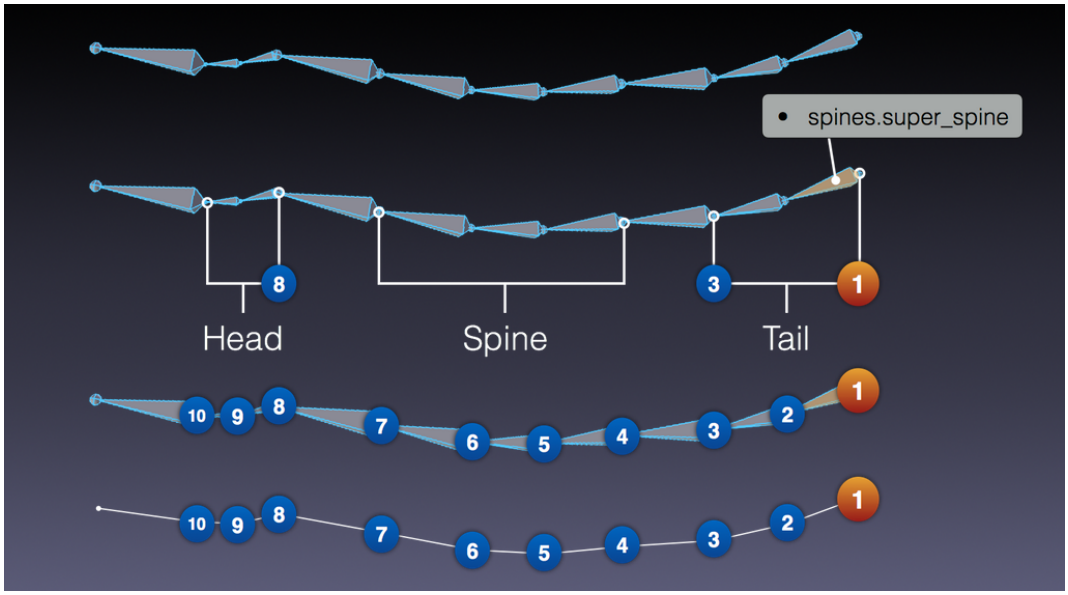
When generating a tail, specifies which local axis rotations should be replicated along the chain.

Assign Tweak Layers

If enabled, allows placing the Tweak controls in different bone collections from the IK bones.



Spine default bones.



Spine with tail bones.

spines.basic_spine

Defines a bendy and stretchy b-bones spine.

Pivot Position (integer)
: Defines the pivot position for torso and hips.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

FK Controls
: Specifies whether to generate an FK control chain.

Assign FK Layers
: If enabled, allows placing the FK chain in different bone collections from the IK bones.

spines.basic_tail

Defines a bendy and stretchy b-bones tail.

X, Y, Z (Boolean)
: Specifies which local axis rotations should be replicated along the chain from each control bone to the following one.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

spines.super_head

Defines a head rig with follow torso controls.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

Limbs

These rig types handle generation of different kind of limbs and their features, like fingers.

limbs.simple_tentacle

Will create a simple bendy and stretchy b-bones tentacle chain, which can optionally replicate local rotation from preceeding bones to the subsequent ones for use in cases like fingers.

Requirement: A chain of at least two connected bones.

Automation Axis (X, Y, Z, None)
: Enables the automation on the selected axis. Multiple axis or none can be selected holding `Shift-LMB`. When enabled the subsequent control bones will copy the local rotations from the previous ones. The option is accessible in the controls of the final rig as a Copy Rotation constraint and can be disabled even after rig is generated, or at animation time.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the main controls.

limbs.super_finger

Will create a bendy and stretchy finger chain with a master control bone that controls the rotation of all joints through its scale.

Requirement: A chain of at least two connected bones.

Bend Rotation Axis (Automatic, X, Y, Z, -X, -Y, -Z)
: Defines the automatic rotation axis to be linked to the scale of the master bone.

B-Bone Segments (integer)

Defines the number of b-bone segments each tweak control will be split into.

IK Control

Generates a very simple IK mechanism with only one control.

IK starts its work with the shape of the finger defined by FK controls and adjusts it to make the fingertip touch the IK control. It is designed as a tool to temporarily keep the fingertip locked to a surface it touches, rather than a fully featured posing system.

To improve performance, the switchable parent for the IK control contains only one option beside None. Thus it is advised to add a 'held object' control using the basic.raw_copy rig to act as the common parent for the fingers with a fully functional parent switch.

IK Local Location

Specifies the value of the Local Location option for IK controls, which decides if the location channels are aligned to the local control orientation or world.

Assign Tweak Layers

If enabled, allows placing the Tweak controls in different bone collections from the main controls.

Assign Extra IK Layers

If enabled, allows placing the extra IK control in different bone collections from the main controls.

Note

Rotation Axis (Bend Rotation Axis in the case of limbs.super_finger) affects the roll of the generated bones. Automatic mode recalculates the generated bones roll while any of the Manual modes copy the roll of the meta-rig bones.

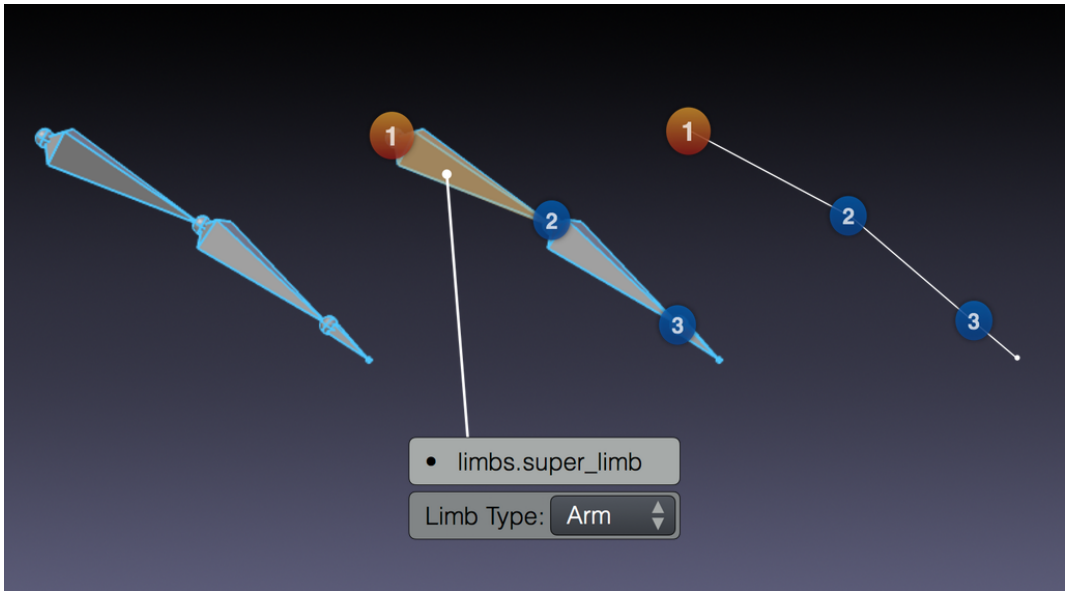
limbs.super_limb

A backwards compatibility wrapper around limbs.arm, limbs.leg and limbs.paw.

limbs.arm

Will create a fully featured bendy and stretchy arm depending on the user-defined options.

Requirement: A chain of three connected bones (upper_arm, forearm, hand).



Arm required bones.

IK Wrist Pivot
: Generates an extra child of the hand IK control that rotates around the tail of the hand bone.

Rotation Axis (Automatic, X, Z)
: Defines the bend axis for the IK chain. FK chains will have a totally free degree of rotation on all axes.

Limb Segments (integer)
: Defines the number of additional tweak controls each limb bone will have on the final rig.

B-Bone Segments (integer)
: Defines the number of b-bone segments each tweak control will be split into.

Custom IK Pivot
: Generates an extra control for the end of the IK limb that allows rotating it around an arbitrarily placed pivot.

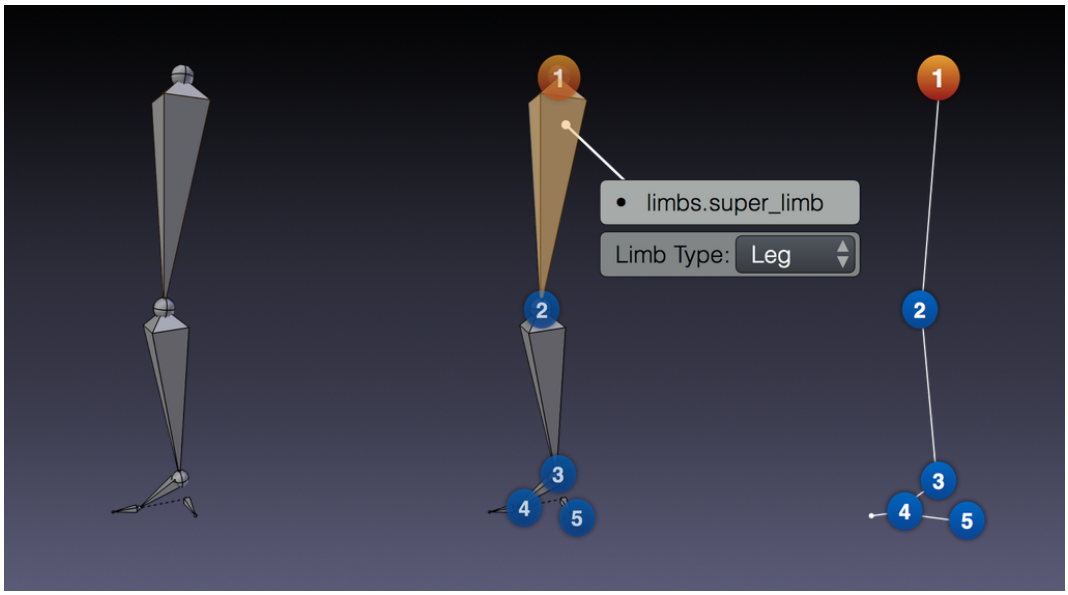
Assign FK Layers
: If enabled, allows placing the FK chain in different bone collections from the IK bones.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

limbs.leg

Will create a fully featured bendy and stretchy leg depending on the user-defined options.

Requirement: A chain of four connected bones (thigh, shin, foot, toe) with one unconnected child of the foot to be used as the heel pivot.



Leg required bones.

Foot Pivot (Ankle, Toe, Ankle & Toe)

Specifies where to put the pivot location of the main IK control, or whether to generate an additional pivot control at the base of the toe.

Separate IK Toe

Specifies that two separate toe controls should be generated for IK and FK instead of sharing one bone. This is necessary to get fully correct IK-FK snapping in all possible poses.

Toe Tip Roll

Generates a slider to switch the heel control to pivot on the tip rather than the base of the toe (for roll this obviously only applies on forward roll).

Rotation Axis (Automatic, X, Z)

Defines the bend axis for the IK chain. FK chains will have a totally free degree of rotation on all axes.

Limb Segments (integer)

Defines the number of additional tweak controls each limb bone will have on the final rig.

B-Bone Segments (integer)
: Defines the number of b-bone segments each tweak control will be split into.

Custom IK Pivot
: Generates an extra control for the end of the IK limb that allows rotating it around an arbitrarily placed pivot.

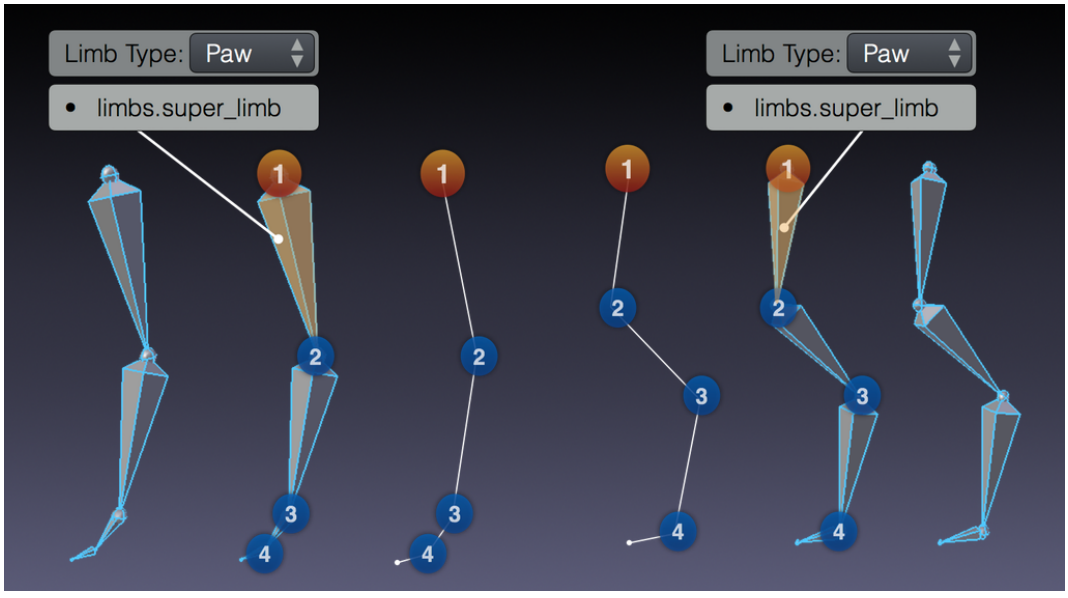
Assign FK Layers
: If enabled, allows placing the FK chain in different bone collections from the IK bones.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

limbs.paw

Will create a fully featured bendy and stretchy paw depending on the user-defined options.

Requirement: A chain of four or five connected bones (thigh, shin, paw, *optional* digit, toe).



Front/Rear paw required bones.

Rotation Axis (Automatic, X, Z)
: Defines the bend axis for the IK chain. FK chains will have a totally free degree of rotation on all axes.

Limb Segments (integer)
: Defines the number of additional tweak controls each limb bone will have on the final rig.

B-Bone Segments (integer)
: Defines the number of b-bone segments each tweak control will be split into.

Custom IK Pivot
: Generates an extra control for the end of the IK limb that allows rotating it around an arbitrarily placed pivot.

Assign FK Layers
: If enabled, allows placing the FK chain in different bone collections from the IK bones.

Assign Tweak Layers
: If enabled, allows placing the Tweak controls in different bone collections from the IK bones.

limbs.front_paw

Derivative of limbs.paw with extended IK suitable for use in front paws. The additional IK limits the degree of change in the angle between shin and paw bones (2nd and 3rd) as the main IK control moves and rotates.

For best results, the shin bone should not be parallel to either thigh or paw in rest pose, i.e. there should be some degree of bend in all joints of the paw.

Heel IK Influence

Influence of the extended IK. At full rotating the main IK control or digit bone would not affect the rotation of the paw bone, while lower values provide some blending.

limbs.rear_paw

Derivative of limbs.paw with extended IK suitable for use in rear paws. The additional IK tries to maintain thigh and paw bones (1st and 3rd) in a nearly parallel orientation as the main IK control moves and rotates.

For best results, thigh and paw bones should start nearly parallel in the rest pose.

limbs.super_palm

Will create a palm system based on the distance between palm bones.

Requirement: At least two bones child of the same parent. The property has to be set on the inner palm bones (think it as index's metacarpus), the rig control will appear on the last palm bone (think it as pinky's metacarpus).

Both Sides
: Generates controls on both sides of the palm, with influence on inner bones blended between them.

Primary Rotation Axis (X, Z)
: Defines the automatic rotation axis to be used on the palm bones.

limbs.spline_tentacle

This rig type implements a flexible tentacle with an IK system using the Spline IK constraint. The control bones define control points of a Bezier curve, and the bone chain follows the curve.

The curve control points are sorted into three groups: start, middle and end. The middle controls are always visible and active, while the other two types

can be shown and hidden dynamically using properties; when enabled they appear next to the corresponding permanent start/end control and can be moved from there.

Extra Start Controls
: Specifies the number of optional start controls to generate.

Middle Controls
: Specifies the number of middle controls to generate.

Extra End Controls
: Specifies the number of optional end controls to generate.

Tip Control:
: Specifies how the curve stretching and the final control bone work:

  Stretch To Fit
  : Stretches the whole bone chain to fit the length of the curve defined by the controls.

    An end twist control is generated to control the twist along the chain.

  Direct Tip Control
  : Generates an IK end control, which directly controls the final bone of the chain similar to how regular IK works for limbs, as well as controlling the end of the bezier curve. The middle bones of the chain stretch to follow the curve and cover the gap.

    The rig automatically deduces twist of up to 180 degrees based on the orientation of the end control. Higher amounts of twist have to be dialled in through an End Twist Estimate slider to avoid flipping.

  Manual Squash & Stretch
  : This mode allows full manual control over the chain scaling, while the chain covers as much of the curve as it can given its current length.

The start control of the chain manages its uniform squash & stretch scale, while the end twist control manages both the twist of the chain, as well as its scale at the tip (blended gradually along the length).

Radius Scaling

Allows scaling the controls to control the thickness of the chain through the curve.

Maximum Radius

Specifies the maximum scale allowed by the Radius Scaling feature.

FK Controls

Generates an FK control chain and IK-FK snapping.

Assign FK Layers

If enabled, allows placing the FK chain in different bone collections from the IK bones.

Faces

faces.super_face

Will create a face system based on the bones child to the parent that has the property set on it.

Requirement: All the face bones bundled in the `faces.super_face` sample had to be present and child of the master bone that has the Rigify-type *face* property set.

Note

This rig type is being deprecated in favor of a new modular skin and face rigging system.

Skin

These rigs implement a flexible system for rigging skin using multiple interacting B-Bone chains. This is developed as the base for a new modular Rigify face rig. These are the main ideas of the system:

Generic B-Bone Chain

One core idea of the system is that most of the deformation should be implemented using a standard powerful B-Bone chain rig. These chains support advanced behavior by interacting with other rig components. This is in contrast to having multiple domain-specific rigs that each generate their own deform chains.

The implementation provides two versions of the chain rig: skin.basic_chain merely attaches B-Bones to the controls with no automation added to the controls themselves. The skin.stretchy_chain rig in addition interpolates motion of the end (and an optional middle) controls to the other controls of the chain.

Automatic Control Merging

The deformation part of the system consists of chains of one or more B-Bones connecting control points (nodes). Whenever controls for two chains would completely overlap, they are automatically merged.

For each merged control, one of the chains is selected as the owner, based on heuristic factors like parent depth from root, presence of `.T`/`.B` `.L`/`.R` symmetry markers, and even alphabetical order as the last resort. This can be overridden by an explicit priority setting in cases when it guesses wrong.

The owner and its parents determine additional automation that is placed on the control. As a special case, if a control is merged with its `.T`/`.B` `.L`/`.R` symmetry counterparts (detected purely by naming), the automation from all of the symmetry siblings of the owner is averaged.

Parent Controllers

Rather than simply using the parent meta-rig bone (ORG) as parent for controls and chain mechanisms, the new system includes an interface for parent rigs. It explicitly provide parent bones and generate control parent automation mechanisms for their child chain controls by inheriting from the appropriate base and overriding methods.

This allows implementing rigs that integrate and manage their child chains in intelligent ways in order to add extra automation specific to certain areas. The base skin system includes one simple example skin.transform.basic rig, which translates its child control points according to its control bone transformation.

Custom Rigging

Finally, the new system provides ways to integrate with custom automation directly included in the meta-rig via two extra rig components.

The skin.anchor rig generates a single control with inherited constraints etc., similar to basic.super_copy. However, it also integrates into the skin system as a zero length chain with highest priority. This allows overriding the normal behavior by providing a control point under full control of the user, which other chains would automatically attach to.

The skin.glue rig on the other hand will attach itself to the control that is generated at its position (it is an error if there is none). It can be used to read the position of the control from custom rigging in the meta-rig, or inject constraints into the control bone. It is possible to also detect the control at the tail of the glue bone and use it as target in the constraints, thus copying transformation between the controls.

skin.basic_chain

This is the basic chain rig, which bridges controls with B-Bones but does not add any automation to the controls themselves.

When controls are merely moved, the chains behave as if using standard automatic handles, but rotating and optionally scaling the controls will adjust the result.

B-Bone Segments

Specifies the number of segments to use. Setting this to 1 disables all advanced behavior and merely bridges the points with a Stretch To bone.

Merge Parent Rotation and Scale

This can be enabled to let the chain respond to rotation and scale induced by parents of controls owned by other chains that this chain's control merged into.

Use Handle Scale

Enables using control scale to drive scale and/or easing of the B-Bone.

Connect With Mirror

Specifies whether the ends of the chain should smoothly connect when merging controls with its `.T`/`.B` `.L`/`.R` symmetry counterpart. The relevant option must be enabled on both chains to work.

Connect Matching Ends

Specifies whether the end of the chain should connect to the opposite end of a different chain when merging controls. Thus forming a continuous smooth chain in the same direction. The relevant options must be enabled on both chains.

Sharpen Corner

Specifies whether the rig should generate a mechanism to form a sharp corner at the relevant connected end, depending on the angle formed by adjacent control locations. When the control angle becomes sharper than the specified value, ease starts reducing from 1 to 0.

Orientation

Specifies that the controls should be oriented the same as the selected bone, rather than being aligned to the chain.

Copy To Selected
: Copy to selected rigs that have the same option. Thus allowing to indiscriminately selecting bones without assigning unnecessary values.

Chain Priority
: Allows overriding the heuristic used to select the primary owner when merging controls.

skin.stretchy_chain

This rig extends the basic chain with automation that propagates movement of the start and end, and an optional middle control, to other controls. This results in stretching the whole chain when moving one of the ends, rather than just the immediately adjacent B-Bones.

Middle Control Position
: Specifies the position of the middle control within the chain; disabled when zero.

Falloff
: Specifies the influence falloff curves of the start, middle and end controls. Zero results in linear falloff, increasing widens the influence, and -10 disables the influence propagation from that control completely.

Spherical Falloff
: Toggle buttons to change the shape of the falloff curve from a power curve that at falloff 1 forms a parabola $(1 - x^{2^f})$ to a curve forming a circle $((1 - x^{2^f})^{2^{-f}})$.

Falloff Along Chain Curve
: Computes the falloff curve along the length of the chain, instead of projecting on the straight line connecting its start and end points.

Propagate Twist
: Specifies whether twist of the chain should be propagated to control points between main controls.

Propagate Scale
: Specifies whether perpendicular scaling of the chain should be propagated to control points between main controls.

Propagate to Controls
: Allows other chains to see propagated twist and scale via *Merge Parent Rotation and Scale* when their controls are merged into this chain, instead of it being completely local to this chain.

Primary Control Layers
: Optionally specifies bone collections for the end controls.

Secondary Control Layers
: Optionally specifies bone collections for the middle control, falling back to *Primary Control Layers* if not set.

The main controls with active falloff have the effect of *Merge Parent Rotation and Scale* automatically enabled just for them.

skin.anchor

This rig effectively acts as a zero-length chain with highest priority, ensuring that it becomes the owner when merging controls with other chains. And also allowing one to input custom automation influence into the skin system.

All constraints on the meta-rig bone are moved to the created control.

Generate Deform Bone
: Creates a deformation bone parented to the control.

Suppress Control
: Makes the control a hidden mechanism bone to hide it from the user.

Widget Type
: Selects which widget to generate for the control.

Relink Constraints

Operates the same as in basic.raw_copy, except all constraints are moved from ORG to the control bone.

Orientation
: Specifies the bone used to orient the control, like for other chains.

skin.glue

This rig is in concept similar to skin.anchor, but instead of overriding controls, it is used to read or adjust the state of controls generated by other rigs. The head of the bone must overlap a control of another skin rig.

The rig sets up its ORG bone to read the state of the control, while moving all constraints that were originally on the bone to the control.

Glue Mode
: Specifies how the ORG bone is connected to the skin control.

  Child Of Control
  : Makes the ORG bone a child of the control bone.

  Mirror Of Control
  : Makes the ORG bone a sibling of the control with a Copy Transforms constraint from the control. The resulting local space transformation is the same as control's local space.

  Mirror With Parents
  : Parents the ORG bone to the parent automation a control owned by the glue rig would have had, while making it follow the actual control. This includes both direct and parent-induced motion of the control into the local space transformation of the bone.

  Deformation Bridge
  : Other than adding glue constraints to the control, the rig acts as a one segment basic deform chain. This is convenient when a pair of controls need to be bridged both with glue and a deform bone.

Relink Constraints
: Operates the same as in basic.raw_copy, except all constraints are moved from ORG to the control bone.

Use Tail Target
: Relinks `TARGET` or any constraints with an empty target bone and no relink specification to reference the control located at the tail of the glue bone.

Target Local With Parents
: Switches the tail target to operate similarly to *Mirror With Parents*.

Add Constraint
: Allows to add a typical glue constraints with specific *Influence*, as if it were at the start of the ORG bone constraint stack.

skin.transform.basic

This rig provides a simplistic parent controller, which uses regular translation, rotation, or scale to modify locations but not orientations or scale of its child chain controls.

Generate Control
: Specifies whether to generate a visible control, or use the transformation of the ORG bone as a part of more complex and specific rig setup.

Face

These rig types implement components of a modular face.

face.basic_tongue

Generates a simple tongue, extracted from the original PitchiPoy super_face rig.

B-Bone Segments (integer)
: Defines the number of b-bone segments each tweak control will be split into.

Primary Control Layers
: Optionally specifies bone collections for the main control.

face.skin_eye

Implements a skin system parent controller that manages two skin chains for the top and bottom eyelids in addition to generating the eye rotation mechanism.

The rig must have two child skin chains with names tagged with `.T` and `.B` symmetry to mark the top and bottom eyelid, which are connected at their ends forming eye corners. The chains are rigged to follow the surface of the eye and twist to its normal.

In addition, it creates target controls for aiming the eye, including a master control shared by all eyes under the same parent rig. The eyelids are rigged to follow the movement of the eyeball with adjustable influence.

Eyeball and Iris Deforms
: Generates deform bones for the eyeball and the iris, the latter copying XZ scale from the eye target control. The iris is located at the tail of the

ORG bone.

Eyelid Detach Option
: Generates a slider to disable the mechanism that keeps eyelid controls stuck to the surface of the eye.

Split Eyelid Follow Slider
: Generates two separate sliders for controlling the influence of the eye rotation on X and Z eyelid motion.

Eyelids Follow Default
: Depending on *Split Eyelid Follow Slider*, specifies the default values for the split follow sliders, or fixed factors to be multiplied with the single common follow influence slider value.

face.skin_jaw

Implements a skin system parent controller that manages one or more loops of mouth skin chains in response to the movement of jaw and mouth controls.

The rig must have one or more child chain loops, each formed by four skin chains tagged with `.T`/`.B` and `.L`/`.R` symmetrical names.

The lip loops are sorted into layers based on the distance from corners to the common center and rigged with blended influence of the jaw and the master mouth control. Other child rigs become children of the jaw.

Bottom Lip Influence
: Specifies the influence of the jaw on the inner bottom lip with mouth lock disabled.

Locked Influence
: Specifies the influence of the jaw on both lips of locked mouth.

Secondary Influence Falloff
: Specifies the factor by which influence fades away with each successive lip loop (for bottom lip loops the blend moves away from inner bottom

lip to full jaw influence).

Feature Sets

Rigify allows third party developers to implement sub-addons, called *Feature Sets*, which can provide new Meta-Rigs and Rig Types. Similar to regular add-ons, they can be installed from zip-files through Rigify settings.

These are some examples of *Feature Sets* currently provided by past and current Rigify developers:

Cessen's Rigify Extensions
: This feature set provides the original Rigify rigs by Nathan Vegdahl, minimally ported and repackaged to work without switching Rigify to legacy mode. Note that their names were changed, so meta-rigs designed for legacy mode aren't directly compatible.

Experimental Rigs by Alexander Gavrilov
: Rig experiments, some of which might be included in Rigify in the future. Examples include limbs with an extra IK system based at knee/elbow, a spline based tentacle, and more.

You can install these packages by clicking *Code* ▸ *Download ZIP*, and then install the downloaded file through Rigify settings.

Developer documentation is available on the Blender Developer Documentation.

System

Important

Work In Progress

These add-ons relate to showing information about objects and scenes.

- Manage UI Translations

Manage UI Translations

Todo

Enabling Add-on

1. Open Blender and go to Add-ons section of the Preferences.
2. Search “Manage UI Translations” and check the *Enable Extension* checkbox.

Description

See Blender translation guide in the Developer Handbook.

Reference

Category::	System
Description::	Allows managing UI translations directly from within Blender (update main po-files, update scripts’ translations, etc.).
Location::	*Topbar ▸ File menu*, Text editor, any UI control
File::	ui_translate folder
Author::	Bastien Montagne
Note::	This add-on is bundled with Blender.

Advanced

This chapter covers advanced use (topics which may not be required for typical usage).

- Using Blender From The Command Line
- Scripting & Extending Blender
- Creating Extensions
- Application Templates
- Keymap Customization
- Working Limits
- Operators
- Blender's Directory Layout
- Deploying Blender in Production
- Appendices

Using Blender From The Command Line

The *Console Window*, also called a *Terminal*, is an operating system text window that displays messages about Blender's operations, status, and internal errors.

When Blender is manually started from a terminal, Blender output is shown in the *Console Window* it is started from.

Use Cases:

- For rendering animation.
- For automation and batch processing which require launching Blender with different arguments.
- For Python development, to see the output of the `print()` function.
- If Blender exits unexpectedly, the messages may indicate the cause or error.
- When troubleshooting, to see the output of `--debug` messages.

See: Launching from the Command Line for specific instructions on launching Blender from the command line.

Tip

Closing the *Console Window* will also close Blender, losing any unsaved work.

Launching from the Command Line

- Linux
- macOS

- Windows

Arguments

- Render Options
- Cycles Render Options
- Format Options
- Animation Playback Options
- Window Options
- Python Options
- Network Options
- Logging Options
- Debug Options
- GPU Options
- Misc Options

Sub-Commands

- Extensions Command Line Arguments
  - Package Management
  - Repository Management
  - Extension Creation

Workflows

- Rendering From The Command Line
  - Single Image
  - Animation
  - Cycles

Launching from the Command Line

- Linux
  - Quick Start
  - Details
- macOS
  - Quick Start
  - Details
- Windows
  - Quick Start
  - Details

Linux

Quick Start

Open a terminal, then go to the directory where Blender is installed, and run Blender like this:

```
cd <blender installation directory>
./blender
```

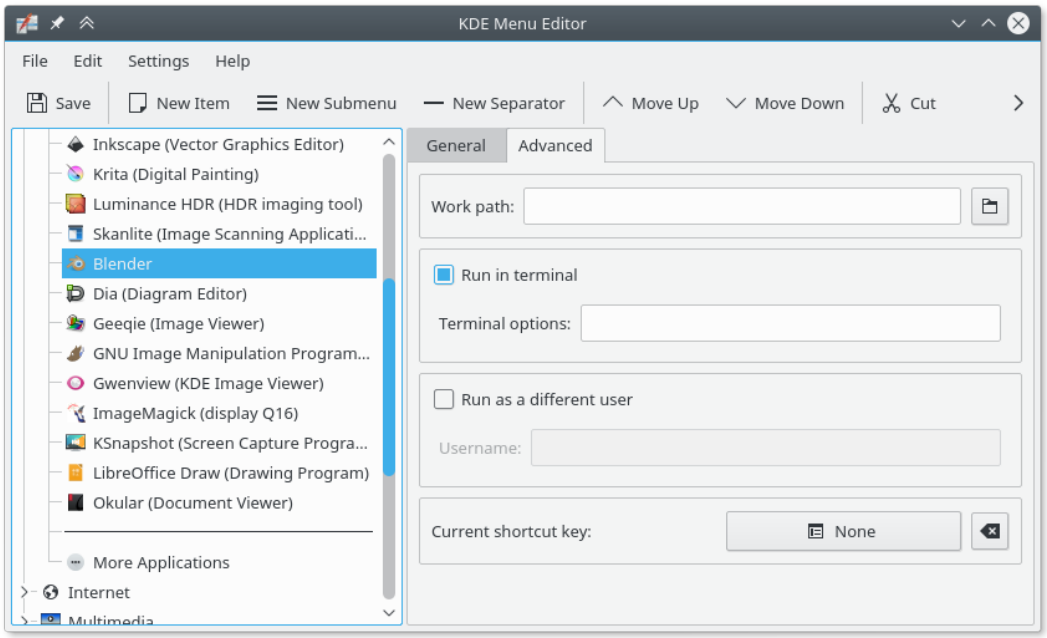
If you have Blender installed in your `PATH` (usually when Blender is installed through a distribution package), you can simply run:

```
blender
```

Details

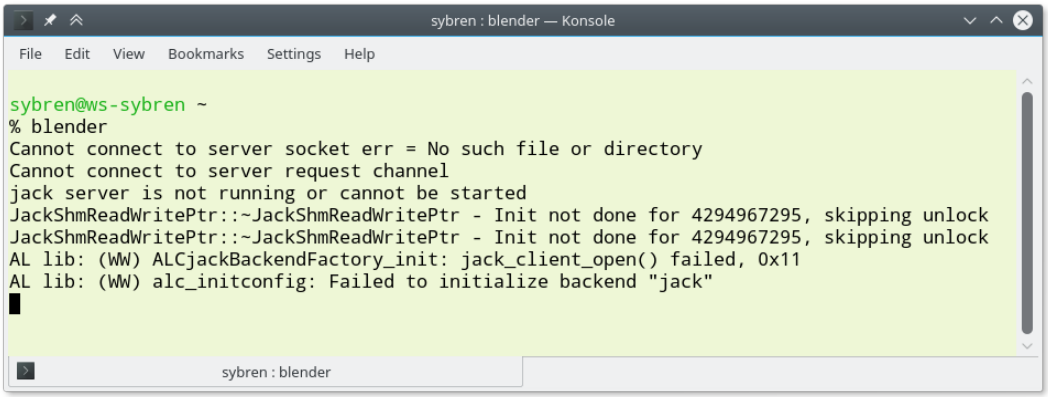
Depending on your desktop environment setup, a Blender icon may appear on your desktop or an entry for Blender is added to your menu after you install Blender.

Many desktop environments support the ability to “Run in terminal”.



Configuring the KDE menu icon to start Blender from a terminal.

This screenshot shows Blender started from a Linux Terminal and the resulting text output written to it:



Starting Blender from a Linux terminal window.

macOS

Quick Start

Open the terminal application, and run the executable within the app bundle, with commands like this:

```
cd /Applications/Blender.app/Contents/MacOS
./Blender
```

If you need to do this often, you can add this directory to your `PATH`.

For that you can run the following procedure:

1. Open up a Terminal.
2. Run the following command: `sudo nano /etc/paths`.
3. Enter your password, when prompted.
4. Go to the bottom of the file, and enter `/Applications/Blender.app/Contents/MacOS`.
5. Enter `Ctrl-X` to quit.
6. Enter `Y` to save the modified buffer.

If you then open a new terminal, the following command will work:

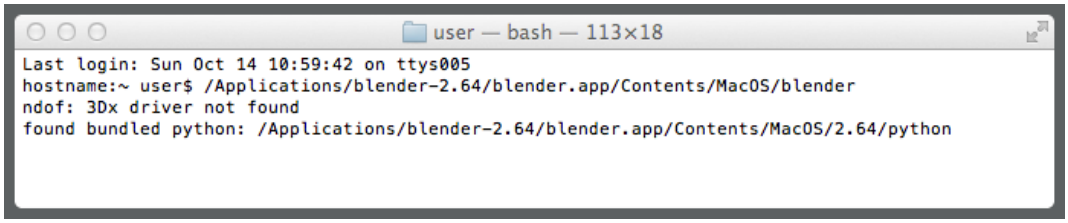
```
Blender
```

Details

macOS uses "files" with the `.app` extension called *applications*. These files are actually folders that appear as files in Finder. In order to run Blender you will have to specify that path to the Blender executable inside this folder, to get all output printed to the terminal. You can start a terminal from *Applications* ▸ *Utilities*. The path to the executable in the `.app` folder is `./Blender.app/Contents/MacOS/Blender`.

If you have Blender installed in the Applications folder, the following command can be used:

```
/Applications/Blender.app/Contents/MacOS/Blender
```



Starting Blender from a macOS console window.

Windows

Quick Start

Open the Command Prompt, go to the directory where Blender is installed, and then run Blender:

```
cd c:\<blender installation directory>
blender
```

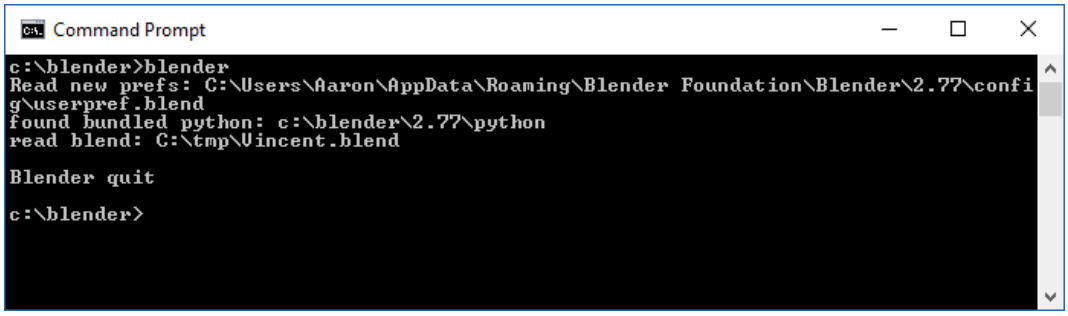
You can also add the Blender folder to your system `PATH` so that you do not have to `cd` to it each time.

Details

When Blender is started on a Microsoft Windows operating system, the *Console Window* (called the Command Prompt) is first created as a separate window on the desktop. The main Blender window will also appear and the *Console Window* will then be toggled off. To display the console again, go to *Window* ‣ *Toggle System Console*.

To start Blender from the command line you need to open an instance of Command Prompt. To do this, type `OSKey-R` then type `cmd`; this will open the Command Prompt window. You then need to find the path to the Blender executable. If you installed Blender via the installer then it can be found here:

```
C:\Program Files\Blender Foundation\Blender\blender.exe
```



Blender's Console Window on Microsoft Windows.

The screenshot shows the Blender *Console Window* on Microsoft Windows directly after starting Blender and then a short while later after opening a file along with the relevant messages.

Command Line Arguments

Blender 4.3
Usage: `blender [args ...] [file] [args ...]`

Render Options

`-b`, `--background`
Run in background (often used for UI-less rendering).

The audio device is disabled in background-mode by default and can be re-enabled by passing in `-setaudo Default` afterwards.

`-a`, `--render-anim`
Render frames from start to end (inclusive).

`-S`, `--scene` `<name>`
Set the active scene `<name>` for rendering.

`-f`, `--render-frame` `<frame>`
Render frame `<frame>` and save it.

- `+<frame>` start frame relative, `-<frame>` end frame relative.
- A comma separated list of frames can also be used (no spaces).
- A range of frames can be expressed using `..` separator between the first and last frames (inclusive).

`-s`, `--frame-start` `<frame>`
Set start to frame `<frame>`, supports +/- for relative frames too.

`-e`, `--frame-end` `<frame>`
Set end to frame `<frame>`, supports +/- for relative frames too.

`-j`, `--frame-jump` `<frames>`
Set number of frames to step forward after each rendered frame.

`-o`, `--render-output <path>`
: Set the render path and file name. Use `//` at the start of the path to render relative to the blend-file.

  The `#` characters are replaced by the frame number, and used to define zero padding.

  - `animation_##_test.png` becomes `animation_01_test.png`
  - `test-######.png` becomes `test-000001.png`

  When the filename does not contain `#`, the suffix `####` is added to the filename.

  The frame number will be added at the end of the filename, eg:

  ```
  blender -b animation.blend -o //render_ -F PNG -x 1 -a
  ```

  `//render_` becomes `//render_####`, writing frames as `//render_0001.png`

`-E`, `--engine <engine>`
: Specify the render engine. Use `-E help` to list available engines.

`-t`, `--threads <threads>`
: Use amount of `<threads>` for rendering and other operations [1-1024], 0 to use the systems processor count.

Cycles Render Options

Cycles add-on options must be specified following a double dash.

`--cycles-device <device>`
: Set the device used for rendering. Valid options are: `CPU` `CUDA` `OPTIX` `HIP` `ONEAPI` `METAL`.

  Append +CPU to a GPU device to render on both CPU and GPU.

  Example:

```
blender -b file.blend -f 20 -- --cycles-device OPTIX
```

`--cycles-print-stats`
: Log statistics about render memory and time usage.

Format Options

`-F`, `--render-format` `<format>`
: Set the render format. Valid options are: `TGA` `RAWTGA` `JPEG` `IRIS` `AVIRAW` `AVIJPEG` `PNG` `BMP` `HDR` `TIFF`.

  Formats that can be compiled into Blender, not available on all systems: `OPEN_EXR` `OPEN_EXR_MULTILAYER` `FFMPEG` `CINEON` `DPX` `JP2` `WEBP`.

`-x`, `--use-extension` `<bool>`
: Set option to add the file extension to the end of the file.

Animation Playback Options

`-a` `<options>` `<file(s)>`
: Instead of showing Blender's user interface, this runs Blender as an animation player, to view movies and image sequences rendered in Blender (ignored if `-b` is set).

  Playback Arguments:

  `-p` `<sx>` `<sy>`
  : Open with lower left corner at `<sx>`, `<sy>`.

  `-m`
  : Read from disk (Do not buffer).

  `-f` `<fps>` `<fps_base>`
  : Specify FPS to start with.

  `-j` `<frame>`
  : Set frame step to `<frame>`.

`-s <frame>`
: Play from `<frame>`.

`-e <frame>`
: Play until `<frame>`.

`-c <cache_memory>`
: Amount of memory in megabytes to allow for caching images during playback. Zero disables (clamping to a fixed number of frames instead).

Window Options

`-w`, `--window-border`
: Force opening with borders.

`-W`, `--window-fullscreen`
: Force opening in full-screen mode.

`-p`, `--window-geometry <sx> <sy> <w> <h>`
: Open with lower left corner at `<sx>`, `<sy>` and width and height as `<w>`, `<h>`.

`-M`, `--window-maximized`
: Force opening maximized.

`-con`, `--start-console`
: Start with the console window open (ignored if `-b` is set), (Windows only).

`--no-native-pixels`
: Do not use native pixel size, for high resolution displays (MacBook `Retina`).

`--no-window-focus`
: Open behind other windows and without taking focus.

Python Options

`-y`, `--enable-autoexec`
: Enable automatic Python script execution.

`-Y`, `--disable-autoexec`
: Disable automatic Python script execution (Python-drivers & startup scripts), (default).

`-P`, `--python` `<filepath>`
: Run the given Python script file.

`--python-text` `<name>`
: Run the given Python script text block.

`--python-expr` `<expression>`
: Run the given expression as a Python script.

`--python-console`
: Run Blender with an interactive console.

`--python-exit-code` `<code>`
: Set the exit-code in [0..255] to exit if a Python exception is raised (only for scripts executed from the command line), zero disables.

`--python-use-system-env`
: Allow Python to use system environment variables such as `PYTHONPATH` and the user site-packages directory.

`--addons` `<addon(s)>`
: Comma separated list (no spaces) of add-ons to enable in addition to any default add-ons.

Network Options

`--online-mode`
: Allow internet access, overriding the preference.

`--offline-mode`
: Disallow internet access, overriding the preference.

Logging Options

`--log <match>`
: Enable logging categories, taking a single comma separated argument. Multiple categories can be matched using a `.*` suffix, so `--log "wm.*"` logs every kind of window-manager message. Sub-string can be matched using a * prefix and suffix, so `--log "*undo*"` logs every kind of undo-related message. Use "^" prefix to ignore, so `--log "*,^wm.operator.*"` logs all except for `wm.operators.*` Use "*" to log everything.

`--log-level <level>`
: Set the logging verbosity level (higher for more details) defaults to 1, use -1 to log all levels.

`--log-show-basename`
: Only show file name in output (not the leading path).

`--log-show-backtrace`
: Show a back trace for each log message (debug builds only).

`--log-show-timestamp`
: Show a timestamp for each log message in seconds since start.

`--log-file <filepath>`
: Set a file to output the log to.

Debug Options

`-d`, `--debug`
: Turn debugging on.

  - Enables memory error detection
  - Disables mouse grab (to interact with a debugger in some cases)

- Keeps Python's `sys.stdin` rather than setting it to None

`--debug-value <value>`
: Set debug value of `<value>` on startup.

`--debug-events`
: Enable debug messages for the event system.

`--debug-ffmpeg`
: Enable debug messages from FFmpeg library.

`--debug-handlers`
: Enable debug messages for event handling.

`--debug-libmv`
: Enable debug messages from libmv library.

`--debug-cycles`
: Enable debug messages from Cycles.

`--debug-memory`
: Enable fully guarded memory allocation and debugging.

`--debug-jobs`
: Enable time profiling for background jobs.

`--debug-python`
: Enable debug messages for Python.

`--debug-depsgraph`
: Enable all debug messages from dependency graph.

`--debug-depsgraph-eval`
: Enable debug messages from dependency graph related on evaluation.

`--debug-depsgraph-build`
: Enable debug messages from dependency graph related on graph construction.

`--debug-depsgraph-tag`

Enable debug messages from dependency graph related on tagging.

`--debug-depsgraph-no-threads`
Switch dependency graph to a single threaded evaluation.

`--debug-depsgraph-time`
Enable debug messages from dependency graph related on timing.

`--debug-depsgraph-pretty`
Enable colors for dependency graph debug messages.

`--debug-depsgraph-uid`
Verify validness of session-wide identifiers assigned to ID data-blocks.

`--debug-ghost`
Enable debug messages for Ghost (Linux only).

`--debug-wintab`
Enable debug messages for Wintab.

`--debug-gpu`
Enable GPU debug context and information for OpenGL 4.3+.

`--debug-gpu-force-workarounds`
Enable workarounds for typical GPU issues and disable all GPU extensions.

`--debug-gpu-compile-shaders`
Compile all statically defined shaders to test platform compatibility.

`--debug-gpu-scope-capture`
Capture the GPU commands issued inside the give scope name.

`--debug-gpu-renderdoc`
Enable Renderdoc integration for GPU frame grabbing and debugging.

`--debug-wm`
Enable debug messages for the window manager, shows all operators in search, shows keymap errors.

`--debug-xr`
: Enable debug messages for virtual reality contexts. Enables the OpenXR API validation layer, (OpenXR) debug messages and general information prints.

`--debug-xr-time`
: Enable debug messages for virtual reality frame rendering times.

`--debug-all`
: Enable all debug messages.

`--debug-io`
: Enable debug messages for I/O (Collada, …).

`--debug-fpe`
: Enable floating-point exceptions.

`--debug-exit-on-error`
: Immediately exit when internal errors are detected.

`--debug-freestyle`
: Enable debug messages for Freestyle.

`--disable-crash-handler`
: Disable the crash handler.

`--disable-abort-handler`
: Disable the abort handler.

`--verbose <verbose>`
: Set the logging verbosity level for debug messages that support it.

`-q, --quiet`
: Suppress status printing (warnings & errors are still printed).

GPU Options

`--gpu-backend`

Force to use a specific GPU backend. Valid options: `vulkan` (experimental), `metal`, `opengl`.

`--gpu-compilation-subprocesses`
Override the Max Compilation Subprocesses setting (OpenGL only).

Misc Options

`--open-last`
Open the most recently opened blend file, instead of the default startup file.

`--app-template <template>`
Set the application template (matching the directory name), use `default` for none.

`--factory-startup`
Skip reading the `startup.blend` in the users home directory.

`--enable-event-simulate`
Enable event simulation testing feature `bpy.types.Window.event_simulate`.

`--env-system-datafiles`
Set the `BLENDER_SYSTEM_DATAFILES` environment variable.

`--env-system-scripts`
Set the `BLENDER_SYSTEM_SCRIPTS` environment variable.

`--env-system-extensions`
Set the `BLENDER_SYSTEM_EXTENSIONS` environment variable.

`--env-system-python`
Set the `BLENDER_SYSTEM_PYTHON` environment variable.

`-noaudio`
Force sound system to None.

`-setaudio`
: Force sound system to a specific device. `None` `Default` `SDL` `OpenAL` `CoreAudio` `JACK` `PulseAudio` `WASAPI`.

`-c`, `--command <command>`
: Run a command which consumes all remaining arguments. Use `-c help` to list all other commands. Pass `--help` after the command to see its help text.

  This implies `--background` mode.

`-h`, `--help`
: Print this help text and exit.

`/?`
: Print this help text and exit (Windows only).

`-r`, `--register`
: Register blend-file extension for current user, then exit (Windows & Linux only).

`--register-allusers`
: Register blend-file extension for all users, then exit (Windows & Linux only).

`--unregister`
: Unregister blend-file extension for current user, then exit (Windows & Linux only).

`--unregister-allusers`
: Unregister blend-file extension for all users, then exit (Windows & Linux only).

`-v`, `--version`
: Print Blender version and exit.

`--`

End option processing, following arguments passed unchanged. Access via Python's `sys.argv`.

Argument Parsing

Arguments must be separated by white space, eg:

```
blender -ba test.blend
```

…will exit since `-ba` is an unknown argument.

Argument Order

Arguments are executed in the order they are given. eg:

```
blender --background test.blend --render-frame 1 --render-
output "/tmp"
```

…will not render to `/tmp` because `--render-frame 1` renders before the output path is set.

```
blender --background --render-output /tmp test.blend --render-
frame 1
```

…will not render to `/tmp` because loading the blend-file overwrites the render output that was set.

```
blender --background test.blend --render-output /tmp --render-
frame 1
```

…works as expected.

Environment Variables

BLENDER_USER_RESOURCES::
: Replace default directory of all user files. Other `BLENDER_USER_*` variables override when set.

BLENDER_USER_CONFIG::
: Directory for user configuration files.

BLENDER_USER_SCRIPTS::
: Directory for user scripts.

BLENDER_USER_EXTENSIONS::
: Directory for user extensions.

BLENDER_USER_DATAFILES::
: Directory for user data files (icons, translations, ..).

BLENDER_SYSTEM_RESOURCES::
: Replace default directory of all bundled resource files.

BLENDER_SYSTEM_SCRIPTS::
: Directory to add more bundled scripts.

BLENDER_SYSTEM_EXTENSIONS::
: Directory for system extensions repository.

BLENDER_SYSTEM_DATAFILES::
: Directory to replace bundled datafiles.

BLENDER_SYSTEM_PYTHON::
: Directory to replace bundled Python libraries.

OCIO:: Path to override the OpenColorIO configuration file.

TEMP:: Store temporary files here (MS-Windows).

TMPDIR:: Store temporary files here (UNIX Systems). The path must reference an existing directory or it will be ignored.

Extensions Command Line Arguments

Command for managing Blender extensions.

options:

`-h, --help` show this help message and exit

subcommands:

Package Management

list::	List all packages.
sync::	Synchronize with remote repositories.
update::	Upgrade any outdated packages.
install::	Install packages.
install-file::	Install package from file.
remove::	Remove packages.

Repository Management

repo-list::	List repositories.
repo-add::	Add repository.
repo-remove::	Remove repository.

Extension Creation

build::	Build a package.
validate::	Validate a package.
server-generate::	Create a listing from all packages.

Package Management

Subcommand: `list`

usage:

```
blender --command extension list [-h] [-s]
```

List packages from all enabled repositories.

options:

`-h, --help`	show this help message and exit
`-s, --sync`	Sync the remote directory before performing the action.

Subcommand: `sync`

usage:

```
blender --command extension sync [-h]
```

Download package information for remote repositories.

options:

`-h, --help`	show this help message and exit

Subcommand: `update`

usage:

```
blender --command extension update [-h] [-s]
```

Download and update any outdated packages.

options:

`-h, --help`	show this help message and exit
`-s, --sync`	Sync the remote directory before performing the action.

Subcommand: `install`

usage:

```
blender --command extension install [-h] [-s] [-e] [--no-prefs]
                                    PACKAGES
```

positional arguments:

PACKAGES:: The packages to operate on (separated by `,` without spaces).

options:

`-h, --help`	show this help message and exit
`-s, --sync`	Sync the remote directory before performing the action.
`-e, --enable`	Enable the extension after installation.
`--no-prefs`	Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `install-file`

usage:

```
blender --command extension install-file [-h] -r REPO [-e] [--
no-prefs]
                                         FILE
```

Install a package file into a user repository.

positional arguments:

FILE:: The packages file.

options:

`-h, --help`	show this help message and exit
`-r` *`REPO`*`, --repo` *`REPO`*	The repository identifier.
`-e, --enable`	Enable the extension after installation.

`--no-prefs` Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `remove`

usage:

```
blender --command extension remove [-h] [--no-prefs] PACKAGES
```

Disable & remove package(s).

positional arguments:

PACKAGES:: The packages to operate on (separated by `,` without spaces).

options:

`-h, --help` show this help message and exit

`--no-prefs` Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Repository Management

Subcommand: `repo-list`

usage:

```
blender --command extension repo-list [-h]
```

List all repositories stored in Blender's preferences.

options:

`-h, --help` show this help message and exit

Subcommand: `repo-add`

usage:

```
blender --command extension repo-add [-h] [--name NAME]
                                     [--directory DIRECTORY]
                                     [--url URL]
                                     [--access-token
ACCESS_TOKEN]
                                     [--source SOURCE]
                                     [--cache BOOLEAN] [--
clear-all]
                                     [--no-prefs]
                                     ID
```

Add a new local or remote repository.

positional arguments:

ID:: The repository identifier.

options:

`-h, --help` show this help message and exit

`--name` *NAME* The name to display in the interface (optional).

`--directory` *DIRECTORY* The directory where the repository stores local files (optional). When omitted a directory in the users directory is automatically selected.

`--url` *URL* The URL, for remote repositories (optional). When omitted the repository is considered "local" as it is not connected to an external repository, where packages may be installed by file or managed manually.

`--access-token` *ACCESS_TOKEN*

	The access token to use for remote repositories which require a token.
`--source` *`SOURCE`*	The type of source in (‘USER’, ‘SYSTEM’). System repositories are managed outside of Blender and are considered read-only.
`--cache` *`BOOLEAN`*	Use package cache (default=1).
`--clear-all`	Clear all repositories before adding, simplifies test setup.
`--no-prefs`	Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Subcommand: `repo-remove`

usage:

```
blender --command extension repo-remove [-h] [--no-prefs] ID
```

Remove a repository.

positional arguments:

ID:: The repository identifier.

options:

`-h, --help`	show this help message and exit
`--no-prefs`	Treat the user-preferences as read-only, preventing updates for operations that would otherwise modify them. This means removing extensions or repositories for example, wont update the user-preferences.

Extension Creation

Subcommand: `build`

usage:

```
blender --command extension build [-h] [--source-dir
SOURCE_DIR]
                                  [--output-dir OUTPUT_DIR]
                                  [--output-filepath
OUTPUT_FILEPATH]
                                  [--valid-tags
VALID_TAGS_JSON]
                                  [--split-platforms] [--
verbose]
```

Build a package in the current directory.

options:

`-h, --help` show this help message and exit

`--source-dir` *`SOURCE_DIR`*
: The package source directory containing a `blender_manifest.toml` manifest.
Default's to the current directory.

`--output-dir` *`OUTPUT_DIR`*
: The package output directory.
Default's to the current directory.

`--output-filepath` *`OUTPUT_FILEPATH`*
: The package output filepath (should include a `.zip` extension).
Defaults to `{id}-{version}.zip` using values from the manifest.

`--valid-tags` *`VALID_TAGS_JSON`*
: Reference a file path containing valid tags lists.
If you wish to reference custom tags a `.json` file can be used. The contents must be a

dictionary of lists where the `key` matches the extension type.

For example:

```
{"add-ons": ["Example",
"Another"], "theme": ["Other",
"Tags"]}
```

To disable validating tags, pass in an empty path `--valid-tags=""`.

`--split-platforms` Build a separate package for each platform. Adding the platform as a file name suffix (before the extension).
This can be useful to reduce the upload size of packages that bundle large platform-specific modules (`*.whl` files).

`--verbose` Include verbose output.

Subcommand: `validate`

usage:

```
blender --command extension validate [-h]
                                     [--valid-tags
VALID_TAGS_JSON]
                                     [SOURCE_PATH]
```

Validate the package meta-data in the current directory.

positional arguments:

SOURCE_PATH:: The package source path (either directory containing package files or the package archive).
This path must containing a `blender_manifest.toml` manifest.
Defaults to the current directory.

options:

`-h, --help` show this help message and exit

`--valid-tags` *`VALID_TAGS_JSON`*
Reference a file path containing valid tags lists.
If you wish to reference custom tags a `.json` file can be used. The contents must be a dictionary of lists where the `key` matches the extension type.

For example:

```
{"add-ons": ["Example",
"Another"], "theme": ["Other",
"Tags"]}
```

To disable validating tags, pass in an empty path `--valid-tags=""`.

Subcommand: `server-generate`

usage:

```
blender --command extension server-generate [-h] --repo-dir
REPO_DIR
                                            [--repo-config
REPO_CONFIG]
                                            [--html]
                                            [--html-template
HTML_TEMPLATE_FILE]
```

Generate a listing of all packages stored in a directory. This can be used to host packages which only requires static-file hosting.

options:

`-h, --help` show this help message and exit

`--repo-dir` *`REPO_DIR`*
The remote repository directory.

`--repo-config` *`REPO_CONFIG`*

An optional server configuration to include information which can't be detected. Defaults to `blender_repo.toml` (in the repository directory).
This can be used to defined blocked extensions, for example

```
schema_version = "1.0.0"

[[blocklist]]
id = "my_example_package"
reason = "Explanation for why this extension was blocked"
[[blocklist]]
id = "other_extenison"
reason = "Another reason for why this is blocked"
```

`--html`
: Create a HTML file (`index.html`) as well as the repository JSON to support browsing extensions online with static-hosting.

`--html-template` *`HTML_TEMPLATE_FILE`*
: An optional HTML file path to override the default HTML template with your own.
The following keys will be replaced with generated contents:

  - `${body}` is replaced the extensions contents.
  - `${date}` is replaced the creation date.

Rendering From The Command Line

In some situations we want to increase the render speed, access Blender remotely to render something or build scripts that use the command line.

One advantage of using the command line is that we do not need a graphical display (no need for X server on Linux for example) and consequently we can render via a remote shell (typically SSH).

- See Command Line Arguments for a full list of arguments (for example to specify which scene to render, the end frame number, etc.), or simply run:

```
blender --help
```

- See Command Line Launching for specific instructions on launching Blender from the command line.

Note

Arguments are executed in the order they are given!

The following command will not work, since the output and extension are set after Blender is told to render:

```
blender -b file.blend -a -x 1 -o //render
```

The following command will behave as expected:

```
blender -b file.blend -x 1 -o //render -a
```

Always position `-f` or `-a` as the last arguments.

Single Image

```
blender -b file.blend -f 10
```

`-b`
: Render in the background (without UI).

`file.blend`
: Path to the blend-file to render.

`-f 10`
: Render only the 10th frame.

```
blender -b file.blend -o /project/renders/frame_##### -F
OPEN_EXR -f -2
```

`-o /project/renders/frame_#####`
: Path of where to save the rendered image, using five padded zeros for the frame number.

`-F OPEN_EXR`
: Override the image format specified in the blend-file and save to an OpenEXR image.

`-f -2`
: Render only the second last frame.

Warning

Arguments are case sensitive! `-F` and `-f` are not the same.

Animation

```
blender -b file.blend -a
```

`-a`

Render the whole animation using all the settings saved in the blend-file.

```
blender -b file.blend -E CYCLES -s 10 -e 500 -t 2 -a
```

`-E CYCLES`
: Use the "Cycles Render" engine. For a list of available render engines, run `blender -E help`.

`-s 10 -e 500`
: Set the start frame to `10` and the end frame to `500`.

`-t 2`
: Use only two threads.

Cycles

In addition to the options above, which apply to all render engines, Cycles has additional options to further control its behavior. See Cycles Render Options

Scripting & Extending Blender

- Introduction
  - General Information
  - Getting Started
  - Extending Blender
- Scripting & Security
  - Scripts in Blend-Files
  - Controlling Script Execution
- Add-on Tutorial
  - Intended Audience
  - Documentation Links
  - What is an Add-on?
  - Your First Add-on
  - Your Second Add-on
  - Conclusions

Introduction

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. Python combines remarkable power with very clear syntax.

Python scripts are a versatile way to extend Blender functionality. Most areas of Blender can be scripted, including animation, rendering, import and export, object creation and automating repetitive tasks.

To interact with Blender, scripts can make use of the tightly integrated API.

General Information

Links that are useful while writing scripts:

- Python.org – General information about Python.
- Blender Python API – Official API documentation. Use this for referencing while writing scripts.
- API Introduction – A short introduction to get you started with the API. Contains examples.

Links that deal with distributing your scripts:

- Sharing scripts – Information on how to share your scripts and get them included in the official Blender distribution.
- Creating Add-ons – Add-ons are used to encapsulate and distribute scripts.
- Add-ons project – Project to maintain a central repository of extensions to Blender.

Getting Started

Manual links

The following links take you from the basics to the more advanced concepts of Python scripting for Blender.

- Text Editor
- Python Console
- Info Editor

External links

Here are external links containing a lot of good information to start learning how to write scripts for Blender:

- Python API: Quickstart
- CG Cookie: Blender 2.8 Python Scripting Superpowers for Non-Programmers
- Olav3D Tutorials: 3D Programming for Beginners Using Python
- Blender Artists: Python Support Forum

Extending Blender

Add-ons

Add-ons are scripts that enable Blender to gain extra functionality; they can be enabled from the Preferences.

Outside of the Blender executable, there are hundreds of add-ons written by many people:

- Officially supported add-ons are bundled with Blender.
- Other add-ons are provided by Blender Extensions which aren't part of official releases. Many of them work reliably and are very useful but are not yet ensured to be stable for release.

See also

See Add-ons for documentation on add-ons included with Blender.

Scripts

Apart from add-ons, there are several other types of scripts that extend Blender's functionality:

Modules:: Utility libraries for import into other scripts.
Presets:: Settings for Blender's tools and key configurations.
Startup:: These files are imported when starting Blender. They define most of Blender's UI, as well as some additional core operators.
Custom Scripts:: In contrast to add-ons, they are typically intended for one-time execution via the Text Editor.

Saving your own Scripts

File Location

All scripts are loaded from the `scripts` folder of the local, system and user paths.

You can setup an additional search path for scripts in File Paths *Preferences* ▸ *File Paths*.

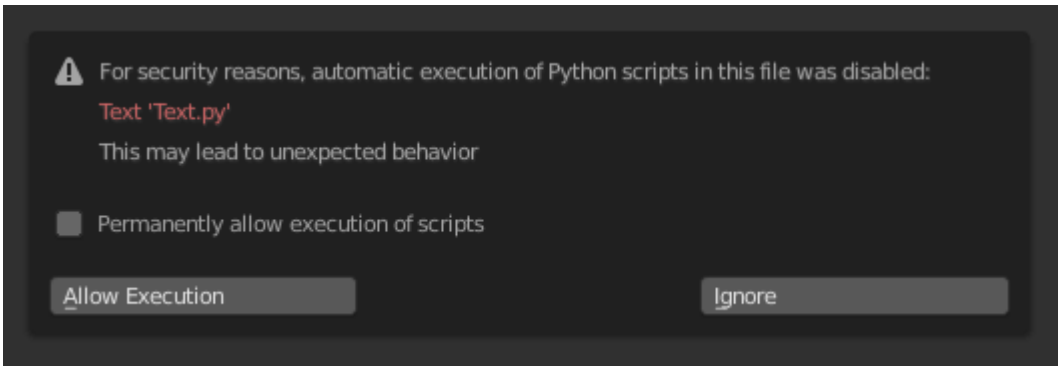
Installation

Add-ons are conveniently installed through Blender in the Preferences. Click the *Install…* button and select the `.zip` file.

Scripting & Security

The ability to include Python scripts within blend-files is valuable for advanced tasks such as rigging and automation. However, it poses a security risk since Python does not restrict what a script can do. Therefore, you should only run scripts from sources you know and trust. Automatic execution is disabled by default; however, some blend-files need this to function properly.

When a blend-file tries to execute a script and is not allowed, a dialog will appear. In it, you can choose to *Allow Execution* or to *Ignore* the scripts.



An Auto-run warning in the Info editor's header.

Scripts in Blend-Files

Auto Execution

Here are the different ways blend-files may automatically run scripts.

Registered Text-Blocks
: A text data-block can have its *Register* option enabled which means it will load on start.

Animation Drivers

Python expressions can be used to *Drive* values and are often used in more advanced rigs and animations.

Manual Execution

There are other ways scripts in a blend-file may execute that require user interaction (therefore will run even when auto execution is off), but you should be aware that this is the case since it is not necessarily obvious.

- Running a script in the Text editor.
- Rendering with Freestyle, because Freestyle uses scripts to control line styles.

Controlling Script Execution

Blender provides a number of ways to control whether scripts from a blend-file are allowed to automatically execute.

First, the File Browser has the option **Trusted Source** which you can use on a case-by-case basis to control auto execution. Since you may forget to set this, or may open a file without going through the File Browser, you can change the default (described next).

Setting Defaults

In the Preferences, there is the toggle to Auto Run Python Scripts. This means the **Trusted Source** option in the File Browser will be enabled by default, and scripts can run when blend-files are loaded without using the File Browser. Once enabled you have the option to exclude certain directories; a typical configuration would be to trust all paths except for the download directory.



The Auto Run Python Scripts checkbox.

Command Line

You may want to perform batch rendering or some other task from the command line, running Blender without an interface. In this case, the Preferences are still used but you may want to override them:

- Enable with `-y` or `--enable-autoexec`
- Disable with `-Y` or `--disable-autoexec`

Example

To render an animation in background mode, allowing drivers and other scripts to run:

```
blender --background --enable-autoexec my_movie.blend --render-anim
```

Note

These command-line arguments can be used to start a regular Blender instance and will still override the Preferences.

Add-on Tutorial

Intended Audience

This tutorial is designed to help technical artists or developers learn to extend Blender. An understanding of the basics of Python is expected for those working through this tutorial.

Prerequisites

Before going through the tutorial you should…

- Be familiar with the basics of working in Blender.
- Know how to run a script in Blender's Text editor.
- Have an understanding of Python primitive types (integer, Boolean, string, list, tuple, dictionary, and set).
- Be familiar with the concept of Python modules.
- Have a basic understanding of classes (object orientation) in Python.

Suggested reading before starting this tutorial.

- Dive Into Python sections (1, 2, 3, 4, and 7).
- Blender API Quickstart to help become familiar with Blender/Python basics.

To best troubleshoot any error message Python prints while writing scripts, you run Blender from a terminal. See Use The Terminal.

Tip

You can enable Developer Extras in the preferences to enable features that make developing add-ons easier.

Documentation Links

While going through the tutorial, you may want to look into our reference documentation.

- Blender API Overview: This document is rather detailed but helpful if you want to know more on a topic.
- `bpy.context` API reference – Handy to have a list of available items your script may operate on.
- `bpy.types.Operator` – The following add-ons define operators, these docs give details and more examples of operators.

What is an Add-on?

An add-on is simply a Python module with some additional requirements so Blender can display it in a list with useful information.

To give an example, here is the simplest possible add-on:

```
bl_info = {
    "name": "My Test Add-on",
    "blender": (2, 80, 0),
    "category": "Object",
}
def register():
    print("Hello World")
def unregister():
    print("Goodbye World")
```

`bl_info`
: is a dictionary containing add-on metadata such as the title, version and author to be displayed in the Preferences add-on list. It also specifies the minimum Blender version required to run the script; older versions won't display the add-on in the list.

`register`
: is a function which only runs when enabling the add-on, this means the module can be loaded without activating the add-on.

`unregister`
: is a function to unload anything setup by `register`, this is called when the add-on is disabled.

Notice this add-on does not do anything related to Blender (the **`blender_api:bpy`** module is not imported for example).

This is a contrived example of an add-on that serves to illustrate the point that the base requirements of an add-on are simple.

An add-on will typically register operators, panels, menu items, etc, but it's worth noting that *any* script can do this, when executed from the Text editor or even the interactive console – there is nothing inherently different about an add-on that allows it to integrate with Blender, such functionality is just provided by the **`blender_api:bpy`** module for any script to access.

So an add-on is just a way to encapsulate a Python module in a way a user can easily utilize.

Note

Running this script within the Text editor won't print anything, to see the output it must be installed through the Preferences. Messages will be printed when enabling and disabling.

Your First Add-on

The simplest possible add-on above is useful as an example but not much else. This next add-on is simple but shows how to integrate a script into Blender using an `Operator` which is the typical way to define a tool accessed from menus, buttons and keyboard shortcuts.

For the first example we will make a script that simply moves all objects in a scene.

Write the Script

Add the following script to the Text editor in Blender:

```
import bpy

scene = bpy.context.scene
for obj in scene.objects:
    obj.location.x += 1.0
```

Click the Run Script button, all objects in the active scene are moved by 1.0 unit.

Write the Add-on (Simple)

This add-on takes the body of the script above, and adds it to an operator's `execute()` function.

```
bl_info = {
    "name": "Move X Axis",
    "blender": (2, 80, 0),
    "category": "Object",
}

import bpy


class ObjectMoveX(bpy.types.Operator):
    """My Object Moving Script"""      # Use this as a tooltip
for menu items and buttons.
    bl_idname = "object.move_x"        # Unique identifier for
buttons and menu items to reference.
    bl_label = "Move X by One"         # Display name in the
interface.
    bl_options = {'REGISTER', 'UNDO'}  # Enable undo for the
operator.

    def execute(self, context):        # execute() is called
when running the operator.

        # The original script
        scene = context.scene
        for obj in scene.objects:
```

```
            obj.location.x += 1.0

        return {'FINISHED'}            # Lets Blender know the
operator finished successfully.

def menu_func(self, context):
    self.layout.operator(ObjectMoveX.bl_idname)

def register():
    bpy.utils.register_class(ObjectMoveX)
    bpy.types.VIEW3D_MT_object.append(menu_func)  # Adds the
new operator to an existing menu.

def unregister():
    bpy.utils.unregister_class(ObjectMoveX)


# This allows you to run the script directly from Blender's
Text editor
# to test the add-on without having to install it.
if __name__ == "__main__":
    register()
```

Note

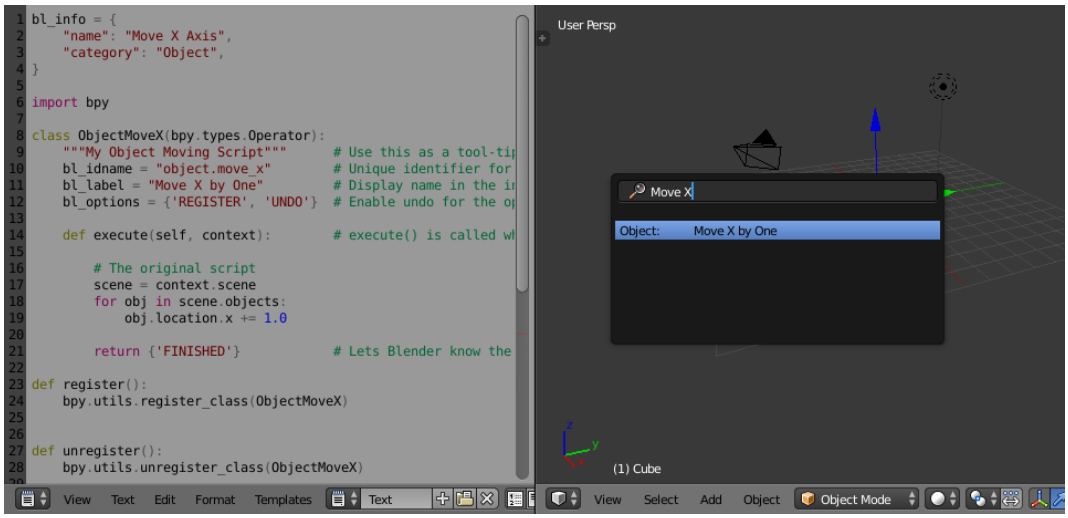
`bl_info` is split across multiple lines, this is just a style convention used to more easily add items.

Note

Rather than using `bpy.context.scene`, we use the `context.scene` argument passed to `execute()`. In most cases these will be the same. However in some cases, operators will be passed a custom context so script authors should prefer the `context` argument passed to operators.

To test the script, you can copy and paste it into Blender's Text editor and run it. This will execute the script directly and call register immediately.

However running the script won't move any objects. For this, you need to execute the newly registered operator.



Operator Search menu.

Open the Operator Search menu and type in "Move X by One" (the `bl_label`), then `Return`.

The objects should move as before.

Keep this add-on open in Blender for the next step - Installing.

Install the Add-on

Once you have your add-on within in Blender's Text editor, you will want to be able to install it so it can be enabled in the Preferences to load on startup.

Even though the add-on above is a test, let's go through the steps anyway so you know how to do it for later.

To install the Blender text as an add-on, you will first have to save it on drive. Take care to obey the naming restrictions that apply to Python modules and end with a `.py` extension.

Once the file is on drive, you can install it as you would for an add-on downloaded online.

Open the *Preferences* ‣ *Add-ons* ‣ *Install…* and select the file.

Now the add-on will be listed and you can enable it by pressing the checkbox, if you want it to be enabled on restart, press *Save as Default*. The operator can be run in the same way as described in the previous section.

When the add-on is enabled, Blender executes the code and runs the `register()` function. When the add-on is disabled, Blender runs the `unregister()` function.

Note

The destination of the add-on depends on your Blender configuration. When installing an add-on the source and destination paths are printed in the console. You can also find add-on path locations by running this in the Python Console:

```
import addon_utils
print(addon_utils.paths())
```

More is written on this topic here: Directory Layout.

Your Second Add-on

For our second add-on, we will focus on object instancing – this is – to make linked copies of an object in a similar way to what you may have seen with the Array modifier.

Write the Script

As before, first we will start with a script, develop it, then convert it into an add-on.

```
import bpy
from bpy import context

# Get the current scene
scene = context.scene

# Get the 3D cursor location
cursor = scene.cursor.location

# Get the active object (assume we have one)
obj = context.active_object

# Now make a copy of the object
obj_new = obj.copy()

# The new object has to be added to a collection in the scene
scene.collection.objects.link(obj_new)

# Now we can place the object
obj_new.location = cursor
```

Now try copying this script into Blender and run it on the default Cube. Make sure you click to move the 3D cursor before running as the duplicate will appear at the cursor's location.

After running, notice that when you go into *Edit Mode* to change the Cube – all of the copies change. In Blender, this is known as *Linked Duplicates*.

Next, we're going to do this in a loop, to make an array of objects between the active object and the cursor.

```
import bpy
from bpy import context

scene = context.scene
cursor = scene.cursor.location
obj = context.active_object

# Use a fixed value for now, eventually make this user
adjustable
total = 10
```

```
# Add 'total' objects into the scene
for i in range(total):
    obj_new = obj.copy()
    scene.collection.objects.link(obj_new)

    # Now place the object in between the cursor
    # and the active object based on 'i'
    factor = i / total
    obj_new.location = (obj.location * factor) + (cursor * (1.0
- factor))
```

Try running this script with the active object and the cursor spaced apart to see the result.

With this script you'll notice we're doing some math with the object location and cursor, this works because both are 3D **mathutils.Vector** instances, a convenient class provided by the **mathutils** module which allows vectors to be multiplied by numbers and matrices.

If you are interested in this area, read into **mathutils.Vector** – there are many handy utility functions such as getting the angle between vectors, cross product, dot products as well as more advanced functions in **mathutils.geometry** such as Bézier spline interpolation and ray-triangle intersection.

For now we will focus on making this script an add-on, but it's good to know that this 3D math module is available and can help you with more advanced functionality later on.

Write the Add-on

The first step is to convert the script as-is into an add-on:

```
bl_info = {
    "name": "Cursor Array",
    "blender": (2, 80, 0),
    "category": "Object",
}

import bpy
```

```
class ObjectCursorArray(bpy.types.Operator):
    """Object Cursor Array"""
    bl_idname = "object.cursor_array"
    bl_label = "Cursor Array"
    bl_options = {'REGISTER', 'UNDO'}

    def execute(self, context):
        scene = context.scene
        cursor = scene.cursor.location
        obj = context.active_object

        total = 10

        for i in range(total):
            obj_new = obj.copy()
            scene.collection.objects.link(obj_new)

            factor = i / total
            obj_new.location = (obj.location * factor) +
(cursor * (1.0 - factor))

        return {'FINISHED'}

def register():
    bpy.utils.register_class(ObjectCursorArray)


def unregister():
    bpy.utils.unregister_class(ObjectCursorArray)


if __name__ == "__main__":
    register()
```

Everything here has been covered in the previous steps, you may want to try run the add-on still and consider what could be done to make it more useful.

The two of the most obvious missing things are – having the total fixed at 10, and having to access the operator with Operator Search is not very

convenient.

Both these additions are explained next, with the final script afterwards.

Operator Property

There are a variety of property types that are used for tool settings, common property types include: int, float, vector, color, Boolean and string.

These properties are handled differently to typical Python class attributes because Blender needs to display them in the interface, store their settings in keymaps and keep settings for reuse.

While this is handled in a fairly Pythonic way, be mindful that you are in fact defining tool settings that are loaded into Blender and accessed by other parts of Blender, outside of Python.

To get rid of the literal 10 for `total`, we'll use an operator property. Operator properties are defined via bpy.props module, this is added to the class body:

```
# moved assignment from execute() to the body of the class...
total: bpy.props.IntProperty(name="Steps", default=2, min=1,
max=100)

# and this is accessed on the class
# instance within the execute() function as...
self.total
```

These properties from **bpy.props** are handled specially by Blender when the class is registered so they display as buttons in the user interface. There are many arguments you can pass to properties to set limits, change the default and display a tooltip.

See also

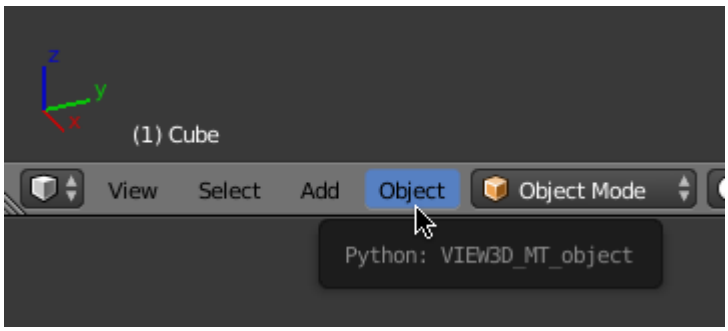
bpy.props.IntProperty()

This document doesn't go into details about using other property types. However, the link above includes examples of more advanced property usage.

Menu Item

Add-ons can add to the user interface of existing panels, headers and menus defined in Python.

For this example we'll add to an existing menu.



Menu Identifier.

To find the identifier of a menu, first enable *Python Tooltips* in the preferences. Then you can hover your mouse over the menu item and the identifier is displayed.

The method used for adding a menu item is to append a draw function into an existing class:

```
def menu_func(self, context):
    self.layout.operator(ObjectCursorArray.bl_idname)

def register():
    bpy.utils.register_class(ObjectCursorArray)
    bpy.types.VIEW3D_MT_object.append(menu_func)
```

For docs on extending menus, see: **bpy.types.Menu**.

Keymap

In Blender, add-ons have their own keymaps so as not to interfere with Blender's built-in keymaps.

In the example below, a new object mode **bpy.types.KeyMap** is added, then a **bpy.types.KeyMapItem** is added to the keymap which references our newly added operator, using `Shift-Ctrl-T` as the key shortcut to activate it.

```
# store keymaps here to access after registration
addon_keymaps = []

def register():

    # handle the keymap
    wm = bpy.context.window_manager
    km = wm.keyconfigs.addon.keymaps.new(name='Object Mode',
space_type='EMPTY')

    kmi = km.keymap_items.new(ObjectCursorArray.bl_idname, 'T',
'PRESS', ctrl=True, shift=True)
    kmi.properties.total = 4

    addon_keymaps.append((km, kmi))


def unregister():

    # handle the keymap
    for km, kmi in addon_keymaps:
        km.keymap_items.remove(kmi)
    addon_keymaps.clear()
```

Notice how the keymap item can have a `total` setting different than the default set by the operator, this allows you to have multiple keys accessing the same operator with different settings.

Note

While `Shift-Ctrl-T` is not a default Blender key shortcut, it is hard to make sure add-ons will not overwrite each other's keymaps. Thus at least

take care when assigning keys that they do not conflict with important functionality of Blender (see also is key free add-on).

For API documentation on the functions listed above, see:

- **bpy.types.KeyMaps.new()**
- **bpy.types.KeyMap**
- **bpy.types.KeyMapItems.new()**
- **bpy.types.KeyMapItem**

Bringing It All Together

```
bl_info = {
    "name": "Cursor Array",
    "blender": (2, 80, 0),
    "category": "Object",
}

import bpy


class ObjectCursorArray(bpy.types.Operator):
    """Object Cursor Array"""
    bl_idname = "object.cursor_array"
    bl_label = "Cursor Array"
    bl_options = {'REGISTER', 'UNDO'}

    total: bpy.props.IntProperty(name="Steps", default=2,
min=1, max=100)

    def execute(self, context):
        scene = context.scene
        cursor = scene.cursor.location
        obj = context.active_object

        for i in range(self.total):
            obj_new = obj.copy()
            scene.collection.objects.link(obj_new)
```

```
            factor = i / self.total
            obj_new.location = (obj.location * factor) +
(cursor * (1.0 - factor))

        return {'FINISHED'}


def menu_func(self, context):
    self.layout.operator(ObjectCursorArray.bl_idname)

# store keymaps here to access after registration
addon_keymaps = []

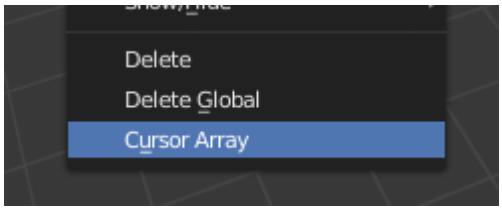

def register():
    bpy.utils.register_class(ObjectCursorArray)
    bpy.types.VIEW3D_MT_object.append(menu_func)

    # handle the keymap
    wm = bpy.context.window_manager
    # Note that in background mode (no GUI available),
keyconfigs are not available either,
    # so we have to check this to avoid nasty errors in
background case.
    kc = wm.keyconfigs.addon
    if kc:
        km = wm.keyconfigs.addon.keymaps.new(name='Object
Mode', space_type='EMPTY')
        kmi = km.keymap_items.new(ObjectCursorArray.bl_idname,
'T', 'PRESS', ctrl=True, shift=True)
        kmi.properties.total = 4
        addon_keymaps.append((km, kmi))

def unregister():
    # Note: when unregistering, it's usually good practice to
do it in reverse order you registered.
    # Can avoid strange issues like keymap still referring to
operators already unregistered...
    # handle the keymap
    for km, kmi in addon_keymaps:
        km.keymap_items.remove(kmi)
    addon_keymaps.clear()
```

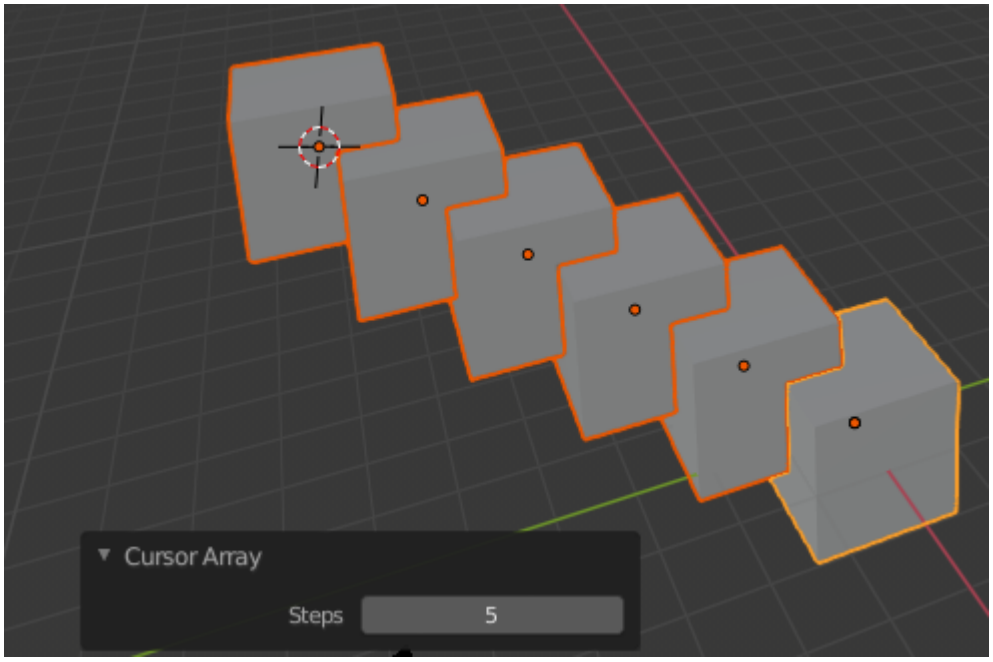
```
    bpy.utils.unregister_class(ObjectCursorArray)
    bpy.types.VIEW3D_MT_object.remove(menu_func)


if __name__ == "__main__":
    register()
```



In the menu.

Run the script (or save it and add it through the Preferences like before) and it will appear in the *Object* menu.



Operator Property.

After selecting it from the menu, you can choose how many instances of the cube you want create.

Note

Directly executing the script multiple times will add the menu each time too. While not useful behavior, there is nothing to worry about since add-ons will not register themselves multiple times when enabled through the Preferences.

Conclusions

Add-ons can encapsulate certain functionality neatly for writing tools to improve your workflow or for writing utilities for others to use.

While there are limits to what Python can do within Blender, there is certainly a lot that can be achieved without having to dive into Blender's C/C++ code.

The example given in the tutorial is limited, but shows the Blender API used for common tasks that you can expand on to write your own tools.

Further Reading

Blender comes with commented templates which are accessible from the Text editor's header. If you have specific areas you want to see example code for, this is a good place to start.

Here are some sites you might like to check on after completing this tutorial.

- Blender/Python API Overview – For more background details on Blender/Python integration.
- How to Think Like a Computer Scientist – Great info for those who are still learning Python.
- Blender Development – Blender Development, general information and helpful links.
- Blender Developer Forum – Forum where people ask Python development questions.

Creating Extensions

Extensions are **add-ons** or **themes** used to extend the core functionality of Blender. They are shared in online platforms, and can be installed and updated from within Blender.

The official extensions platform for the Blender project is extensions.blender.org. Other third party sites can also be supported, as long as they follow the Extensions Platform specification.

- Getting started
- Compatible licenses
- Supported tags
- Add-ons
- Python Wheels
- Creating a Repository

See also

For extension add-on guidelines (requirements for `extensions.blender.org`), refer to the Developer Handbook.

For the extension settings, and how to manage them, refer to the User Preferences.

For managing extensions from the command line, refer to Extension Command Line Arguments.

How to Create Extensions

Creating an extension takes only a few steps:

1. Open the directory containing the add-on code or theme file.
2. Add a blender_manifest.toml file with all the required meta-data (`name, maintainer, ...`).
3. Use the Blender command-line tool to build the extension .zip file.

How to publish to the Blender Extensions Platform:

- Install from Disk to test if everything is working well.
- Upload the .zip file (this step requires Blender ID).

The extension will be held for review, and published once the moderation team approves it.

Extension files

An extension is shared as a `.zip` archive containing a manifest file and other files. The expected files depend on the extension type.

Add-on extension

Add-ons need at least the manifest and an `__init__.py` file, while more complex add-ons have a few different .py files or wheels together.

```
my_extension-0.0.1.zip
├─ __init__.py
├─ blender_manifest.toml
└─ (...)
```

Theme extension

A theme extension only needs the manifest and the .xml theme file.

```
my_extension-0.0.1.zip
├─ blender_manifest.toml
└─ theme.xml
```

Note

Extensions can optionally have all its files inside a folder (inside the archive). This is a common behavior when saving a repository as ZIP from version-control platforms.

Manifest

A manifest is a file with all the meta-data required for an extension to be processed. This example is a good starting point to the `blender_manifest.toml` that should be inside the `.zip`.

```
schema_version = "1.0.0"

# Example of manifest file for a Blender extension
# Change the values according to your extension
id = "my_example_extension"
version = "1.0.0"
name = "My Example Extension"
tagline = "This is another extension"
maintainer = "Developer name <email@address.com>"
# Supported types: "add-on", "theme"
type = "add-on"

# # Optional: link to documentation, support, source files, etc
# website = "https://extensions.blender.org/add-ons/my-example-
package/"

# # Optional: tag list defined by Blender and server, see:
# #
https://docs.blender.org/manual/en/dev/advanced/extensions/tags
.html
# tags = ["Animation", "Sequencer"]

blender_version_min = "4.2.0"
```

```
# # Optional: Blender version that the extension does not
support, earlier versions are supported.
# # This can be omitted and defined later on the extensions
platform if an issue is found.
# blender_version_max = "5.1.0"

# License conforming to https://spdx.org/licenses/ (use "SPDX:
prefix)
#
https://docs.blender.org/manual/en/dev/advanced/extensions/lice
nses.html
license = [
  "SPDX:GPL-3.0-or-later",
]
# # Optional: required by some licenses.
# copyright = [
#   "2002-2024 Developer Name",
#   "1998 Company Name",
# ]

# # Optional: list of supported platforms. If omitted, the
extension will be available in all operating systems.
# platforms = ["windows-x64", "macos-arm64", "linux-x64"]
# # Other supported platforms: "windows-arm64", "macos-x64"

# # Optional: bundle 3rd party Python modules.
# #
https://docs.blender.org/manual/en/dev/advanced/extensions/pyth
on_wheels.html
# wheels = [
#   "./wheels/hexdump-3.3-py3-none-any.whl",
#   "./wheels/jsmin-3.0.1-py3-none-any.whl",
# ]

# # Optional: add-ons can list which resources they will
require:
# # * files (for access of any filesystem operations)
# # * network (for internet access)
# # * clipboard (to read and/or write the system clipboard)
# # * camera (to capture photos and videos)
# # * microphone (to capture audio)
# #
# # If using network, remember to also check
```

```
`bpy.app.online_access`
# #
https://docs.blender.org/manual/en/dev/advanced/extensions/addo
ns.html#internet-access
# #
# # For each permission it is important to also specify the
reason why it is required.
# # Keep this a single short sentence without a period (.) at
the end.
# # For longer explanations use the documentation or detail
page.
#
# [permissions]
# network = "Need to sync motion-capture data to server"
# files = "Import/export FBX from/to disk"
# clipboard = "Copy and paste bone transforms"

# # Optional: advanced build settings.
# #
https://docs.blender.org/manual/en/dev/advanced/extensions/comm
and_line_arguments.html#command-line-args-extension-build
# [build]
# # These are the default build excluded patterns.
# # You only need to edit them if you want different options.
# paths_exclude_pattern = [
#   "__pycache__/",
#   "/.git/",
#   "/*.zip",
# ]
```

Required values:

blender_version_min::
: Minimum supported Blender version - use at least `4.2.0`.

id::
: Unique identifier for the extension.

license::
: List of licenses, use SPDX license identifier.

maintainer::
: Maintainer of the extension.

name::
: Complete name of the extension.

schema_version::
: Internal version of the file format - use `1.0.0`.

tagline::

One-line short description, up to 64 characters - cannot end with punctuation.

type:: "add-on", "theme".

version:: Version of the extension - must follow semantic versioning.

Optional values:

blender_version_max:: Blender version that the extension does not support, earlier versions are supported.

website:: Website for the extension.

copyright:: Some licenses require a copyright, copyrights must be "Year Name" or "Year-Year Name".

tags:: List of tags. See the list of available tags.

platforms:: List of supported platforms. If omitted, the extension will be available in all operating systems. The available options are ["windows-x64", "windows-arm64", "macos-x64", "macos-arm64", "linux-x64"]

wheels:: List of relative file-paths Python Wheels.

permissions:: Add-ons can list which resources they require. The available options are *files*, *network*, *clipboard*, *camera*, *microphone*. Each permission should be followed by an explanation (short single-sentence, up to 64 characters, with no end punctuation).

Optional values for "build":

These values are only used by the build sub-command.

paths:: A list of file-paths relative to the manifest to include when building the package.

paths_exclude_pattern:: A list of file-path patterns to exclude include when building the package.

The pattern matching is compatible with gitignore.
Note that setting this value isn't supported when `paths` is also declared.

If the `[build]` table isn't declared the following default is used:

```
[build]
paths_exclude_pattern = [
  "__pycache__/",
  ".*",
  "*.zip",
]
```

Reserved:

These values **must not** be declared in a TOML and are reserved for internal use.

- `[build.generated]`

Note

All the values present in the manifest file must be filled (i.e., cannot be empty, nor text `""`, nor list `[]`).

If you don't want to set one of the optional values just exclude it from the manifest altogether.

Command-line

Extensions can be built, validated & installed via command-line.

To build the package defined in the current directory use the following commands:

```
blender --command extension build
```

See build docs.

—

To validate the manifest without building the package:

```
blender --command extension validate
```

You may also validate a package without having to extract it first.

```
blender --command extension validate add-on-package.zip
```

See validate docs.

See also

Extensions Command Line Arguments.

Third party extension sites

If you want to host the extensions yourself, see the Creating an Extensions Repository docs.

Extension Licenses

The Blender Extensions Platform only supports free and open source extensions compliant with the Blender's license:

- For add-ons, the required license is GNU General Public License v3.0 or later.
- For themes, the recommended license is GNU General Public License v3.0 or later, but any GPL-compatible license is supported.
- For assets used in add-ons, the required license is Public Domain (CC0).

Extensions Tags

A different set of tags is available for the different extensions types. This is the list of the tags currently supported by the Blender Extensions Platform:

Add-ons

- 3D View
- Add Curve
- Add Mesh
- Animation
- Bake
- Camera
- Compositing
- Development
- Game Engine
- Geometry Nodes
- Grease Pencil
- Import-Export
- Lighting
- Material
- Modeling
- Mesh
- Node
- Object
- Paint
- Pipeline
- Physics
- Render
- Rigging
- Scene
- Sculpt
- Sequencer
- System

- Text Editor
- Tracking
- User Interface
- UV

Themes

- Dark
- Light
- Colorful
- Inspired By
- Print
- Accessibility
- High Contrast

Add-ons

Add-ons let you extend Blender's functionality using Python. Most of the time you can get add-ons as part of the Extensions system.

Tip

If the Add-on does not activate when enabled, check the Console window for any errors that may have occurred.

Internet Access

If the add-on needs to use internet, it must check for the (read-only) property `bpy.app.online_access`. This option is controlled by Preferences, which can be overriding via command-line (`--offline-mode` / `--online-mode`).

For better error messages, you can check also for `bpy.app.online_access_overriden`, to determine whether users can turn Allow Online Access on the preferences, or not.

Note

Add-ons that follow this setting will only connect to the internet if enabled. However, Blender cannot prevent third-party add-ons from violating this rule.

Bundle Dependencies

For add-ons to be bundled as extensions, they must be self-contained. That means they must come with all its dependencies. In particular 3rd party

Python modules.

There are a few options for this:

Bundle with Python Wheels.
: This is the recommended way to bundle dependencies.

Bundle other add-ons together.
: This is recommended if an add-on depends on another add-on.

  Make sure that both the individual and the combined add-on check for already registered types (Operators, Panels, …). This avoids duplication of operators and panels on the interface if the add-ons are installed as a bundle and individually.

Bundle with Vendorize
: This can be used as a way to bundle a pure Python dependencies as a sub-module.

  This has the advantage of avoiding version conflicts although it requires some work to setup each package.

Local Storage

Add-ons must not assume their own directory is user writable since this may not be the case for "System" repositories. Writing files into the add-on's directory also has the down side that upgrading the extension will remove all files.

Add-ons which need their own user directory should use a utility function provided for this purpose:

```
extension_directory =
bpy.utils.extension_path_user(__package__, path="",
create=True)
```

If you wish create subdirectories, this can be done with the `path` argument.

This directory will be kept between upgrades but will be removed if the extension is uninstalled.

Legacy vs Extension Add-ons

With the introduction of Extensions in Blender 4.2, the old way of creating add-ons is considered deprecated. While the changes are rather small they impact existing add-ons.

In order to allow a smooth transition process, the so-called legacy add-ons will continue to be supported by Blender. They can be installed via Install legacy Add-on button in the User Preferences.

All add-on maintainers are urged to convert the add-ons they want to share, so they are future proof and can support features like updating from the extensions platform.

Converting a Legacy Add-on into an Extension

1. Create a manifest file.
2. Remove the `bl_info` information (this is now in the manifest).
3. Replace all references to the module name to `__package__`.
4. Make all module imports to use relative import.
5. Use wheels to pack your external Python dependencies.
6. Remember to test it thoroughly.

Note

For testing it is important to install the extension from disk and check if everything is working well. This will get you as close to the final experience as possible.

Extensions and Namespace

The legacy add-ons would use their module name to access the preferences. This could lead to a name clash when extensions with the same name (from different repositories) would be installed. To prevent this conflict, the repository name is now part of the namespace.

For example, now instead of `kitsu` the module name would be `bl_ext.{repository_module_name}.kitsu` instead.

This has a few implications for preferences and module imports.

User Preferences and `__package__`

Add-ons can define their own preferences which use the package name as an identifier. This can be accessed using `__package__`.

This was already supported in the legacy add-ons, but not reinforced. As such this can break backward compatibility.

Before:

```
class KitsuPreferences(bpy.types.AddonPreferences):
    bl_idname = "kitsu"
    # ... snip ...

# Access with:
addon_prefs = bpy.context.preferences.addons["kitsu"]
```

Now:

```
class KitsuPreferences(bpy.types.AddonPreferences):
    bl_idname = __package__
    # ... snip ...

# Access with:
addon_prefs = bpy.context.preferences.addons[__package__]
```

Sub-packages
: An add-on that defines sub-packages (sub-directories with their own `__init__.py` file) that needs to use this identifier will have to import the

top-level package using a relative import.

```
from .. import __package__ as base_package
```

Then `base_package` can be used instead of `__package__`. The `..` imports relative to the packages parent, sub-sub-packages must use `...` and so on.

Note

- The value of `__package__` will vary between systems so it should never be replaced with a literal string.
- Extensions must not manipulate the value of `__package__` as this may cause unexpected behavior or errors.

Relative Imports

before::	`from kitsu import utils`
now::	`from . import utils`

Importing packages within the add-on module need to use relative paths. This is a standard Python feature and only applicable for add-ons that have multiple folders.

This was already supported in the legacy add-ons, but not reinforced. As such this can break backward compatibility.

Python Wheels

Python wheels (`*.whl`) are the standard way of distributing Python modules. They are supported in Blender to make self-contained Python Extensions.

Guidelines

- By convention, always locate the files under `./wheels/`.

Requirements

- Wheels must be bundled unmodified from Python's package index.
- Wheels must include their dependencies.
- Wheels filenames must match Python's binary distribution specification: see docs. *Wheels downloaded from Python's package index will follow this convention.*
- Use forward slashes as path separators when listing them on the manifest.

How to Bundle Wheels

Python wheels (`*.whl`) can be bundled using the following steps.

Downloading Wheels

Download the wheel to the directory `./wheels/`.

For wheels that are platform independent this example downloads `jsmin`:

```
pip wheel jsmin -w ./wheels
```

For wheels that contain binary compiled files, wheels for all supported platforms should be included:

This example downloads `pillow` - the popular image manipulation module.

```
pip download pillow --dest ./wheels --only-binary=:all: --
python-version=3.11 --platform=macosx_11_0_arm64
pip download pillow --dest ./wheels --only-binary=:all: --
python-version=3.11 --platform=manylinux_2_28_x86_64
pip download pillow --dest ./wheels --only-binary=:all: --
python-version=3.11 --platform=win_amd64
```

The available platform identifiers are listed on pillow's download page.

Update the Manifest

In `blender_manifest.toml` include the wheels as a list of paths, e.g.

```
wheels = [
   "./wheels/pillow-10.3.0-cp311-cp311-
macosx_11_0_arm64.whl",
   "./wheels/pillow-10.3.0-cp311-cp311-
manylinux_2_28_x86_64.whl",
   "./wheels/pillow-10.3.0-cp311-cp311-win_amd64.whl",
]
```

Now installing the package will extract the wheel into the extensions own `site-packages` directory.

Running

Once the extension has been installed you can check the module is being loaded by importing it in the Python console and printing it's location:

```
import PIL
print(PIL.__file__)
```

Platform Builds

Wheels can severely impact the size of an extension. To mitigate this, it is possible to build different extension zip files for each unique required platform.

For this you need to use the `--split-platforms` option from the build command.

```
blender --command extension build --split-platforms
```

Example

Manifest file excerpt:

```
id = "my_addon_with_wheels"
version = "1.0.0"

platforms = ["windows-x64", "macos-x64"]
wheels = [
   "./wheels/pillow-10.3.0-cp311-cp311-macosx_11_0_arm64.whl",
   "./wheels/pillow-10.3.0-cp311-cp311-
manylinux_2_28_x86_64.whl",
   "./wheels/pillow-10.3.0-cp311-cp311-win_amd64.whl",
]
```

Generated files from `--split-platforms`:

- my_addon_with_wheels-1.0.0-windows_x64.zip
- my_addon_with_wheels-1.0.0-macos_x64.zip

Note

Even though there is a Linux-only wheel present, no Linux zip file is generated. This happens because the `platforms` field only has Mac and Windows.

Creating an Extensions Repository

Third party sites that wish to support extensions in Blender can do so in several ways, in order of complexity:

- Static: Host a static JSON file listing all the packages of your repository.
- Dynamic: Serve the JSON file on-demand based on the Blender version and platform.
- Platform: Fork the entire Extensions Website to create your own.

Tags and Translations

If you are planning to use a different set of tags than the ones used by Blender Extensions Platform, remember to submit a pull request to tags.py.

This way they can be shown translated within Blender.

Sub-Pages

- Static Repository
- Dynamic Repository

Creating a Static Extensions Repository

To host your own extensions and leverage Blender update system all that is required is to host a static JSON file that is generated by Blender.

JSON

To generate a valid JSON file use the server generate Blender command-line tool:

```
blender --command extension server-generate --repo-dir=/path/to/packages
```

This creates an `index.json` listing from all the .zip extensions found in the *–repo-dir* location.

For more details, read the generated JSON API.

Testing

To test the generated repository, add a new Remote Repository from the Preferences:

- Get Extensions → Repositories → [+] → Add Remote Repository
- Paste the location of the generated JSON as the URL. So the example `/path/to/packages` would use the:
  - `file:///path/to/packages/index.json` on Linux and macOS.
  - `file:///C:/path/to/packages/index.json` on Windows.
  - `file://HOST/share/path/to/packages/index.json` for network shares on Windows.

Tip

Open `file:///` in a web browser and navigate to the repository location and copy that as the remote repository URL.

Extensions Listing HTML

The `server-generate` command can optionally create a simple website using the `--html` argument.

```
blender --command extension server-generate --repo-
dir=/path/to/packages --html
```

This creates an `index.html` file ready to use, listing extensions which can be dropped into Blender for installation.

Download Links

In order to support drag and drop for installing from a remote repository, there are a few optional ways to prepare the URLs.

The only strict requirement is that the download URL must end in `.zip`.

You can pass different arguments to the URL to give more clues to Blender about what to do with the dropped URL.

repository:: Link to the JSON file to be used to install the repository in Blender. Supports relative URLs.

platforms:: Comma-separated list of supported platforms. If omitted, the extension will be available in all operating systems.

blender_version_min:: Minimum supported Blender version, e.g., `4.2.0`.

blender_version_max:: Blender version that the extension does not support, earlier versions are supported.

Tip

The more details you provide, the better the user experience.

With the exception of the `repository`, all the other parameters can be extracted from the extensions manifest. Those arguments are to be encoded as part of the URL.

Expected format:

```
{URL}.zip?repository={repository}&blender_version_min=
{version_min}&blender_max={version_max_exclusive}&platforms=
{platform1,platform2}
```

Example from self-hosted repository:

```
http://my-site.com/my-addon.zip?
repository=.%2Findex.json&blender_version_min=4.2.0&platform
s=windows-x64
```

Example from the Extensions Platform:

```
https://extensions.blender.org/download/sha256:57a6a5f39fa2c
c834dc086a27b7b2e572c12fd14f8377fb8bd1c7022df3d7ccb/add-on-
amaranth-v1.0.23.zip?
repository=%2Fapi%2Fv1%2Fextensions%2F&blender_version_min=4
.2.0&platforms=linux-x64%2Cmacos-x64
```

Note

`%2F` and `%2C` are simply the url-encoded equivalent of `/` and `,` respectively.

Creating a Dynamic Extensions Repository

A dynamic repository allows you to serve a smaller JSON file with only the latest version of the extensions which are compatible with the query parameters. This is only relevant for repositories which contain multiple version of multiple extensions.

For small or personal repositories it is simpler and recommended to use static repositories instead.

Listing

To setup a dynamic extensions repository, follow the steps for static repositories, since the format and the listing are the same.

Query Parameters

When Blender fetches the extensions listing it passes the following arguments to make sure only compatible extensions are listed:

- `platform`
- `blender_version`

These arguments are passed as parameters to the server via a query URL:

URL:: `https://extensions.blender.org/api/v1/extensions/`

query URL:: `https://extensions.blender.org/api/v1/extensions/?blender_version=4.2.0&platform=linux-x64`

Access Token

Some repositories may require authentication. The user can specify an `access token` for a repository, which is passed along with the API request from Blender.

This is passed to the servers via an Authorization header:

```
curl -i https://extensions.blender.org/api/v1/extensions/ \
     -H "Accept: application/json" \
     -H "Authorization: Bearer abc29832befb92983423abcaef93001"
```

Application Templates

Usage

Application templates are a feature that allows you to define a re-usable configuration that can be selected to replace the default configuration, without requiring a separate Blender installation or overwriting your personal settings.

Application templates can be selected from the splash screen or *File ▸ New* submenu. When there are no templates found the menu will not be displayed on the splash screen.

New application templates can be installed from the Blender Menu. If you would like to keep the current application template active on restarting Blender, save your preferences.

Motivation

In some cases it's not enough to write a single script or add-on, and expect someone to replace their preferences and startup file, install scripts and change their keymap.

The goal of application templates is to support switching to a customized configuration without disrupting your existing settings and installation. This means people can build their own *applications* on top of Blender that can be easily distributed.

Details

An application template may define its own:

Startup File
: The default file to load with this template.

Preferences
: Only certain preferences from a template are used:

  - Themes.
  - Add-ons.
  - Keymaps.
  - Viewport lighting.

Splash Screen
: Templates may provide their own splash screen image.

Python Scripts
: While templates have access to the same functionality as any other scripts, typical operations include:

  - Modifying and replacing parts of the user interface.
  - Defining new menus, keymaps and tools.
  - Defining a custom add-on path for template specific add-ons.

Templates also have their own user configuration, so saving a startup file while using a template won't overwrite your default startup file.

Directory Layout

Templates may be located in one of two locations within the `scripts` directory.

Template locations:

```
{BLENDER_USER_SCRIPTS}/startup/bl_app_templates_user
{BLENDER_SYSTEM_SCRIPTS}/startup/bl_app_templates_system
```

User configuration is stored in a subdirectory:

Without a template:

```
./config/startup.blend
./config/userpref.blend
```

With a template:

```
./config/{APP_TEMPLATE_ID}/startup.blend
./config/{APP_TEMPLATE_ID}/userpref.blend
```

See Blender's Directory Layout for details on script and configuration locations.

Hint

Troubleshooting Paths

When creating an application template, you may run into issues where paths are not being found. To investigate this you can log output of all of Blender's path look-ups.

Example command line arguments that load Blender with a custom application template (replace `my_app_template` with the name of your own template):

```
blender --log "bke.appdir.*" --log-level -1 --app-template
my_app_template
```

You can then check the paths where attempts to access `my_app_template` are made.

Command Line Access

Using the command-line arguments you can setup a launcher that opens Blender with a specific app template:

```
blender --app-template my_template
```

Template Contents

Each of the following files can be used for application templates but are optional.

`startup.blend`
: Factory startup file to use for this template.

`userpref.blend`
: Factory preferences file to use for this template. When omitted preferences are shared with the default Blender configuration.

  (As noted previously, this is only used for a subset of preferences).

`splash.png`
: Splash screen to override Blender's default artwork (not including header text). Note, this image must be a `1000x500` image.

`__init__.py`
: A Python script which must contain `register` and `unregister` functions.

Note

Bundled blend-files `startup.blend` and `userpref.blend` are considered *Factory Settings* and are never overwritten.

The user may save their own startup/preferences while using this template which will be stored in their user configuration, but only when the template includes its own `userpref.blend` file.

The original template settings can be loaded using: *Load Template Factory Settings* from the file menu in much the same way *Load Factory Settings* works.

Template Scripts

While app templates can use Python scripts, they simply have access to the same APIs available for add-ons and any other scripts.

As noted above, you may optionally have an `__init__.py` in your app template. This has the following advantages:

- Changes can be made to the startup or preferences, without having to distribute a blend-file.
- Changes can be made dynamically.

  You could for example – configure the template to check the number of processors, operating system and memory, then set values based on this.
- You may enable add-ons associated with your template.

On activation a `register` function is called, `unregister` is called when another template is selected.

As these only run once, any changes to defaults must be made via handler. Two handlers you are likely to use are:

- `bpy.app.handlers.load_factory_preferences_post`
- `bpy.app.handlers.load_factory_startup_post`

These allow you to define your own “factory settings”, which the user may change, just as Blender has it’s own defaults when first launched.

This is an example `__init__.py` file which defines defaults for an app template to use.

```
import bpy
from bpy.app.handlers import persistent

@persistent
def load_handler_for_preferences(_):
    print("Changing Preference Defaults!")
    from bpy import context
```

```
    prefs = context.preferences
    prefs.use_preferences_save = False

    kc = context.window_manager.keyconfigs["blender"]
    kc_prefs = kc.preferences
    if kc_prefs is not None:
        kc_prefs.select_mouse = 'RIGHT'
        kc_prefs.spacebar_action = 'SEARCH'
        kc_prefs.use_pie_click_drag = True

    view = prefs.view
    view.header_align = 'BOTTOM'


@persistent
def load_handler_for_startup(_):
    print("Changing Startup Defaults!")

    # Use smooth faces.
    for mesh in bpy.data.meshes:
        for poly in mesh.polygons:
            poly.use_smooth = True

    # Use material preview shading.
    for screen in bpy.data.screens:
        for area in screen.areas:
            for space in area.spaces:
                if space.type == 'VIEW_3D':
                    space.shading.type = 'MATERIAL'
                    space.shading.use_scene_lights = True


def register():
    print("Registering to Change Defaults")

bpy.app.handlers.load_factory_preferences_post.append(load_hand
ler_for_preferences)

bpy.app.handlers.load_factory_startup_post.append(load_handler_
for_startup)

def unregister():
    print("Unregistering to Change Defaults")
```

```
bpy.app.handlers.load_factory_preferences_post.remove(load_hand
ler_for_preferences)

bpy.app.handlers.load_factory_startup_post.remove(load_handler_
for_startup)
```

Keymap Customization

Keys

Available Keys

When customizing keymaps it's useful to use keys which won't conflict with Blender's default keymap.

Here are keys which aren't used and aren't likely to be used in the future.

F-Keys (`F5` - `F8`)
: These F-keys (including modifier combination) have been intentionally kept free for users to bind their own keys to.

`OSKey` (also known as the `Windows-Key`, `Cmd` or `Super`)
: Blender doesn't use this key for any bindings.

  macOS is an exception, where `Cmd` replaces `Ctrl` except in cases it would conflict with the system's key bindings.

Modifier Double Click
: Binding modifier keys as primary keys is supported, to avoid conflicts with regular usage you can bind them to double click.

Multi-Action Keys

Click/Drag

It's possible to configure a single key to perform multiple operations using *Click* event instead of *Press*. Then you may bind *Drag* to a separate action.

This is useful for mixing actions where one uses a drag event, e.g: Toggle a setting using with `Tab`, drag to open a pie menu showing all options related

to the setting.

This is used in the default keymap in the 3D Viewport, `Alt-MMB` dragging in different directions rotates the view.

Common Operations

This section lists useful generic operations which can be used.

Key Bindings for Pop-Ups

Menus and panels can be assigned key shortcuts, even if they're only accessible from submenus elsewhere.

Open a Pop-up Menu (`wm.call_menu`)
: Open any menu on key press.

Open a Pie Menu (`wm.call_menu_pie`)
: Open any pie menu on key press.

Open a Panel (`wm.call_panel`)
: Open a pop-up panel (also known as a pop-over).

Menu & Panel Identifiers

To find the `name` of a menu, enable the preference *Interface ▸ Display ▸ Python Tooltips*.

Then hover the cursor over the popover button or menu item. For submenus you will need to use the back arrow to prevent the submenu from opening and gaining focus.

Key Bindings for Properties

There are many properties you might want to bind a key with. To avoid having to define operators for each property, there are generic operators for this purpose:

Operators for adjusting properties begin with `wm.context_`.

Some of these include:

- `wm.context_toggle` toggle a Boolean property.
- `wm.context_cycle_enum` cycle an enum property forwards or backwards.
- `wm.context_menu_enum` show a pop-up menu for an enum property.
- `wm.context_pie_enum` show a pie menu for an enum property.
- `wm.context_scale_float` scale a number (used for increasing / decreasing brush size for example).
- `wm.context_toggle_enum` toggle between two options of an enum.
- `wm.context_modal_mouse` moving the cursor to interactively change a value.

See bpy.ops.wm for a complete list.

Each of these operators has a `data_path` setting to reference the property to change.

To find the `data_path`, basic Python knowledge is needed.

For example, you can use the Python Console to access a Boolean property you wish to map to a key:

```
bpy.context.object.show_name
```

To bind this to a key, add a new keymap item using the operator `wm.context_toggle` with `data_path` set to `object.show_name` (notice the `bpy.context` prefix is implicit).

See bpy.context for other context attributes.

The Python API documentation can be used to find properties or you may use the Python Console's auto-complete to inspect available properties.

Working Limits

Space

While object positions, vertex locations are not clamped, larger values become increasingly imprecise. To get an idea of the precision you can work with using different scales. Here's a table of scales and their associated accuracy:

10::	1/1,048,576th
100::	1/131,072th
1,000::	1/16,384th
10,000::	1/1,024th
100,000::	1/128th
1,000,000::	1/16th

Hint

For a rough rule of thumb, values within -5,000/+5,000 are typically reliable (range of 10,000). Internally *single precision* floating-point calculations are used.

Time

The maximum number of frames for each scene is currently 1,048,574, and allows for continuous shots for durations of:

24 fps::	12 hours, 8 minutes.
25 fps::	11 hours, 39 minutes.
30 fps::	9 hours, 42 minutes.
60 fps::	4 hours, 51 minutes.

Note

In practice, a finished work is typically composed of output from many scenes. So this limit does not prevent you from creating longer works.

Text Fields

Fixed strings are used internally, and while it is not useful to list all limits, here are some common limits. *Used for data-block names, modifiers, vertex groups, UV layers…*

directory::	767
file-name::	255
file-path::	1023
identifier::	63

Note

Multi-byte encoding means some Unicode characters use more than a single ASCII character.

Operators

Operator Cheat Sheet

Reference

Menu:: *Help ▸ Operator Cheat Sheet*
Context:: Enable Developer Extras

Creates a text file in the Text Editor that gives a list of all operators and their default values in Python syntax, along with the generated docs. This is a good way to get an overview of all Blender's operators.

See also

Blender's API documentation

System Operators

Reload Scripts

Reference

Mode:: All Modes
Menu:: *Topbar ▸ Blender ▸ System ▸ Reload Scripts*

Reloads all scripts found in the scripts data folder; any changes that have been made in the Text Editor will be lost!

Memory Statistics

Reference

Mode:: `--debug-memory`
Menu:: *Topbar ▸ Blender ▸ System ▸ Memory Statistics*

This operator which can be found by searching "Memory Statistics" with the Operator Search will print useful information about memory objects, their size and user count.

Important

To fully use this operator run Blender from the console with `--debug-memory`.

Debug Menu

Reference

Mode:: All Modes
Menu:: *Topbar ▸ Blender ▸ System ▸ Debug Menu*

This operator brings up a menu to set Blender into a certain debug mode.

See the source code for a description of what each value does.

Tip

Developers can search the code for `G.debug_value` to find other possible uses for this operator.

Note

Additional debug options are available by launching Blender in debug mode or setting `bpy.app.debug = True`.

Redraw Timer

Reference

Mode::	All Modes
Menu::	*Topbar ▸ Blender ▸ System ▸ Redraw Timer*

This operator brings up a menu with a list of tests to benchmark UI render times along with undo/redo functions.

Clean Up Space-Data

Reference

Mode::	All Modes
Menu::	*Topbar ▸ Blender ▸ System ▸ Clean Up Space-data*

Removes unused settings for invisible editors.

Blender's Directory Layout

This page documents the different directories used by Blender.

This can be helpful for troubleshooting, automation and customization.

User Directories

User directories store preferences, startup file, installed extensions, presets and more. By default these use the standard configuration folders for each operating system.

Linux

```
$HOME/.config/blender/4.3/
```

If the `$XDG_CONFIG_HOME` environment variable is set:

```
$XDG_CONFIG_HOME/blender/4.3/
```

macOS

```
/Users/$USER/Library/Application Support/Blender/4.3/
```

Windows

```
%USERPROFILE%\AppData\Roaming\Blender Foundation\Blender\4.3\
```

Portable Installation

When running Blender from a portable drive, it's possible to keep the configuration files on the same drive to take with you.

To enable this, create a folder named `portable` at the following locations:

- Windows: Next to the Blender executable, in the unzipped folder
- Linux: Next to the Blender executable, in the unzipped folder
- macOS: Inside the application bundle at `Blender.app/Contents/Resources`

This folder will then store preferences, startup file, installed extensions and presets.

Environment Variables

The `BLENDER_USER_RESOURCES` environment variable can be set to a custom directory to replace the default user directory.

System Directories

System directories store files that come bundled with Blender and are required for it to function. This includes scripts, presets, essential assets and more.

Linux

Archive downloaded from blender.org:

```
./4.3/
```

Linux distribution packages:

```
/usr/share/blender/4.3/
```

macOS

```
./Blender.app/Contents/Resources/4.3/
```

Windows

Zip file downloaded from blender.org:

`./4.3/`

Installer downloaded from blender.org:

`%ProgramFiles%\Blender Foundation\Blender\4.3\`

Microsoft Store installation:

`%ProgramFiles%\WindowsApps\BlenderFoundation.Blender<HASH>\Blender\4.3\`

Environment Variables

`BLENDER_SYSTEM_SCRIPTS` and `BLENDER_SYSTEM_EXTENSIONS` environment variables can be used to bundle additional scripts and extensions, that are not part of the regular Blender installation.

Other `BLENDER_SYSTEM` environment variables can override other system paths, though are not commonly used in practice.

Path Layout

`./autosave`
: Autosave blend-file location. (Windows only, temp directory used for other systems.)

  Located in user directories.

`./config`
: User configuration and session info.

  Located in user directories.

  `./config/startup.blend`
  : Blend file to load on startup.

  `./config/userpref.blend`
  : User preferences.

`./config/bookmarks.txt`
: File Browser bookmarks.

`./config/recent-files.txt`
: Recent file menu list.

`./config/{APP_TEMPLATE_ID}/startup.blend`
: Startup file for an application template.

`./config/{APP_TEMPLATE_ID}/userpref.blend`
: User preferences file for an application template.

`./datafiles`
: Data files loaded at runtime.

  Located in both user and system directories. User data files either override or add to system data files.

  `./datafiles/colormanagement`
  : Default OpenColorIO configuration.

  `./datafiles/fonts`
  : User interface fonts.

  `./datafiles/studiolights`
  : Studio light images for 3D viewport.

`./extensions`
: Extension repositories.

  Located in both user and system directories. Repositories are loaded from both directories.

`./scripts`
: Add-ons, presets, templates, user interface, startup scripts.

  Located in both user and system directories. Scripts are loaded from both directories.

`./scripts/addons/*.py`
: Python add-ons which may be enabled in the Preferences include import/export format support, render engine integration and many handy utilities.

`./scripts/addons/modules/*.py`
: Modules for add-ons to use (added to Python's `sys.path`).

`./scripts/addons_core/*.py`
: The add-ons directory which is used for bundled add-ons.

`./scripts/addons_core/modules/*.py`
: Modules for `addons_core` to use (added to Python's `sys.path` when it found).

`./scripts/modules/*.py`
: Python modules containing our core API and utility functions for other scripts to import (added to Python's `sys.path`).

`./scripts/startup/*.py`
: Scripts which are automatically imported on startup.

  `./scripts/startup/bl_app_templates_user/{APP_TEMPLATE_ID}`
  : Application templates installed in user directories.

  `./scripts/startup/bl_app_templates_system/{APP_TEMPLATE_ID}`
  : pplication templates automatically loaded from system directories.

`./scripts/presets/{preset}/*.py`
: Presets used for storing user-defined settings for cloth, render formats, etc.

`./scripts/templates_py/*.py`
: Example scripts which can be accessed from *Text Editor ▸ Templates ▸ Python*.

`./scripts/templates_osl/*.osl`
: Example OSL shaders which can be accessed from *Text Editor* ▸ *Templates* ▸ *Open Shading Language*.

`./python`
: Bundled Python distribution.

  Located in system directories.

Local Cache Directory

The cache directory is used to store persistent caches locally. Currently it is only used for the indexing of Asset Libraries. The operating system is not expected to clear this automatically.

The following path will be used:

- **Linux::** `$XDG_CACHE_HOME/blender/` if `$XDG_CACHE_HOME` is set, otherwise `$HOME/.cache/blender/`
- **macOS::** `/Library/Caches/Blender/`
- **Windows::** `%USERPROFILE%\AppData\Local\Blender Foundation\Blender\Cache\`

Temporary Directory

The temporary directory is used to store various files at run-time (including render layers, physics cache, copy-paste buffer and crash logs).

The temporary directory is selected based on the following priority:

- User Preference (see File Paths).
- Environment variables (`TEMP` on Windows, `TMP` & `TMP_DIR` on other platforms).
- The `/tmp/` directory.

Deploying Blender in Production

This page contains tips for setting up Blender in environments such as animation studios and schools.

These environments often have special requirements regarding security, automated deployment and customization.

Installing Blender

Blender downloads can be extracted to any directory on the system, as a self contained installation. Multiple Blender versions can co-exist on the same system, and deployment can be automated using standard file management tools.

New Blender versions may add, remove or change functionality that affects the results of production files. For a given project, it is advisable to use a single LTS version of Blender. LTS versions receive bug fixes for two years.

Working Offline

For security or other reasons, workstation may not have internet access.

By default Blender does not access the internet, however this can be enabled in the System preferences with the Online Access option.

Working offline can be enforced by running with the `--offline-mode` command line argument. Users will then be unable to enable online access in the preferences.

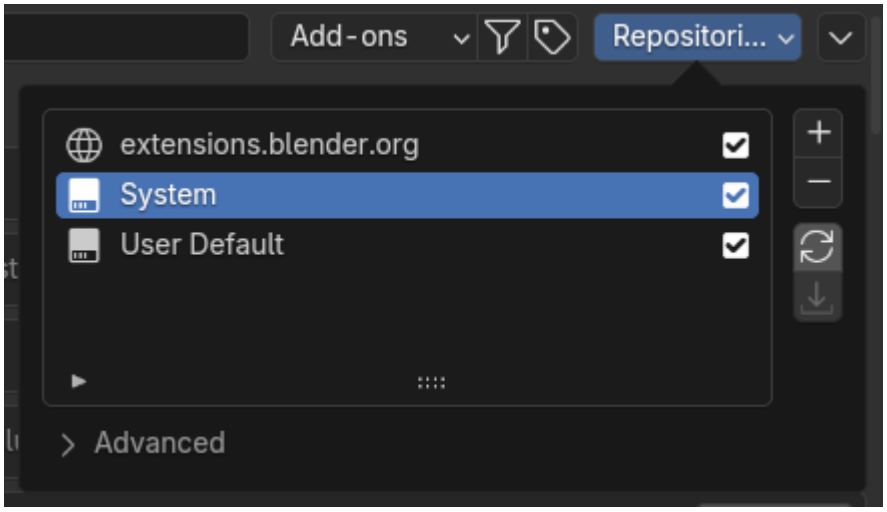
Note

Add-ons that follow this setting will only connect to the internet if enabled. However, Blender cannot prevent third-party add-ons from

violating this rule.

Bundling Extensions

When working offline or in a more controlled environment, it may be useful to provide a set of extensions to all users. For this there is a default read-only System repository. This repository can for example be located on a read-only network drive or in a system directory.



System repository

The `$BLENDER_SYSTEM_EXTENSIONS` environment variable controls the default location. This should point to a directory, within which a `system` directory should exist.

Extensions packages should be extracted in this `system` directory, with a resulting path like this:

```
$BLENDER_SYSTEM_EXTENSIONS/system/my-
addon/blender_manifest.toml
```

In the Extensions preferences, it's possible to manually set a custom directory for the default System repository, or to create multiple repositories.

Bundling Scripts

Besides extensions, it's possible to bundle scripts for presets, application templates, legacy add-ons, as well as scripts run on startup.

Script directories can be manually added in the File Paths preferences. The `$BLENDER_SYSTEM_SCRIPTS` can also be used to add a script directory without modifying the preferences.

These script directories are expected to contain specific directories like `presets`, `addons` and `startup` for different types of scripts. See Path Layout for a complete list.

Startup Scripts

The Blender Python API can be used to customize Blender. This includes changing preferences, changing the startup file and adding UI elements.

For example, a script can enable add-ons for every user.

`$BLENDER_SYSTEM_SCRIPTS/startup/enable_addons.py`

```
def register():
    import addon_utils
    addon_utils.enable("my-addon")

def unregister():
    pass

if __name__ == "__main__":
    register()
```

Application Templates

Application Templates can be used to set up Blender for particular tasks or projects, separate from the default configuration. When creating a new file the user can choose the template.

The files are expected to be placed in the system script directories like this:

```
$BLENDER_SYSTEM_SCRIPTS/startup/bl_app_templates_system/MyTemplate/__init__.py
$BLENDER_SYSTEM_SCRIPTS/startup/bl_app_templates_system/MyTemplate/startup.blend
```

Legacy Add-ons

Add-ons that have not been converted to become an extension yet need to be placed in the `addons` script directory.

For example, an add-on could be located at:

```
$BLENDER_SYSTEM_SCRIPTS/addons/simple_addon.py
$BLENDER_SYSTEM_SCRIPTS/addons/complex_addon/__init__.py
```

VFX Platform

Blender follows the VFX reference platform, which means it is able to run on the same systems as other VFX software and exchange image, volume and scene files with them.

Python Version

Blender and the bpy module are only compatible with a single Python version. This makes it possible for add-ons and VFX software in general to only have to target a single Python version.

Blender bundles a complete Python installation and does not interact with the system Python by default. This can be changed with the `--python-use-system-env` command line argument, if care is taken to set up a compatible Python version.

Appendices

This chapter covers far more detailed explanations about some Blender tools (which may not be required for typical usage).

- Rotation Modes

Rotation Modes

Blender lets you define rotations in several ways. Each one of them has a series of advantages and drawbacks; there is no best rotation mode, as each one is suitable for specific cases.

In all of these modes, positive angle values mean counter-clockwise rotation direction, while negative values define clockwise rotation.

Though you can rotate elements using the global or local transform orientations, these axes are not suitable to define rotations, as the effect of each of them cannot be isolated from the other two.

Take, for instance, any three values for X, Y and Z rotation. Perform each one of these using global or local axes. Depending on the order in which you perform these, you will end up with different final orientations. So proper rotation coordinate systems are needed.

Euler Modes

The axes system used for performing Euler rotations is the so called Euler gimbal. A gimbal is a particular set of three axes. The special thing about this is that the axes have a hierarchical relationship between them: one of the axes is at the top of the hierarchy, and has one of the other two axes as its immediate child; at the same time, this child axis is the parent of the remaining axis, the one at the very bottom of the hierarchy.

Which axis is on top, which one in the middle and which at the bottom, depends on the particular Euler gimbal: there are six types of them, as there are six possible combinations: XYZ, XZY, YXZ, YZX, ZXY and ZYX Euler rotation modes. These modes are named using the letters of the axes in order, starting from the axis at the bottom of the hierarchy, and finishing with the one on top.

The main problem of these systems comes when they lose their relative perpendicularity. And this happens when the axis in the middle rotates, causing the axis at the bottom to rotate with it. It keeps getting worse when this bottom axis approaches 90° (or equivalent angles). In that case, it will remain aligned with the axis on top of the hierarchy. In that moment we have just lost one axis of rotation. This can cause discontinuous interpolations when animating. This particular loss of axis is known as the "gimbal lock".

Hint

The actual configuration of the gimbal axes can be seen in the 3D Viewport by enabling the *Rotate* object gizmo and setting it to *Gimbal* (from the gizmos button in the header). At the same time, rotation mode should be set to any of the Euler modes for the active object.

Now you can perform a rotation around the axis in the middle (e.g. in *XYZ Euler* mode that is the Y axis), and see how easy it is to end up having a gimbal with just two axes. In the specific case of the *XYZ Euler* mode with gimbal lock, a rotation around the X axis will have the same effect as rotating around the Z axis, meaning, in practice, that no X axis rotations can be performed.

One advantage of this mode is that animation curves are easy to understand and edit. However, special attention must be done when the middle axis approaches values close to 90° (or equivalent angles).

Axis Angle Mode

This mode lets us define an axis (X, Y, Z) and a rotation angle (W) around that axis.

If we define the rotation using interactive rotations (with the rotation gizmo), the values of X, Y and Z will not exceed 1.0 in absolute value, and W will be comprised between 0 and 180 degrees.

If you wish to define rotations above 180° (e.g. to define multiple revolutions), you will need to edit the W value directly, but as soon as you perform an interactive rotation, that value will be adjusted again. Same thing goes for axis values.

This system is suitable for elements revolving around a fixed axis, or to animate one of the elements at a time (either the axis or the angle). The problem might come when animating (interpolating) both components at the same time: axis and angle. The resulting effect might not be as expected.

The *Gimbal* gizmo in this rotation mode shows a set of three orthogonal axes in which the Z axis goes along the defined rotation axis, i.e. it points towards the direction defined by the (X, Y, Z) point.

The axis-angle system is free from gimbal lock, but animation curves in this mode are not intuitive at all when animating axis and angle at the same time, in which case they are difficult to understand and edit.

Quaternion Mode

In this mode, rotations are also defined by four values (X, Y, Z and W). X, Y and Z also define an axis, and W an angle, but it does it quite differently from axis-angle. The important thing here is the relation between all four values.

To describe it in an intuitive way, let's take the effect of the X coordinate: what it does is to rotate the element around the X axis up to 180 degrees. The same goes for Y and Z. The effect of W is to avoid those rotations and leave the element with zero rotation. The final orientation is a combination of these four effects.

As the relation between components is what defines the final orientation, multiplying or dividing all four numbers by a constant value will yield the very same rotation.

This mode is ideal for interpolating between **any** pair of orientations. It doesn't suffer from gimbal lock or any interpolation undesired effect. The only drawback is that you cannot interpolate between two orientations that are at a distance greater than than 180°, as the animation will take the shortest path between them. Thus to animate a revolving element you must set up many intermediate keyframes, 180° from each other at most.

The *Gimbal* gizmo in this mode is equivalent to the *Local* one, and doesn't have any special meaning.

The animation curves in this mode are not intuitive, so they are also difficult to understand and edit.

More about Quaternions

This section is not really useful for 3D artists, but it can be suitable for the curious or the scientist.

Quaternions are a number system extending the complex numbers. They represent a four component vector, whose components are called, in Blender, X, Y, Z and W. When rotating interactively in quaternion mode, the so called norm (length) of the quaternion will remain constant. By definition, the norm of a quaternion equals 1.0 (that's a **normalized** quaternion). When you select the quaternion mode in Blender, the XYZW components describe a normalized quaternion.

Note

The norm of a quaternion *q* is defined mathematically as:

\[\lvert q \rvert = \sqrt{X^2 + Y^2 + Z^2 + W^2}\]

However, if one of the quaternion components is locked during the interactive transformation using the proper lock button, the norm will not remain unchanged, as that blocked component will not be able to adjust itself to keep the unit norm.

Hint

Interactive rotations with the gizmo don't change the norm of the current quaternion. Editing a single XYZW component individually you can change the norm. To make the norm 1.0 again you can switch to any rotation mode and back again into quaternion.

The rotation components of a quaternion keep a tight relation with those of axis-angle. To find a correspondence, first of all we must deal with the normalized version of the quaternion, that is, one whose norm equals 1.0. To normalize a quaternion, just divide each one of its components by its norm. As we have seen before, dividing all four values by the same number gives the same orientation.

Once we have calculated the components of the normalized quaternion, the relation with the axis-angle components is as follows:

- X, Y and Z mean exactly the same as in axis-angle: they just define an axis around which the rotation takes place.
- W can be used to retrieve the actual rotation around the defined angle. The following formula applies (provided that the *quaternion is normalized*): $W = \cos(\frac{a}{2})$, where *a* is actually the rotation angle we are looking for. That is: $a = 2 \arccos{W}$.

Other Considerations

In axis-angle and quaternion modes we can lock rotations in interactive modes in a per component basis, instead of doing it by axis. To do so we can activate this locking ability using the lock buttons next to the corresponding *Rotation* transform buttons.

Regarding rotation animations, all keyframes must be defined in the same rotation mode, which must be the selected rotation mode for the object throughout the entire animation.

Troubleshooting

- Startup
- 3D Viewport
- Graphics Hardware
- Crashes
- Python Errors
- Recovering Data

Compatibility

Some applications which integrate themselves into your system can cause problem's with Blender.

Here is a list of known compatibility issues.

Startup

There are some common causes for problems when using Blender. If Blender fails to start, there are a few things to check for:

- See if your computer meets the minimum requirements.
- Confirm that your graphics card is supported and that the drivers are up to date (see Troubleshooting Graphics Hardware).
- Make sure any antivirus software is not preventing Blender from starting.

If you cannot find a solution to your problem here, try asking the community for help.

Common Startup Messages

The *Blender Console Window* can display many different types of status and error messages. Some messages simply inform the user what Blender is doing, but have no real impact on Blender's ability to function. Other messages can indicate serious errors that will most likely prevent Blender carrying out a particular task and may even make Blender non-responsive or shut down completely. The *Blender Console Window* messages can also originate internally from within the Blender code or from external sources such as Python scripts.

`found bundled python: {DIR}`
: This message indicates that Blender was able to find the Python library for the Python interpreter embedded within Blender. If this folder is missing or unable to be found, it is likely that an error will occur, and this message will not appear.

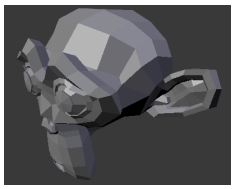
`Read prefs: {DIR}/userpref.blend`
: The preferences use this path.

3D Viewport

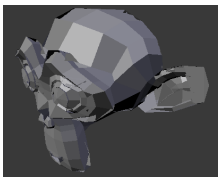
Rendering

Depth Buffer Glitches

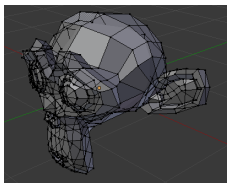
Sometimes when setting a large clipping range will allow you to see both near and far objects, but reduces the depth precision resulting in artifacts.



Model with no clipping artifacts.



Model with clipping artifacts.



Mesh with artifacts in Edit Mode.

To avoid this:

- Increase the near clipping when working on large scenes.
- Decrease the far clipping when objects are not viewed at a distance.

When perspective is disabled only the far Clip End is used, very high values can still result in artifacts. This is **not** specific to Blender, all graphics applications have these same limitations.

Objects Invisible in Camera View

If you have a large scene, viewing it through Camera View may not display all of the objects in the scene. One possibility may be that the clipping distance of the camera is too low. The camera will only show objects that fall within the clipping range.

Performance

Slow Rendering

There are a couple of reasons why you may be experiencing a slow viewport.

Old Hardware
: Sometimes your hardware, mainly your graphics card, may be too slow to keep up with your model.

Upgrade Graphics Driver
: In some cases, slow selection is resolved by using updated drivers.

Slow Selection

Blender uses OpenGL for selection, some graphics card drivers are slow at performing this operation.

This becomes especially problematic on dense geometry.

Possible Solutions:

GPU Depth Picking (Preferences)
: See *Preferences ▸ Viewport ▸ Selection*.

  This option is enabled by default, disabling it may give a better performance at the cost of selection accuracy.

Upgrade Graphics Driver
: In some cases, slow selection is resolved by using updated drivers. *It is generally good to use recent drivers when using 3D software*.

Select Centers (Workaround)
: In *Object Mode*, holding `Ctrl` while selecting uses the object center point. While this can be useful on its own, it has the side effect of not relying on OpenGL selection.

Change Display Mode (Workaround)

Using *Wireframe* display mode can be used to more quickly select different objects.

Note

Obviously, the workarounds listed here are not long term solutions, but it is handy to know if you are stuck using a system with poor OpenGL support.

Ultimately, if none of these options work out it may be worth upgrading your hardware.

Viewport Playback Frame Rate Limited

Having the viewport playback clamped to a maximum of 60 FPS is typically caused by the VSYNC setting on your GPU, for higher frame rates you may have to disable VSYNC functionality although this may be of limited us since frames rendered may be more than your GPU and monitor are able to display.

VSYNC is configured as part of your GPU driver options which vary depending on your system & GPU combination.

Navigation

Lost in Space

When navigating your scene, you may accidentally navigate away from your scene and find yourself with a blank viewport. There are two ways to fix this:

1. Select an object in the Outliner, then zoom to that object with *View* ▸ *Frame Selected* or `NumpadPeriod`.
2. Use `Home` to fit all objects into the 3D Viewport.

Invisible Limit Zooming In

Sometimes when navigating you may be trying to zoom in but it seems that you have hit a limit to how far you can zoom. This is because Blender uses a central point to orbit around.

In practice this is good for modeling an object which you rotate about a lot to see from all sides (think of a potter using a wheel). However, this makes it awkward to explore a scene or model an object from the 'inside', for example.

Solutions

- Use View Dolly.
- Use Walk/Fly Navigation.
- Use Auto Depth and Zoom to Mouse Position. These tools will make sure the distance is always the value under the mouse cursor,
- Use Zoom Region as it also resets the center point when zooming.
- Center the view around the mouse cursor `Alt-MMB`. This will take the position under the cursor and make it your viewpoint center.
- Use an NDOF, also known as a 3D mouse, see configuring peripherals for more information.

Tools

Invalid Selection

There are times when selection fails under some configurations, often this is noticeable in mesh *Edit Mode*, selecting vertices/edges/faces where random elements are selected.

Internally Blender uses OpenGL for selection, so the graphics card driver relies on giving correct results.

Possible Solutions:

Disable Multisampling

This is by far the most common cause of selection issues.

There are known problems with some graphics cards when using multisampling.

You can disable this option by turning multisampling off in your graphics card driver options.

Change Anti-Aliasing Sample Settings

Depending on your OpenGL configuration, some specific sample settings may work while others fail.

Unfortunately finding working configuration involves trial & error testing.

Upgrade Graphics Driver

As with any OpenGL-related issues, using recent drivers can resolve problems.

However, it should be noted that this is a fairly common problem and remains unresolved with many drivers.

Troubleshooting Graphics Hardware

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

- Windows
  - Nvidia
  - AMD
  - Intel
  - Other GPU
- Linux
  - Nvidia
  - AMD
  - Intel
  - Other GPU
- macOS
  - Nvidia
  - AMD
  - Intel
  - Other GPU

Troubleshooting Windows Hardware

- Nvidia
  - Drivers
  - Laptops
  - Common Problems
  - Information
- AMD
  - Drivers
  - Laptops
  - Common Problems
  - Information
- Intel
  - Drivers
  - Laptops
  - Compatibility
  - Common Problems
  - Information
  - Legacy Intel HD 4000/5000
- Other GPU
  - Drivers
  - Laptops
  - Common Problems
  - Information

Windows – Nvidia

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Windows drivers are provided by the graphics card manufacturer (Nvidia). Windows update automatically installs graphics drivers, or your computer manufacturer may provide its own version of the graphics drivers.

However, these are not always the latest version or may have been corrupted in some way. We recommend to always use the official drivers.

Download Latest Nvidia Drivers

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Windows – AMD

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Windows drivers are provided by the graphics card manufacturer (AMD). Windows update automatically installs graphics drivers, or your computer manufacturer may provide its own version of the graphics drivers.

However, these are not always the latest version or may have been corrupted in some way. We recommend to always use the official drivers.

Download Latest AMD Drivers

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Windows – Intel

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Windows drivers are provided by the graphics card manufacturer (Intel). Windows update automatically installs graphics drivers, or your computer manufacturer may provide its own version of the graphics drivers.

However, these are not always the latest version or may have been corrupted in some way. We recommend to use the official drivers.

Download Latest Intel Drivers

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Compatibility

In some cases Blender may crash on startup. Running Blender in compatibility mode can help in fixing this issue. To enable compatibility mode, RMB on the Blender executable and select *Properties ▸ Compatibility* and enable *Run this program in compatibility mode*. Confirm the changes with *Apply*.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).

- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Legacy Intel HD 4000/5000

When running on Intel 3, 4 or 5th gen iGPU the latest Intel driver will crash on startup. In order to start Blender try to install a previous version of the driver. Drivers that are known to work are:

- 20.19.15.4835
- 20.19.15.4963
- 20.19.15.5063

Drivers that are known to fail are:

- 20.19.15.5144
- 20.19.15.5166
- 20.19.15.5171

Download older Intel Drivers

Windows – Other GPU

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Windows drivers are provided by the graphics card manufacturer. Windows update automatically installs graphics drivers, or your computer manufacturer may provide its own version of the graphics drivers.

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Troubleshooting Linux Hardware

- Nvidia
  - Drivers
  - Laptops
  - Common Problems
  - Information
- AMD
  - Drivers
  - Laptops
  - Common Problems
  - Information
- Intel
  - Drivers
  - Common Problems
  - Information
- Other GPU
  - Drivers
  - Laptops
  - Common Problems
  - Information

Linux – Nvidia

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Linux, graphics drivers are usually installed as a package by your Linux distribution. Installing the latest drivers is typically done by upgrading packages or the distribution as a whole. Some distributions provide multiple packages for multiple drivers versions, giving you the choice to install newer versions.

For Nvidia there are open source (Nouveau) and closed source (by Nvidia) graphics drivers. Blender functions best with the closed source drivers as they are more optimized and complete. Linux graphics drivers can be downloaded from Nvidia's website, however in most cases the ones from your Linux distribution are fine and make things easier. Manually downloading drivers is mostly useful to get the very latest version, for example for a GPU that was only recently released.

NVidia Website

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better

performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.

- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Linux – AMD

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Linux, graphics drivers are usually installed as a package by your Linux distribution. Installing the latest drivers is typically done by upgrading packages or the distribution as a whole. Some distributions provide multiple packages for multiple drivers versions, giving you the choice to install newer versions.

AMD drivers are open source, except for the OpenCL support which is available as part of Pro drivers. Installing packages through your Linux distribution is usually best. AMD also provides graphics drivers for download on their website if you need the latest version.

AMD Drivers and Support Website

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.

- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Linux – Intel

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Linux, graphics drivers are usually installed as a package by your Linux distribution. Installing the latest drivers is typically done by upgrading packages or the distribution as a whole. Some distributions provide multiple packages for multiple drivers versions, giving you the choice to install newer versions.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information

about your graphics card, vendor and driver version.

Linux – Other GPU

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On Linux, graphics drivers are usually installed as a package by your Linux distribution. Installing the latest drivers is typically done by upgrading packages or the distribution as a whole. Some distributions provide multiple packages for multiple drivers versions, giving you the choice to install newer versions.

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Troubleshooting macOS Hardware

- Nvidia
  - Drivers
  - Common Problems
  - Information
- AMD
  - Drivers
  - Common Problems
  - Information
- Intel
  - Drivers
  - Common Problems
  - Information
- Other GPU
  - Drivers
  - Common Problems
  - Information

macOS – Nvidia

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On macOS graphics drivers are built into the operating system and the only way to get newer drivers is to upgrade macOS as a whole to the latest version.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

macOS – AMD

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On macOS graphics drivers are built into the operating system and the only way to get newer drivers is to upgrade macOS as a whole to the latest version.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

macOS – Intel

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On macOS graphics drivers are built into the operating system and the only way to get newer drivers is to upgrade macOS as a whole to the latest version.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

macOS – Other GPU

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

On macOS graphics drivers are built into the operating system and the only way to get newer drivers is to upgrade macOS as a whole to the latest version.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences* ▸ *System* ▸ *Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.

Crashes

The most common causes of Blender crashes:

- Running out of memory.
- Issues with graphics hardware or drivers.
- Bugs in Blender.

Firstly, you may be able to recover your work with *File ▸ Recover ▸ Auto Save*….

To prevent the problem from happening again, you can check that the graphics drivers are up to date (Troubleshooting Graphics Hardware), upgrade your machine's hardware (the RAM or graphics card), and disable some options that are more memory intensive:

- Reduce undo steps *Preferences ▸ System ▸ Memory & Limits ▸ Undo Steps*.
- Using multisample anti-aliasing also increases the memory usage and makes the display slower.
- On Linux, the Window Manager (KDE and Gnome for example) may be using hardware accelerated effects (e.g. window shadows and transparency) that are using up the memory that Blender needs. Try disabling the desktop effects or switch to a lightweight Window Manager.

To check memory usage by Blender:

- On Windows, use Task Manager and sort by Memory.
- On macOS, use Activity Monitor.app and open Memory tab. Alternatively, run `top -o MEM`.
- On Linux, run `top -o %MEM`.

Crash Log

When Blender crashes, it writes out a text file which contains information that may help identify the cause of the crash. Usually, this file is written in the Temporary Directory directory.

This file contains a log of tools used up until the crash as well as some other debug information. When reporting bugs about crashes it can be helpful to attach this file to your reports, especially when others are unable to reproduce the crash.

Windows

On a crash, a file is written based on the name of the currently loaded blend-file, so `test.blend` will create a file called `test.crash.txt`.

Batch scripts are provided in Blender installation directory which may be run to obtain the Blender debug log and system info text files:

- `blender_debug_log.cmd` is used in most cases.
- `blender_debug_gpu.cmd` and `blender_debug_gpu_workaround.cmd` log GPU-related errors.
- `blender_factory_startup.cmd` starts Blender with default settings which is recommended for debugging.

If the crash happens in Blender module, stack trace is also written to a file named `blender.crash.txt`. The path to that file can be found at the end of `blender_debug_log.txt` file.

macOS

After crash, macOS Crash Reporter shows a window with backtrace after some time, or when Blender is opened again. Copy the text in the crash report and save it in a text file. That file should be attached to the bug report while following other bug reporting guidelines.

Some `.crash` files can also be found in `~/Library/Logs/DiagnosticReports/` with the name of format: `Blender_YYYY-MM-DD-HHMMSS_MACNAME.crash`. If a report is present

corresponding to the time of crash, that file can also provide hints about cause of the crash. Alternatively, Console.app can be used to navigate all “User Reports” (see sidebar in the app).

Linux

On a crash, a file named `blender.crash.txt` is written in `/tmp` directory.

Note

More logs can be obtained by running Blender from Command Line and using `--factory-startup --debug-all` flags. See Launching from the Command Line and Command Line Arguments.

Python Errors

Precompiled Libraries

While not common practice, Python add-ons can be distributed with their own precompiled libraries. Unlike regular Python scripts, these are not portable between different platforms.

It is possible the library is incompatible with your Blender installation (attempting to load a library built for a different version of Python, or loading a 32-bit library on a 64-bit system).

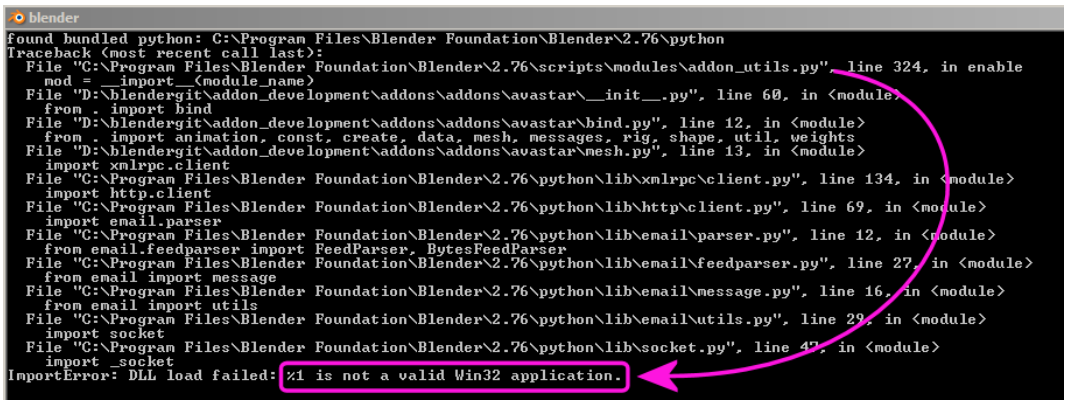
If the add-on contains `.pyd` or `.so` files, check that the distribution is compatible with your operating system.

Platform Specific

Windows

Mixed Python Libraries (DLLs)

If Python is raising errors or you have an add-on that just fails when enabled with an error, e.g: `... is not a valid Win32 application.`.



A Python traceback.

This may be caused by some inconsistency in the Python libraries. While Blender comes with its own bundled Python interpreter, duplicate, incompatible libraries can cause problems.

To find out which Python Library caused the Problem check the error message.

This is normally reported somewhere around the bottom line of the traceback. With the error above you see the problem is caused while trying to import `_socket`. This corresponds to either a file named `_socket.py` or `_socket.pyd`.

To help troubleshoot this problem, the following script can be pasted into the Text editor and run to check for duplicate libraries in your search path. (The output will show in Command Line Window.)

```
import os
import sys

# Change this based on the library you wish to test
test_lib = "_socket.pyd"

def GetSystemDirectory():
    from ctypes import windll, create_string_buffer, sizeof
    GetSystemDirectory = windll.kernel32.GetSystemDirectoryA
    buffer = create_string_buffer(260)
    GetSystemDirectory(buffer, sizeof(buffer))
    return os.fsdecode(buffer.value)

def library_search_paths():
    return (
        # Windows search paths
        os.path.dirname(sys.argv[0]),
        os.getcwd(),
        GetSystemDirectory(),
        os.environ["WINDIR"],  # GetWindowsDirectory
        *os.environ["PATH"].split(";"),

        # regular Python search paths
```

```
        *sys.path,
        )

def check_library_duplicate(libname):
    paths = [p for p in library_search_paths()
             if os.path.exists(os.path.join(p, libname))]

    print("Library %r found in %d locations:" % (libname,
len(paths)))
    for p in paths:
        print("- %r" % p)

check_library_duplicate(test_lib)
```

Recovering Data

Computer crashes, power outages, or simply forgetting to save can result in the loss or corruption of your work. You can use Blender's *Auto Save* feature to reduce the chance of losing files when such events occur.

There are options to save a backup of your files like *Auto Save* that saves your file automatically over time, and *Save on Quit*, which saves your blend-file automatically when you exit Blender. In addition to these functions being enabled by default, the *Save on Quit* functionality cannot be disabled.

Note

For your actions, there are options like *Undo*, *Redo* and an *Undo History*, used to roll back from mistakes under normal operation, or return back to a specific action. See Undo & Redo.

Recovering Save Versions

By default Blender keeps an additional backup when saving files. So saving renames the previously saved file with a `.blend1` extension instead of overwriting it.

This file can be used to revert to a previous state.

See Save Versions to configure the number of versions kept.

Recovering Auto Saves

Last Session

Reference

Menu:: *File ▸ Recover ▸ Last Session*

The *Recover Last Session* will open the `quit.blend` file that is saved into the Temporary Directory when you quit Blender under normal operation (see Blender Session). Note that files in your temporary directory may be deleted when you reboot your computer (depending on your system configuration).

Auto Save

Reference

Menu:: *File ▸ Recover ▸ Auto Save*

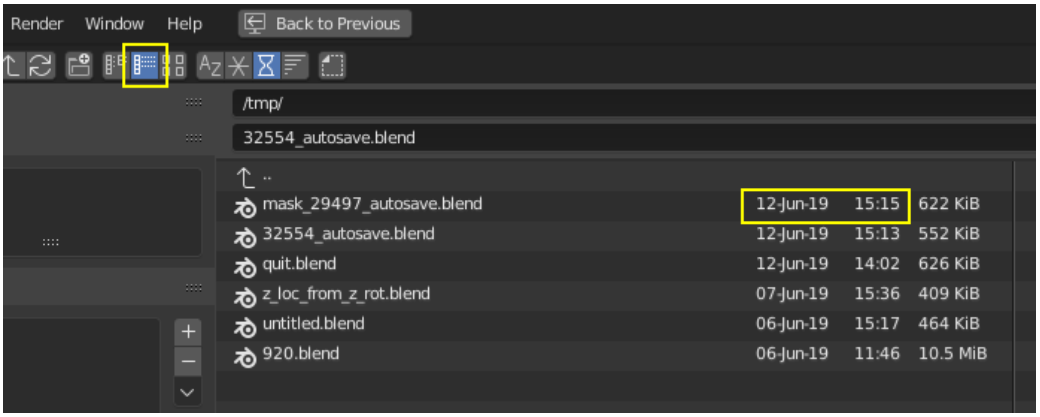
The *Recover Auto Save* allows you to open the *Auto Saved* file. You will have to navigate to your Temporary Directory. The *Auto Saved* files are named using a random number and have a blend extension.

See Auto Save Preferences to configure auto-save.

Trusted Source
: When enabled, Python scripts and drivers that may be included in the file will be run automatically. Enable this only if you created the file yourself, or you trust that the person who gave it to you did not include any malicious code with it. See Python Security to configure default trust options.

Tip

Enable the detailed list view when browsing auto-saved files to show which is the most recent.



File Browser displaying a vertical list.

Warning

When recovering an *Auto Saved* file, any changes made since the last *Auto Save* will be lost. Only one *Auto Saved* file exists for each `.blend` file, i.e. Blender does not keep older versions. Therefore, you will only be able to restore the most recent *Auto Save* file.

Glossary

This page lists definitions for terms used in Blender and this manual.

Action Safe

Area of the screen visible on most devices. Place content inside it to ensure it does not get cut off.

Active

When many items are selected, the last selected item will be the active one. Used in situations where the interface only shows options for one item at a time.

See also selection states.

Aliasing

Rendering artifacts in the form of jagged lines.

Alpha Channel

Additional channel in an image for transparency.

Straight Alpha

Method where RGBA channels are stored as (R, G, B, A) channels, with the RGB channels unaffected by the alpha channel. This is the alpha type used by paint programs such as Photoshop or Gimp, and used in common file formats like PNG, BMP or Targa. So, image textures or output for the web are usually straight alpha.

Premultiplied Alpha

Method where RGBA channels are stored as (R × A, G × A, B × A, A), with the alpha multiplied into the RGB channel.

This is the natural output of render engines, with the RGB channels representing the amount of light that comes toward the viewer, and

alpha representing how much of the light from the background is blocked. The OpenEXR file format uses this alpha type. So, intermediate files for rendering and compositing are often stored as premultiplied alpha.

Conversion (Straight/Premultiplied) Alpha

Conversion between the two alpha types is not a simple operation and can involve data loss, as both alpha types can represent data that the other cannot, though it is often subtle.

Straight alpha can be considered to be an RGB color image with a separate alpha mask. In areas where this mask is fully transparent, there can still be colors in the RGB channels. On conversion to premultiplied alpha, this mask is *applied* and the colors in such areas become black and are lost.

Premultiplied alpha, on the other hand, can represent renders that are both emitting light and letting through light from the background. For example, a transparent fire render might be emitting light, but also letting through all light from objects behind it. On converting to straight alpha, this effect is lost.

Channel Packed

A separate image map is stored for each color and alpha channel. Channel packing is commonly used by game engines to save memory and to optimize memory access.

Ambient Light

The light that comes from the surrounding environment as a whole.

Ambient Occlusion

A ratio of how much Ambient Light a surface point would be likely to receive. If a surface point is under a foot or table, it will end up much darker than the top of someone's head or the tabletop.

Animation

Simulation of motion.

Anti-Aliasing

Is the technique of minimizing Aliasing, by e.g. rendering multiple samples per pixel.

Armature

An Object consisting of Bones. Used to Rig characters, props, etc.

Asset

Curated data-blocks that are meant for reuse, usually contained in an Asset Library. See also Asset Libraries.

Note that there are other meanings of the word “asset” – sometimes this is used more generically, and refers to any “useful thing”, like images, models, materials, and more.

Asset Catalog

Container for assets, similar to what a directory is for files. See also Asset Catalogs.

Asset Library

Directory on drive, registered in the list of asset libraries in the preferences. See also Asset Libraries and Current File Asset Library.

Asset Metadata

Asset-related information, such as its catalog, description, author, preview, and tags. See Asset Details Region.

Attribute

A generic term to describe data stored per-element in a geometry data-block.

Axis

A reference line which defines coordinates along one cardinal direction in n-dimensional space.

Axis Angle

Rotation method where X, Y, and Z correspond to the axis definition, while W corresponds to the angle around that axis, in radians.

Baking

The process of computing and storing the result of a potentially time-consuming calculation so as to avoid needing to calculate it again.

Bevel

The operation to chamfer or bevel edges of an object.

Bézier

A computer graphics technique for generating and representing curves.

Bit Depth

The exponent value (with base two) for how many colors can be represented within a single color channel. A higher bit depth will allow more possible colors, reducing banding, and increasing precision. Yet a higher bit depth will increase memory usage exponentially.

Blend Modes
Color Blend Modes

Methods for blending two colors together.

See also Blend Modes on Krita docs.

Blender Session
Session

The timespan of a Blender instance. The session begins with starting an instance of Blender and ends with closing it.

In some cases, loading a new file may be considered beginning a new session. If so, the documentation should mention that.

Bone

The building block of an Armature. Made up of a Head, Tail and Roll Angle which define a set of local axes and a point of rotation at the

Head. Also see Pose Bone.

Bone Collection

Collection of bones of an Armature, identified by its name. Bone collections can be used to organise bones and toggle their visibility. See Bone Collections.

Boolean

A type of logic dealing with binary true/false states.

See also Boolean Modifier.

Bounding Box

The box that encloses the shape of an object. The box is aligned with the local space of the object.

Bump Mapping

Technique for simulating slight variations in surface height using a grayscale “heightmap” texture.

BVH
Bounding Volume Hierarchy

A hierarchical structure of geometric objects.

See also Bounding Volume Hierarchy on Wikipedia.

Caustics

The optical phenomenon of light concentration focused by specular reflections or refracting objects. In example observable on light passing through a glass of water onto a table or the pattern at the bottom of a swimming pool.

In rendering this refers to diffuse reflected light paths after a glossy or refraction bounce.

See also Caustics on Wikipedia.

Child

An Object that is affected by its Parent.

Chroma
Chrominance

In general, a resulting image color decomposition, where its (*L* or *Y*) luminance channel is separated. There are two different contexts whereas this term is used:

Video Systems

Refers to the general color decomposition resulting in *Y* (Luminance) and *C* (Chrominance) channels, whereas the chrominance is represented by: U = (Blue minus Luminance) and V = (Red minus Luminance).

Matte Compositing

Refers to a point in the color gamut surrounded by a mixture of a determined spectrum of its RGB neighboring colors. This point is called *Chroma key* and this key (a chosen color) is used to create an *Alpha Mask*. The total amount of gamut space for this chrominance point is defined by users in a circular or square-shaped format.

Chromaticities

The coordinates of the Primaries on the CIE 1931 xy chromaticity diagram.

Clamp
Clamping

Limits a variable to a range. The values over or under the range are set to the constant values of the range's minimum or maximum.

Collection

A device for organizing objects. See also Collections.

Color Gamut

A gamut traditionally refers to the volume of color a particular color model/space can cover. In many instances, it is illustrated via a 2D model using CIE Yxy coordinates.

Color Model

A mechanism for representing colors as numbers.

RGB

An additive system where three primaries; red, green, and blue are combined to make other colors.

HSV

Three values often considered as more intuitive (human perception) than the RGB system. In this model, colors are represented as Hue, Saturation, and Value.

HSL

Similar to *HSV* except the colors are represented as Hue, Saturation, and Luminance.

YUV

Luminance-Chrominance standard used in broadcasting analog PAL (European) video.

YCbCr

Luminance-ChannelBlue-ChannelRed component video for digital broadcast use, whose standards have been updated for HDTV and commonly referred to as the HDMI format for component video.

Color Space

A coordinate system in which a vector represent a color value. This way the color space defines three things:

- The exact color of each of the Primaries
- The White Point
- A transfer function

The color spaces supported by Blender depend on the active OCIO config. The default supported color spaces are described in detail here: Default OpenColorIO Configuration

sRGB

A color space that uses the Rec .709 Primaries and a D65 white point, and 2.2 gamma correction value as the transfer function.

Concave Face

Face in which one vertex is inside a triangle formed by other vertices of the face.

See also Convex and concave polygons on Wikipedia.

Constraint

A way of controlling one Object with data from another.

Convex Face

Face where, if lines were drawn from each vertex to every other vertex, all lines would remain in the face. Opposite of a Concave Face.

Coplanar

Refers to any set of elements that are all aligned to the same 2D plane in 3D space.

Crease

Property of an Edge. Used to define the sharpness of edges in Subdivision Surface meshes.

Current File Asset Library

Asset library that is not a directory on drive, but only reflects the assets in the current blend-file. This library is available regardless of the location of the blend-file. See The Current File Asset Library.

Curve

A type of object defined in terms of a line interpolated between Control Vertices. Available types of curves include Bézier, NURBS and Poly.

Curve Segment

The part of a curve connecting two adjacent control points.

Cyclic

Often referring to an object being circular. This term is often associated with Curve.

Data User

An existing Blender object, which is using its own data, or linked data (data owned and controlled by another Blender object).

Dielectric Material

A material for real world objects that are electrical insulators such as plastics, wood, glass, ect. Essentially this summarizes any material that is solid and non metallic.

Diffuse Light

Even, directed light coming off a surface. For most things, diffuse light is the main lighting we see. Diffuse light comes from a specific direction or location and creates shading. Surfaces facing towards the light source will be brighter, while surfaces facing away from the light source will be darker.

Directional Light

The light that has a specific direction, but no location. It seems to come from an infinitely far away source, like the sun. Surfaces facing the light are illuminated more than surfaces facing away, but their location does not matter. A directional light illuminates all objects in the scene, no matter where they are.

Displacement Mapping

A method for distorting vertices based on an image or texture. Similar to Bump Mapping, but instead operates on the mesh's actual geometry.

This relies on the mesh having enough geometry to represent details in the image.

Display Referenced

Refers to an image whose Luminance channel is limited to a certain range of values (usually 0-1). The reason it is called display referenced is because a display cannot display an infinite range of values. So, the term Scene Referenced must go through a transfer function to be converted from one to the other.

DOF

Depth of Field

The distance in front of and behind the subject which appears to be in focus. For any given lens setting, there is only one distance at which a subject is precisely in focus, but focus falls off gradually on either side of that distance, so there is a region in which the blurring is tolerable. This region is greater behind the point of focus than it is in front, as the angle of the light rays change more rapidly; they approach being parallel with increasing distance.

Double Buffer

Technique for rendering and displaying content on the screen. Blender uses two buffers (images) to render the interface, the content of one buffer is displayed while rendering occurs on the other buffer. When rendering is complete, the buffers are switched.

Edge

Straight segment (line) that connects two Vertices, and can be part of a Face.

Edge Loop

Chain of Edges belonging to consecutive Quads. An edge loop ends at a pole or a boundary. Otherwise, it is cyclic.

Edge Ring

Path of all Edges along a Face Loop that share two faces belonging to that loop.

Elastic

Objects that are able to spontaneously return to their original shape after all outside forces are removed from the object.

Elasticity

The amount a material is elastic versus inelastic.

Empty

An Object without any Vertices, Edges or Faces.

Euler
Euler Rotation

Rotation method where rotations are applied to each of the X, Y, Z axes in a specific order.

Euler orders in Blender are most intuitive when read backwards: *XYZ Euler* is similar to rotating around *Local Z* using the *Rotate* tool in the 3D Viewport, followed by *Local Y* and then *Local X*.

F-Curve

A curve that holds the animation values of a specific property.

Face

Mesh element that defines a piece of surface. It consists of three or more Edges.

Face Loop

Chain of consecutive Quads. A face loop stops at a Triangle or N-gon (which do not belong to the loop), or at a boundary. Otherwise, it is cyclic.

Face Normal

The normalized vector perpendicular to the plane that a Face lies in. Each face has its own normal.

Fake User

A special Data User, a program construct that is used to mark an object (e.g. material) to be saved in a blend-file, even when no Real User is using the object. Objects that are not used by any Data User are not included in saved blend-files.

Field of View

The area in which objects are visible to the camera. Also see Focal Length.

Fireflies

Rendering artifacts encountered with path tracing resulting from improbable samples that contribute very high values to pixels.

FK
Forward Kinematics

The process of determining the movement of interconnected segments or bones of a body or model in the order from the parent bones to the child bones. Using forward kinematics on a hierarchically structured object, you can move the upper arm then the lower arm and hand go along with the movement. Without forward kinematics the lower arm and hand would disconnect from upper arm and would move independently in space.

See also Inverse Kinematics.

Focal Length

The distance required by a lens to focus collimated light. Defines the magnification power of a lens. Also see Field of View.

Frame Types

In video compression, a frame can be compressed by several different algorithms. These algorithms are known as *picture types* or *frame types*

and there are three major types: **I**, **P**, and **B** frames.

I-frames

The least compressible but don't require other video frames to decode.

P-frames

Use data from previous frames to decompress and are more compressible than I-frames.

B-frames

Use both previous and forward frames for data reference to get the highest amount of compression.

Gamma

An operation used to adjust the brightness of an image.

See also Gamma correction on Wikipedia.

Geodesic

Relating to the shortest possible path between two points on a curved surface.

Geometric Center

The mean average of the positions of all vertices making up the object.

Gimbal

A pivoted support that allows the rotation of an object about a single axis.

See also Gimbal on Wikipedia.

Gimbal Lock

The limitation where axes of rotation can become aligned, losing the ability to rotate on an axis (typically associated with Euler Rotation).

- See also Gimbal lock on Wikipedia.

- See also Gimbal lock on Stack Exchange.

Global Illumination

A superset of Radiosity and ray tracing. The goal is to compute all possible light interactions in a given scene, and thus, obtain a truly photorealistic image. All combinations of diffuse and specular reflections and transmissions must be accounted for. Effects such as color bleeding and caustics must be included in a global illumination simulation.

Global Space

See World Space.

Glossy Map

See Roughness Map.

HDRI
High Dynamic Range Image

A set of techniques that allow a far greater dynamic range of exposures than normal digital imaging techniques. The intention is to accurately represent the wide range of intensity levels found in real scenes, ranging from direct sunlight to the deepest shadows.

See also HDRI on Wikipedia.

Head

A subcomponent of a Bone. The point of rotation for the bone has X, Y, and Z coordinates measured in the Local Space of the Armature object. Used in conjunction with the Tail to define the local Y axis of the bone in Pose Mode. The larger of the two ends when displayed as an Octahedron.

Hue

A shade of light out of the color spectrum.

IK

Inverse Kinematics

The process of determining the movement of interconnected segments or bones of a body or model in the order from the child bones to the parent bones. Using inverse kinematics on a hierarchically structured object, you can move the hand then the upper and lower arm will automatically follow that movement. Without inverse kinematics the hand would come off the model and would move independently in space.

See also Forward Kinematics.

Interpolation

The process of calculating new data between points of known value, like Keyframes.

IOR
Index of Refraction

A property of transparent materials. When a light ray travels through the same volume it follows a straight path. However, if it passes from one transparent volume to another, it bends. The angle by which the ray is bent can be determined by the IOR of the materials of both volumes.

Keyframe

A frame in an animated sequence drawn or otherwise constructed directly by the animator. In classical animation, when all frames were drawn by animators, the senior artist would draw these frames, leaving the "in between" frames to an apprentice. Now, the animator creates only the first and last frames of a simple sequence (keyframes); the computer fills in the gap.

Keyframing

Inserting Keyframes to build an animated sequence.

Lattice

A type of object consisting of a non-renderable three-dimensional grid of vertices.

See also Lattice Modifier.

Light Bounces

Refers to the reflection or transmission of a light ray upon interaction with a material. See also Light Paths.

Local Space

A 3D coordinate system that originates (for Objects) at the Object Origin. or (for Bones) at the Head of the Bone.

Compare to World Space.

Luminance

The intensity of light either in an image/model channel, or emitted from a surface per square unit in a given direction.

Manifold

Manifold meshes, also called 'water-tight' meshes, define a closed non-self-intersecting volume (see also Non-manifold). A manifold mesh is a mesh in which the structure of the connected faces in a closed volume will always point the normals (and their surfaces) to the outside or to the inside of the mesh without any overlaps. If you recalculate those normals, they will always point at a predictable direction (to the outside or to the inside of the volume). When working with non-closed volumes, a manifold mesh is a mesh in which the normals will always define two different and non-consecutive surfaces. A manifold mesh will always define an even number of non-overlapped surfaces.

MatCap

Stands for "material capture", using an image to represent a complete material including lighting and reflections.

Matte
Mask

A grayscale image used to include or exclude parts of an image. A matte is applied as an Alpha Channel, or it is used as a mix factor when

applying Color Blend Modes.

Mesh

Type of object consisting of Vertices, Edges and Faces.

Micropolygons

A polygon roughly the size of a pixel or smaller.

MIP
Mip-map
Mip-mapping

'MIP' is an acronym of the Latin phrase 'multum in parvo', meaning 'much in little'. Mip-maps are progressively lower resolution representations of an image, generally reduced by half squared interpolations using Anti-Aliasing. Mip-mapping is the process used to calculate lower resolutions of the same image, reducing memory usage to help speed visualization, but increasing memory usage for calculations and allocation. Mip-mapping is also a process used to create small anti-aliased samples of an image used for texturing. The mip-mapping calculations are made by CPUs, but modern graphic processors can be selected for this task and are way faster.

See the mip-map option present in the System Preferences.

MIS
Multiple Importance Sampling

A process of estimating the direction of light rays to improve sampling quality.

See Multiple Importance Sampling and also Importance sampling on Wikipedia.

Modifiers

A non-destructive operation that is applied on top of some sort of data.

Motion Blur

The phenomenon that occurs when we perceive a rapidly moving object. The object appears to be blurred because of our persistence of vision. Simulating motion blur makes computer animation appear more realistic.

Multisampling

Rendering multiple samples per pixel, for Anti-Aliasing.

N-gon

A Face that contains more than four Vertices.

NDOF
3D Mouse

A general term used to describe a 3D mouse, or any input devices which supports more degrees of freedom than a conventional 2D input device, see: Touchpad.

Non-manifold

Non-Manifold meshes essentially define geometry which cannot exist in the real world. This kind of geometry is not suitable for several types of operations, especially those where knowing the volume (inside/outside) of the object is important (refraction, fluids, Boolean operations, or 3D printing, to name a few). A non-manifold mesh is a mesh in which the structure of a non-overlapped surface (based on its connected faces) will not determine the inside or the outside of a volume based on its normals, defining a single surface for both sides, but ended with flipped normals. When working with non-closed volumes, a non-manifold mesh will always determine at least one discontinuity in the normal directions, either by an inversion of a connected loop, or by an odd number of surfaces. A non-manifold mesh will always define an odd number of surfaces.

There are several types of non-manifold geometry:

- Some borders and holes (edges with only a single connected face), as faces have no thickness.

- Edges and vertices not belonging to any face (wire).
- Edges connected to three or more faces (interior faces).
- Vertices belonging to faces that are not adjoining (e.g. two cones sharing the vertex at the apex).

See also: Select Non-Manifold tool.

Nonlinear Animation

Animation technique that allows the animator to edit motions as a whole, not just the individual keys. Nonlinear animation allows you to combine, mix, and blend different motions to create entirely new animations.

Normal

The normalized vector perpendicular to a surface.

Normals can be assigned to vertices, faces and modulated across a surface using Normal Mapping.

See also Normals on Wikipedia.

Normal Mapping

Is similar to Bump Mapping, but instead of the image being a grayscale heightmap, the colors define in which direction the normal should be shifted, the three color channels being mapped to the three directions X, Y and Z. This allows more detail and control over the effect.

NURBS
Non-uniform Rational Basis Spline

A computer graphics technique for generating and representing curves and surfaces.

Object

Container for a type (mesh, curve, surface, metaball, text, armature, lattice, empty, camera, light) and basic 3D transform data (Object Origin).

Object Center
Object Origin

A reference point used to position, rotate, and scale an Object and to define its Local Space coordinates.

Octahedron

An eight-sided figure commonly used to depict the Bones of an Armature.

OpenGL

The graphics system used by Blender (and many other graphics applications) for rendering 3D graphics, often taking advantage of hardware acceleration.

See also OpenGL on Wikipedia.

Operator

An executable action that is completed the moment they're initiated. See Operators as described in the user interface section.

Overscan

The term used to describe the situation. when not all of a televised image is present on a viewing screen.

See also Overscan on Wikipedia.

Panel

A user interface element that contains buttons. Panels are collapsible to hide there contents and can often be rearranged. See Panels as described in the user interface section.

Parent

An Object that affects its Child objects.

Parenting

Creating a Parent-Child relationship between two objects.

Particle System

Technique that simulates certain kinds of fuzzy phenomena, which are otherwise very hard to reproduce with conventional rendering techniques. Common examples include fire, explosions, smoke, sparks, falling leaves, clouds, fog, snow, dust, meteor tails, stars, and galaxies, or abstract visual effects like glowing trails, magic spells. Also used for things like fur, grass or hair.

Phong

Local illumination model that can produce a certain degree of realism in three-dimensional objects by combining three elements: diffuse, specular and ambient for each considered point on a surface. It has several assumptions – all lights are points, only surface geometry is considered, only local modeling of diffuse and specular, specular color is the same as light color, ambient is a global constant.

Pivot Point

The pivot point is the point in space around which all rotation, scaling and mirror transformations are centered.

See also the Pivot Point docs.

Pixel

The smallest unit of information in a 2D raster image, representing a single color made up of red, green, and blue channels. If the image has an Alpha Channel, the pixel will contain a corresponding fourth channel.

Point Cloud

A list of points in 3D space.

Pole

Vertex where three, five, or more edges meet. A vertex connected to one, two, or four edges is not a pole.

Pose Bone

Pose-specific properties of a Bone, such as its location / rotation / scale relative to the Armature's rest pose. Its properties are stored on the Object, and thus can be different for each user of the Armature. The Pose Bone also stores constraints.

Pose Mode

Used for Posing, Keyframing, Weight Painting, Constraining and Parenting the Bones of an Armature.

Posing

Moving, Rotating and Scaling the Pose Bones of an Armature to achieve an aesthetically pleasing pose for a character.

Premultiplied Alpha

See Alpha Channel.

Primaries

In color theory, primaries (often known as primary colors) are the abstract lights, using an absolute model, that make up a Color Space.

Primitive

A basic object that can be used as a basis for modeling more complicated objects.

Procedural Texture

Computer generated (generic) textures that can be configured via different parameters.

Projection

In computer graphics, there are two common camera projections used.

Perspective

A *perspective* view is geometrically constructed by taking a scene in 3D and placing an observer at point *O*. The 2D perspective scene is built by placing a plane (e.g. a sheet of paper) where the 2D scene is to be rendered in front of point *O*, perpendicular to the viewing

direction. For each point *P* in the 3D scene a *PO* line is drawn, passing by *O* and *P*. The intersection point *S* between this *PO* line and the plane is the perspective projection of that point. By projecting all points *P* of the scene you get a perspective view.

Orthographic

In an *orthographic* projection, you have a viewing direction but not a viewing point *O*. The line is then drawn through point *P* so that it is parallel to the viewing direction. The intersection *S* between the line and the plane is the orthographic projection of the point *P*. By projecting all points *P* of the scene you get the orthographic view.

Proxy

For video editing, a proxy is a smaller version of the original file, typically using an optimized video codec and lower resolution version (faster to load) that stands in for the main image or video.

When proxies are built, editing functions like scrubbing and scrolling and compositing is much faster but gives lower resolution and slightly imprecise result.

Quad
Quadrilateral
Quadrangle

Face that contains exactly four Vertices.

Quaternion
Quaternion Rotation

Rotation method where rotations are defined by four values (X, Y, Z, and W). X, Y, and Z also define an Axis, and W an angle, but it is quite different from Axis Angle.

Quaternion values can be interpreted geometrically as defining a point on a unit sphere in 4D space. Moving along any *great circle* of the sphere represents rotating around a fixed axis, with one full circle matching two full rotations.

Radiosity

A global lighting method that calculates patterns of light and shadow for rendering graphics images from three-dimensional models. One of the many different tools which can simulate diffuse lighting in Blender.

See also Radiosity (computer graphics) on Wikipedia.

Random Seed
Seed

Blender uses pseudo random number generators, which produce numbers that appear to be random, but given the same initial condition, they will always produce the exact same sequence of numbers.

This is a critical feature to get reproducible and/or stable effects (otherwise e.g. your hair simulation would change every time you re-run it, without any way to control the outcome).

The **seed** is a number that represents the initial condition of a random generator, if you change its seed, it will produce a new sequence of pseudo-random numbers.

See also Random seed on Wikipedia.

Ray Tracing

Rendering technique that works by tracing the path taken by a ray of light through the scene, and calculating reflection, refraction, or absorption of the ray whenever it intersects an object in the world. More accurate than Scanline, but much slower.

Real User

A Blender object, which is a Data User. Opposite of Fake User, which is only a program construct.

Refraction

The change in direction of a wave due to a change in velocity. It happens when waves travel from a medium with a given Index of Refraction to a medium with another. At the boundary between the

media, the wave changes direction; its wavelength increases or decreases but frequency remains constant.

Render

The process of computationally generating a 2D image from 3D geometry.

Resource

External files such as images, sounds, fonts and volumes files that can be packed into a blend-file.

RGB

A color model based on the traditional primary colors, Red/Green/Blue. RGB colors are also directly broadcasted to most computer monitors.

Rig

A system of relationships that determine how something moves. The act of building of such a system.

Roll
Roll Angle

The orientation of the local X and Z axes of a Bone. Has no effect on the local Y axis as local Y is determined by the location of the Head and Tail.

Rolling Shutter

In real CMOS cameras the sensor is read out with scanlines and hence different scanlines are sampled at a different moment in time. This, for example, make vertical straight lines being curved when doing a horizontal camera pan. See also Rolling Shutter on Wikipedia.

Roughness Map

A grayscale texture that defines how rough or smooth the surface of a material is. This may also be known as a Glossy Map.

Saturation

Also known as colorfulness, saturation is the quantity of hue in the color (from desaturated – a shade of gray – to saturated – brighter colors).

Scanline

Rendering technique. Much faster than Ray Tracing, but allows fewer effects, such as reflections, refractions, motion blur and focal blur.

Scene Referenced

An image whose Luminance channel is not limited.

See also Display Referenced.

Shading

Process of altering the color of an object/surface in the 3D scene, based on its angle to lights and its distance from lights to create a photorealistic effect.

Smoothing

Defines how Faces are shaded. Faces can be either solid (faces are rendered flat) or smooth (faces are smoothed by interpolating the normal on every point of the face).

Specular Light

A light which is reflected precisely, like a mirror. Also used to refer to highlights on reflective objects.

SSS
Subsurface Scattering

Mechanism of light transport in which light penetrates the surface of a translucent object, is scattered by interacting with the material, and exits the surface at a different point. All non-metallic materials are translucent to some degree. In particular, materials such as marble, skin, and milk are extremely difficult to simulate realistically without taking subsurface scattering into account.

Straight Alpha

See Alpha Channel.

Subdiv
Subdivision Surface

A method of creating smooth higher poly surfaces which can take a low polygon mesh as input.

See also Catmull-Clark subdivision surface on Wikipedia.

Subdividing

Technique for adding more geometry to a mesh. It creates new vertices on subdivided edges, new edges between subdivisions and new faces based on new edges. If new edges cross a new vertex is created at their crossing point.

Swing
Swing and Twist

Refers to decomposition of an arbitrary rotation into a sequence of two single axis rotations: a *swing* rotation that aims a chosen axis in its final direction using the shortest possible rotation path, followed by a *twist* rotation around that axis.

This decomposition is available through Driver Variables and inputs of the Transformation constraint. The Damped Track constraint produces a pure *swing* rotation.

In the Quaternion representation the *swing* rotation always has 0 as the X/Y/Z component corresponding to the selected axis, while *twist* always has 0 as the other two components.

Tail

A subcomponent of a Bone. Has X, Y and Z coordinates measured in the Local Space of the armature object. Used in conjunction with the Head to define the local Y axis of a bone in Pose Mode. The smaller of the two ends when displayed as an Octahedron.

Tangent

A line that intersects a surface at exactly one point, a tangent is perpendicular to a Normal.

Tessellation

The tiling of a plane using one or more geometric shapes usually resulting in Micropolygons.

Texture

Specifies visual patterns on surfaces and simulates physical surface structure.

Texture Space

The bounding box to use when using *Generated* mapping to add a Texture to an image.

Timecode

A coded signal on videotape or film giving information about the frame number and time the frame was recorded. Timecodes are used to sync media between different recording devices, including both audio and video.

Title Safe

Area of the screen visible on all devices. Place text and graphics inside this area to make sure they do not get cut off.

Topology

The arrangement of *Vertices*, *Edges*, and *Faces* which define the shape of a mesh. See Vertex, Edge, and Face.

Transform
Transformation

The combination of location, rotation, and scale. Can be expressed in World Space or Local Space.

Transformation Matrix

A matrix that is used to represent the Transformation of an item.

Triangle

Face with exactly three Vertices.

UV Map

Defines a relation between the surface of a mesh and a 2D texture. In detail, each face of the mesh is mapped to a corresponding face on the texture. It is possible and often common practice to map several faces of the mesh to the same or overlapping areas of the texture.

Value

The brightness of the color (dark to light).

Vertex
Vertices

A point in 3D space containing a location. Vertices are the terminating points of Edges.

Vertex Group

Collection of Vertices. Vertex groups are useful for limiting operations to specific areas of a mesh.

Voxel

A cubic 3D equivalent to the square 2D pixel. The name is a combination of the terms “Volumetric” and “Pixel”. Used to store smoke and fire data from physics simulations.

Walk Cycle

In animation, a walk cycle is a character that has just the walking function animated. Later on in the animation process, the character is placed in an environment and the rest of the functions are animated.

Weight Painting

Assigning Vertices to a Vertex Group with a weight of 0.0 - 1.0.

White Point

A reference value for white light when all primaries of a color model are combined evenly.

A white point is defined by a set of CIE illuminates which correspond to a color temperature. For example, D65 corresponds to 6500 K light and D70 corresponding to 7000 K.

World Space

A 3D coordinate system that originates at a point at the origin of the world. Compare to Local Space.

Z-buffer

Raster-based storage of the distance measurement between the camera and the surface points. Surface points which are in front of the camera have a positive Z value and points behind have negative values. The Z-depth map can be visualized as a grayscale image.

Index

Symbols | A | B | C | D | E | F | G | H | I | K | L | M | N | O | P | Q | R | S | T | U | V | W | Z

Symbols

3D Mouse

A

Action Safe
Active
Add-ons, [1]
Add-ons Extensions
Aliasing
Alpha Channel
Ambient Light
Ambient Occlusion
Animation
Anti-Aliasing
Armature
Asset
Asset Catalog
Asset Library
Asset Metadata
Attribute
Axis
Axis Angle

B

Baking
Bevel
Bézier
Bit Depth
Blend Modes
Blender Session
Bone
Bone Collection
Boolean
Bounding Box
Bounding Volume Hierarchy
bssrdf()
    built-in function
built-in function
    bssrdf()
Bump Mapping
BVH

C

Caustics
Child
Chroma
Chromaticities
Chrominance
Clamp
Clamping
Collection
Color Blend Modes
Color Gamut
Color Model
Color Space
Compositor Nodes
- Alpha Convert
- Alpha Over
- Anti-Aliasing
- Bilateral Blur
- Blur
- Bokeh Blur
- Bokeh Image
- Box Mask
- Brightness/Contrast
- Channel Key
- Chroma Key
- Color Balance
- Color Correction
- Color Key
- Color Ramp
- Color Space
- Color Spill
- Combine Color
- Combine XYZ
- Composite
- Corner Pin

Concave Face
Constraint
- Bone Constraints
- Object Constraints
- Rigid Body Constraints

Convex Face
Coplanar
Crease
Current File Asset Library
Curve
Curve Segment
Cyclic

Crop
Cryptomatte
Cryptomatte (Legacy)
Defocus
Denoise
Despeckle
Difference Key
Dilate/Erode
Directional Blur
Displace
Distance Key
Double Edge Mask
Ellipse Mask
Exposure
File Output
Filter
Flip
Gamma
Glare
Hue Correct
Hue/Saturation/Value
ID Mask
Image
Inpaint
Invert Color
Keying
Keying Screen
Kuwahara
Lens Distortion
Levels
Luminance Key
Map Range
Map UV
Map Value
Mask
Math
Mix

Movie Clip
Movie Distortion
Normal
Normalize
Pixelate
Plane Track Deform
Posterize
Render Layers
RGB
RGB Curves
RGB TO BW
Rotate
Scale
Scene Time
Separate Color
Separate XYZ
Set Alpha
Split
Stabilize 2D
Sun Beams
Switch
Switch View
Texture
Time Curve
Tone Map
Track Position
Transform
Translate
Vector Blur
Vector Curves
Viewer
Z Combine

D

Data User

Directional Light

Depth of Field
Dielectric Material
Diffuse Light
Displacement Mapping
Display Referenced
DOF
Double Buffer

E

Edge
Edge Loop
Edge Ring
Editors
- 3D Viewport
- Asset Browser
- Compositor
- Dope Sheet
- Drivers Editor
- File Browser
- Geometry Node Editor
- Graph Editor
- Image Editor
- Info Editor
- Movie Clip Editor
- NLA Editor
- Outliner
- Preferences
- Properties
- Python Console
- Shader Editor
- Spreadsheet
- Text Editor
- Texture Node Editor
- Timeline
- UV Editor
- Video Sequencer

Elastic
Elasticity
Empty
Euler
Euler Rotation
Extensions, [1]

F

- **F-Curve**
- F-Curve Modifiers
  - Built-in Function Modifier
  - Cycles Modifier
  - Envelope Modifier
  - Generator Modifier
  - Limits Modifier
  - Noise Modifier
  - Stepped Interpolation Modifier
- **Face**
- **Face Loop**
- **Face Normal**
- **Fake User**
- **Field of View**
- **Fireflies**
- **FK**
- **Focal Length**
- Force Fields
  - Boid
  - Charge
  - Curve Guide
  - Drag
  - Fluid Force
  - Force
  - Harmonic
  - Lennard-Jones
  - Magnetic
  - Texture
  - Turbulence
  - Vortex
  - Wind
- **Forward Kinematics**
- **Frame Types**

G

- **Gamma**
- **Geodesic**
- **Geometric Center**
- Geometry Nodes
  - 3D Cursor
  - Accumulate Field
  - Active Camera
  - Active Element
  - Align Euler to Vector
  - Align Rotation to Vector
  - Arc
  - Attach Hair Curves to Surface
  - Attribute Reference
  - Attribute Statistic
  - Axis Angle to Rotation
- **Gimbal**
- **Gimbal Lock**
- **Global Illumination**
- **Global Space**
- **Glossy Map**
- Grease Pencil Modifiers
  - Armature Modifier
  - Array Modifier
  - Build Modifier
  - Dot Dash Modifier
  - Envelope Modifier
  - Hook Modifier
  - Hue/Saturation Modifier
  - Lattice Modifier
  - Length Modifier
  - Line Art Modifier

- Axis to Rotation
- Bake
- Bézier Segment
- Blackbody
- Blend Hair Curves
- Blur Attribute
- Boolean
- Boolean Math
- Bounding Box
- Braid Hair Curves
- Brick Texture
- Capture Attribute
- Checker Texture
- Clamp
- Clump Hair Curves
- Collection Info
- Color
- Color Ramp
- Combine Color, [1]
- Combine Matrix
- Combine Transform
- Combine XYZ
- Compare
- Cone
- Convex Hull
- Corners of Edge
- Corners of Face
- Corners of Vertex
- Create Guide Index Map
- Cube
- Curl Hair Curves
- Curve Circle
- Curve Handle Position
- Curve Info
- Curve Length
- Curve Line
- Curve of Point
- Mirror Modifier
- Multiple Strokes
- Noise Modifier
- Offset Modifier
- Opacity Modifier
- Outline Modifier
- Shrinkwrap Modifier
- Simplify Modifier
- Smooth Modifier
- Subdivide Modifier
- Texture Mapping Modifier
- Thickness Modifier
- Time Offset Modifier
- Tint Modifier
- Vertex Weight Angle Modifier
- Vertex Weight Proximity Modifier

Grease Pencil Visual Effects

- Blur Visual Effect
- Colorize Visual Effect
- Flip Visual Effect
- Glow Visual Effect
- Pixelate Visual Effect
- Rim Visual Effect
- Shadow Visual Effect
- Swirl Visual Effect
- Wave Distortion Visual Effect

Curve Root
Curve Segment
Curve Spiral
Curve Tangent
Curve Tilt
Curve Tip
Curve to Mesh
Curve to Points
Curves to Grease Pencil
Cylinder
Deform Curves on Surface
Delete Geometry
Dial Gizmo
Displace Hair Curves
Distribute Points in Volume
Distribute Points on Faces
Domain Size
Dual Mesh
Duplicate Elements
Duplicate Hair Curves
Edge Angle
Edge Neighbors
Edge Paths to Curves
Edge Paths to Selection
Edge Split
Edge Vertices
Edges of Corner
Edges of Vertex
Edges to Face Groups
Endpoint Selection
Euler to Rotation
Evaluate at Index
Evaluate on Domain
Extrude Mesh
Face Area
Face Group Boundaries

Face Neighbors
Face of Corner
Face Set
Fields
Fill Curve
Fillet Curve
Flip Faces
Float Curve
Float To Integer
For Each Element
Frizz Hair Curves
Gabor Texture
Generate Hair Curves
Geometry Proximity
Geometry to Instance
Gizmos
Gradient Texture
Grease Pencil to Curves
Grid
Hair
Hair Attachment Info
Hair Curves Noise
Handle Type Selection
Hash Value
Ico Sphere
ID
Image
Image Info
Image Texture
Index
Index of Nearest
Index Switch
Inspection
Instance on Points
Instance Rotation
Instance Scale
Instance Transform

Instances
Instances to Points
Integer
Integer Math
Interpolate Curves
Interpolate Hair Curves
Invert Matrix
Invert Rotation
Is Edge Smooth
Is Face Planar
Is Face Smooth
Is Spline Cyclic
Is Viewport
Join Geometry
Join Strings
Linear Gizmo
Magic Texture
Map Range
Material
Material Index
Material Selection
Math
Matrix Determinant
Menu Switch
Merge by Distance
Merge Layers
Mesh Boolean
Mesh Circle
Mesh Island
Mesh Line
Mesh to Curve
Mesh to Points
Mesh to Volume
Mix
Mix Color
Mix Vector
Mouse Position

Multiply Matrices
Musgrave Texture
Named Attribute
Noise Texture
Normal
Object Info
Offset Corner in Face
Offset Point in Curve
Pack UV Islands
Points
Points of Curve
Points to Curves
Points to Vertices
Points to Volume
Position
Project Point
Quadratic Bézier
Quadrilateral
Quaternion to Rotation
Radius
Random Value
Raycast
Realize Instances
Redistribute Curve Points
Remove Named Attribute
Repeat
Replace Material
Replace String
Resample Curve
Restore Curve Segment
Length
Reverse Curve
RGB Curves
Roll Hair Curves
Rotate Euler
Rotate Hair Curves
Rotate Instances

Rotate Rotation
Rotate Vector
Rotation
Rotation to Euler
Rotation to Quaternion
Sample Curve
Sample Index
Sample Nearest
Sample Nearest Surface
Sample UV Surface
Scale Elements
Scale Instances
Scene Time
Selection
Self Object
Separate Color, [1]
Separate Components
Separate Geometry
Separate Matrix
Separate Transform
Separate XYZ
Set Curve Normal
Set Curve Radius
Set Curve Tilt
Set Face Set
Set Geometry Name
Set Hair Curve Profile
Set Handle Positions
Set Handle Type
Set ID
Set Instance Transform
Set Material
Set Material Index
Set Point Radius
Set Position
Set Selection
Set Shade Smooth

Set Spline Cyclic
Set Spline Resolution
Set Spline Type
Shortest Edge Paths
Shrinkwrap Hair Curves
Simulation
Slice String
Smooth Hair Curves
Sort Elements
Special Characters
Spline Length
Spline Parameter
Spline Resolution
Split To Instances
Star
Store Named Attribute
Straighten Hair Curves
String
String Length
String to Curves
Subdivide Curve
Subdivide Mesh
Subdivision Surface
Switch
Tools
Transform Direction
Transform Geometry
Transform Gizmo
Transform Point
Translate Instances
Transpose Matrix
Triangulate
Trim Curve
Trim Hair Curves
UV Sphere
UV Unwrap
Value

Value to String
Vector
Vector Curves
Vector Math
Vector Rotate
Vertex Neighbors
Vertex of Corner
Viewer
Viewport Transform
Volume Cube
Volume to Mesh
Voronoi Texture
Warning
Wave Texture
White Noise Texture

H

HDRI
Head
High Dynamic Range Image
Hue

I

IK
Index of Refraction
Interpolation
Inverse Kinematics
IOR

K

Keyframe
Keyframing

L

Lattice
Light Bounces

- Licenses
- **Local Space**
- **Luminance**

M

- **Manifold**
- **Mask**
- **MatCap**
- **Matte**
- **Mesh**
- **Micropolygons**
- **MIP**
- **Mip-map**
- **Mip-mapping**
- **MIS**
- Modeling
- Modeling Modifiers
  - Armature Modifier
  - Array Modifier
  - Bevel Modifier
  - Boolean Modifier
  - Build Modifier
  - Cast Modifier
  - Cloth Modifier
  - Collision Modifier
  - Curve Modifier
  - Data Transfer Modifier
  - Decimate Modifier
  - Displace Modifier
  - Dynamic Paint Modifier
  - Edge Split Modifier
  - Explode Modifier
  - Fluid Modifier
  - Geometry Nodes Modifier
  - Hook Modifier
  - Laplacian Deform Modifier
- Modeling Transform
- **Modifiers**
  - F-Curve Modifiers
  - Grease Pencil Modifiers
  - Modeling Modifiers
  - Video Sequencer Modifiers
- **Motion Blur**
- **Multiple Importance Sampling**
- **Multisampling**

Lattice Modifier
Mask Modifier
Mesh Cache Modifier
Mesh Deform Modifier
Mesh Sequence Cache Modifier
Mesh to Volume
Mirror Modifier
Multiresolution Modifier
Normal Edit Modifier
Ocean Modifier
Particle Instance Modifier
Particle System Modifier
Remesh Modifier
Screw Modifier
Shrinkwrap Modifier
Simple Deform Modifier
Skin Modifier
Smooth By Angle Modifier
Smooth Corrective Modifier
Smooth Laplacian Modifier
Smooth Modifier
Soft Body Modifier
Solidify Modifier
Subdivision Surface Modifier
Surface Deform Modifier
Triangulate Modifier
UV Project Modifier
UV Warp Modifier
Vertex Weight Edit Modifier
Vertex Weight Mix Modifier
Vertex Weight Proximity Modifier

- Volume Displace
- Volume to Mesh
- Warp Modifier
- Wave Modifier
- Weighted Normal Modifier
- Weld Modifier
- Wireframe Modifier

N

- **N-gon**
- **NDOF**
- Nodes
  - Compositing Nodes
  - Geometry Nodes
  - Shader Nodes
  - Texture Nodes
- **Non-manifold**
- **Non-uniform Rational Basis Spline**
- **Nonlinear Animation**
- **Normal**
- **Normal Mapping**
- **NURBS**

O

- **Object**
- **Object Center**
- Object Constraints
  - Action Constraint
  - Armature Constraint
  - Camera Solver Constraint
  - Child Of Constraint
  - Clamp To Constraint
  - Copy Location Constraint
  - Copy Rotation Constraint
  - Copy Scale Constraint
  - Copy Transforms Constraint
  - Damped Track Constraint
  - Floor Constraint
- **Object Origin**
- **Octahedron**
- **OpenGL**
- **Operator**
- **Overscan**

Follow Path Constraint
Follow Track Constraint
Inverse Kinematics Constraint
Limit Distance Constraint
Limit Location Constraint
Limit Rotation Constraint
Limit Scale Constraint
Locked Track Constraint
Maintain Volume Constraint
Object Solver Constraint
Pivot Constraint
Shrinkwrap Constraint
Spline IK Constraint
Stretch To Constraint
Track To Constraint
Transform Cache Constraint
Transformation Constraint

P

Panel
Parent
Parenting
Particle System
Phong
Pivot Point
Pixel
Point Cloud
Pole
Pose Bone
Pose Mode
Posing
Premultiplied Alpha
Primaries
Primitive
Procedural Texture
Projection
Proxy

Q

Quad
Quadrangle
Quadrilateral
Quaternion
Quaternion Rotation

R

Radiosity
Random Seed
Ray Tracing
Real User
Refraction
Render
Resource
RGB
Rig
Rigid Body Constraints
    Fixed Constraint
    Generic Constraint
    Generic Spring Constraint
    Hinge Constraint
    Motor Constraint
    Piston Constraint
    Point Constraint
    Slider Constraint
Roll
Roll Angle
Rolling Shutter
Roughness Map

S

Saturation
Scanline
Scene Referenced
Seed
Session
Shader Nodes
    Combine XYZ
    Separate XYZ
Smoothing
Specular Light
SSS
Straight Alpha
Subdiv
Subdividing
Subdivision Surface
Subsurface Scattering

Shading

Swing
Swing and Twist

T

Tags
Tail
Tangent
Tessellation
Texture
Texture Space
Timecode

Title Safe
Topology
Transform
    Modeling Transform
Transformation
Transformation Matrix
Triangle

U

User Interface

UV Map

V

Value
Vertex
Vertex Group
Vertices
Video Sequencer Modifiers
    Brightness/Contrast Modifier
    Color Balance Modifier
    Curves Modifier
    Hue Correct Modifier
    Mask Modifier
    Sound Equalizer Modifier
    Tone Map Modifier
    White Balance Modifier

Voxel

W

Walk Cycle
Weight Painting
White Point
World Space

Z

Z-buffer

Contribute Documentation

The Blender Manual is a community driven effort to which anyone can contribute. Whether you like to fix a tiny spelling mistake or rewrite an entire chapter, your help with the Blender manual is most welcome!

If you find an error in the documentation, please report the problem.

Get involved in discussions through any of the project Contacts.

Getting Started

The following guides lead you through the process.

- Installing Dependencies
- Building the Manual
- Editing the Manual
- Pull Requests

Where to help

- Todo List

Guidelines

- Writing Style Guide
- Markup Style Guide
- Commit Guidelines
- Templates
- Maintenance

Translations

- Contribute
- Style Guide

Contacts

Project Page
: An overview of the documentation project.

Documentation Forum
: A forum based discussions on writing and translating documentation. This includes the user manual, Wiki, release notes, and code docs.

Chat
: `#docs` channel for informal discussions in real-time.

Project Workboard
: Manage tasks such as bugs, todo lists, and future plans.

Installing Dependencies

This section documents how to install the software used to generate the manual. The installation is different for each operating system, instructions have been written for:

- Linux
- macOS
- Windows

Installation on Linux

This guide covers the following topics:

1. Installing Dependencies
2. Downloading the Repository
3. Setting up the Build Environment

Installing Dependencies

Below are listed the installation commands for popular Linux distributions.

For the appropriate system, run the command in a terminal:

Debian/Ubuntu:

```
sudo apt-get install python3 python3-pip git git-lfs
git lfs install --skip-repo
```

Redhat/Fedora:

```
sudo dnf install python python-pip git git-lfs
git lfs install --skip-repo
```

Arch Linux:

```
sudo pacman -S python python-pip git git-lfs
git lfs install --skip-repo
```

Downloading the Repository

Simply check out the Blender Manual's repository using:

```
cd ~
git clone https://projects.blender.org/blender/blender-
manual.git
```

The repository will now be downloaded which may take a few minutes depending on your internet connection.

Setting up the Build Environment

Tip

It is recommended to setup and activate a virtual Python environment where dependencies will be installed:

```
python3 -m venv .venv
source .venv/bin/activate
```

Repeat the `source .venv/bin/activate` command to re-activate the virtual environment, whenever you open a new terminal to build the documentation.

This step may be required on some distributions that enforce PEP 668.

- Open a Terminal window.
- Enter the `blender-manual` folder which was just added by `git clone`:

  ```
  cd ~/blender-manual
  ```

- Inside that folder is a file called `requirements.txt` which contains a list of all the dependencies we need. To install these dependencies, we can use the `pip3` command:

  ```
  pip3 install -r requirements.txt
  ```

Note

Every now and then you may want to make sure your dependencies are up to date using:

```
pip3 install -r requirements.txt --upgrade
```

Installation on macOS

This guide covers the following topics:

1. Installing Dependencies
2. Downloading the Repository
3. Setting up the Build Environment

Note

This guide relies heavily on command-line tools. It assumes you are the least familiar with the macOS Terminal application.

Installing Dependencies

Install those packages or make sure you have them in your system.

- PIP
- Git
- Git LFS

When using Homebrew, run the following commands in the terminal:

```
python3 -m ensurepip
brew install git git-lfs
git lfs install
```

Downloading the Repository

Simply check out the Blender Manual's repository using:

```
cd ~
git clone https://projects.blender.org/blender/blender-
manual.git
```

The repository will now be downloaded which may take a few minutes depending on your internet connection.

Setting up the Build Environment

Tip

It is recommended to setup and activate a virtual Python environment where dependencies will be installed:

```
python3 -m venv .venv
source .venv/bin/activate
```

Repeat the `source .venv/bin/activate` command to re-activate the virtual environment, whenever you open a new terminal to build the documentation.

- Open a Terminal window.
- Enter the `blender-manual` folder which was just added by `git clone`:

  ```
  cd ~/blender-manual
  ```

- Inside that folder is a file called `requirements.txt` which contains a list of all the dependencies we need. To install these dependencies, we can use the `pip` command:

  ```
  pip install -r requirements.txt
  ```

Note

Every now and then you may want to make sure your dependencies are up to date using:

```
pip install -r requirements.txt --upgrade
```

Installation on Windows

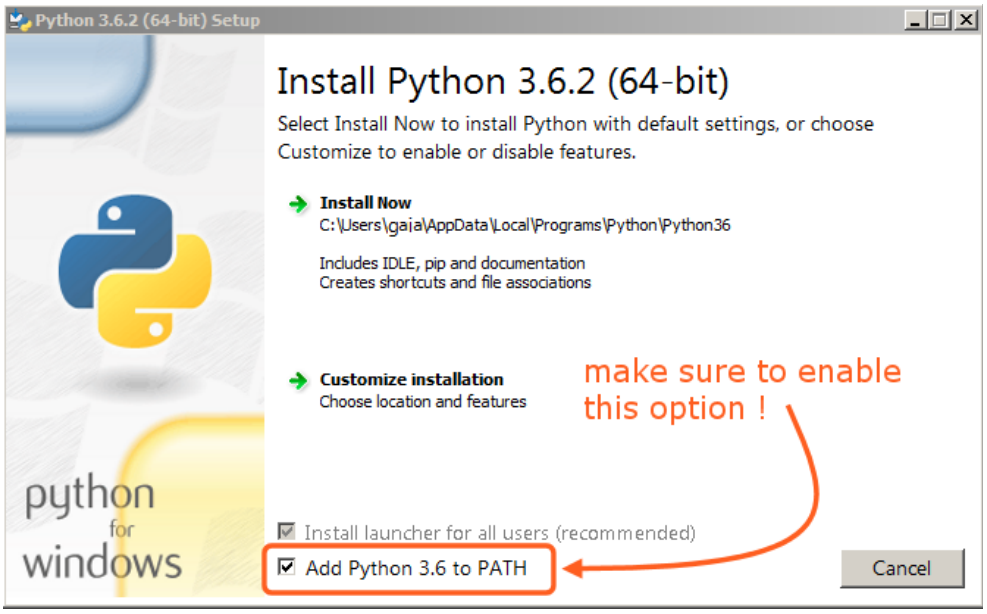
This guide covers the following topics:

1. Installing Python (used to “convert” the source files to HTML)
2. Installing Git and Downloading the Repository
3. Setting up the Build Environment

Installing Python

1. Download the Python installation package for Windows. In this guide version 3.9.x is used.

2. Install Python with the installation wizard. Please make sure that you enable the “Add Python to PATH” option:



   The option must be enabled so you can build the manual with the make script.

   All other settings can remain as set by default.

Installing Git and Downloading the Repository

In this guide, we will use the official Git client, though any Git client will do.

1. Download and install Git for Windows.

2. Simply check out the Blender Manual's repository using:

   ```
   cd ~
   git lfs install
   git clone https://projects.blender.org/blender/blender-
   manual.git
   ```

3. The repository will now be downloaded which may take a few minutes depending on your internet connection.

Note

This process can be completed using a graphical Git client, in that case you will just use the repository address in the URL field provided by your client:

```
https://projects.blender.org/blender/blender-manual.git
```

Setting up the Build Environment

Tip

It is recommended to setup and activate a virtual Python environment where dependencies will be installed:

```
python3 -m venv .venv
.venv/Scripts/activate
```

Repeat the `.venv/Scripts/activate` command to re-activate the virtual environment, whenever you open a new terminal to build the

documentation.

- Open a Command Prompt. (Run as Administrator)
- Enter the `blender-manual` folder which was just added by `git clone`:

  ```
  cd C:\blender-manual
  ```

- Inside that folder is a file called `requirements.txt` which contains a list of all the dependencies we need. Install all the dependencies using Python's `pip` command:

  ```
  pip install -r requirements.txt
  ```

- If all goes well, you should see the following message when it is finished:

  ```
  Successfully installed Jinja2 MarkupSafe Pygments Sphinx
  docutils sphinx-rtd-theme Cleaning up...
  ```

During the setup, some warnings may be shown, but do not worry about them. However, if any errors occur, they may cause some problems.

Note

Every now and then you may want to make sure your dependencies are up to date using:

```
pip install -r requirements.txt --upgrade
```

Building the Manual

Converting the RST-files into pretty HTML pages is simple.

Open a terminal or Command Prompt in the `~/blender-manual` directory and simply run:

```
make
```

Tip

On Microsoft Windows you can simply open the `make.bat` file to easily run the command without having to open the Command Prompt and typing commands.

This is the command you should use when building the docs, however, other commands are available by typing `make help`. This command will convert the RST-files into HTML pages and automatically open your default web browser to view the result. The command will continue to run and watch for changes made to the RST-files and refresh the HTML pages as necessary.

Note

The converted pages can also be viewed manually by browsing the build directory: `~/blender-manual/build/html`. For example to open the home page, open `build/html/index.html` to read the manual.

The building process may take several minutes the first time (or after any major changes), but for subsequent changes it should only take a few seconds.

Editing the Manual

You can modify the manual by editing local text files. These files are kept in sync with those online via a repository, based on this the server will update the online manual.

The manual is written in the reStructuredText (RST) markup language and can be edited using a plain text editor. For a local preview, you convert (build) the manual source files from RST into HTML web pages.

Update

Firstly, make sure that your local copy of the manual is up to date with the online repository using:

```
make update
```

Writing

You can now edit the documentation files, which are the `.rst` files inside the `manual` folder using a text editor of your choice.

Be sure to check the Writing Style Guide for conventions and Markup Style Guide to learn how to write in the reStructuredText markup language.

Happy writing!

Bigger Changes

If you are going to add or overhaul a section, be sure to check carefully that it does not already exist. In some places, the docs are so disorganized that sections may be duplicated or in a strange location. In the case that you find a duplicate or out of place section, create an issue explaining the issue, and optionally include a revision (actual changes).

Before you make any edits that are not simple and plainly justified (for example, moving folders around), you should verify with a manual maintainer that your contribution is along the community's vision for the manual. This ensures the best use of your time and good will as it is otherwise possible that, for some reason, your changes will conflict and be rejected or need time-consuming review. For example, another person may be already working on the section you wish to change, the section may be scheduled for deletion or to be updated according to a planned change to Blender.

Communicating early and frequently is the key to have a productive environment, to not waste people's effort and to attain a better Blender manual as a result.

Getting Help/Answers

If you are in doubt about functionality that you wish to document, you should pose your questions to the Blender developers responsible for that area or ask at the unofficial user support channel in the `#docs` or `#blender-coders` channel in Chat.

Blender itself has a system of module owners with developer and artist members who are responsible for the design and maintenance of the assigned Blender areas. See the module pages for more information.

Preview

To view your changes, build the manual as instructed. Keep in mind that you can also build only the chapter you just edited to view it quickly. Open the generated `.html` files inside the `build/html` folder using your web browser, or refresh the page if you have it open already.

Upload

When you are happy with your changes, you can upload them, so that they will be in the online manual. At first, this is done by submitting patches so

that someone can review your changes and give you feedback. After, you can commit your changes directly. This process is described in detail in the next section.

Pull Requests

This page describes the tools used for code contribution and review.

Reviews are a key measure to assure changes are of a good quality. They help preventing bugs, design consistencies, or potential maintenance problems. And having your work reviewed also generally keeps you on your toes.

Note

Writers who have been given commit access can commit to the main repository without needing to fork the repository.

See Commit Guidelines if this applies to you.

One Time Setup

This assumes you have the Blender manual repository already checked out on your computer, following the install instructions.

Fork

1. Go to Blender repository and click the Fork button.
2. Confirm the fork with the default settings.
3. Now you will have to add your personal fork as a remote in your local git repository. Click *SSH* to see the correct URL, and then add it like this:

   ```
   git remote add me git@projects.blender.org:
   <USERNAME>/blender-manual.git
   ```

Note

In order to push to the fork repository, you need an SSH key. If you don't already have the file `~/.ssh/id_rsa.pub`, there's a simple command to generate such keys which works on Linux, macOS, and in Git Bash on Windows:

```
ssh-keygen
```

This command will generate a private key id_rsa and a public key id_rsa.pub in `~/.ssh`. The private key must never be shown or sent to anyone else to avoid compromising your account, but the public key is safe to share.

The contents of `~/.ssh/id_rsa.pub` can be copied and pasted into the account settings on projects.blender.org, after clicking "Add Key". Any name for the SSH key is ok.

Workflow

The workflow for working with pull requests can be found in the Blender Developer's Documentation.

Note, some text in the above guideline is focused on the main Blender repository, however, the workflow is the same for any Blender project.

Guidelines for Reviewers

- The pull request text should be usable as the git commit message (see the guidelines for details).
- Be explicit when some changes are to be addressed before committing, without the need for a review iteration.
- If the pull request is not approved the author is expected to make another iteration.

- If the change needs agreement on the design task first, put the pull request on hold by adding a `WIP:` prefix in the title, indicating the author considers the pull request not ready to be merged. No review is expected unless the author specifically asks for it.
- Writers are expected to reply to pull requests in 3 working days.
- Add relevant modules/projects to the labels.
- Encourage new writes to do review, it's a good way to learn and important to grow the project.

Tips

- To get the patch file, add .patch to the end of the URL of the pull request. Example:

  ```
  https://projects.blender.org/blender/blender-manual/pulls/104892.patch
  ```

- Checkout a pull request into a detached head (not leaving behind a branch). Example:

  ```
  git fetch -q origin +refs/pull/104892/head: ; git checkout -qf FETCH_HEAD
  ```

Todo List

This page provides a list of changes that need to be made to the manual. This is a great place for new contributors to start but also check the documentation workboard.

Writing Style Guide

Primary Goals

The main goals for this manual are as follows:

User Focused
: The manual is written for people educated in computer graphics, who understand the basics of 3D and/or know other 3D software. While some areas of computer graphics are highly technical, this manual shall be kept understandable by non-technical users.

Complete
: The manual provides detailed functional description of all features, tools and options in Blender. While there is a canonical source of truth for each of Blender's key areas, this does not mean we have to document every small detail. The manual should provide information on what a feature is, how to use it, and its purpose. More background information should be provided when necessary to give deeper understanding of a 3D pipeline.

Concise
: Computer graphics is an incredibly interesting field, there are many rules, exceptions to the rules, and interesting details. Expanding into details can add unnecessary content, so keep the text concise and relevant to the topic at hand.

Maintainable
: Keep in mind that Blender has frequent releases, so try to write content that will not have to be redone the moment some small change is made. This also helps a small documentation community to maintain the manual.

Content Guidelines

In order to maintain a consistent writing style within the manual, please keep this page in mind and only deviate from it when you have a good reason to do so.

Rules of thumb:

- Spell checking is *strongly* recommended.
- Use American English (e.g: modeling and not modelling, color and not colour) also for formatting numbers (e.g: 2,718.28 and not 2 718,28).
- Take care about grammar, appropriate wording and use simple English.
- Keep sentences short and clear, resulting in text that is easy to read, objective and to the point.
- Including why or how an option might be useful is a good idea.
- If you are unsure about how a feature works, ask someone else or find out who developed it and ask them.

To be avoided:

- Avoid writing in first person perspective, about yourself or your own opinions.

- Avoid weasel words and being unnecessarily vague, e.g:

  "Reloading the file will probably fix the problem"
  "Most people do not use this option because …"

- Avoid including specific details such as:

  "Blender has 23 different kinds of modifiers."
  "Enabling previews adds 65536 bytes to the size of each blend-file (unless it is compressed)."

  These details are not useful for users to memorize and they become quickly outdated.

- Avoid documenting bugs.

  Blender often has 100's of bugs fixed between releases, so it is not realistic to reference even a fraction of them from the manual, while

keeping this list up to date.

Issues that are known to the developers and are not going to be resolved before the next release can be documented as *Known Limitations*. In some cases, it may be best to include them in the troubleshooting section.

- Avoid product placements, i.e. unnecessarily promoting software or hardware brands. Keep content vendor-neutral where possible.

- Avoid technical explanations about the mathematical/algorithmic implementation of a feature if there is a simpler way to explain it.

  (E.g. explaining how mesh smoothing algorithms work is unnecessary, but the blending types of a Mix node do need a mathematical explanation.)

- Avoid repetition of large portions of text. Simply explain it once, and from then on refer to that explanation.

  For general terminology, consider defining a `:term:` in the glossary.

- Avoid enumerations of similar options, such as listing every preset or every frame rate in a select menu.

  Their contents may be summarized or simply omitted. – Such lists are only showing what is already *obvious* in the interface and end up being a lot of text to read and maintain.

- Avoid documenting changes in Blender between releases, that is what the release notes are for. We only need to document the current state of Blender.

- Unless the unit a value is measured in is obscure and unpredictable, there is no need to mention it.

- Do not simply copy the tooltips from Blender. – People will come to the manual to learn *more* than is provided by the UI.

As a last resort you can add comment (which is not shown in the HTML page, but useful for other editors):

```
.. TODO, how does this tool work? ask Joe Blogg's.
```

Glossary

This section is specifically about the Glossary section, where we define common terms in Blender and computer graphics.

Rules of thumb:

- Define the term before providing any further information.
- Avoid using constructs such as “it is” or “xyz is” before the definition.
- Avoid repeating the term immediately or using it in the definition.
- Avoid adding terms not found in Blender’s interface or the manual.
- Avoid overly long entries. If an explanation of a complex term is needed, supplement with external links.
- Avoid duplicating documentation; if explaining the term is the primary focus of another section of the manual (e.g. if the term is the name of a tool), either just link to that section, or avoid creating a glossary entry entirely.
- URL references are to be added at the end, formatted as follows, e.g:

  ```
  See also `OpenGL <https://en.wikipedia.org/wiki/OpenGL>`__
  on Wikipedia.
  ```

Examples

This entry:

```
Displacement Mapping
   Uses a grayscale heightmap, like Bump Mapping,
```

```
   but the image is used to physically move the vertices of the
mesh at render time.
   This is of course only useful if the mesh has large amounts
of vertices.
```

Would be written like this instead, putting a definition first:

```
Displacement Mapping
   A method for distorting vertices based on an image.
   Similar to Bump Mapping, but instead operates on the mesh's
actual geometry.
   This relies on the mesh having enough geometry.
```

This entry:

```
Doppler Effect
   The Doppler effect is the change in pitch that occurs
   when a sound has a velocity relative to the listener.
```

Would be written more like this, avoiding the immediate repetition of the term:

```
Doppler Effect
   Perceived change in pitch that occurs
   when the source of a sound is moving relative to the
listener.
```

This entry:

```
Curve
   It is a class of objects.
   In Blender there are Bézier curves and NURBS curves.
```

Would be written more like this, avoiding the “it is”:

```
Curve
   A type of object defined in terms of a line interpolated
between Control Vertices.
   Available types of curves include Bézier and NURBS.
```

Markup Style Guide

This page covers the conventions for writing and use of the reStructuredText (RST) markup syntax.

Conventions

- Three space indentation.
- Lines should be less than 120 characters long.
- Use italics for button/menu names.

Other loose conventions:

- Avoid Unicode characters.
- Avoid heavily wrapped text (i.e. sentences can have their own lines).

Headings

```
##################
  Document Part
##################

****************
Document Chapter
****************

Document Section
================

Document Subsection
-------------------

Document Subsubsection
^^^^^^^^^^^^^^^^^^^^^^
```

```
Document Paragraph
""""""""""""""""""""""
```

Note

Parts should only be used for contents or index pages.

Note

Each `.rst` file should only have one chapter heading (`*`) per file.

Text Styling

See the overview on ReStructuredText for more information on how to style the various elements of the documentation and on how to add lists, tables, pictures and code blocks. The Sphinx reference provides more insight additional constructs.

The following are useful markups for text styling:

```
*italic*
**bold**
``literal``
```

Interface Elements

- `:kbd:`LMB`` – keyboard and mouse shortcuts.
- `*Mirror*` – interface labels.
- `:menuselection:`3D Viewport --> Add --> Mesh --> Monkey`` – menus.

Code Samples

There is support for syntax highlighting if the programming language is provided, and line numbers can be optionally shown with the `:linenos:`

option:

```
.. code-block:: python
   :linenos:

   import bpy
   def some_function():
       ...
```

Images

Figures should be used to place images:

```
.. figure:: /images/interface_window-system_splash_current.png

   Image caption.
```

For consistency, and since it would be good to ensure screenshots are all a similar size when floated next to text, writers should take screenshots in the following manner:

1. Prepare the area you would like to capture making sure to use the default theme and setting. (In some cases you may not want to use the default settings e.g. if some options are hidden behind a checkbox.)
2. Zoom to the maximum zoom level (hold `NumpadPlus` or `Ctrl-MMB` or similar).
3. Zoom out eight zoom levels (`NumpadMinus` – eight times).
4. In some cases you will want to leave a small margin around the thing you are trying to capture. This should be around 30px but does not have to be exact.

This can be applied to several parts of the interface but might not work for all cases.

Files

No Caps, No Gaps
: Lower case filenames underscore between words.

Sort Usefully
: Order naming with specific identifiers at the end.

Format
: Use `.png` for images that have solid colors such as screenshots of the Blender interface, and `.jpg` for images with a high amount of color variance, such as sample renders and photographs.

  Do not use animated `.gif` files, these are hard to maintain, can be distracting and are usually large in file size. Instead use a video if needed (see Videos below).

Location
: Place the image in the `manual/images` folder. Use no other subfolders.

Naming
: For naming files use underscores to separate chapters and sections, and use dashes to separate sections that are two or more words. So for image files should look like: `chapter_subsection_sub-subsection_id.png`, e.g:

  - `interface_splash_current.png`
  - `interface_undo-redo_last.png`
  - `interface_undo-redo_repeat-history-menu.png`

  Do not use special characters or spaces!

Usage Guides

- Avoid specifying the resolution of the image, so that the theme can handle the images consistently and provide the best layout across different screen sizes.

- When documenting a panel or section of the UI, it is better to use a single image that shows all of the relevant areas (rather than multiple images for each icon or button) placed at the top of the section you are writing, and then explain the features in the order that they appear in the image.

Note

It is important that the manual can be maintained long term, UI and tool options change so try to avoid having a lot of images (when they are not especially necessary). Otherwise, this becomes too much of a maintenance burden.

Videos

Videos can be embedded from Blender's self-hosted PeerTube instance which can be found at video.blender.org. To embed a video using the following directive:

```
.. peertube:: ID
```

The `ID` is found in the video's URL, e.g:

The ID for `https://video.blender.org/videos/watch/47448bc1-0cc0-4bd1-b6c8-9115d8f7e08c` is `47448bc1-0cc0-4bd1-b6c8-9115d8f7e08c`.

To get a new video uploaded, contact a Documentation Project Administrator or include the uploaded video in your Pull Request description.

Usage Guides

- Avoid adding videos that rely on voice or words, as this is difficult to translate.
- Do not embed video tutorials as a means of explaining a feature, the writing itself should explain it adequately. (Though you may include a link to the video at the bottom of the page under the heading `Tutorials`).

Useful Constructs

- `|BLENDER_VERSION|` – Resolves to the current Blender version.
- `:abbr:`SSAO (Screen Space Ambient Occlusion)`` – Abbreviations display the full text as a tooltip for the reader.
- `:term:`Manifold`` – Links to an entry in the Glossary.

Cross References and Linkage

You can link to another document in the manual with:

```
:doc:`The Title </section/path/to/file>`
```

To link to a specific section in another document (or the same one), explicit labels are available:

```
.. _sample-label:

[section or image to reference]

Some text :ref:`Optional Title <sample-label>`
```

Linking to a title in the same file:

```
Titles are Targets
==================

Body text.

Implicit references, like `Titles are Targets`_
```

Linking to the outside world:

```
`Blender Website <https://www.blender.org>`__
```

Context Sensitive Manual Access

It is possible to link to a specific part of the manual from in Blender by opening the context menu (right click) of a property or operator and selecting *Online Manual*. In order for this to work, this needs to be accounted for in the documentation. To link a property or operator to a

specific part of the manual you need to add an external reference link tag whose ID matches Blender's RNA tag. The easiest way to find out what the tag for a property is to open the context menu of the property/operator and select *Online Python Reference* to extract the tag from the URL. Some examples of how this looks in the RST document are given below:

```
.. _bpy.types.FluidDomainSettings.use_fractions:

Fractional Obstacles
   Enables finer resolution in fluid / obstacle regions (second
order obstacles)...

   .. _bpy.types.FluidDomainSettings.fractions_distance:

   Obstacle Distance
      Determines how far apart fluid and obstacles are...
```

For an operator:

```
.. _bpy.ops.curve.subdivide:

Subdivide
=========
```

Icons

Blender's icons can be included as inline text using:

```
`:bl-icon:`<icon_name>`
```

Further Reading

To learn more about reStructuredText, see:

Sphinx RST Primer
: Good basic introduction.

Docutils reStructuredText Reference
: Links to reference and user documentation.

Commit Guidelines

Access to directly submit changes is limited to people with commit access to the repository. Once you are provided with commit access you can start committing directly instead of creating a patch file.

You can make commits from your Git client or using the Git command line tool. The following command will create a commit and send it to the central repository:

```
git commit -m "This is what I did"
git push
```

If you leave out `-m "message"`, you will be prompted to type the message in a text editor.

Tip

You should make sure you are always on the latest revision before committing. You may not be able to commit directly if there are conflicting changes in the latest revision.

To avoid this update your local repository before committing (run `make update`).

See also

Blender's Git usage guide

See also

See release branch for documentation on how to make commits to a specific release branch and how to create merge commits.

Writing a Good Commit Message

When making changes to the manual that directly relate to a specific commit (change) in Blender, it is helpful to make the title of the commit the same as the commit made to Blender. It is requested that you include the commit hash of the commit made to the Blender source code.

For example, the commit rBM8473 includes a descriptive indicative of the changes made along with the hash `rBa71d2b260170`. The hash can be extracted from the URL provided in the Documentation task for a specific upcoming release.

Other more general changes do not have to follow the above policy however, it is still important to make the description clear about what changes you made and why. It can be helpful to prefix the commit title with a prefix word such as `Cleanup:` or `Fix:` when you are making general cleanups or fixes respectively.

Writing good commit messages helps administrators keep track of changes made and ensures all new features are properly documented.

Templates

The following guide provides patterns for interface elements and directories.

Operator Menus

Each operator should receive its own heading or page based on the length of the content. At the start should be a reference admonition documenting the context of the operator:

```
.. admonition:: Reference
   :class: refbox

   :Mode:      Edit Mode
   :Menu:      :menuselection:`Curve --> Snap`
   :Shortcut:  :kbd:`Shift-S`
```

Panels

Panels should be documented by their own heading, nested panels should use decreasing heading levels. Each panel could have its own page based on the length of documentation and/or the amount of panels. Expanded menus that toggle what properties are presented to the user should be treated like subpanels:

```
Panel Title
===========

Nested Panel Title
------------------
```

Properties

Properties should be documented using definition lists. Properties that are hidden based on other properties should used nested definitions:

```
Property
   Property description.

   Hidden Property
      Hidden property description.
```

Select menus should be documented using the following syntax:

```
Menu Label
   General description of the menu.

   :Menu Item: Menu Item Definition.
   :Menu Item: Menu Item Definition.
   :Menu Item: Menu Item Definition.
```

Nodes

Nodes should always have three headings inputs, properties and outputs with a note of absence if the node has none. At the end of the page can be an optional example(s) section:

```
**********
World Node
**********

.. figure:: /images/render_shader-nodes_output_world_node.png
   :align: right

   The World node.

Introduction and general use case(s).


Inputs
======

This node has no inputs.
```

```
Properties
==========

This node has no properties.


Outputs
=======

This node has no outputs.


Example
=======
```

Directory Layout

Sections should be generally structured as follows:

- `directory_name/`
  - `index.rst` (contains links to internal files)
  - `introduction.rst`
  - `section_1.rst`
  - `section_2.rst`

For example:

- `rendering/`
  - `index.rst`
  - `cycles/`
    - `index.rst`
    - `introduction.rst`
    - `materials/`
      - `index.rst`
      - `introduction.rst`
      - `volumes.rst`

The idea is to enclose all the content of a section inside of a folder. Ideally every section should have an `index.rst` (containing the TOC for that section) and an `introduction.rst` (introducing) to the contents of the section.

Table of Contents

By default, a table of contents should show two levels of depth:

```
.. toctree::
   :maxdepth: 2

   introduction.rst
   perspective.rst
   depth_of_field.rst
```

Maintenance

Adding/Removing/Moving Files

When RST-files are added or removed the corresponding locale files are added or removed automatically by the update script. However, if files need to be moved please use this Python script:

```
python tools/utils_maintenance/rst_remap.py start
```

RST-files can then be freely moved and the remap script will move the locale file after:

```
python tools/utils_maintenance/rst_remap.py finish
```

It is best to avoid moving/renaming files as this breaks URLs and without this script translators will lose all their work in these files. Please ask an administrator if you think something should be renamed/moved.

Note

This script also works for image file names.

Release Checklist

- Create a release branch (`blender-3.2-release/`)
- Update the splash image: `interface_splash_current.png` in the release branch.
- Increase the `conf.py: blender_version` variable in the trunk version.

Contribute

On this page French (`fr`) is used for examples. However, it can be replaced with other languages codes. So, be sure to change the `/fr` suffixes in this guide to the language you are translating!

To see which languages are currently available, you can check the online interface, or browse the underlying git repository.

Simple Contribution

The preferred way to contribute to the translation effor is the use the web-based interface, currently a Weblate instance.

Simple enhancement suggestions can be contributed by any user, even without loging in. Suggestions will be reviewed by the translating team before they get published.

Weblate also comes with new helping tools to improve coherance of translations, like the glossary.

Advanced Operations

If for some reasons the web-based translation interface does not work well for you, you can still download PO file from it, and upload back it later.

Warning

You will have to deal with potential conflicts yourself if some update happened in the mean time. Direct commit to the git repository for translations is not possible anymore.

Note

There is a known issue with the current tool behind the web interface, which will make heavy processing like upload and integration of a PO file take several minutes, with the web page staying in refresh mode for the whole time. If it takes more than ten minutes, it will even apparently fail with a server timeout error message. There is usually no actual problem though, so no need to re-try uploading the PO file then, refreshing the page after a few minutes should be enough to see the contribution in the web interface.

Note

First of all, it is assumed that you have the manual already building. If you have not done this already go back too the Getting Started section.

Installing

Language Files

From the directory containing your checkout of the manual run:

```
make checkout_locale
```

You will be prompted to type in the language folder you want to download. In the case of this example we will use `fr`. Pressing `Return` will confirm this selection.

It will take a few minutes to download but once complete it will create a `locale/fr` subdirectory.

You should have a directory layout like this:

```
blender-manual
   |- locale/
   |  |- fr/
   |  |  |- LC_MESSAGES/
   |- manual/
```

Note

When running Git from the command line (such as updating), you will need to change directory to `locale` first rather than the `blender-manual` directory.

The PO language files themselves can also be downloaded from the web interface, `Files` menu, on each dedicated language page of the `Manual` comnponent.

A PO Editor

To make edit the PO files you will need to install a PO editor. We recommend that you use Poedit, however any PO editor will do.

Note

For Linux users, you will have to check with your distribution's software center for a version of Poedit. This editor is only a recommendation. There are others, such as Kate and Kwrite, that could offer syntax highlighting and basic tools for text editing, e.g. letter case transposes. Other platforms can use some text editors supporting the syntax highlighting for PO files, or allowing you to create a custom one (such as Notepad++ on Windows).

Building with Translations

Building

Now you can build the manual with the translation applied:

On Linux and macOS run:

```
make -e BF_LANG=fr
```

On Windows run:

```
set BF_LANG=fr
make html
```

Now you will have a build of the manual with translations applied.

Editing Translation Files

Now you can edit the PO translation files, in the `LC_MESSAGES` folder you have two files:

- `blender_manual.po` – This is the main translation file that you will be editing.
- `sphinx.po` – This translation file is much smaller and contains translations for the website theme.

To edit these files open them up in your translation editor, i.e. Poedit. Once in your editor you will see a list of texts, each of these items represent some part of the user manual. You may need to adjust your editor to sort the list in a way that makes sense for example “by source”.

You can now select an untranslated string and your editor will have an input box to add the translation. The modified `.po` files can now be submitted back to the web-based interface.

Tip

Make sure that you Building with Translations to catch any syntax errors you may make while translating. These errors will be displayed as warnings while building the manual.

Maintenance

Keeping Track of Fuzzy Strings

When the manual is updated, those translations which are outdated will be marked as fuzzy. To keep track with that, you can use a tool we created for that task.

You can do this by running:

```
make report_po_progress
```

This will only give a quick summary however, you can get more information by running:

```
python tools/translations/report_translation_progress.py
locale/fr/
```

You should get a list of all the files with information about the number of empty and fuzzy strings. For more options see:

```
python tools/translations/report_translation_progress.py --help
```

Updating PO Files

As the original manual changes, the templates will need updating. Note, doing this is not required, as administrator usually update the files for all languages at once. This allows all languages to be on the same version of the manual. However, if you need to update the files yourself, it can be done as follows:

```
make update_po
```

The updated templates can then be committed to the repository.

See also

A guide how to add a new language can be found in the Adding a Language.

Style Guide

This page covers conventions concerning the translations.

Note

- We expect our readers to use the English version of Blender, not a translated one.
- The translations are licensed under the same License as the original.

Should I Translate... ?

Maybe

Hyperlinks
: Can be translated, but only as an addition, not as a replacement. See also Adding Text.

Technical Terms
: Only translate these, when the localized expression is common! See also Technical Terms.

Text you are not sure you understood
: Simply mark the text as fuzzy and/or add a comment. The next translator might understand it.

Never

Images
: You probably will not find the original scene if it is a screenshot of a file and it is too much load on the server (and too much work for you).

Menu and button names

We expect our readers to use the English UI.

Text you do not understand
: Do not translate it! It will do more harm than good!

Technical Terms

In general, the technical terms used in computer graphics are quite new or even downright neologisms invented for the needs, so they do not always have a translation in your language. Moreover, a large part of Blender users use its English interface.

As a result, unless a term has an evident translation, you should preferably use the English one, *putting it in italic*. You can then find a translation for it, which you will use from times to times (e.g. to avoid repetitions…). This is also valid in the other way: even when a term has a straightforward translation, do not hesitate to use its English version from times to times, to get the reader used with it…

If a term is definitively not translatable, simply use the English one, but make sure its glossary entry is translated.

In the glossary, the English term is written first (to maintain alphabetic order) with the translated entry following in parenthesis, when appropriate.

Adding Text

Generally, **you should always translate exactly what is in the text**, and avoid providing updates or extra information.

But sometimes that is necessary, for example when talking about the manual itself: To a foreign reader it is not clear, that they can contribute English text only, whereas this is obvious to an English reader.

In these (rare) cases you can and should provide extra information.

Keeping Pages Up To Date

When the manual is updated, those translations which are outdated will be marked as fuzzy. To keep track with that, you can use a tool we created for that task, see How to install it.

Not Found (404)

It seems the page you are trying to find has been either moved or deleted.
You can try one of the following things:

Search

License

Blender itself is released under the GNU General Public License. More info blender.org/about/license.

Except where otherwise noted, the content of the Blender Manual is available under a Creative Commons Attribution-ShareAlike 4.0 International License or any later version. Excluded from the CC-BY-SA are also the used logos, trademarks, icons, source code and Python scripts.

Please attribute the “Blender Documentation Team” and include a hyperlink (online) or URL (in print) to manual. For example:

```
The Blender 4.3 Manual
by the Blender Documentation Team
is licensed under a CC-BY-SA v4.0.
```

See Best practices for attribution for further explanation.

It means, that when contributing to the manual you do not hold exclusive copyright to your text. You are, of course, acknowledged and appreciated for your contribution. However, others can change and improve your text in order to keep the manual consistent and up to date.

If you have questions about the license, feel free to contact the Blender Foundation: foundation (at) blender (dot) org

Previous versions of the Blender Manual are made available under an Open Content License v2.26 – v2.77.

Adding a Language

Preparations

If the language you want to translate has not been started by someone else already and you wish to create a set of new files for the desired language, say ‘fr’ (French), then you must first use the environment you have created, as guided in Getting Started, in particular Installing Dependencies and Building the Manual sections.

This will give you a foundation environment for:

- Creating a new set of translation language from English source.
- Perform `make` command to turn translated texts in po files into html files for testing locally.
- Update changes in English texts which have been added by other contributors.

Below examples show the process to create a new set of files for French, language code `fr`, on Linux platform. Other platforms might vary slightly but should be mainly the same.

1. Create a Blender ID if you have not done so already.
2. Log into projects.blender.org and Create an Issue requesting for commit access in order to transfer changes to the central repository of the translation team.
3. Open an instance of a console application.
4. Change the current working directory to the directory of `blender-manual`, where the instance of `Makefile` resides.

Trying the Make Process to Create HTML Files in English

1. Ensure the previous instance of `build` directory is removed, if any exists:

   ```
   make clean
   ```

2. Convert all the `rst` files into `pot` translation files:

   ```
   make gettext
   ```

3. Create `html` files:

   ```
   make html
   ```

4. After this, you can actually view the created html files locally by opening the `blender-manual/build/html/index.html` file.

Creating the Language Entry in the HTML Menu

1. Create an entry for the language in the html menu by opening file `./build_files/theme/js/version_switch.js` (assuming you are at the `blender-manual` subdirectory).

2. Find the table for the languages in `var all_langs = {..};`.

3. Enter the entry: `"fr": "Français",`, (`"fr": "Français"`). (Notice the Unicode characters.)

4. Commit the updated file:

   ```
   git add ./build_files/theme/js/version_switch.js
   git commit -m "HTML: Add French to language menu"
   ```

5. Push your changes to the upstream repository:

   ```
   git push
   ```

Generating the Set of Files for the Target Language

1. Check out the current translation repository using the command:

   ```
   git clone https://projects.blender.org/blender/blender-manual-translations.git locale
   ```

   This will download all language sets available in the repository into the `locale` directory of your drive. You can go to the `locale` directory to see the hidden subdirectory `.git` within it, together with directories of languages. You'll need to add your own set of files for the language you are trying to translating to.

2. From the `blender-manual` directory to generate a set of files for `fr` language:

   ```
   make update_po
   ```

   These files are still in English only, with all `msgstr` entries blank.

3. Submit new set of files to the central repository:

   ```
   cd locale
   git add fr
   git commit -m "Initial commit language set of files for French"
   ```

Tip

- It is recommended you make two environment variables for these directories, in the `.bashrc` to make it more convenient for changing or scripting batch/shell commands for the process of translation and reviewing results:

  ```
  export BLENDER_MAN_EN=$HOME/<directory to make file directory above>/blender-manual
  export BLENDER_MAN_FR=$BLENDER_MAN_EN/locale
  ```

- Newly generated files will contain some placeholders for authors and revision dates etc. If you find the job of replacing them repetitive, make use of the script `change_placeholders.sh` in the subdirectory `~/blender-manual/tools/util_maintenance`, make a copy of that

to your local `bin` directory and replace all values that were mentioned in the file with your specific details, then after each change to a file, you would do following commands to update the file with your personal details, revision date and time, plus generating the html files for your language, which you can view using your Internet browser:

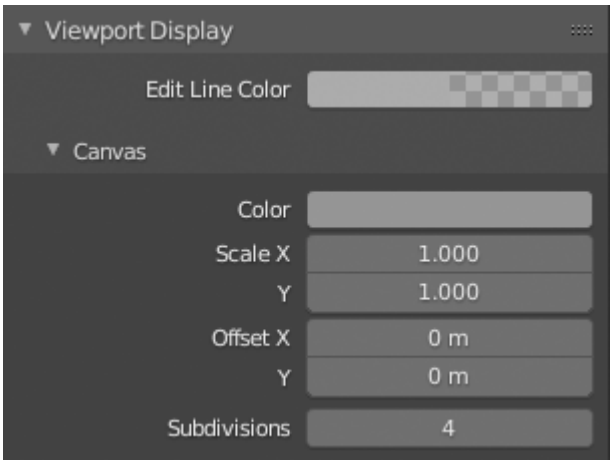
```
$HOME/bin/change_placeholders.sh $BLENDER_MAN_FR
make -d --trace -w -B -e SPHINXOPTS="-D language='fr'"
2>&1
```

Getting Started

- About Blender
  - Who uses Blender?
  - Key Features
  - Further Reading
- Installing Blender
  - System Requirements
  - Download Blender
  - Installation Guides
- Configuring Blender
  - Introduction
  - Configuring Peripherals
  - Defaults
- Help System
  - Tooltips
  - Context-Sensitive Manual Access
  - Help Menu

Viewport Display

Display settings for Edit Lines in *Edit Mode* and *Sculpt Mode*.



Viewport Display panel.

Edit Line Color
: Sets the color of the Edit Lines.

Canvas

In 3D space sometimes is difficult to assess on which plane are you drawing. The Canvas is a display overlay helper that shows a grid at the current *Drawing Plane*. You can enable the Canvas visualization in the Viewport Overlays.

See Drawing Plane for more information.

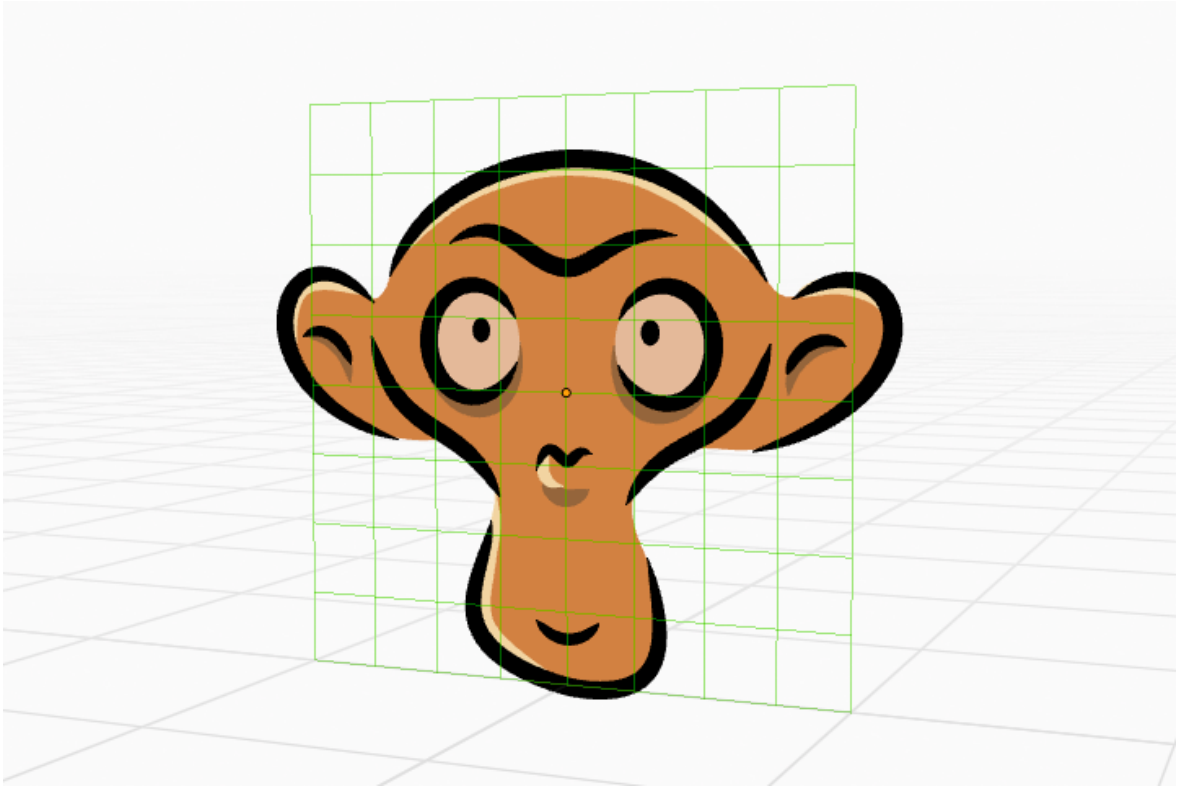
Color
: Color of the Canvas grid lines.

Scale X/Y
: Defines the X and Y scale of the Canvas.

Offset X/Y
: Sets the Canvas position offset from the object's origin.

Subdivisions

Specifies the number of subdivisions to use for the grid.



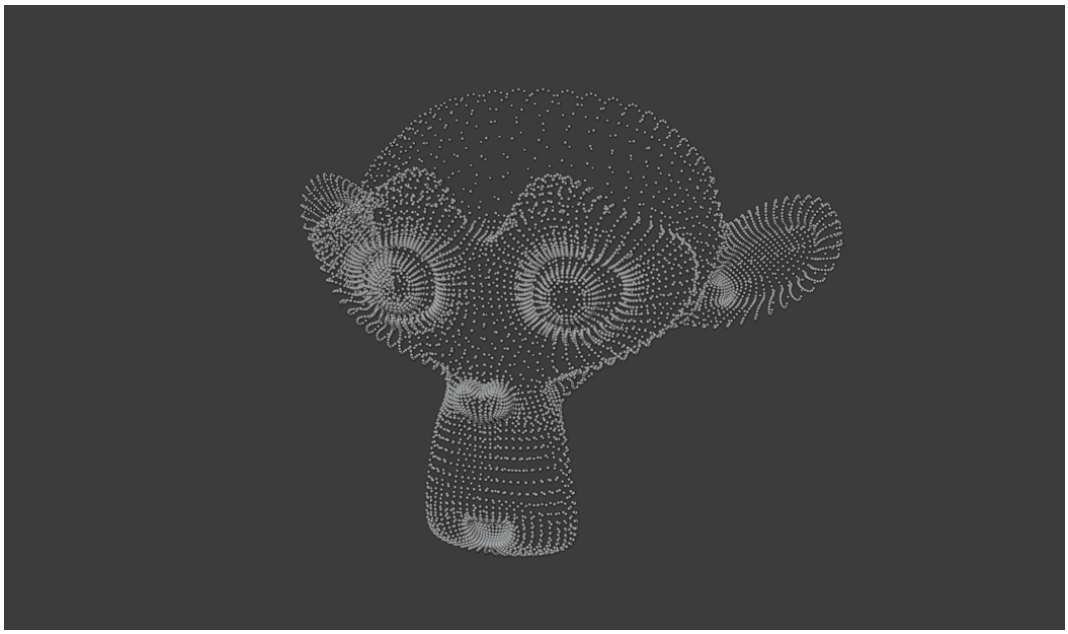
Canvas example on the XZ drawing plane using a green color grid.

Important

This feature is currently experimental and not available in current releases.

Point Cloud

Point clouds can be used to represent 3D scans and in the future can represent particles. Each point can store data in a set of Attributes.



Example of a monkey object represented as a point cloud.

Properties

Attributes

The *Attributes* panel contains different point cloud characteristics such as the position and size of points. Use the List View to manage attributes.

Attribute Types

See also

See Built-In Attributes for information about common attributes.

Name	Type	Domain	Notes
`position`	*Vector*	*Point*	Built-in attribute describing vertex or point locations, in the modifier object's transform space.
`radius`	*Float*	*Point*	The radius of each point.
`color`	*Color*	*Point*	The color of each point.
`id`	*Integer*	*Point*	A unique identifier given to each particle.
`velocity`	*Vector*	*Point*	The speed and direction that the particle is traveling.

Custom Attributes

Custom attribute can be given to particles to hold a custom characteristic.

Name

The name of the attribute.

Data Type

The type of data to store in the attribute.

Float:: Floating-point value
Integer:: 32-bit integer
Vector:: 3D vector with floating-point values
Color:: RGBA color with floating-point precision
Byte Color:: RGBA color with 8-bit precision
String:: Text string

Domain
: The type of element the attribute is stored in. Currently, attributes can only be stored per *Point*.

Custom Properties

See the Custom Properties page for more information.

Editing

Currently, not much can be done with point clouds; however, they can be converted to/from meshes.

Line Style Modifiers

Color

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Color Modifiers
  - Along Stroke
  - Crease Angle
  - Curvature 3D
  - Distance from Camera
  - Distance from Object
  - Material
  - Noise
  - Tangent
- Material
- Noise
- Tangent

Alpha

- Along Stroke
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Alpha Modifiers
  - Along Stroke
  - Crease Angle
  - Curvature 3D
  - Distance from Camera

    - Distance from Object
    - Material
    - Noise
    - Tangent
- Material
- Noise
- Tangent

Thickness

- Along Stroke
- Calligraphy
- Crease Angle
- Curvature 3D
- Distance from Camera
- Distance from Object
- Thickness Modifiers
    - Along Stroke
    - Calligraphy
    - Crease Angle
    - Curvature 3D
    - Distance from Camera
    - Distance from Object
    - Material
    - Noise
    - Tangent
- Material
- Noise
- Tangent

Geometry

- 2D Offset
- 2D Transform
- Backbone Stretcher
- Bézier Curve

- Blueprint
- Guiding Lines
- Geometry Modifiers
  - 2D Offset
  - 2D Transform
  - Backbone Stretcher
  - Bézier Curve
  - Blueprint
  - Guiding Lines
  - Perlin Noise 1D
  - Perlin Noise 2D
  - Polygonization
  - Sampling
  - Simplification
  - Sinus Displacement
  - Spatial Noise
  - Tip Remover
- Perlin Noise 1D
- Perlin Noise 2D
- Polygonization
- Sampling
- Simplification
- Sinus Displacement
- Spatial Noise
- Tip Remover

On macOS graphics drivers are built into the operating system and the only way to get newer drivers is to upgrade macOS as a whole to the latest version.

Blender uses of OpenGL for the 3D Viewport and user interface. The graphics card (GPU) and driver have a big impact on Blender's behavior and performance.

This section lists possible solutions for graphics glitches, problems with EEVEE and Cycles, and crashes related to your GPU.

Drivers

Upgrading to the latest graphics drivers often solves problems. Newer drivers have bug fixes that help Blender function correctly.

Laptops

Laptops often have two GPUs for power saving purposes. One slower onboard GPU (typically Intel) and one faster dedicated GPU for a better performance (AMD or Nvidia).

For the best performance the dedicated GPU should be used for Blender. Which GPU to use for which application can be configured in your graphics driver settings.

If there is a graphics glitch or crash specific to the onboard GPU, then using the dedicated GPU can help avoid that. Or vice versa, if the dedicated GPU causes issues, then using the onboard graphics can help.

Common Problems

Unsupported Graphics Driver Error

This means your graphics card and driver do not have the minimum required OpenGL 3.3 version needed by Blender.

Installing the latest driver can help upgrade the OpenGL version, though some graphics cards are simply too old to run the latest Blender. Using Blender 2.79 or earlier is the only option then.

Crash on Startup

Try running Blender from the command line, to see if any helpful error messages are printed.

On Windows, graphics drivers can sometimes get corrupted. In this case it can help to uninstall all graphics drivers (there may be multiple from Intel, AMD and Nvidia) and perform a clean installation with drivers from the manufacturer's website.

Poor Performance

- Update your graphics drivers (see above).
- On laptops, make sure you are using a dedicated GPU (see above).
- Try lowering quality settings in *Preferences ▸ System ▸ Memory & Limits*.
- Try undoing settings in your graphics drivers, if you made any changes there.

Render Errors

See EEVEE and Cycles documentation respectively.

Wrong Selection in 3D Viewport

See Invalid Selection, Disable Anti-Aliasing.

Virtual Machines

Running Blender inside a virtual machine is known to have problems when OpenGL drawing calls are forwarded to the host operating system.

To resolve this, configure the system to use PCI passthrough.

Information

To find out which graphics card and driver Blender is using, use *Help* ▸ *Save System Info* inside Blender. The OpenGL section will have information about your graphics card, vendor and driver version.