

---

# **Blender User Manual**

***Release 2.92***

**Blender Community**

**Nov 27, 2020**

---

## Contents

---



Welcome to the manual for [Blender](#), the free and open source 3D creation suite.

# CHAPTER 1

---

## Getting Started

---

## 2.1 Getting Started

### 2.1.1 About Blender

#### Introduction

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, VFX shots, and video editing. 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. People and studios around the world use it for hobby projects, 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](https://blender.org/community) for an extensive list of sites.



- It has a small executable, which is optionally portable.

You can download the latest version of Blender [here](#).

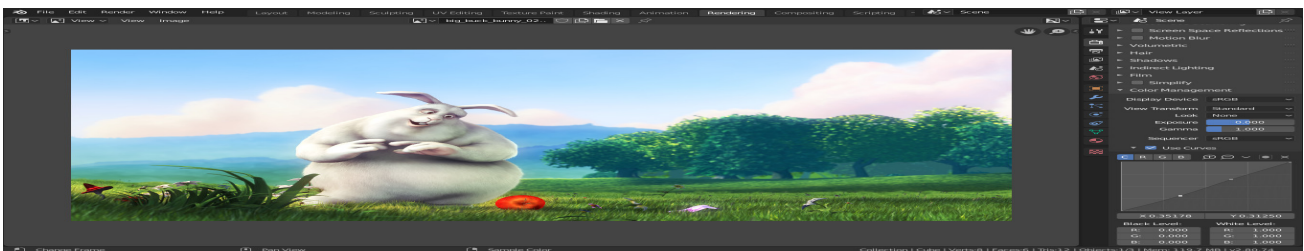


Fig. 1: A rendered image being post-processed.

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, keep your mind open to other artistic and technological areas and you too can become a great artist.

## 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 scene, 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 back end, 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.

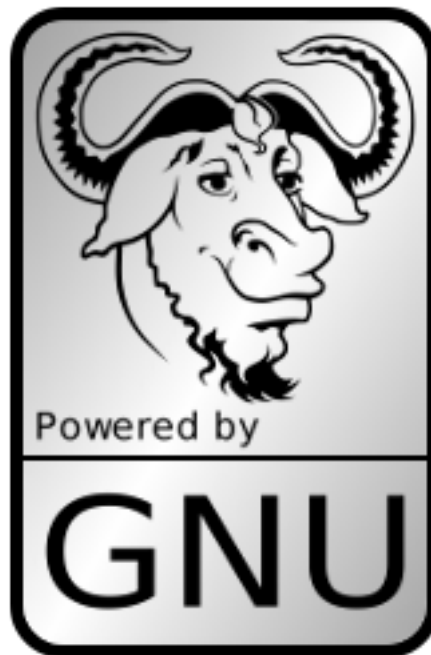
**2.83 - June 2020:** 3D Viewport virtual reality scene inspection, new volume object type, Cycles adaptive sampling, Cycles viewport denoising, sculpting improvements, and much more.

## 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.

## 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 [Blender Chat](#) and [Stack Exchange](#).

There are also more official sources of support, such as [Certified Trainers](#) and the [Blender Cloud](#). If you think you have found an issue with Blender, you can easily [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 [developer.blender.org](#) system
- Various [mailing lists](#)

- [Online Chat](#) (see below)

If you are interested in helping develop Blender, see the [Get Involved](#) page.

## Blender Chat

For real-time discussion, we have [blender.chat](#) which uses [Blender ID](#) for authentication.

You can join these channels:

- [#today](#) For getting answers from 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!](#)

## 2.1.2 Installing Blender

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

### Support 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 two different binary packages; you can choose from a stable release or a daily build. The first has the benefit of being more reliable, the latter provides the newest features, as they are developed, at the cost of stability.

**Latest Stable Release** This is a binary distribution of the latest version of Blender. It is considered stable and without regressions.

**Daily Builds** This is a binary distribution of Blender that is 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.

---

## Installing on Linux

Check the *Downloading Blender* page to find the minimum requirements and where to get Blender (if you have not done so yet).

### Install from blender.org

Download the Linux version for your architecture and uncompress 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 can configure your system by adding a menu entry or shortcut for Blender. You may also associate `blend`-files with Blender so that when selected from the file browser, they will automatically open in Blender. These settings are typically found in conjunction with the Window Manager settings. (Gnome or KDE.)

### 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 CUDA support, for licensing 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](https://blender.org).

### Install from Snap

[Snap](https://snapcraft.io/) 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
```

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

### Running from the Terminal

See *Launching from the terminal*.

## Avoiding Alt+Mouse Conflict

Many Window Managers default to Alt-LMB for moving windows, which is a shortcut that Blender uses to simulate a three button mouse. You can either have this feature disabled *Preferences* → *Input* → *Emulate 3 Button Mouse* or 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 import settings from previous Blender versions and on other quick settings.

## Installing on macOS

Check the *Downloading Blender* page to find the minimum requirements and where to get Blender (if you have not done so yet).

### Install from DMG

Blender for macOS are distributed on 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, before opening Blender for the first time, macOS will request your approval.

---

#### Tip: How to Make a Portable Installation

To keep all configuration files and installed add-ons inside the Blender application bundle, create a folder named `config` in the *LOCAL directory*.

```
./Blender.app/Contents/Resources/2.92/config/
```

---

## Updating on macOS

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

### Updating with DMG

When a 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 import settings from previous Blender versions and on other quick settings.

## Installing on Windows

Check the *Downloading Blender* page to find the minimum requirements and where to get Blender (if you have not done so yet).

Download the zip-file or Windows Installer File.

### Install from Windows Installer File

The Windows installer when run will let you choose where to place Blender and configure Windows to have an entry to the start menu and to open blend-files with Blender. Administrator rights are needed to install Blender on your system.

### 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.

There is no installer to place Blender on the menu, but there is also no need for administrator rights. With this option, it is possible to have multiple versions of Blender without conflicting, as they are not actually installed on the system.

However, if you want a particular version to be registered with your computer the simply run `blender -r` from the *Command Line*.

---

**Tip:** How to Make a Portable Installation

To keep all configuration files and installed add-ons in the executable folder, create a folder named `config` in the *LOCAL directory* of the unzipped folder.

---

## 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*.

### 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 import settings from previous Blender versions and on 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.

## 2.1.3 Configuring Blender

### Introduction

Here are some preferences that you may wish to set initially. The full list and explanation of the Preferences are documented in the section *Preferences*.

## Language

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

See *Translation* for details.

## Input

If you have a compact keyboard without a separate number pad, enable *Preferences* → *Input* → *Keyboard* → *Emulate Numpad*.

If you do not have a middle mouse button, you can enable *Preferences* → *Input* → *Mouse* → *Emulate 3 Button Mouse*.

See *Input Preferences* for details.

## File and Paths

At *Preferences* → *File Paths* you can set options such as what external *Image Editor* to use, such as GIMP or Krita, and the Animation Player.

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 that someone can include inside a blend-file. This would not happen by accident, many users leave this option on to automatically run scripts often used in advanced rigs (such as “Rigify” that controls the skeleton of a human rig).

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.

### 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)
- NDOF Device (also known as *3D Mouse*)
- Graphic Tablet





Fig. 2: Example of Blender's multi-monitor support.

---

**Note:** If you are missing an input device such as a mouse or numpad you can change Blender's *keymap* to emulate these devices. Settings to enable this can be found 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 numeric Numpad on the side of your keyboard, you may want to emulate one (uses the numbers at the top of the keyboard instead, however, removes quick access to layer visibility).

#### See also:

Read more about *Numpad Emulation* in the *Preferences*.

## Non-English Keyboards

If you use a keyboard with a non-English keyboard layout, you still may benefit from switching your computer to the UK or US layout as long as you work with Blender.

---

**Note:** You can also change the default keymap and default hotkeys 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.

---

## NDOF (3D Mouse)

3D mice or NDOF (N-Degrees of Freedom) 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, as well as making *Fly/Walk Navigation* easier to control. The NDOF device can be configured in the *Preference*. These settings can also be accessed using the NDOF-Menu button on the NDOF device to open a pop-up menu to adjust the settings directly from the viewport.

### See also:

See *Input Preference* for more information on configuring peripherals.

## Head-Mounted Displays (Virtual Reality)

HMDs (Head-Mounted Displays) 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.

Table 1: OpenXR compatible platforms.

Platform	Operating System	Notes
Monado	GNU/Linux	<i>Not</i> recommended for general use yet.
Oculus (Rift and Quest)	Windows	Prototype Release, Oculus Link required for Quest.
SteamVR	Windows, GNU/Linux	Developer Preview.
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.

### Oculus

Oculus provides prototype OpenXR support. To use it, Blender has to be started in a special way, as described below.

- Download and install the [Oculus Rift software](#).
- Start Blender by double clicking the `blender_oculus` script inside the installation directory. It will open a command-line window with further information.
- Enable the *VR Scene Inspection add-on* in Blender.

### SteamVR

OpenXR support in SteamVR is a developer preview. It requires SteamVR beta features.

- [Enable SteamVR beta updates](#).
- Launch SteamVR. It should show a prompt for making SteamVR the default OpenXR runtime. Click `Set as default`.
- Enable the *VR Scene Inspection add-on* in Blender.

### Windows Mixed Reality

To check if a PC meets the requirements to run Windows Mixed Reality, 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.

For more information, refer to Windows' [Getting Started Guide for OpenXR](#).

### 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.

## Defaults

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

---

**Note:** These options can always be changed later in the *Preferences*.

---

**Language** The language used for translating 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*.

**Shortcuts** 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 the last major 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 creation software. Read more about this keymap *here*.

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

**Spacebar** 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.

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

**Load Previous Version Settings** Copies preferences and startup files from the previous version of Blender and then loads them.

The settings 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.

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, Workspaces and interface which is displayed at startup and when loading a new file (*File* → *New*).

## Saving Defaults

The user preferences are automatically saved when changed.

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

## Loading Factory Settings

You can revert your own 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.

**See also:**

See *Preferences* for a complete list of options.

## 2.1.4 Help System

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

### Tooltips

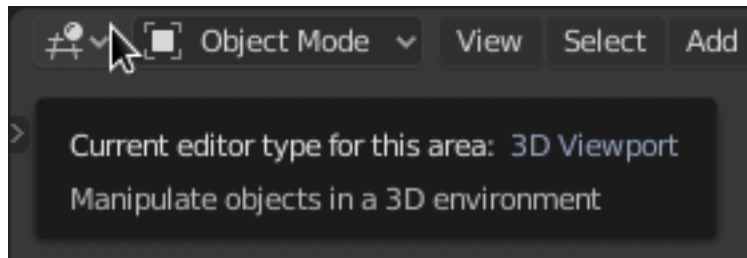


Fig. 3: Tooltip of the Renderer selector in the Info Editor.

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.

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

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

**Python** When *Developer Extras* 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*

**Hotkey** Alt-F1

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

Use the key-shortcut, or context menu to visit pages from this reference manual within Blender. This opens a webpage 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 some tools do 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).

**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.

**Developer Community** Blender's developer forum.

---

**Python API Reference** Python application programming interface (API)

---

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

### Save System Info

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

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

The text file contains sections:

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

**Python** The Python version you are using, showing the paths of the Python programming language.

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

Those directories are configured using the *Preferences* Editor.

**OpenGL** This section shows the OpenGL version, the name of the manufacturer, and lists the capabilities of your hardware and driver.

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



## 2.2 User Interface

### 2.2.1 Splash Screen

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

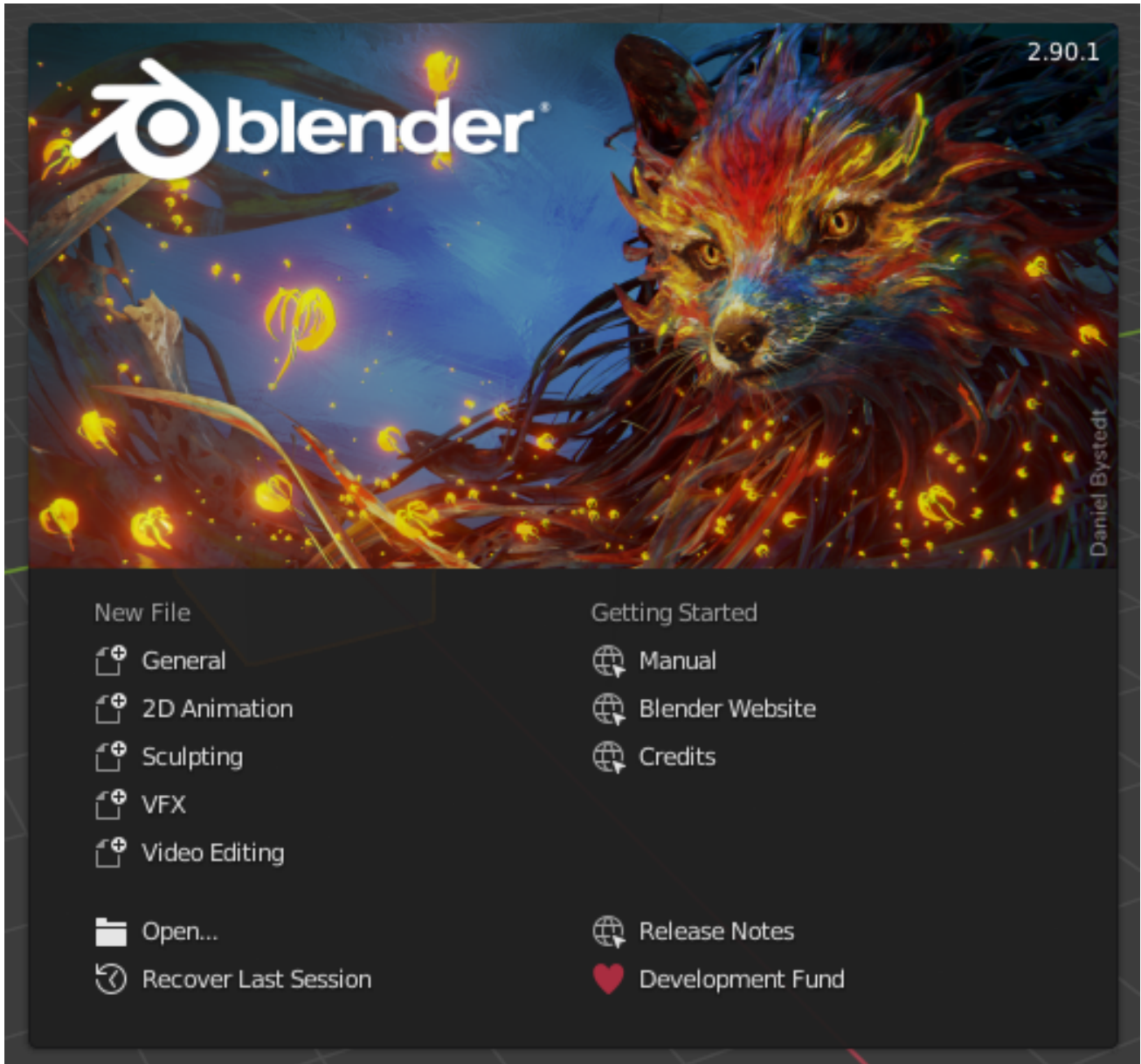


Fig. 4: Blender Splash Screen.

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*.

**Information Region** 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*.

**Links** Links to the official web site. The same links can be found in the *Help Menu* of the *Topbar*.

---

**Note:** When starting Blender for the first time, the Interactive Region contains a *Quick Set Up Process*.

---

## 2.2.2 Window System

### Window System Introduction

After starting Blender and closing the *Splash Screen* the Blender window should look something similar to the image below; as Blender's user interface is consistent across all platforms.

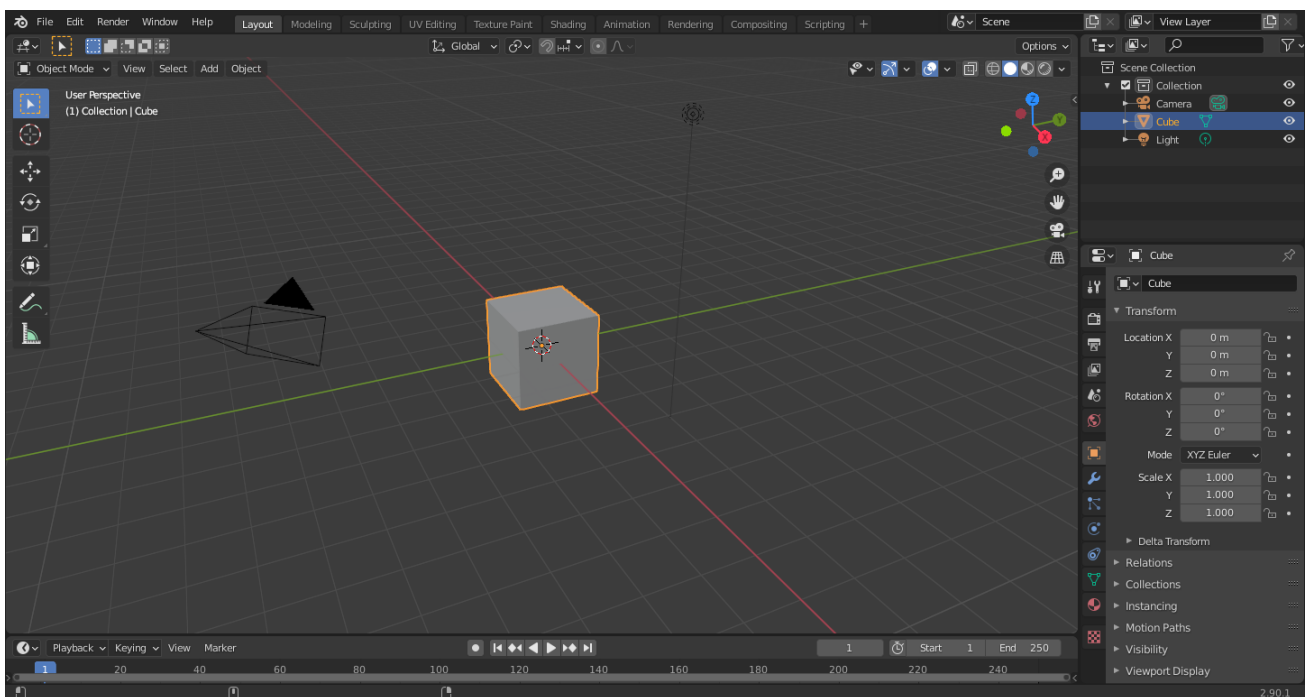


Fig. 5: The default startup Blender window.

Blender's interface is separated into three main parts:

- *Topbar* at the very top.
- *Areas* in the middle.
- *Status Bar* at the bottom.



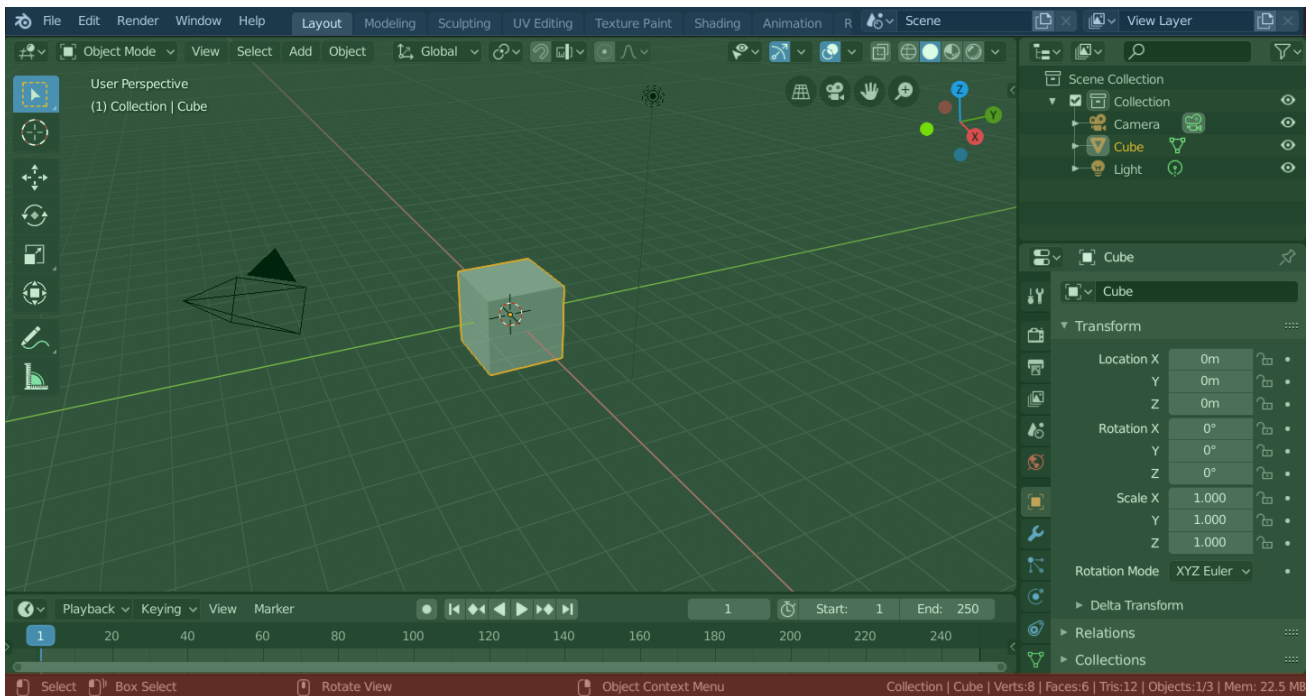


Fig. 6: Blender's default Screen Layout. Topbar (blue), Areas (green) and Status Bar (red).

## Customization

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

## Theme colors

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

## Topbar

## Menus

## App Menu

**Splash Screen** Open the *Splash Screen*.

**About Blender** Opens a menu displaying 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 personal when diagnosing a problem.

**Branch** Optional branch name.

**Release Notes** Open the latest release notes.

**Credits** Open credits website.

**License** Open License website.

**Blender Website** Open main Blender website.

**Blender Store** Open the Blender store.

**Development Fund** Open the developer fund 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 *recently* saved blend-files to open.

**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.

**Link...** Links data from an external blend-file (library) to the current scene. The edition 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 scene. 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*.

**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 Into .blend** This option activates the file packing. If enabled, every time the blend-file is saved, all external files will be saved (packed) in it.

**Pack All Into .blend** Pack all used external files into the blend-file.

**Unpack Into Files** Unpack all files packed into this blend-file to external ones.

**Make All Paths Relative** Make all paths to external files *Relative Paths* to current blend-file.

**Make All Paths Absolute** Make all paths to external files absolute. Absolute ones have 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 All 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 All** Remove all unused data-blocks from the file (cannot be undone). See the *Outliner* for more information.

**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.

**See also:**

*Managing Preferences.*

**Quit Ctrl-Q** Closes Blender and the file is saved into `quit.blend`.

## Edit Menu

**Undo/Redo/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** Restrict select to the current mode.

**Preferences** 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 scenes audio file to a sound file.

**See also:**

- *Rendering audio* for details.

**View Render F11** Toggle show render view.

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

**See also:**

- *Animation player* for details.
- *Animation player* preferences to select different animation players.

**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 active area of whole Blender window.

## Help Menu

See *Help System*.

## Workspaces

These sets of tabs are used to select the current *Workspace*; which are essentially predefined window layouts.

## Scenes & Layers

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

## Workspaces

*Workspaces* are essentially predefined window layouts. Blender's flexibility with *Areas* lets you create customized workspaces for different tasks such as modeling, animating, and scripting. It is often useful to quickly switch between different workspaces within the same file.



Fig. 7: Workspaces are located at the Topbar.

## Controls

**Tabs** Click on the tabs titles to switch between the workspaces. To cycle between workspaces use Ctrl-PageUp and Ctrl-PageDown. Double click 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 objects and contains the following *Editors*:

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

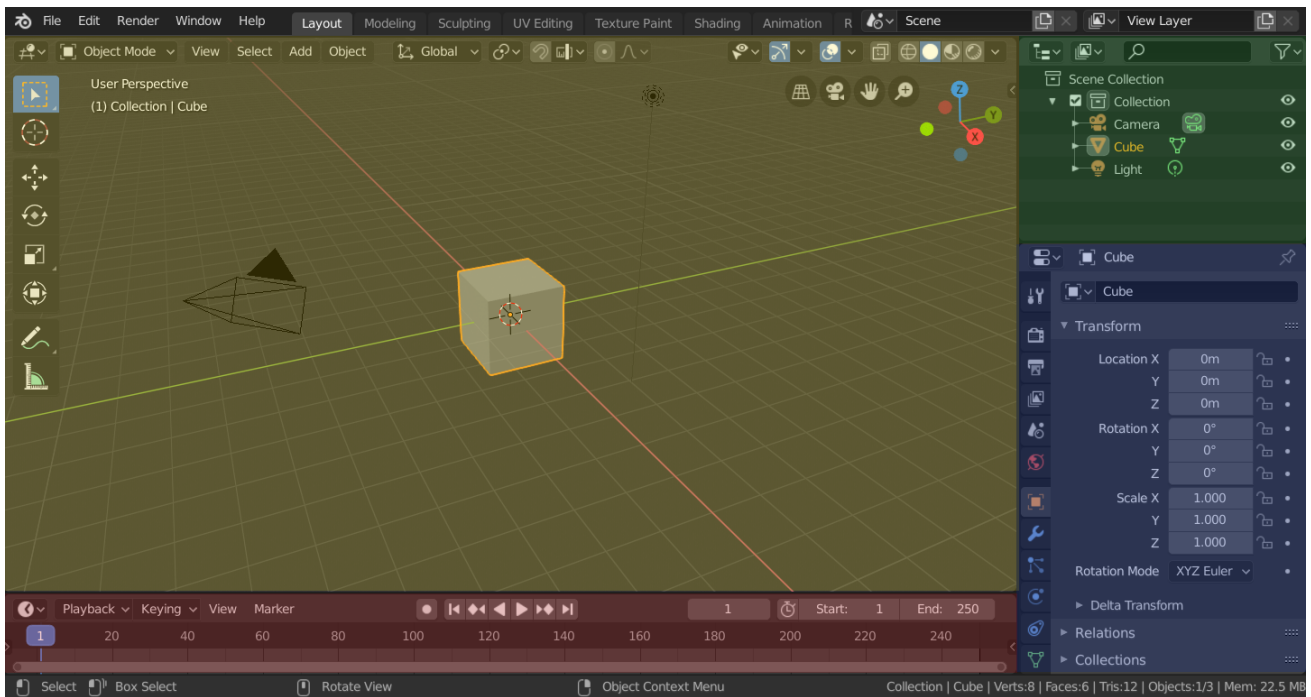


Fig. 8: Blender's 'Layout' Workspace with four editors. 3D Viewport (yellow), Outliner (green), Properties (blue) and Timeline (red).

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** Mapping of image texture coordinates to 3D surfaces.

**Texture Paint** Tools for coloring image textures in the 3D Viewport.

**Shading** Tools for specifying material properties for rendering.

**Animation** Tools for making properties of objects dependent on time.

**Rendering** For viewing and analyzing rendering results.

**Compositing** Combining and post-processing of images and rendering information.

**Scripting** Programming workspace for 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** Tools to create 2D masks for compositing.

**Motion Tracking** Tools to motion track and stabilize footage.

### Video Editing

**Video Editing** Sequence together media into one video.

## Save and Override

The workspaces are saved in the blend-file. When you open a file, enabling the *Load UI* in the File Browser indicates that Blender should use the file’s screen layouts and overriding the current layout. See *Load UI*.

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

## Status Bar

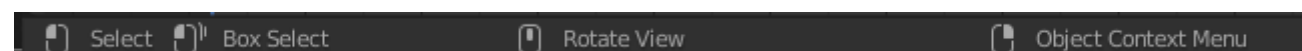
The Status Bar is located at the bottom of the Blender window and displays contextual information such as keyboard shortcuts, result or warning message and statistical information. The Status Bar can be hidden by disabling *Show Status Bar* in Window menu or by dragging from the top edge down.



Fig. 9: 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, pressing Alt shows the hotkeys to change to a desired tool.





## Status Messages

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



**Running Task** The progress of the currently running task is shown when a computation is being performed for example rendering, baking or playback. Hovering the mouse pointer over the progress bar will display a time estimate. The task can be aborted by clicking the cancel button (X icon).

**Report Message** Blender operation results or warnings, such as after saving a file. They disappear after a short time. Click this label to show the full message in the *Info Editor*.

## Resource Information

The right side of the Status Bar displays information about the Blender instance. These can individually be shown or hidden by RMB on the Status Bar area.

Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Objects:0/3 | Mem: 29.7 MiB | v2.81.16

## Scene Statistics

**Collection** Name of the active *Collection*.

**Active Object** Name of the active selected object.

**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.

**Objects** Number of the selected objects and the total count.

**System Memory** Estimate of Blender's RAM consumption. In a single-instance single-machine scenario, this estimate provides a measurement against the hardware limit of the machine.

**Blender Version** The version of Blender that is currently run.

## Areas

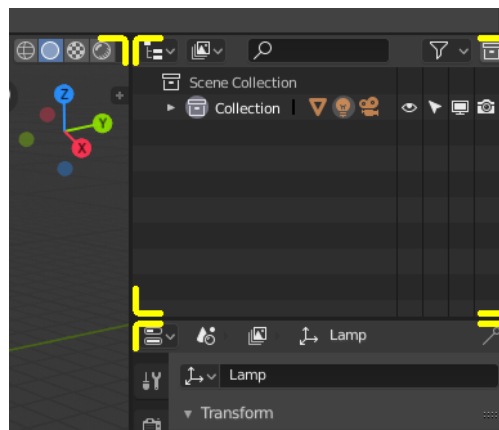


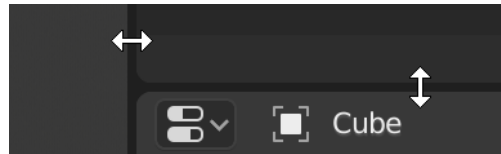
Fig. 10: Area boundaries are indicated by rounded corners (yellow highlights).

The Blender window is divided up into a number of rectangles called Areas. Areas reserve screen space for *Editors*, such as the 3D Viewport, or the Outliner. In general an Editor provides a way

to view and modify your work through a specific part of Blender. All hotkeys you press will affect the contents of the Editor in the Area the mouse pointer is located. Area boundaries are indicated by rounded (beveled) corners.

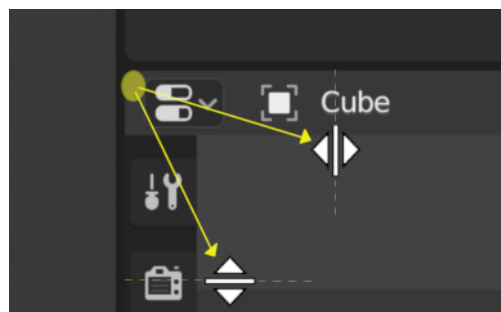
Areas can be customized to match specific tasks called *Workspaces*, which can then be named and saved for later use.

## 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.

## Splitting



Splitting an area will create a new area. Placing the mouse cursor in an area corner will change the cursor to a cross (+) to indicate that pressing down LMB will activate splitting or joining operator. Dragging from area corner **inward** will *split* the area. You define the split direction by dragging either horizontally or vertically.

## Joining

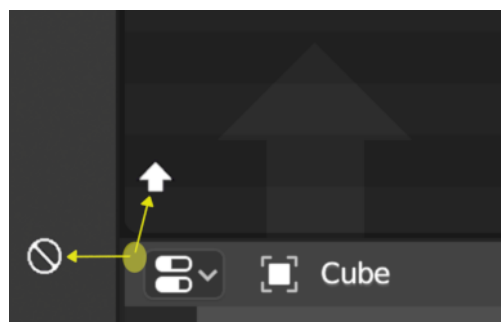


Fig. 11: The Properties is being joined to the Outliner.

Dragging from an area corner **outward** will *join* two areas. The area that will be closed gets a dark overlaid arrow. You can select which area will be closed by moving the mouse over areas.



Release the LMB to complete the join. If you press Esc or RMB before releasing the mouse, the operation will be aborted.

---

**Note:** Areas to be joined must be the same size (width or height) in the direction you wish to join, otherwise nothing will happen. This is so that the combined area remains a rectangle.

---

### Area Options

RMB on the border opens the *Area Options*.

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

**Join Areas** Shows the join direction overlay.

### Swapping Contents

You can swap the contents between two areas with 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 Shift-LMB on the area corner, then drag outward slightly.

The window can be closed with the OS *Close Window* button.

### Toggle Maximize Area

---

#### Reference

**Menu** *View* → *Area* → *Toggle Maximize Area*

**Hotkey** Ctrl-Spacebar

---

The maximized area fill the whole application window. You can maximize an area with *View* → *Area* → *Toggle Maximize Area* menu entry or keyboard shortcut. To return to normal size, use the keyboard shortcut again or the *Back to Previous* button on the Topbar.

---

**Note:** The area your mouse is currently hovering over is the one that will be maximized using the keyboard shortcuts.

---

## Toggle Fullscreen Area

### Reference

**Menu** *View* → *Area* → *Toggle Fullscreen Area*

**Hotkey** Ctrl-Alt-Spacebar

The fullscreen area contains only the main region of the editor. To exit fullscreen use the keyboard shortcut or move the mouse to the top right corner of the area to reveal the return icon.

### 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.

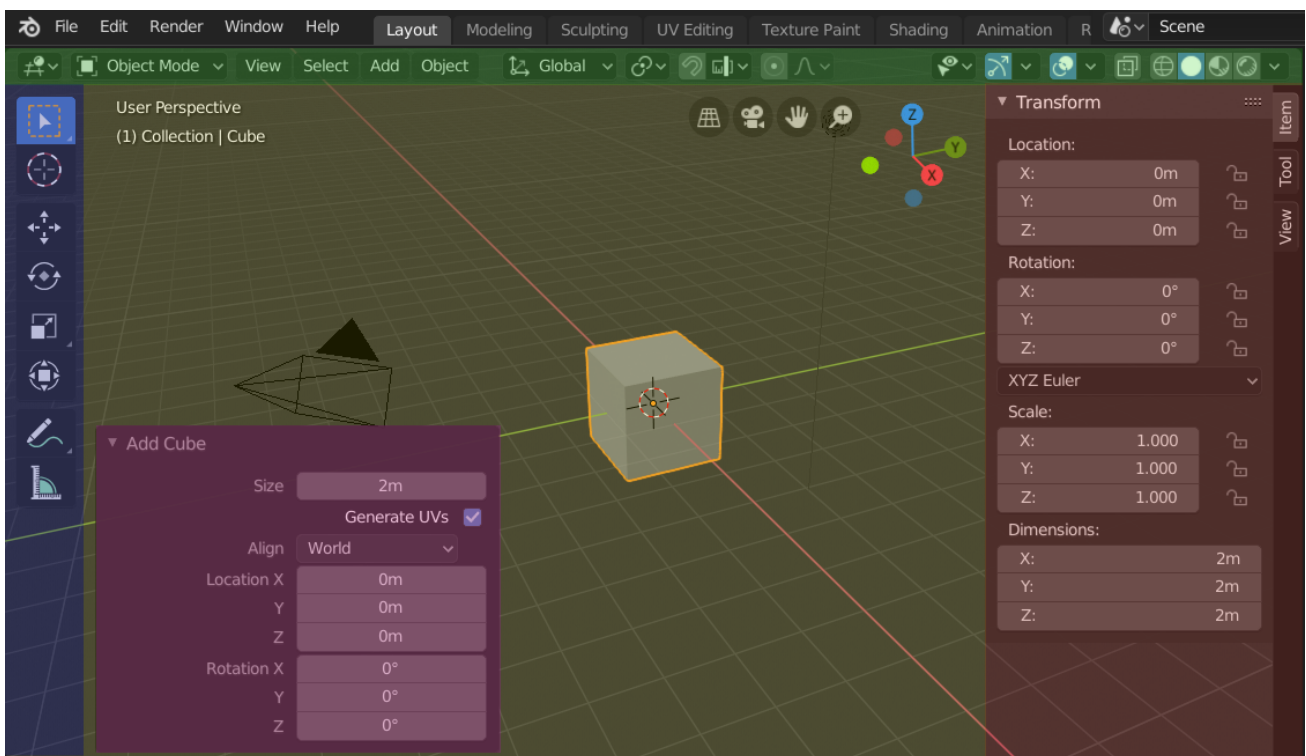


Fig. 12: The regions of the 3D Viewport showing the Sidebar and the Adjust Last Operation panel after adding a Cube.

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.

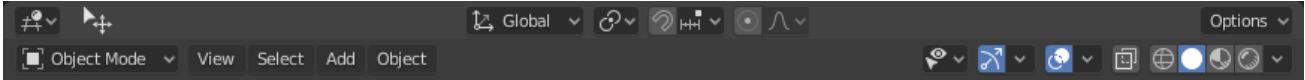


Fig. 13: The Header of the 3D Viewport.

## 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 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.

**Maximize Area** See *Toggle Maximize 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.

This is further documented here: *Toolbar*.

## Tool Settings

The *Tool Settings* (at the top/bottom of the editor area) contains as its name suggests the settings of the active tool. It's visibility can be toggled with the header's context menu just as its position with the *Flip to Bottom/Top* operator.

## Adjust Last Operation

The *Adjust Last Operation* is a region that shows tool options when tools (operators) are run.

This is further documented here: *Adjust Last Operation*.

## 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.

## Arranging

### Scrolling

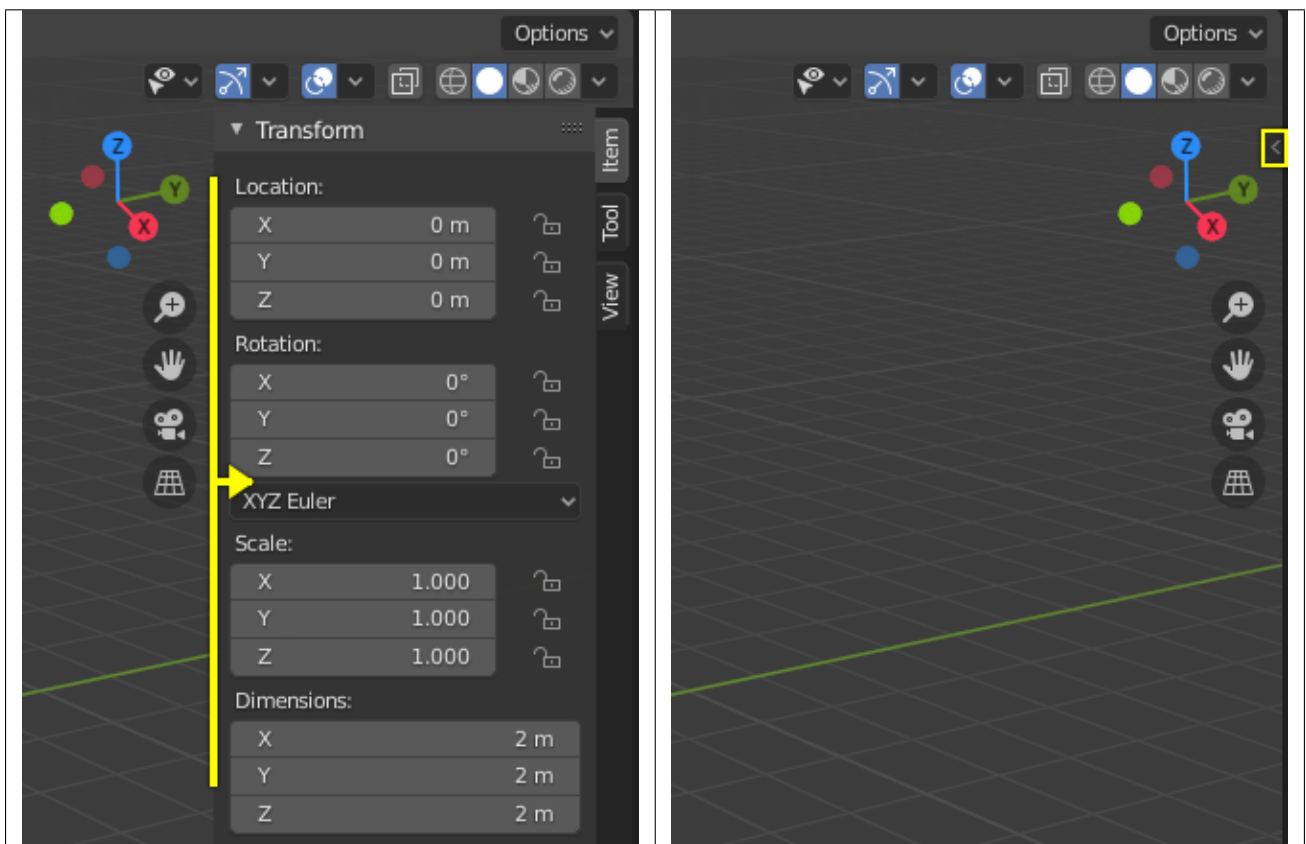
A region can be scrolled vertically and/or horizontally by dragging it with the MMB. If the region has no zoom level, it can be scrolled by using the Wheel, while the mouse hovers over it.

### 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.

Table 2: Hiding and showing the Sidebar.



## Tabs & Panels

### Tabs

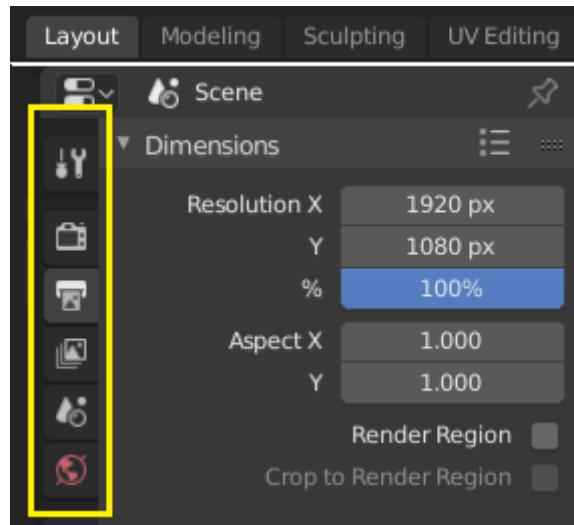


Fig. 14: Top: Horizontal Tab header in the Topbar. Bottom: Vertical Tab header shows tab icons in the Properties.

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

### 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 mouse over tab header icons. (Workspace tabs do not use this keymap. See *Workspace controls*.)

## Panels

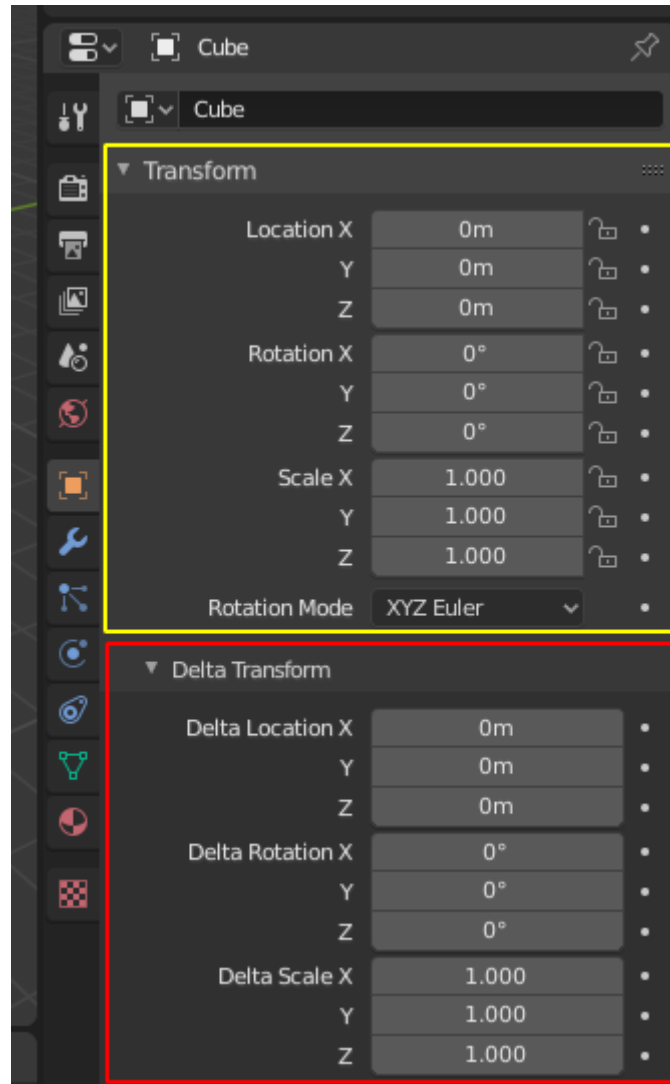


Fig. 15: Panels in Properties.  
A panel is highlighted in yellow and a subpanel in red.

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

### 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 (►).

- A click with the LMB on the panel header expands or collapses it.
- Pressing A expands/collapses the panel under the mouse pointer.
- A Ctrl-LMB click on the header of a specific panel will collapse all other panels and make this the only expanded one.
- A Ctrl-LMB click on the header of a specific panel that contains subpanels will expand / collapse all 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 it with the LMB on the grip widget (:::) located in on the right side of the panel header.

## Pinning

Sometimes it is desirable to view panels from different tabs at the same time. 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 also be pinned by RMB and selecting *Pin*, or you use Shift-LMB on the panel.

## Zoom

The zoom factor of a whole region with panels can be changed by Ctrl-MMB clicking and moving the mouse anywhere within that region or use the NumpadPlus and NumpadMinus to zoom in and out the contents. Pressing Home (Show All) will reset the zooming at the screen/panel focused by the mouse pointer.

## 2.2.3 Keymap

### Common Shortcuts

### Conventions

### 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.

### Mice

This manual refers to mouse buttons as:

**LMB** Left Mouse Button

**RMB** Right Mouse Button

**MMB** Middle Mouse Button

**Wheel** Scrolling the wheel.

## Mouse

Blender's default keymap has two main interaction modes: Right- and left-click-select.

In the past, Blender has used right-click-select to have a more clear distinction between selection and action. In this mode, the RMB (Right Mouse Button) is generally used for selection and the LMB (Left Mouse Button) initiates or confirms actions.

Today, Blender users can choose between the older right-click-select method and left-click-select, which makes Blender feel more like other software.

Video: [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-0	Open file.
Ctrl-S	Save file.
Ctrl-N	New file.
Ctrl-Z	Undo.
Shift-Ctrl-Z	Redo.
Ctrl-Q	Quit.
F1	Help ( <i>context sensitive</i> ).
F2	Rename active item.
F3	Menu Search.
F4	File context menu.
F5 - F8	<i>Reserved for user actions.</i>
F9	Adjust last operation.
F10	<i>Reserved for user actions.</i>
F11	Show render window.
F12	Render the current frame.
Q	Quick access (favorites).
Ctrl-Spacebar	Toggle Maximize Area.
Ctrl-Alt-Spacebar	Toggle Fullscreen Area
Ctrl-PageUp / Ctrl-PageDown	Next/previous Workspace.
Spacebar	User configurable. <b>Play</b> Toggle animation playback. <b>Tools</b> Tool switching with hotkeys (Shift-Spacebar for play). <b>Search</b> Search for actions (Shift-Spacebar for play).
Shift-Ctrl-Spacebar	Playback animation (reverse).

## 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	Edit-mode toggle.
Ctrl-Tab	Mode switching pie menu (toggles Pose Mode for armatures).
1 - 3	Edit mesh vertex/edge/face toggle (Shift extends, Ctrl expands).
AccentGrave	3D Viewport navigation pie menu.
Ctrl-AccentGrave	Toggle gizmos.
Shift-AccentGrave	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.
-------------	--------------

### Preferences

**Select with Mouse Button** Controls which mouse button, either right or left, 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're 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.

**Press** Allows immediate activation, preventing click events being passed to the tool.

**Drag** Allows click events to pass through to the tool, adding a small delay.

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

**Alt Click Tool Prompt** Tapping `Alt` shows a prompt in the Status Bar prompting a second keystroke to activate the tool.

### 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.

**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).**Industry Compatible Keymap**

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

**General**

1 - 9	Mode/Element switching
RMB	Context menu
Tab	Menu Search.
Shift-Tab	Quick access (favorites)
Ctrl-D	Duplicate
P	Set Parent
Return	Rename
Ctrl-Return	Render
B	Proportional Editing / Soft Selection
Ctrl-[	Toggle Toolbar
Ctrl-]	Toggle Sidebar

**Viewport**

Alt-LMB	Orbit View
Alt-MMB	Pan View
Alt-RMB	Zoom View
F1 - F4	Front/Side/Top/Camera Viewpoints
F	Frame Selected
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
]	Select Linked

## Tools

W, E, R	Transform Tools
Q	Box Select
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.

## 2.2.4 Interface Controls

### Buttons and Controls

#### Buttons

##### Operator Buttons

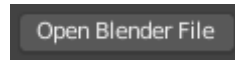


Fig. 16: Operator button.

Operator buttons perform an action when clicked with LMB. Button may show an icon, text, or both.

##### Checkboxes & Toggle Buttons

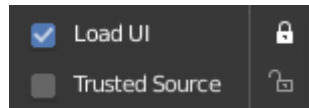


Fig. 17: Checkboxes and Toggle buttons.

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.

#### 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



Fig. 18: Radio buttons.

Radio buttons are used to choose one option from a selection of options. Active button is indicated by a color on the icon background.

#### Cycling

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

## Direction Buttons



Fig. 19: Direction buttons.

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

## Shortcuts

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

## Fields

### Text & Search Fields

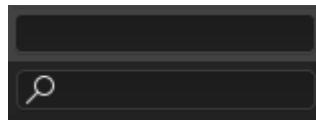


Fig. 20: A text and a search field.

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

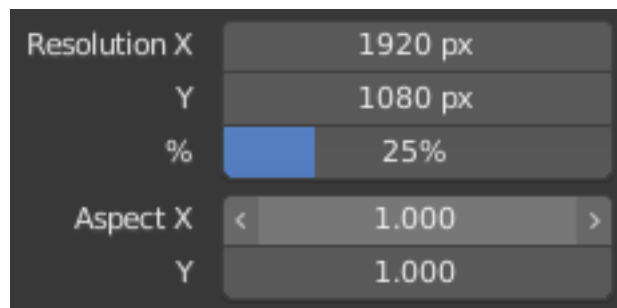


Fig. 21: Number fields.

Number fields store values and units.



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 (only available on first field type). 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 `Ctrl-Tab` to go to the previous field.
- Press `Minus` while hovering over a number field to negate the value.

## Multi-Value Editing

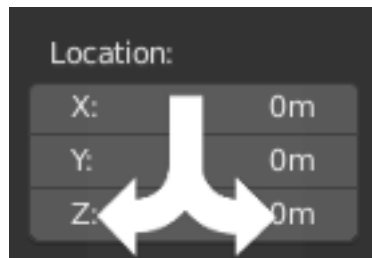


Fig. 22: Multi-value editing.

You can edit multiple number fields at once by pressing down LMB on the first field, and then drag 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.

## 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 access map a value to the current frame, but more complex expressions are also supported, `#fmod(frame, 24) / 24` for example.

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



Fig. 23: Color fields. With and without alpha.

The color field stores a color value shown in its background. Clicking LMB on color fields opens the *Color Picker*. 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 the color with an alpha channel shown over a checker pattern. Colors can be copied to other color fields by being dragged and dropped to another color field.

## Menus

Blender uses a variety of different menus for accessing options and tools. Selecting 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 the menu content is too large to fit on the screen, small menu scrolling indicator triangle appears on bottom or top of menu. Scrolling is done by moving the mouse below or above the scrolling indicator.

## 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 be used to access menu items. (Where 1 is the first menu item, 2 the second, etc. For larger menus Alt-1 the 11th... up to Alt-0 the 20th.)
- Press Return to activate the selected menu item.
- Press Esc to cancel the menu, or move the mouse cursor far from the pop-up, or by LMB clicking anywhere out of it.

## Header Menus

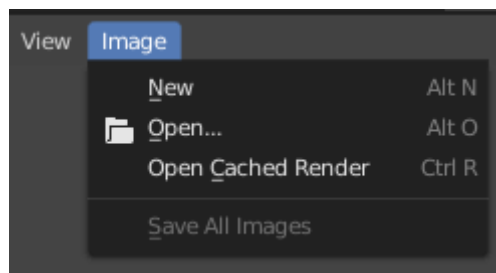


Fig. 24: Image menu in the Header of the Image editor.

Most *headers* exhibit a set of menus, located at the start of the header. Header menus are used to configure the editor and access tools. All menu entries show the relevant shortcut keys, if any.

## 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 select *Collapse Menus*.

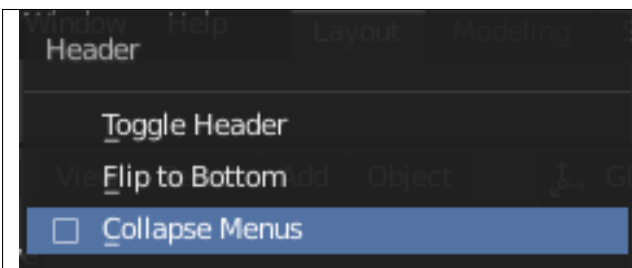


Fig. 25: Right-click on any of the header menus.

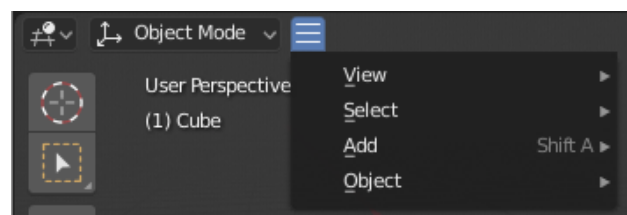


Fig. 26: Access the menu from the collapsed icon.

## Select Menus

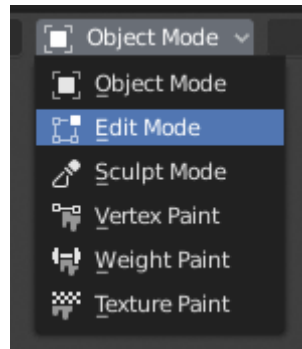


Fig. 27: The 3D Viewport Mode Select menu.

The Select menu (short selector) lets you choose between a set of options. Select menu appears as an icon and/or text with down arrow on the right side of the button. The menu options are shown with LMB click on the button. The selected option is then shown as active on the menu button.

## Popover Menus

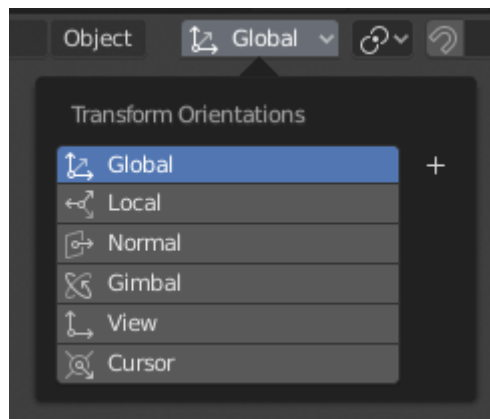


Fig. 28: The Transform Orientations pop-up menu.

Popover menus are overlays. Like Select Menus, pop-up menus also include down arrow on the right side of menu button. However, pop-up menus allow more content to be shown, such as title, list options, buttons, sliders, etc. Popover menus group controls into a menu, which is automatically hidden when mouse pointer leaves menu boundaries (including a margin).

## Context Menu

Context menus are pop-ups opened with the Menu key for editors and RMB for properties. Context menu contents 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. Or when invoked over buttons and properties common options include:

**Single** Set or get single value.

**All** Include all combinations.

**Reset All/Single to Default Value(s)** Replaces the current value by the default (keyboard shortcut Backspace).

**Unset** TODO.

**Copy Data Path** Copies the Python property Data path, relative to the data-block. Useful for Python scripting.

**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, and when "Press a key" appears you must press and/or click the desired shortcut. Press Esc to cancel.

**Change Shortcut** Lets you redefine the shortcut.

**Remove Shortcut** Unlinks the existing shortcut.

**Online Manual** Opens an online page of 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.

**See also:**

*Common Shortcuts.*

## Specials Menu

The Specials pop-up menu contains a context-sensitive list of operators. It is opened by a button with a down arrow on dark background .

## Pie Menus

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

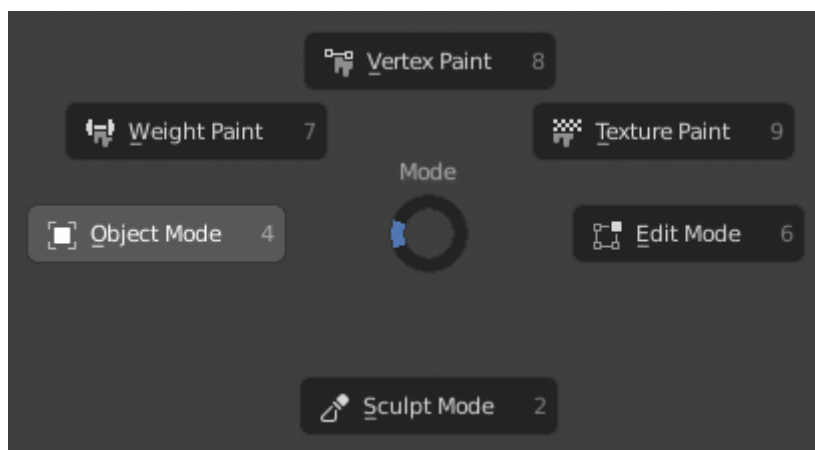


Fig. 29: The 3D Viewport Mode Pie menu.

## Interaction

The pie menu is spawned by a key press, which are listed in the add-on's documentation.

**Tip:** Fastest way to operate a Pie menu is to press down key(s) that invoke the menu, then move mouse slightly towards a selection, and release key(s) to activate the selection.

Releasing the key without moving the mouse will keep the menu open and the user can then move the mouse pointer towards the direction of a pie menu item and select it by clicking. Releasing the key, after moving the mouse towards a pie menu item, will cause the menu to close and the selected menu item to activate.

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. Also number keys can be used to select the items.

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

See *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 pixels color from anywhere within Blender.

**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.

**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*.

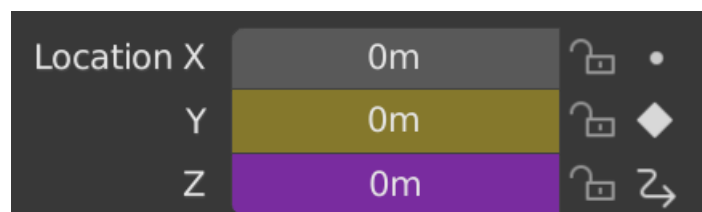


Fig. 30: Decorators indicating different property states.

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 add a keyframe to the current property value and frame.

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

Decorators make it quick and easy to glance over properties and see the state of the property.

**See also:**

*State Colors*

## Extended Controls

### Data-Block Menu

A set of menu buttons used to link *Data-Blocks* to each other. If data-blocks are linked the data will be updated across all of the *data users* when edited.

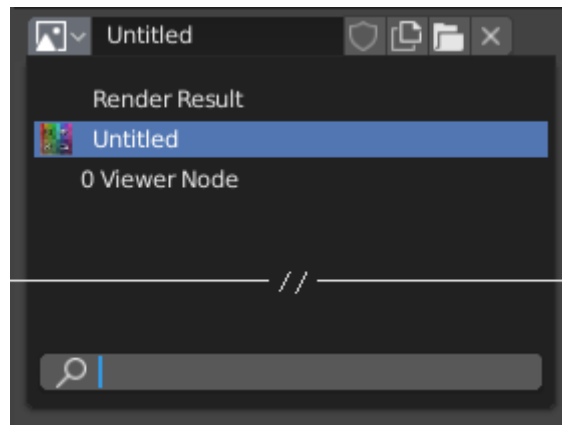


Fig. 31: A data-block menu with a search field.

**Type** Shows an icon indicating the data-block type. It opens up the following pop-up menu. The data-block can be dragged from here e.g. to drag a material onto an object in the 3D Viewport or into a *Data ID* field.

**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.

**Name** Displays the internal name of the linked Data-Block, which can be edited as a regular text field. If a name is already assigned, Blender will add a digit to the name like “.001”.

**User Count** Displays the number of *data users* of the data. Clicking on user count button will make it a single-user copy, with it linked only to the active object/object’s data.

**Fake User (shield icon)** Keeps the data-block saved in the blend-file, even if it has no *Real User*. When activated an “F” will be shown before the name in the list.

**Make Local (chain icon)** Todo <2.79.

**New/Add (files icon)** Creates a new data-block or duplicates the current data-block and applies it.

**Open File (folder icon)** Opens the *File Browser*.

**Unpack File (bin icon)** *Unpack* the file packed into the current blend-file to external ones.



**Unlink Data-block X** 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**

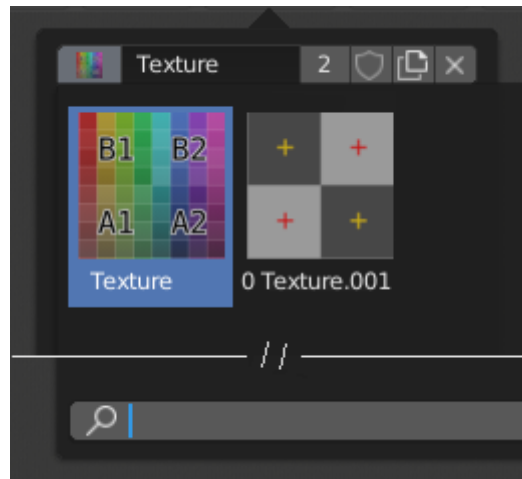


Fig. 32: The Data-Block menu with preview.

In the Tool Settings is a version of the data-block menu with a bigger preview.

**Data ID**

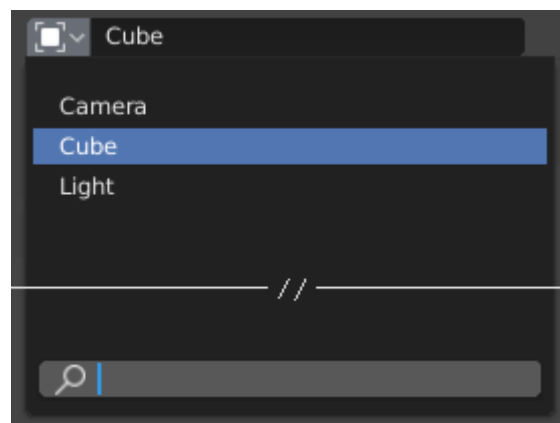


Fig. 33: A Data ID field.

A Data ID is a text field with an icon on the left, which opens a pop-up. Data ID is a unique name for an object. Data ID is used to refer to objects, and therefore Blender does not allow any two objects of same type to have same ID (same name). If Data ID is already in use, Blender will automatically append a number to the end to prevent ID collision (for example “Cube.001”).

Menus showing Data IDs can show the following elements:

**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 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.

**Remove X** Click the X button on the right to remove the reference.

## Sub IDs

Related types of IDs may become available to select a property or child object, depending on the object type.

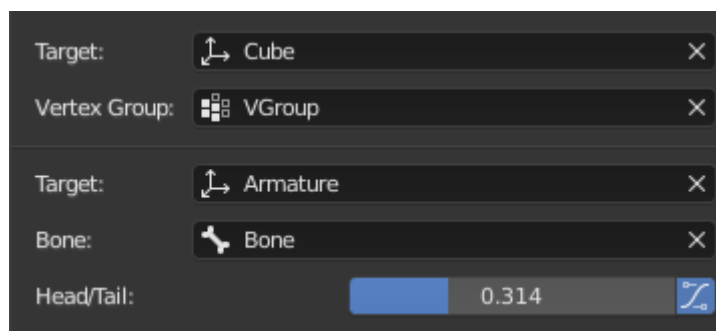


Fig. 34: Sub ID Example.

**Vertex Group** If the selected object in the *Name* field is a mesh or a lattice, an additional field is displayed where a vertex group can be selected.

**Bone** If the selected object in the *Name* field is an armature, a new field is displayed offering the choice to specify an individual bone by entering its name in the *Bone* data ID.

**Head/Tail** If a Bone is set, a new field is displayed offering the choice of whether the head or tail of a Bone will be pointed at. The slider defines where along this bone the point lies interpolating along the bone axis in a straight line. A value of zero will point at the Head/Root of a Bone, while a value of one will point at the Tail/Tip of a Bone.

**Use B-Bone Shape** When the bone is a *bendy bone*, click on this button to make the point follow the curvature of the B-spline between head and tail.

## List Views & Presets

### List Views

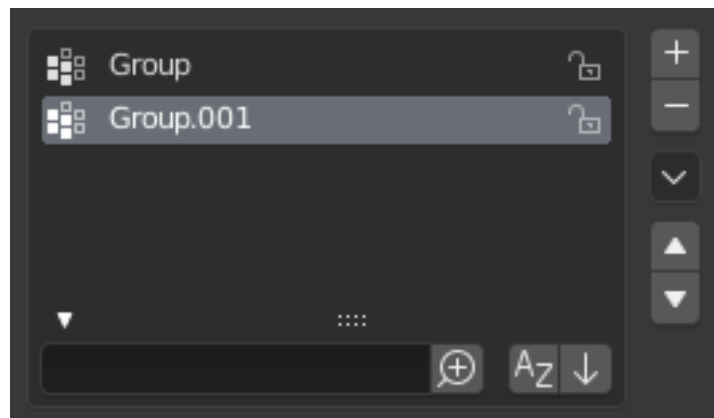


Fig. 35: List view with expanded Filtering Options panel.

This control is useful to manage lists of items. They can be found, for example, in the object data properties. In addition to the main list, there is a Filtering panel on the bottom (hidden by default) and modification buttons on the right.

**Select** To select an item, 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 pressing `Ctrl-LMB` over it.

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

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

**Search** Type part of a list item's name in the filter text field to filter items by part of their name.

**Filter Include** When the magnifying glass icon has a + sign then only items that match the text will be displayed.

**Filter Exclude** When the magnifying glass icon has a - sign then only items that do not match text will be displayed.

**Sort** Sort list items.

**Alphabetical** This button switches between alphabetical and non-alphabetical ordering.

**Inverse** 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 v** A *Specials* menu with tools to operate on list entries.

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

## Presets



Fig. 36: Example Presets menu.

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

**Add +** New presets can be added based on 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.

**Specials** Optional *Specials* menu with tools to operate on list entries.

## Color Picker

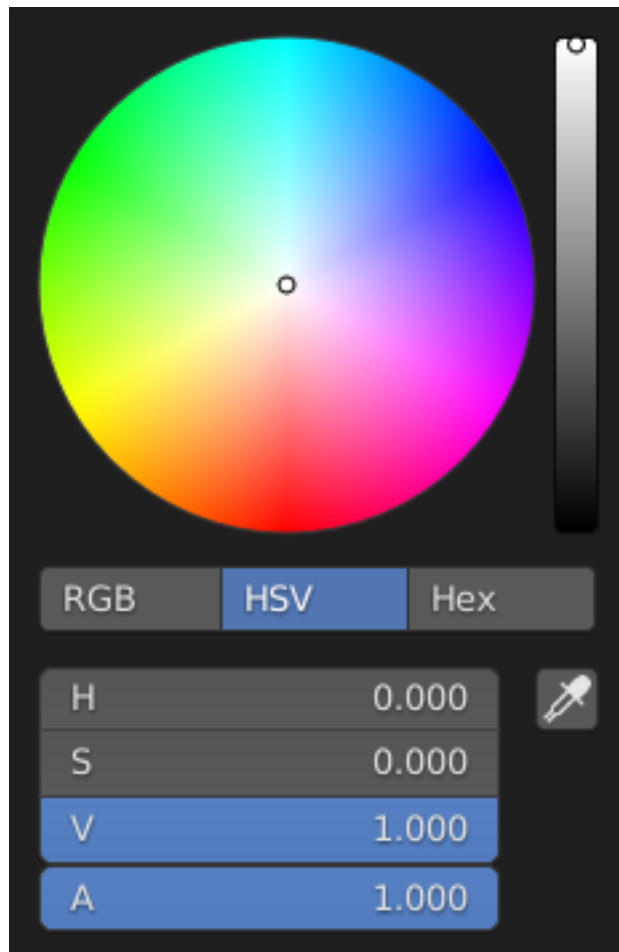


Fig. 37: Circle HSV.

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.

**Color field** Lets you pick the first and second color component. The shape can be selected by the *Types*.

**Color slider** The slider with a gradient in the background lets you define the third color component. It can also be controlled with the *Wheel*.

**Color space** Selects the *Color Space* for the number fields below.

RGB, HSV/HSL, Hex

**Color values** Blender uses (0 to 1.0) values to express colors for RGB and HSV colors.

Hexadecimal (Hex) values are expressed as RRGGBB. Shorthand hex colors are also supported as RGB, e.g. dark-yellow FFCC00, can be written as FC0.

For operations that are capable of using Alpha, another slider “A” is added.

**Eyedropper** The *Eyedropper* (pipette icon) can be used to sample a color value from inside the Blender window.

---

**Note:** In Blender, the *Hex* and HSV/HSL values are automatically *Gamma* corrected; however, for the RGB values, they are in Scene Linear color space, and are therefore not gamma corrected. For more information, see the *Color Management and Exposure* page.

---

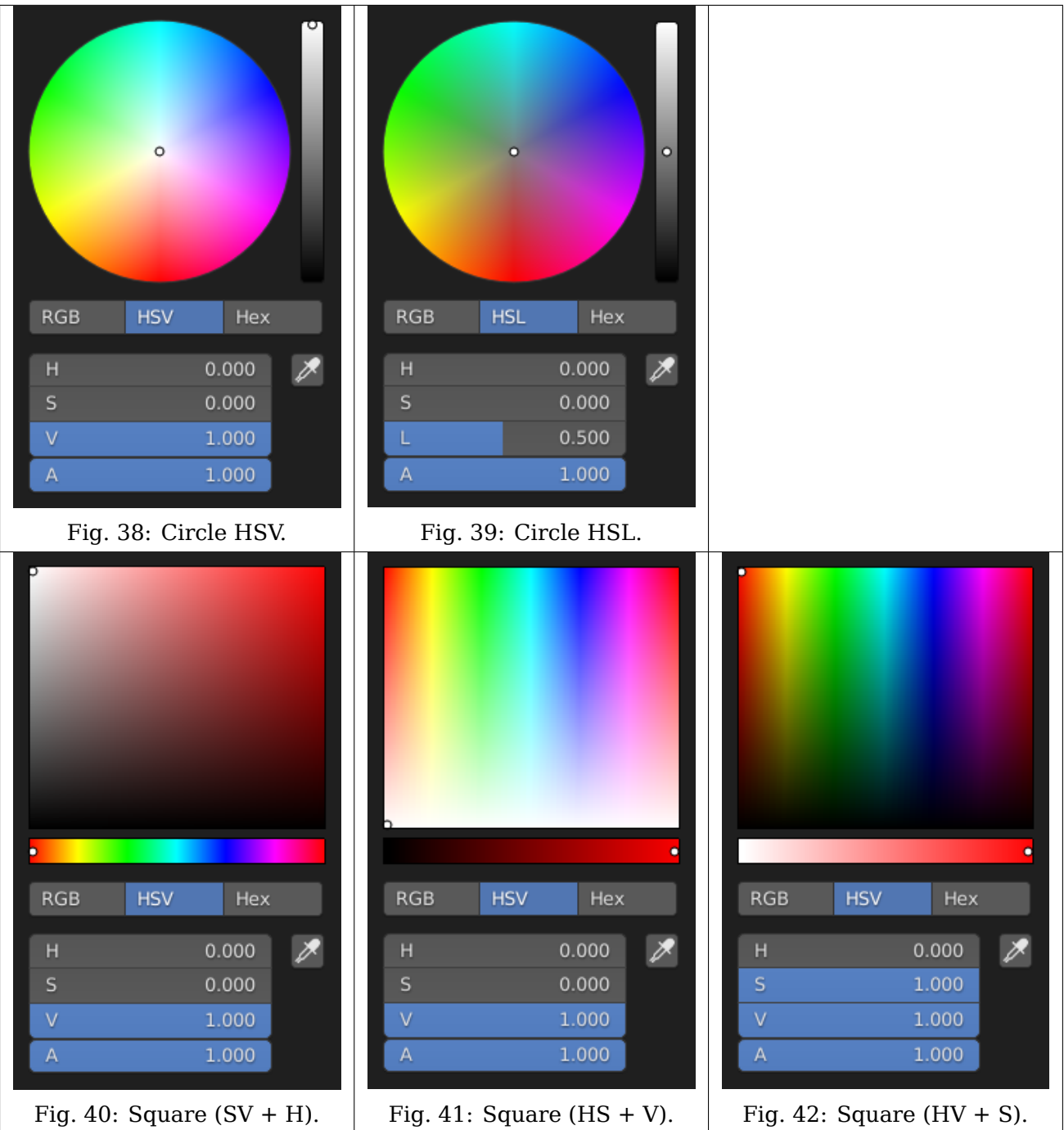
## Types

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

**Circle** The color values ranging from center to the borders. The center is a mix of the colors.

**Square** The Borders of the square are the axis for the two color components, with the center on the bottom right.

Table 3: Color Picker types.



### Shortcuts

- Ctrl-LMB (drag) snaps to hue.
- Shift-LMB (drag) precision motion.
- Wheel adjust the brightness.
- Backspace reset the value.

## Color Ramp Widget

*Color Ramps* enables the user to specify a range of colors based on color stops. Color stops are similar to a mark indicating where exactly the chosen color should be. The interval from each of the stops, added to the ramp, is a result of the color interpolation and chosen interpolation method.

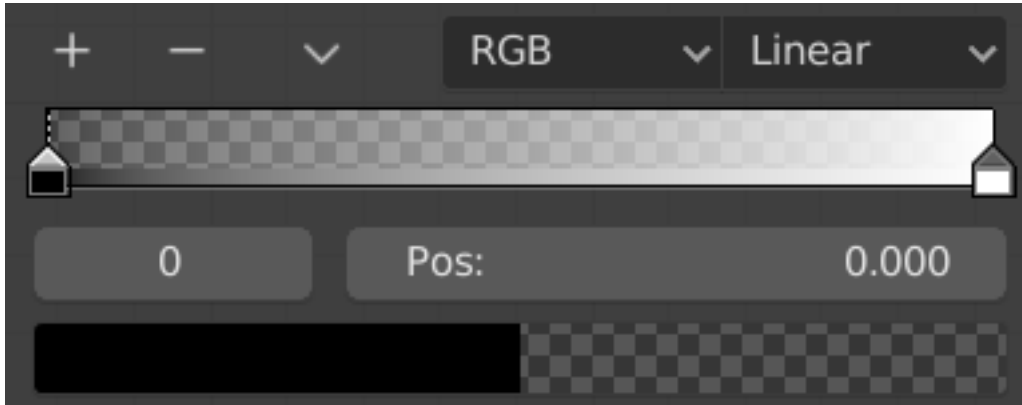


Fig. 43: Color ramp.

### Controls

**Add +** Clicking on this button will add a stop to your color ramp. The stops are added from the last selected stop to the next one, from left to right and they will be placed in the middle of both stops.

**Delete -** Deletes the selected color stop from the list.

**Specials v** Contains more operators for the color ramp.

**Flip Color Ramp** Flips the gradient, inverting the values of the color ramp.

**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** Space between all neighboring stops becomes equal.

**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 Space* 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.

**Interpolation Options** Enables the user to choose the types of calculations for the color interpolation for each color stop.

#### 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 and 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.



Fig. 44: HSV and HSL interpolation options.

**Active Color Stop** Index of the active color stop (shown as a dashed line). Allows you to change the active color when colors may be too close to easily select with the cursor.

**Position** This slider controls the positioning of the selected color stop in the range.

**Color** Opens a color picker for the user to specify color and Alpha for the selected color stop. When a color is using Alpha, the color field is then divided in two, with the left side showing the base color and the right side showing the color with the alpha value.

### Shortcuts

- LMB (drag) moves colors.
- Ctrl-LMB (click) adds a new control point.

## Color Palette



Fig. 45: 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.

### Shortcuts

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

## Curve Widget

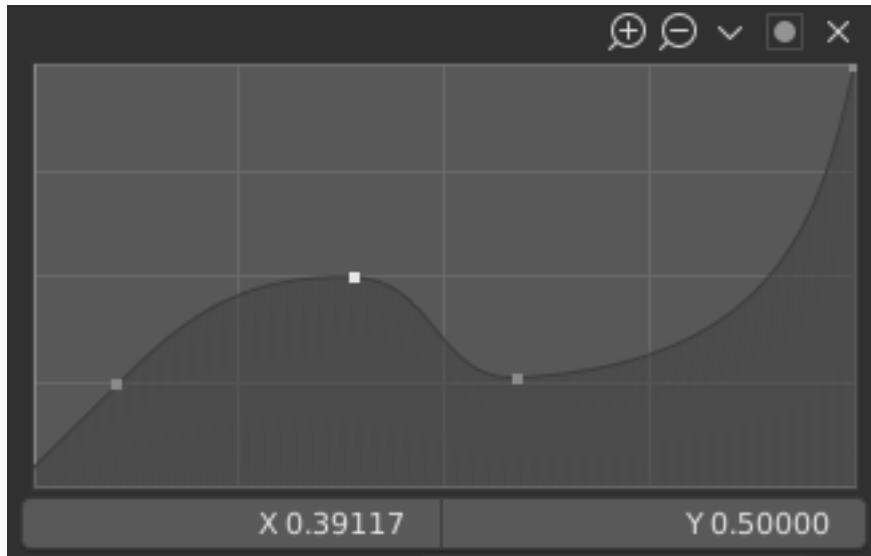


Fig. 46: Curve widget.

The purpose of the *Curve Widget* is to allow the user to modify an input (such as an image) in an intuitive manner by smoothly adjusting the values up and down using the curve.

The input values are mapped to the X axis of the graph, and the output values are mapped to the Y axis.

### 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 X button at the top 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 tools to operate on control points or to set properties.

**Reset View** Resets the view of the curve.

**Handle Options** 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; breaking the tangent at the curve handle, making it an angle.

**Auto Handle** Automatic handles that create smooth curves.

**Auto Clamped Handle** Automatic handles that create smooth curves, which prevents 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.

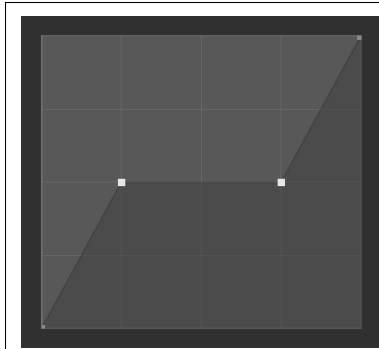


Fig. 47: Vector Handles.

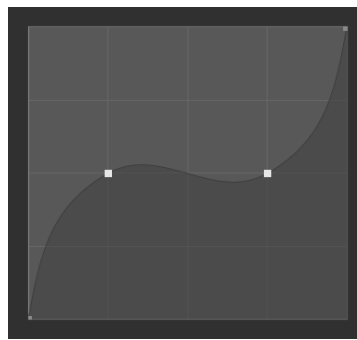


Fig. 48: Auto Handles.

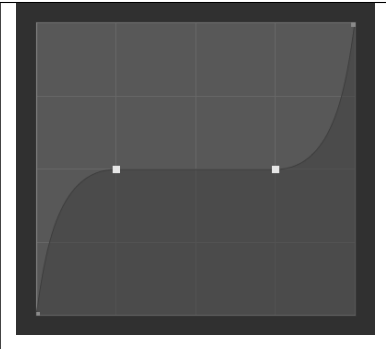


Fig. 49: Auto Clamped Handles.

**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.

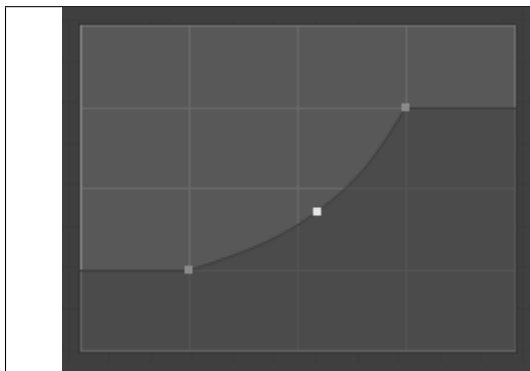


Fig. 50: Extend Horizontal.

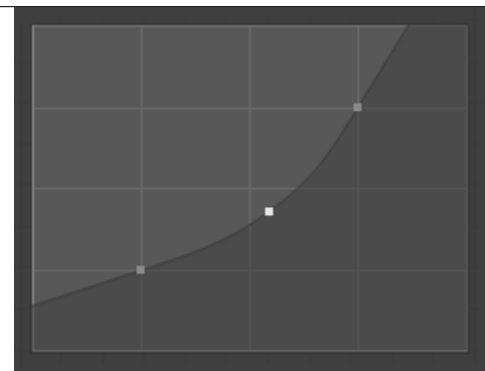


Fig. 51: Extend Extrapolated.

**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.

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

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

**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.

## Search

### Menu Search

---

#### Reference

**Mode** All Modes

**Menu** *Edit* → *Menu Search*

**Hotkey** F3

---

The Menu Search pop-up searches Blender's interface for a desired tool and allows you to execute that tool. It returns a list of matches and shows which menu the tool was found in. Start typing the name of the tool you want to refine the list. When the list is sufficiently narrowed, LMB on the desired tool or navigate with Down and Up, run the tool by pressing Return.

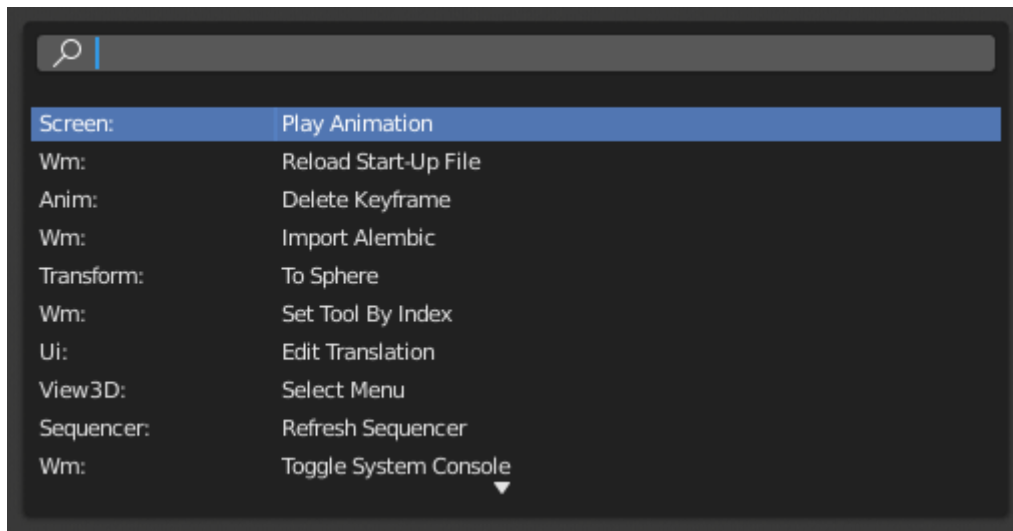


Fig. 52: The Menu Search pop-up.

#### 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 search menu search 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.




## Nodes

### Introduction

The different node editors are used to work with node-based workflows. Each node editor type has its own specific purpose. Therefore, this section only explains how to work with nodes in general. In the list below it shows a list of different types of node trees and where each is documented.



Fig. 53: Example of a node editor.

Icon	Name	Documentation
	Shader Nodes	Documentation is in the <i>Render</i> section.
	Composite Nodes	Documentation can be found in the <i>Compositing</i> section.
	Texture Nodes	Texture Nodes are covered in the <i>UV editor</i> docs.

### Editor Interface

#### Header

The *Header* contains various menus, buttons and options, partially based on the current node tree type.

**View** This menu changes your view of the editor.



Fig. 54: Common node editor header options.

**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.

**Use Pinned** 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** This button allows you to parent node tree e.g. leaving a group.

**Snapping** Change options for snapping node positions to achieve a cleaner node tree layout.

## Toolbar

The *Toolbar* contains a set of tools that can be used in the node editor.

## Sidebar

The *Sidebar* region contains properties for the current 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

**Mode** All Modes

**Tool** *Toolbar*

**Menu** *Add*

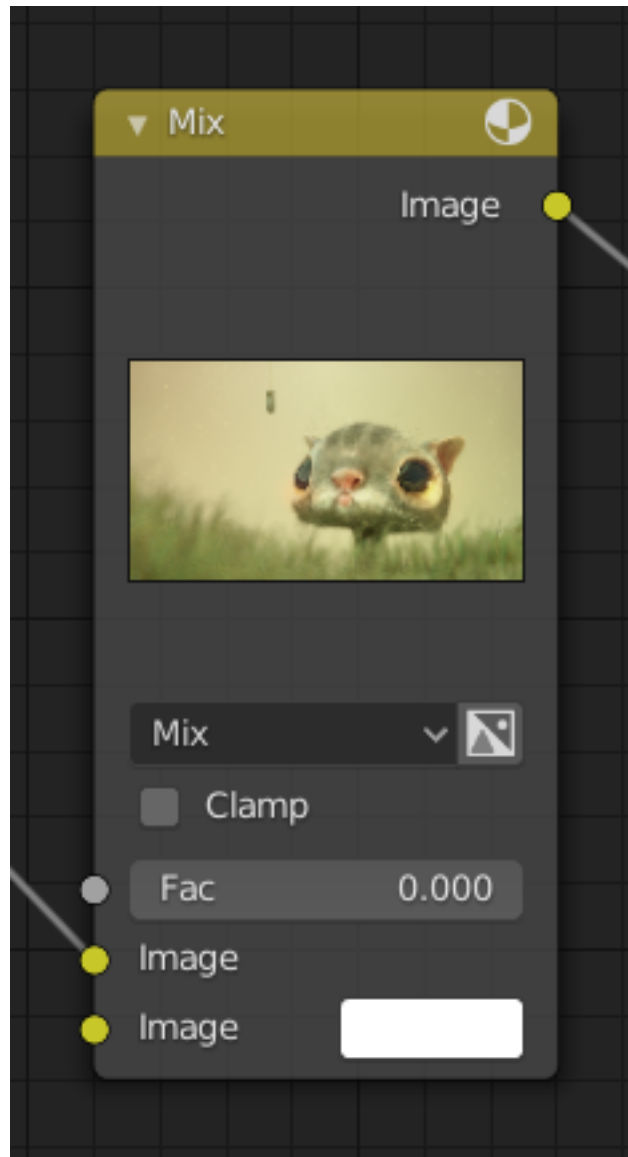
**Hotkey** Shift-A

---

Nodes are added via the *Add* menu or using the Shift-A shortcut.

## 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, Preview and more.



## Title

The *Title* shows the name/type of the node. It can be overridden by changing the value of Label in the *Node* section of the *Sidebar region N*. 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.



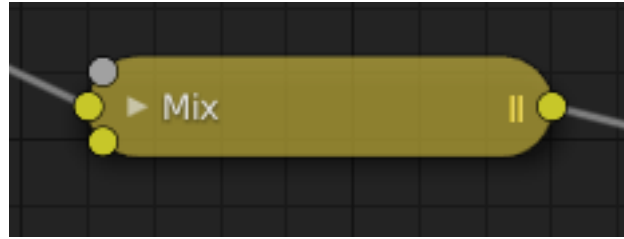
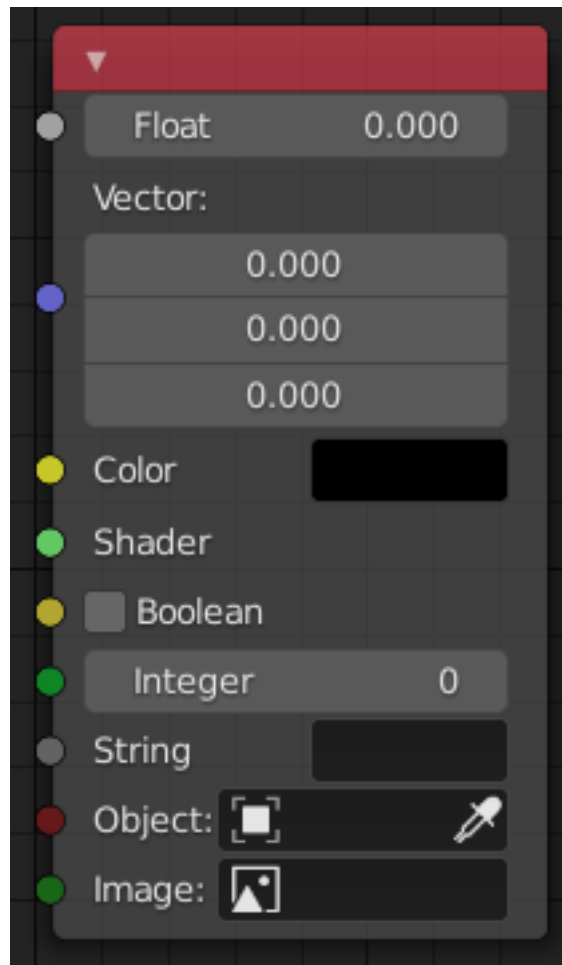


Fig. 55: How a node appears when collapsed.

## Sockets



The *Sockets* input and output values from the node. They appear as little colored circles on either side of the node. Unused sockets can be hidden with `Ctrl-H`. There are two kinds of sockets: *inputs* and *outputs*.

Each socket is color-coded depending on what type of data it handles.

**Float (gray)** Indicates numeric value's information. It can either be a single numerical value or a so-called "value map". (You can think of a value map as a gray-scale map where the different amount of bright/dark reflects the value for each point.) 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. Common use: Alpha maps and value options for a node.

**Vector (blue)** Indicates vector, coordinate and normal information.

**Color (yellow)** Indicates that color information needs to be input or will be output from the node. Depending on the node tree type, the color has an alpha channel or not.

**Shader (bright green)** Used for shaders in *Cycles* and *Eevee*.

**Boolean (soft yellow)** Used to pass a true or false value.

**Integer (lime green)** Used to pass an integer value.

**String (dark gray)** Used to pass a string value.

**Object (dark red)** Used to pass an object data-block.

**Image (dark green)** Used to pass an image data-block.

## Inputs

The *Inputs* are located on 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.

## 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 other socket types either implicitly or explicitly. Implicit conversion can happen automatically without the need of a conversion node.

For example, color and float sockets can both be placed into one another. 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 will default to use radians regardless of the scene *Units*. This happens because the value node has no unit while the angle input does.

Valid conversions:

- Between color and vector - in this case the using individual color channels to store the vector.
- Between color and float - the color data is converted to its gray scale equivalent.
- Color/float/vector to Shader - implicitly converts to color and gives the result of using an emission node.

Explicit conversion requires the use of a conversion node for example 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.

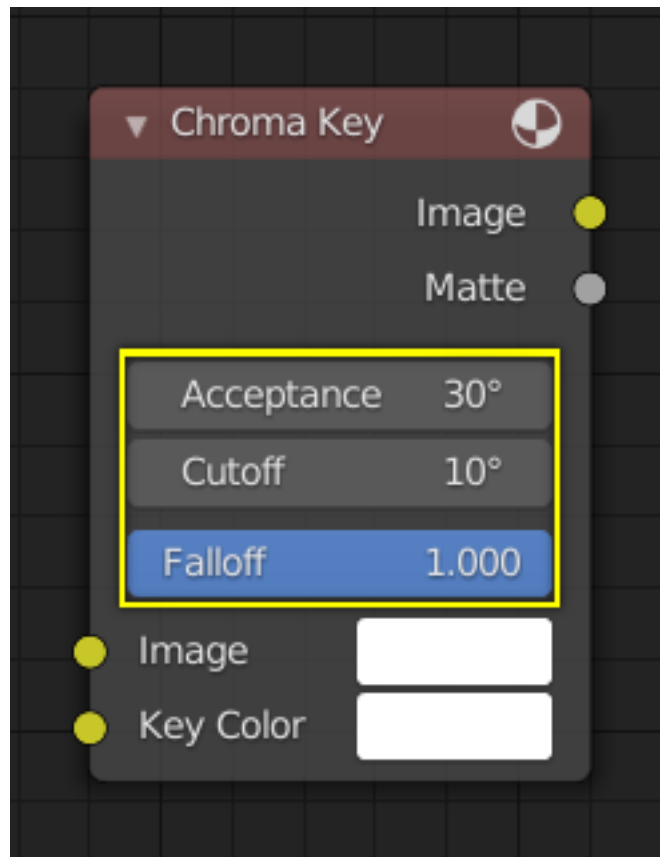


Fig. 56: An example of the controls on the Chroma Key node.

## Preview

On some nodes this shows a preview image of how the output data for a certain channel will appear. Usually it shows color data.

The preview can be toggled using the icon on the very top right-hand corner of the node, next to the title.

## Selecting

**Box Select** Click and drag to box select multiple nodes. Alternatively, B starts the bounding box selection process as well.

**Lasso Select** Ctrl-Alt-LMB click and drag starts a lasso selection.

**Select All A** Select all nodes.

**Deselect All Alt-A** Deselect all nodes.

**Invert Ctrl-I** Invert the selection.

**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 similar nodes to the active node by their *properties*.

**Type** The node type. e.g. all Math nodes.

**Color** The color property.

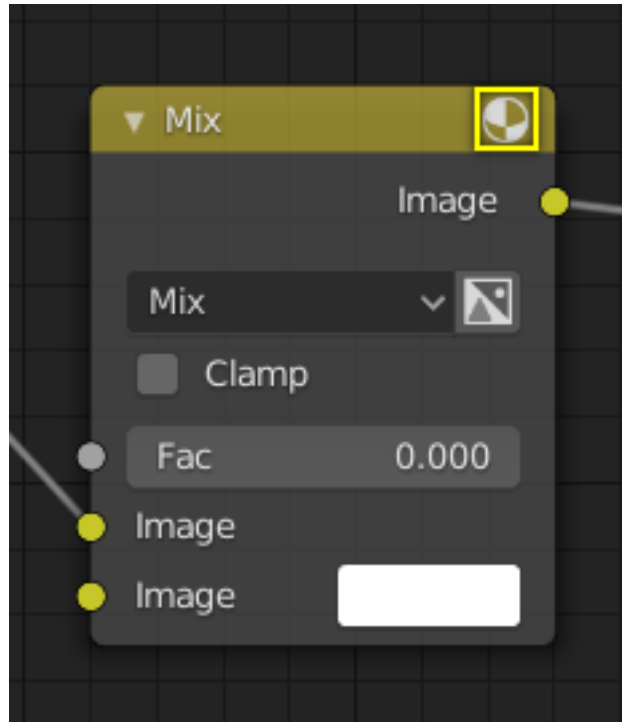


Fig. 57: How a node appears without the preview.

**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 the node, and ensures the node is visible.

**Find Node Ctrl-F** To search for a node. On selecting a node, it activates the node and makes sure the node is visible.

**Select Multiple** Shift-LMB or Shift-RMB used for multiple node selection.

## Arranging Nodes

### Snapping

**Snap** Toggle snapping mode for moving nodes around.

**Snap Node Element Selector** This selector provide the following node elements for snap:

**Grid** Snap to grid background.

**Node X** Snap to left/right node border.

**Node Y** Snap to top/bottom node border.

**Node X/Y** Snap to any node border.

**Snap Target** Which part to snap onto the target.

**Closest** Snap closest point onto target.

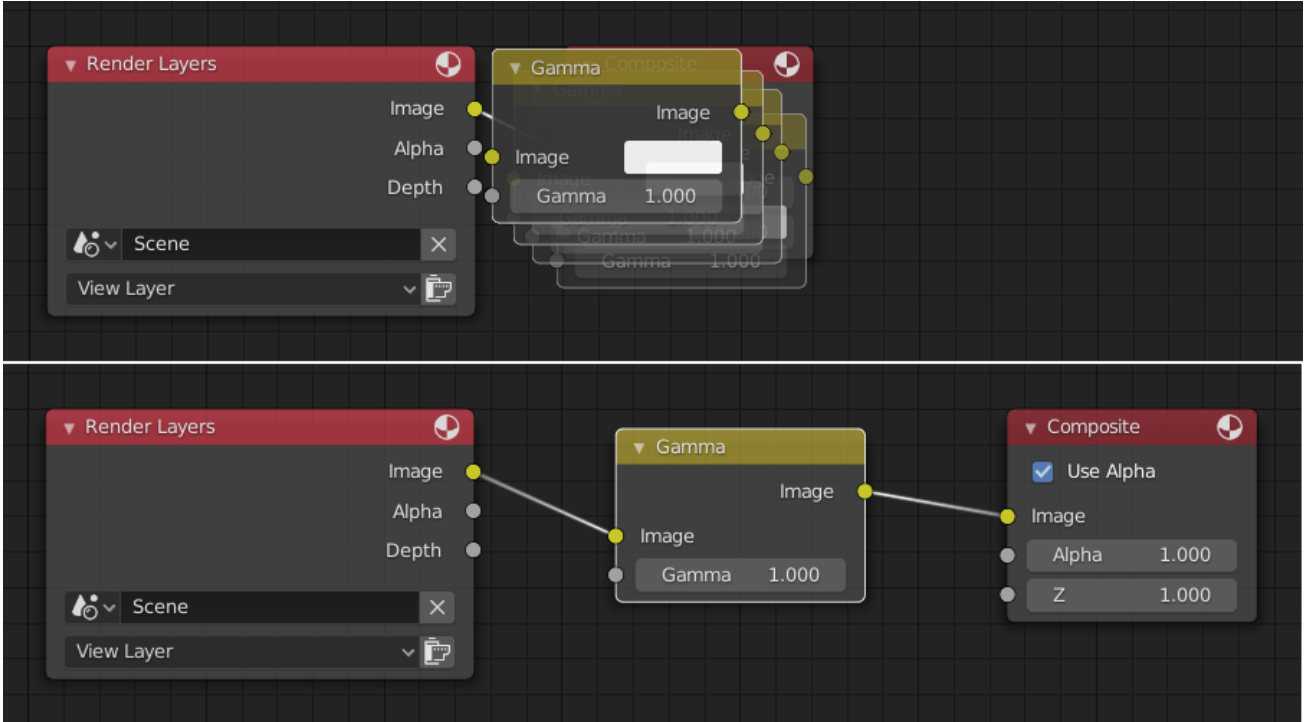
**Center** Snap center onto target.

**Median** Snap median onto target.

**Active** Snap active 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 from the editor's header.

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

**Mode** All Modes

**Menu** *Node* → *Move, Rotate, Resize*

**Hotkey** G, R, S

Move a single node by clicking and dragging it around. A node can be clicked almost anywhere to start dragging. Multiple nodes can be moved after pressing G.

In general it is recommended to arrange your nodes within the view such that the data flows from left to right, top to bottom.

A node can be resized by dragging the edges on the left or right side.

## Connecting Sockets

### Interactively

LMB-click on a socket and drag. You 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, only a single link can be attached to an input 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. 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 from an input socket and let it go keeping it unconnected.

## Cut Links

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Cut Links*

**Hotkey** `Ctrl-RMB`

---

To break a link between sockets click in an empty area, near the link you want to disconnect, and drag: You will see a little cutter icon appearing at your mouse pointer. Move it over the link itself, and release.

**Detach Links Alt-D, Alt-LMB drag** Use Detach Links in order to cut all links attached to selected nodes at once.

## Duplicate

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Duplicate*

**Hotkey** `Shift-D`

---

Click LMB or RMB on the desired node, press **Shift-D** and move the mouse away to see the duplicate of the selected node appearing under the mouse pointer.

---

**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.

---

## Copy/Paste

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Copy*, *Node* → *Paste*

**Hotkey** **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.

---

## Delete

**Delete X**, **Delete** Deletes the selected node(s).

**Delete with Reconnect Ctrl-X** Delete the node(s) without losing the connections.

## Mute

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Toggle Node Mute*

**Hotkey** **M**

---

Muting a node removes the contribution of the node to the node tree, and makes all links pass through that node without change. Links will appear red as an indicator of passing through the muted node.

## Show/Hide

**Hide H** Collapses the node so only the node header is visible. This can also be toggled by clicking the triangle at the top 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.

**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

---

**Note:** The tools are only used in the *Compositor*.

---

**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. This can also be used to recover some information from a failed render. For this to work, *Save Buffers* must be enabled.

## Sidebar

### Item

---

## Reference

**Panel** *Sidebar region* → *Item*

---

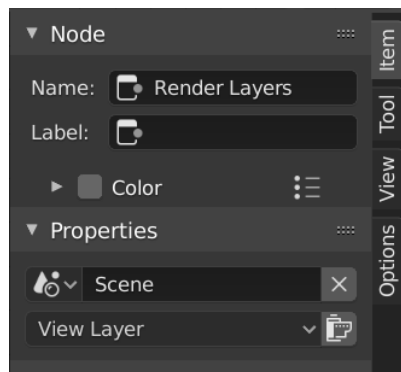


Fig. 58: Item tab with a compositing Render Layers node selected.

### Node

**Name** A unique node identifier inside this node tree.

**Label** Nodes can be given a title by modifying the text field.

### Color

**Color Presets** Colors saved as a preset for reuse in other nodes.

**Color** Color of the node background. Node colors can be used to provide a visual cue.



---

## 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

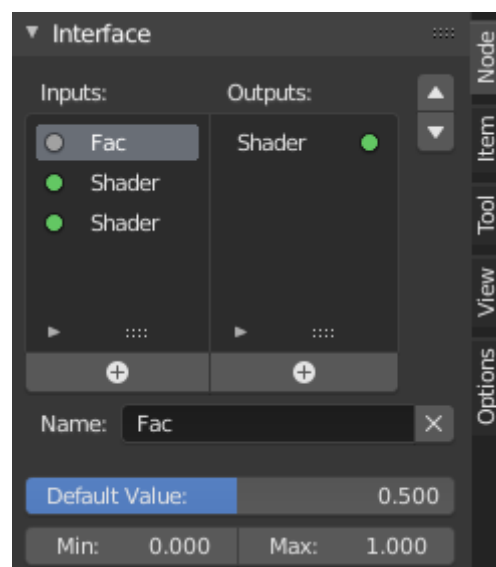


Fig. 59: Example of a node group.

Grouping nodes can simplify a node tree by allowing instancing and hiding parts of the tree. Both material and composite nodes can be grouped.

Conceptually, grouping nodes allows you to specify a *set* of nodes that you can treat as though it were “just one node”. Node groups are similar to functions in programming, they can be reused in many places in a node tree and can be customized by changing the “parameters” of the node group.

As an example: If you have created a material that you would like to use with different inputs e.g. diffuse color: red plastic, green plastic. You could create different materials with *Make Single User* for each different color with a copy of the tree part describing the plastic material. If you like to edit the material you would need to redo the edit on all materials. A better method of reuse is to create node groups, exposing only the variable inputs (e.g. diffuse color).

Also nested node groups are supported. I.e. a node group can be inserted or created inside another 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).

---

## Make Group

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Make Group*

**Hotkey** Ctrl-G

---

To create a node group, select the nodes you want to include, then press Ctrl-G, *Group* → *Make Group*. A node group will have a green title bar. All of the 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 easily distinguish between the two types.

---

**Tip:** What **not** to include in node groups:

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not** include:

**Input nodes** If you include a source node in your group, you will end up having the source node appearing *twice*: once inside the group, and once outside the group in the new material node tree.

**Output node** If you include an output node in the group, there will not be an output socket available *from* the group!

---

## Edit Group

---

### Reference

**Mode** All Modes



**Mode** All Modes

**Panel** *Sidebar region* → *Node* → *Interface*

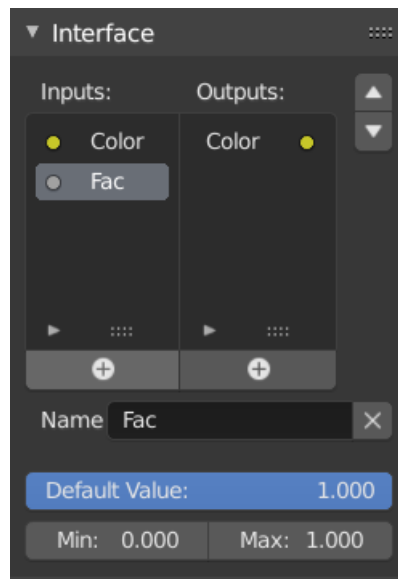


Fig. 61: The interface panel for editing groups.

Sockets can be added, re-ordered, or removed, descriptive names can be added and the details of the input data value defined here.

If you have multiple inputs or outputs, they can be re-ordered by selecting the socket in the list and then moving it up or down with the arrow buttons on the right side of the panel. The larger plus sign buttons below the list will add an unconnected socket of the same type as the selected socket or a value socket if there is no selection. The triangle at the bottom of the list has filtering functions to facilitate finding nodes if the group has a large number of sockets.

## Ungroup

### Reference

**Mode** All Modes

**Menu** *Group* → *Ungroup*

**Hotkey** Ctrl-Alt-G

The Ctrl-Alt-G tool removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread 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.

## Group Insert

### Reference

**Mode** All Modes

**Menu** *Node → Group Insert*

Selecting a set of nodes, ending with the destination group node, and pressing *Node → Group Insert* 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.

## Appending Node Groups

### Reference

**Editor** Topbar

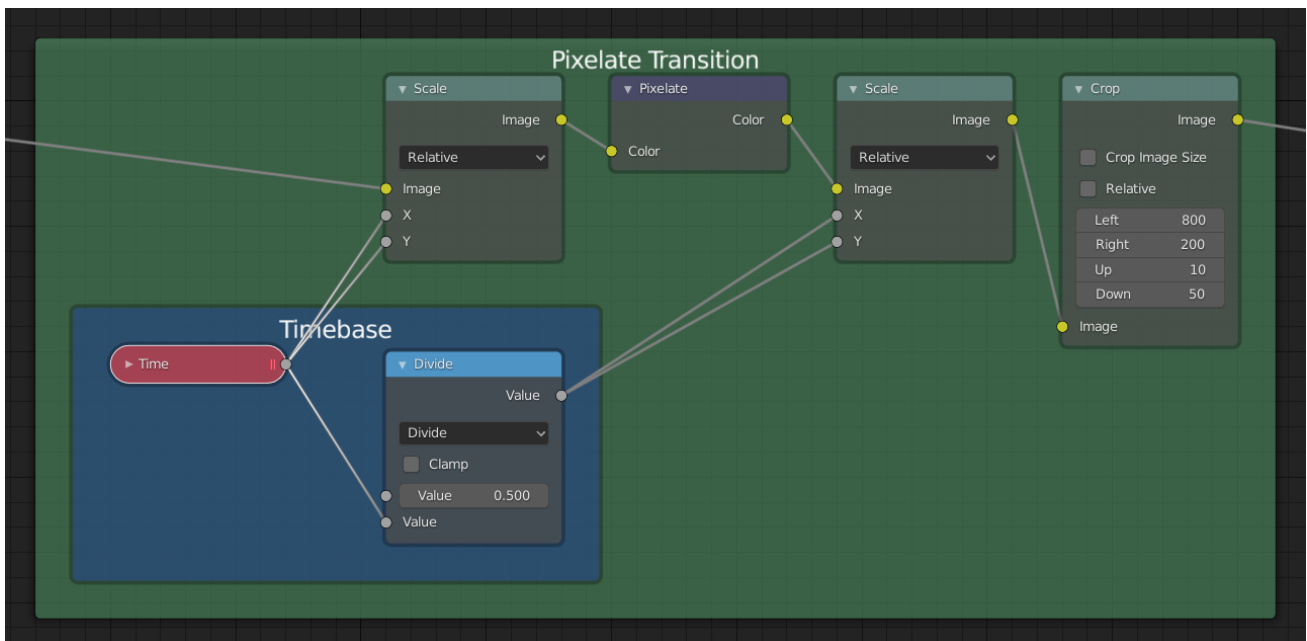
**Mode** All Modes

**Menu** *File → Link/Append*

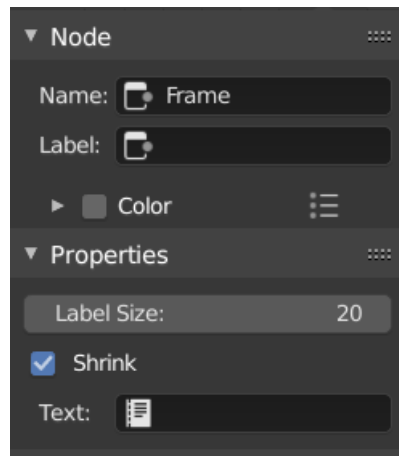
Once you have appended a Node Tree to your blend-file, you can make use of it in a node editor by pressing *Shift-A, Add → Group*, then selecting the appended group. The “control panel” of the Group include the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

### Frame Node

The Frame node is a useful tool 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

**Mode** All Modes

**Menu** *Node* → *Join in new Frame*

**Hotkey** Ctrl-J

---

Make a new frame including the selected nodes.

### Add to Frame

---

#### Reference

**Mode** All Modes

**Hotkey** 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

**Mode** All Modes

**Menu** *Node* → *Remove from Frame*

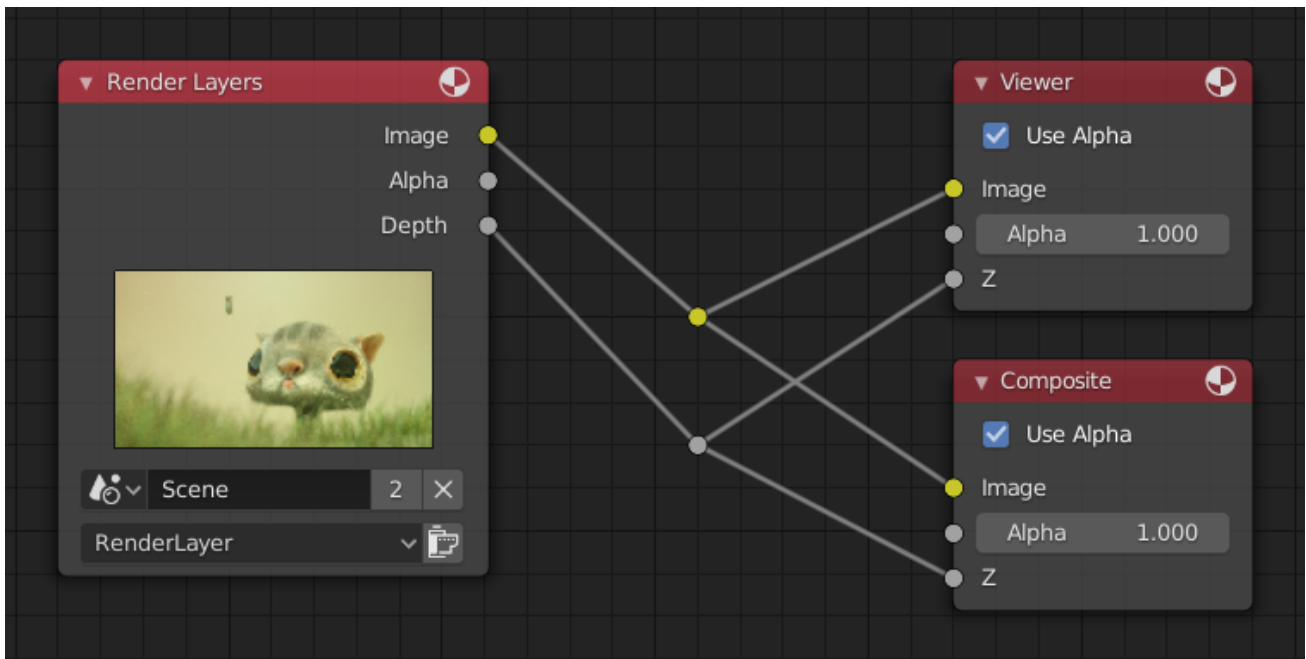
**Hotkey** Alt-P

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.

## 2.2.5 Tools

### 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 which is stored for each space & mode.

Tools may set their own keys which override other keys although typically they use the LMB, sometimes with modifier keys. *Keymaps can be edited from the preferences.*

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

## Toolbar

---

### Reference

#### Hotkey T

---

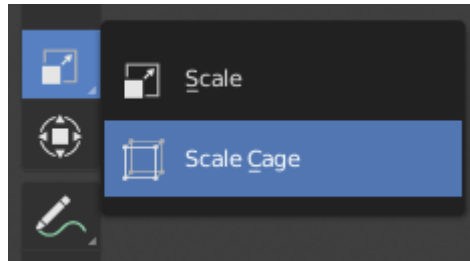


Fig. 62: Button with pop-up menu indicator.

The Toolbar shows buttons for each tool. For tool buttons which have a small triangle in their bottom right corner, a pop-up menu will be revealed when you LMB drag so that you can select other tools of the same 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

#### Hotkey Shift-Spacebar

---

You can switch tools with a toolbar that will pop up at the location of your cursor after pressing Shift-Spacebar. The shortcuts for selecting the tools are displayed on the right.

Alternatively, you can map this action to Spacebar in the Preferences. Then you'll be able use Spacebar like a modifier key (similar to pressing Ctrl or Shift).

Spacebar T for Transform, Spacebar D for Annotate, Spacebar M for measure, etc. See *Spacebar Action*.

## Quick Favorites

---

### Reference

#### Hotkey Q

---

The Quick Favorites menu gather your favorite tools. Any tool or menu can be added to this pop-up menu via the context menu of buttons and menus.



## Changing Tools

Pressing **Alt** opens a tool prompt, shown in the *Status Bar*, for changing the active tool, pressing **Alt** again closes the prompt.

Tools can be changed by pressing the appropriate icon or by pressing **Alt** then pressing the hotkey assigned to the desired tool.

## Fallback Tool

The fallback tool is the default tool in the Toolbar as in the tool at the top of the list. To switch to this tool use **Alt-W** to open a pie menu to choose what the default drag action does.

## 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 cycle through the tools in that group.

## Properties

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

- The *Sidebar* → *Tools* → *Active Tool* panel.
- The *Active Tool* tab in the Properties.
- The *Tool Settings* region.

## 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*

**Hotkey** **Ctrl-Z**

---

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

#### See also:

*Editing Preferences* section on undo to change defaults.

### Redo

---

#### Reference

**Mode** All Modes

**Menu** *Edit* → *Redo*

---

**Hotkey** 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...*

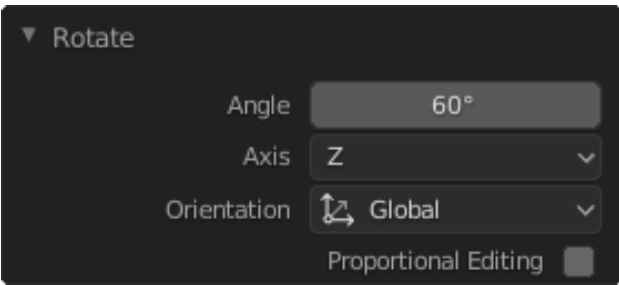
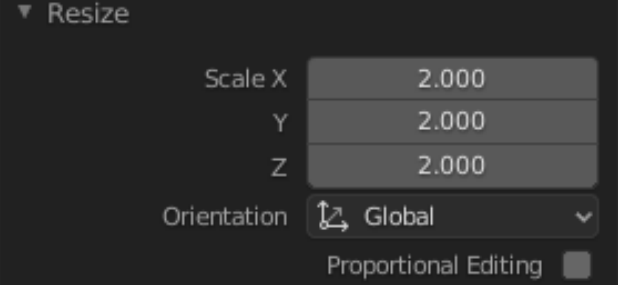
**Hotkey** F9

After an operation is complete you can tweak the parameters of the operation afterwards. In editors that support it, there is a “head-up display” panel in the bottom left based on the last performed operation; dependent on mode and context. 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. 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).

Table 4: Adjust Last Operation.

	
<p>Fig. 63: Rotation (Object Mode, 60 degrees).</p>	<p>Fig. 64: Scale (Edit Mode, Resize face).</p>

**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*

---

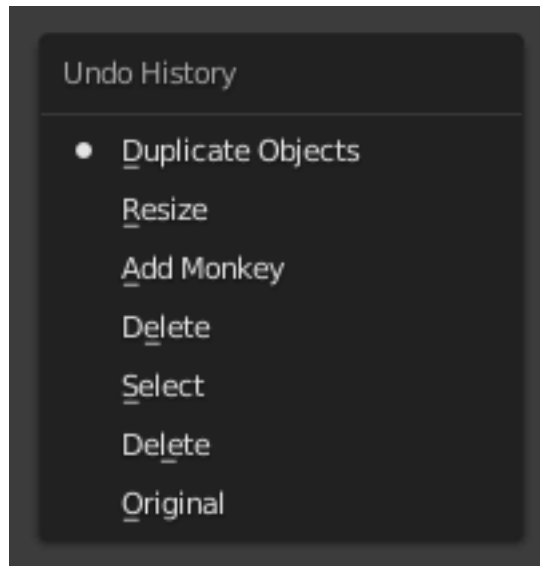


Fig. 65: The Undo History menu.

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

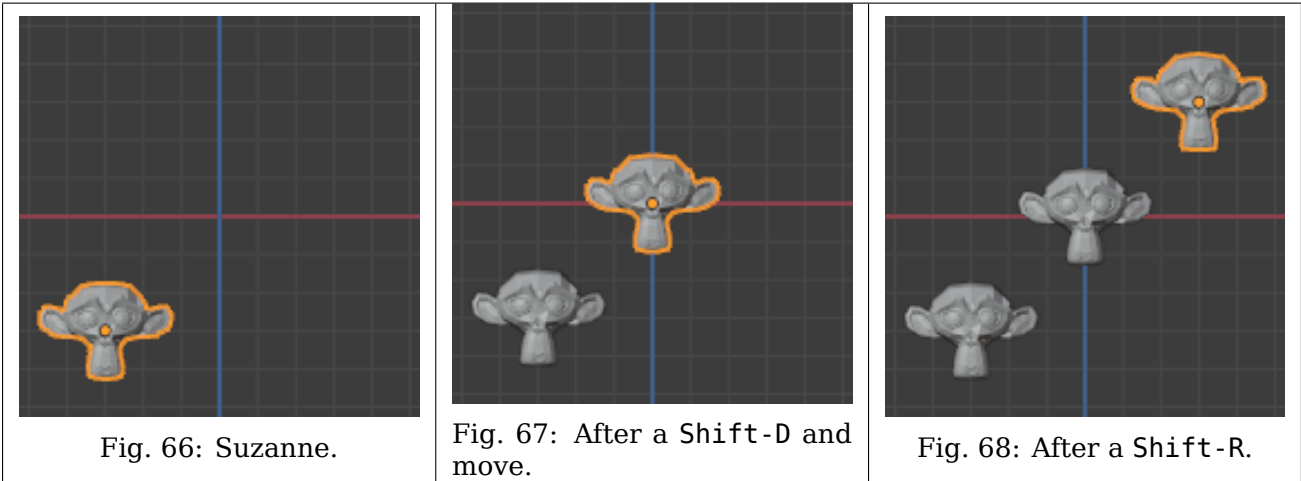
**Panel** *Edit* → *Repeat Last*

**Hotkey** 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 then we moved the object a bit. Using repeat Shift-R, the *Monkey* was also duplicated and moved.



## Repeat History

### Reference

**Mode** All Modes

**Menu** *Edit* → *Repeat History...*

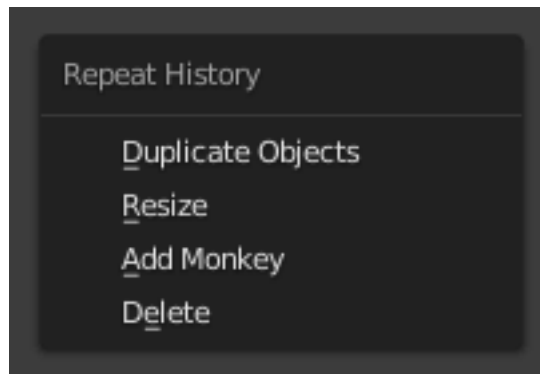


Fig. 69: The Repeat History menu.

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.

### See also:

Troubleshooting section on *Recovering your lost work*.

### Annotate Tool

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

The annotation tool can be activated in the Toolbar on the left side. It has a couple of sub-tools listed below.

**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. The current polygon is finished when Esc is pressed.

**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.

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

## Settings

### Common

There is a panel, *Sidebar* → *View* → *Annotations*, in it multiple annotation layers can be managed.

**Color** Adjusts the color 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*.

**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.

### 3D Editor

When creating new annotations in the 3D Viewport, there is one tool setting.

**Placement** The *Placement* option determines where the line is drawn in 3D space.

**3D Cursor** Draw on an imaginary plane that goes through the 3D cursor and is aligned to your view.

**View** Draw in screen space instead of in 3D space. That means, that the line will stay on the same position in the screen, even when the camera moves or rotates.

**Surface** Project the line on the surface under the mouse.

### 2D Editors

In 2D editors, the *Placement* option does not exist. When the annotation tool is enabled, the settings for managing multiple layers can be found in the *Tool* → *Active Tool* panel in the right Sidebar.

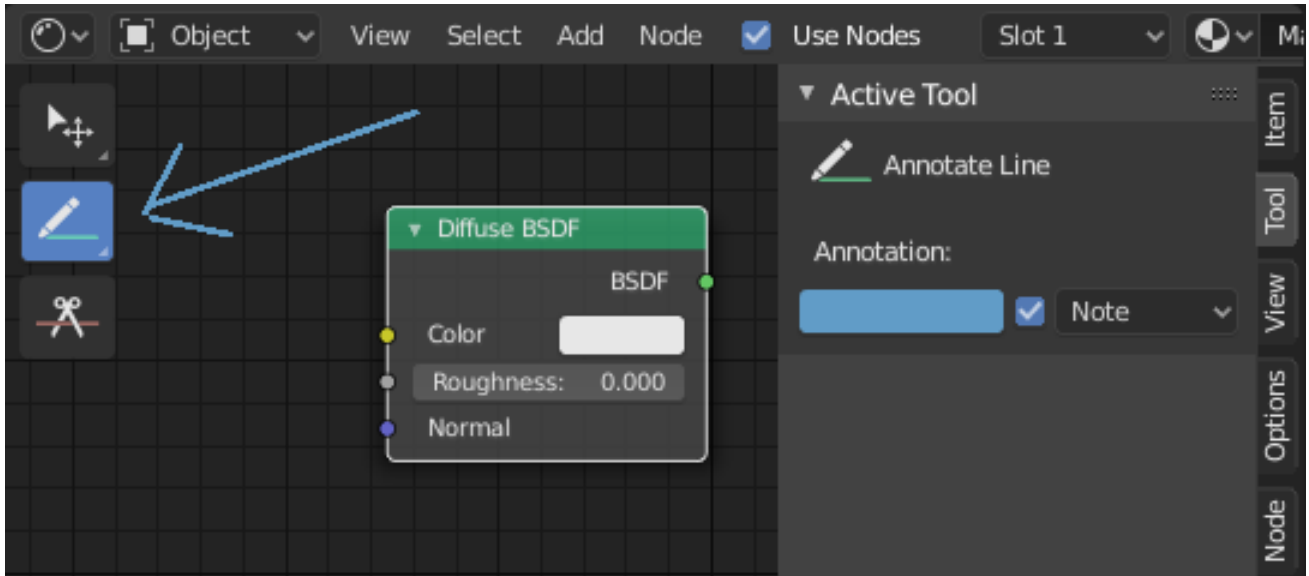


Fig. 70: Annotations tool in a node editor.

## Selecting

By default Blender uses the LMB to select items in the Blender window. Alternatively, the RMB can be used instead by changing the *Preferences*. Blender has several selecting tools that can be used across the different editors.

## Selection Tools

### Tweak

---

## Reference

**Hotkey** LMB

---

Clicking on an item selects it, using modifier keys you can perform other operations. Holding the selection and moving the mouse on interactive items such as objects in the 3D Viewport or keyframes in an animation editor will general move the item with the mouse.

## Box Select

---

## Reference

**Menu** *Select* → *Box Select*

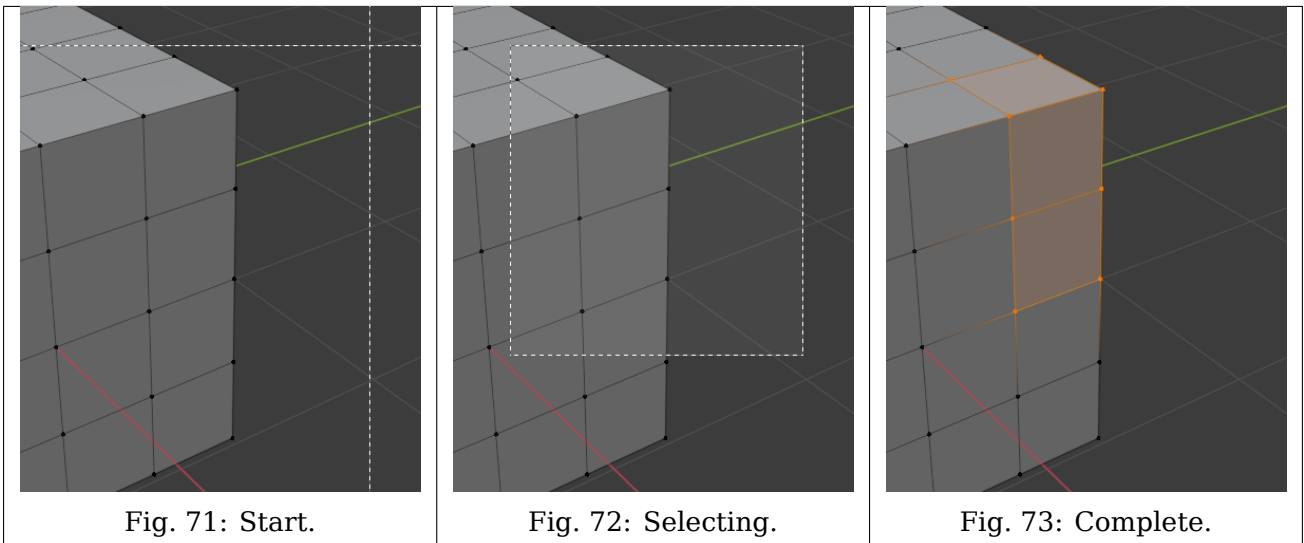
**Hotkey** B

---

To activate the tool, press B or click and drag LMB. With *Select Box* you draw a rectangle while holding down LMB. Any item that lies even partially within this rectangle becomes selected. If any item that was last active appears in the selection it will become active.

For deselecting items, use MMB, or Shift-LMB. To move the selection area hold Ctrl-Spacebar while moving the cursor.

Table 5: Box Select example.



## Circle Select

---

### Reference

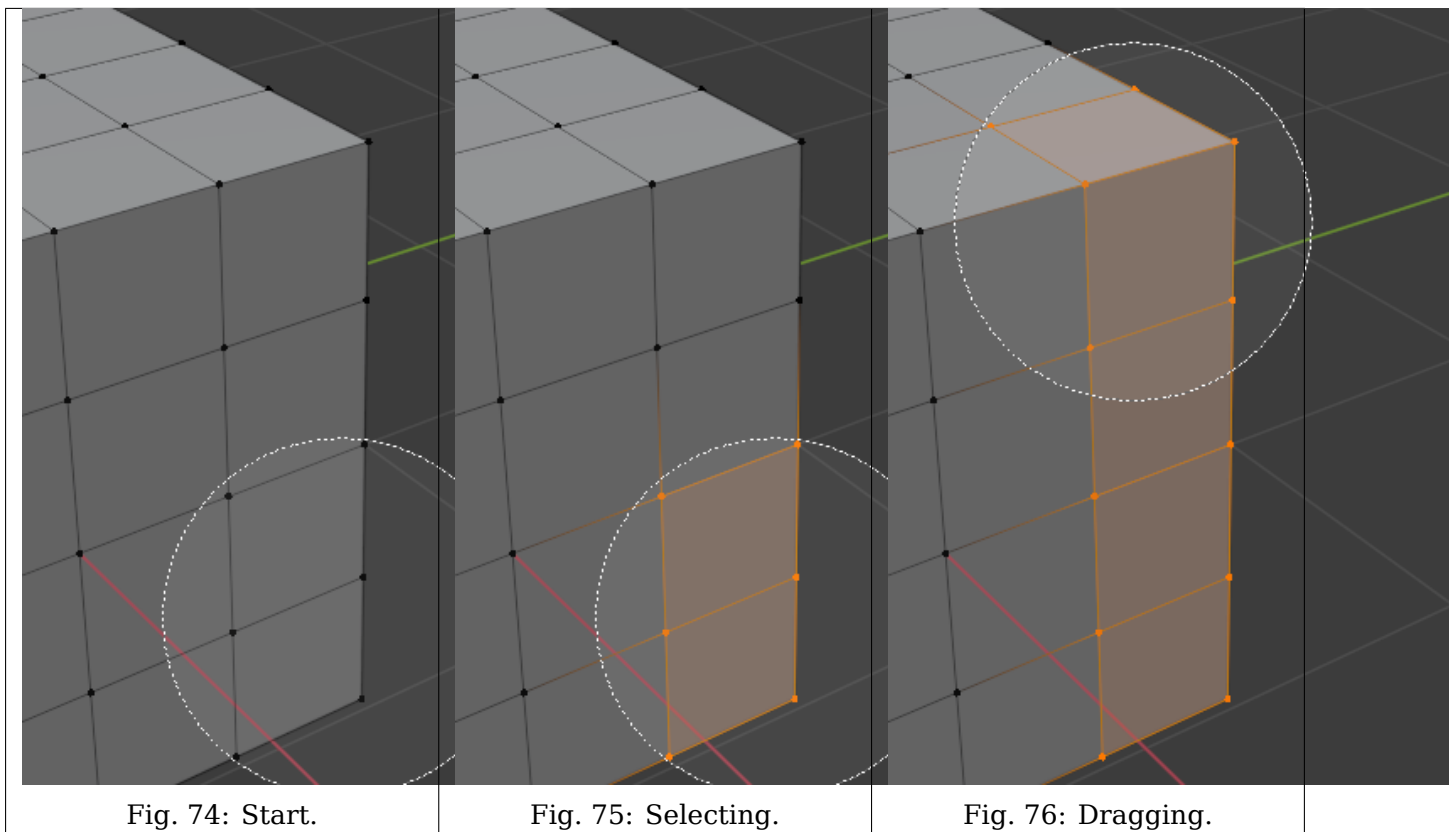
**Menu** *Select* → *Circle Select*

**Hotkey** C

---

*Circle Select* C allows you to select multiple items within a circular area. Move your mouse over any items within the circular area (shown with a dotted circle) while holding LMB to select those items. Alternatively, use MMB to deselect them. When you're done selecting, press RMB or Esc. To change the diameter of the circle, scroll with the Wheel or use the NumpadPlus and NumpadMinus keys.

Table 6: Circle Select example.



## Lasso Select

### Reference

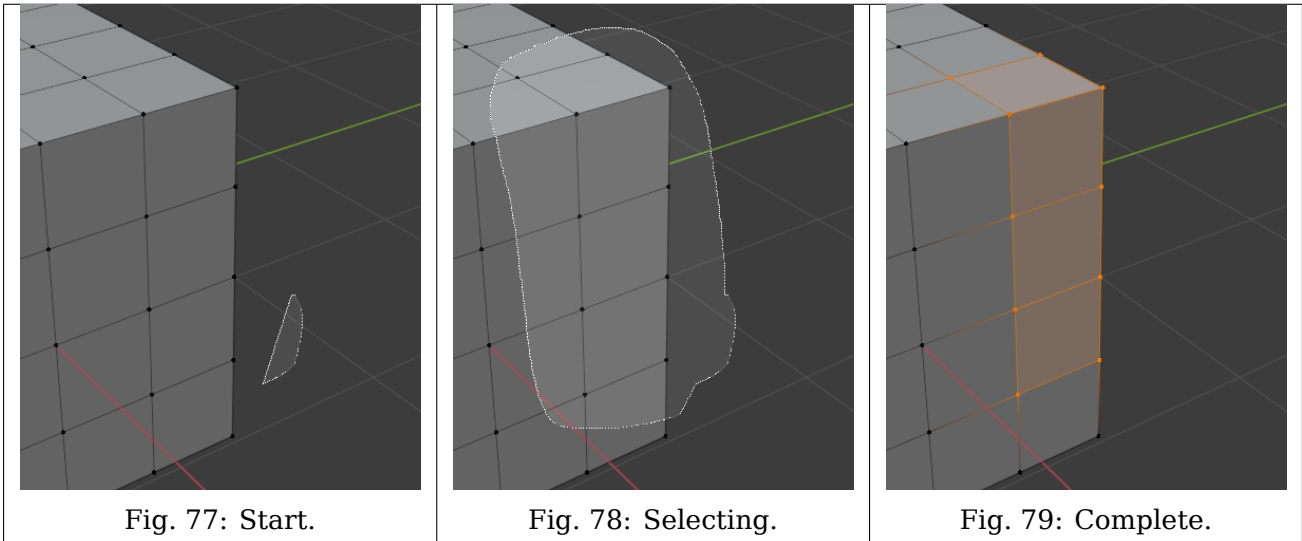
**Hotkey** Ctrl-RMB

*Lasso Select* is used to create a free-form selection. Simply hold Ctrl-RMB while drawing a dotted line around the items you want to select. The shape you draw will be automatically closed by connecting a line from the current position back to the starting point.

*Lasso Select* adds to the previous selection. For deselection, use Shift-Ctrl-RMB. To move the selection area hold Ctrl-Spacebar while moving the cursor.



Table 7: An example of using the *Lasso Select* tool in *Vertex Select Mode*.



## Selecting Modes

### Reference

**Tool** Select Tools

**Panel** *Tool Settings* → *Mode*

Each tool has some sort of mode to configure how the tool interacts with existing selections. Note that not every selection tool supports each of these modes.

**Set** Sets a new selection ignoring any existing selections.

**Extend** Adds newly selected items to the existing selection. The selection can also be extended by Shift-LMB.

**Subtract** Removes newly selected items from the existing selection. Items can be removed from the selection by Shift-LMB already selected items.

**Invert** Selects non-selected items and deselects existing selection. The selection can also be inverted by Ctrl-I.

**Intersect** Selects items that intersect with existing selection.

## 2.3 Editors

Blender provides a number of different editors for displaying and modifying different aspects of data.

The *Editor Type* selector, the first button at the left side of a header, allows you to change the editor in that *area*. Every area in Blender may contain any type of editor and it is also possible to open the same type multiple times.

See *User Interface* for documentation on the general interface.

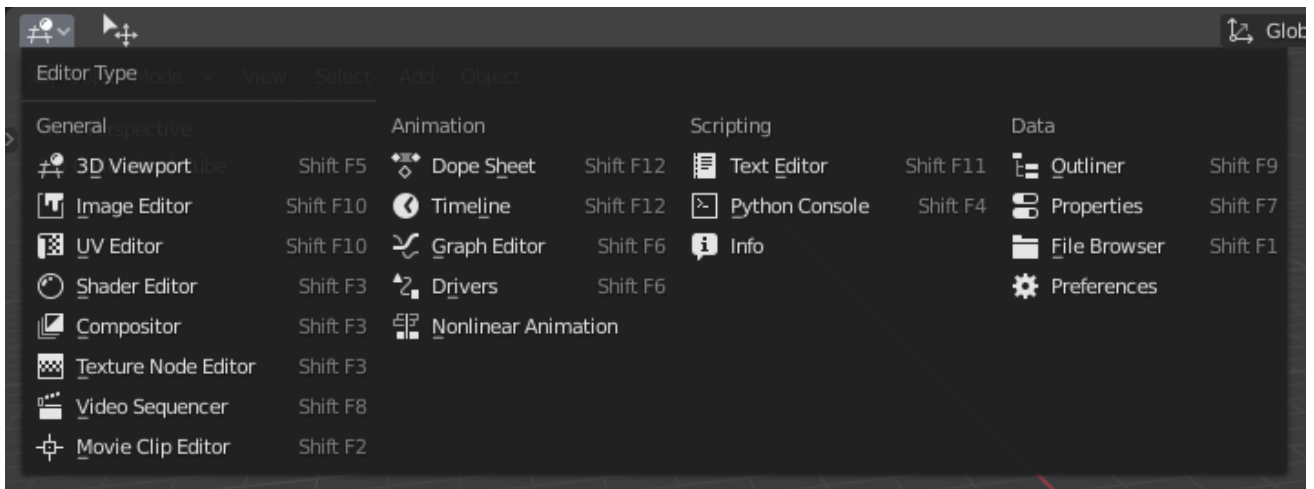


Fig. 80: The Editor Type selector.

## General

### 2.3.1 3D Viewport

#### 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

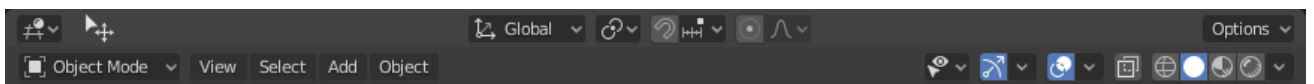


Fig. 81: Object Mode header.

The header contains various menus and controls based on the current *mode*.

Tools and modes in the 3D Viewport header are split in three groups of buttons:

#### Mode & Menus

**Mode** The 3D Viewport has *several modes* used for editing different kinds of data.

**View** This menu offers tools to *navigate* in 3D space.

Other menus depend on the current mode, Object Mode menus listed below:

**Select** Contains tools for *selecting* objects.

**Add** Gives a list of different *objects types* that can be added to a scene.

**Object** This menu appears when in Object Mode. it contains tools to edit *objects*. In Edit Mode, it will change to the appropriate menu with *editing tools*.

#### Transform Controls

**Transform Orientations** Use to select and modify the active *Transform Orientations*.

**Pivot Point** Used to change the reference point (or *Pivot Point*) used by many mesh manipulation tools.

Read more about *Pivot Points*.

**Snapping** Controls the *snapping tools* that help with transforming and modeling objects.

**Proportional Edit** *Proportional Edit*.

## Display & Shading

**Object Type Visibility** Change the *Object Type Visibility* and selectability of objects in the 3D Viewport.

**Viewport Gizmos** Change the way how *gizmos* are displayed in the 3D Viewport.

**Viewport Overlays** Change the way how *overlays* are displayed in the 3D Viewport.

**X-Ray** Show the whole scene transparent. This is a shortcut to the *X-ray* option inside the shading control.

**Viewport Shading** Change the *shading* of the 3D Viewport.

## Toolbar Region

The Toolbar is a context-sensitive region containing 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 selected objects (such as their locations), as well as properties of the editor itself.

See *Sidebar Panels* for more information.

## Startup Scene

After closing the splash, the startup scene is displayed in the 3D View if no other blend-file was loaded. A customized startup scene can be saved as a part of the *startup file*.

## Elements

**Cube** The gray cube in the center of the scene is a *mesh* object. Because the cube is selected it is displayed with an orange outline.

**Object Origin** The *Origin of the object* is displayed as an orange dot and it marks the cube's (relative) position.

**Light** The circles with a thin line to the bottom is a light source illuminating the cube. Lights in: *General Settings*.

**Camera** The pyramid with a big triangle pointing upward is the camera used as point of view for rendering. See also: cameras in *Cycles*.

**3D Cursor** The *3D cursor*, a cross with a red-and-white circle, is used for placing objects in the scene.

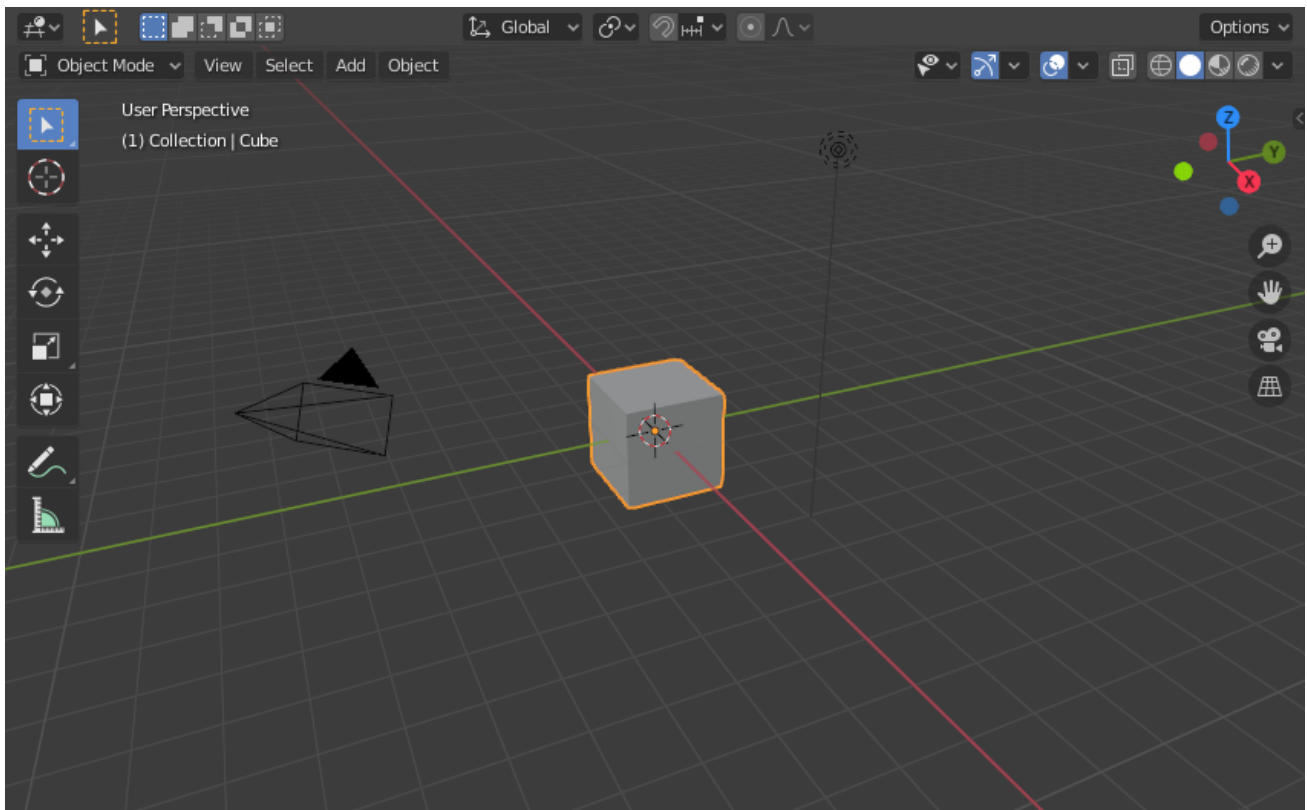


Fig. 82: The Startup scene.

**Grid Floor** The gray squares forming a floor mark the zero height of the world. The red and green lines are the axis of the world coordinate system. They meet at the origin, which is also the position of the *Cube*. The Grid Floor settings are in the *Viewport Overlays* popover.

### Text Info

The visibility and settings of the overlays can be set in the *Viewport Overlays* popover.

**View Name** If the viewport camera is not aligned, the view is named “User” plus the perspective of the viewport camera.

**Playback FPS** Displays the Frames Per Second screen rate, while playing an animation back.

**Object Info** Shown in brackets is the current frame. Followed by the path of the *active object*. And optionally the selected *shape key* and in brackets (<>) the *Markers* name on the current frame. The color of the Object Info is set by the *State Colors* (keyframe only).

## Object Modes

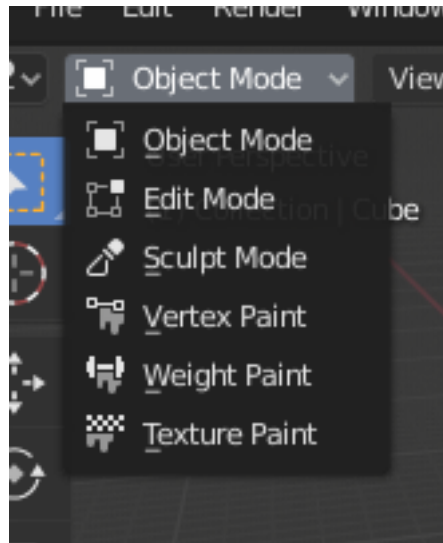


Fig. 83: The Mode select menu.

*Modes* are an object-oriented feature, which means that the available modes vary depending on the selected active object's type - most of them only enable the default *Object Mode* (like cameras, lights, etc.).

Each mode is designed to edit an aspect of the selected object. See Tab. *Blender's Modes* below for details.










You set the current mode in the *Mode* selector of 3D Viewport header (see Fig. *The Mode select menu*).

Modes can affect many things in Blender:

- They can modify the panels and/or controls available in some Properties tabs.
- They can modify the behavior of the whole editor, like e.g. the UV Editor and 3D Viewport.
- They can modify the available header tools (menus and/or menu entries, as well as other controls...). For example, in the 3D Viewport, the *Object* menu in Object Mode changes to a *Mesh* menu in Edit Mode (with an active mesh object!), and a *Paint* menu in Vertex Paint Mode...
- They can modify the available shortcuts.

## Object Mode List

Table 8: Blender’s Modes

Icon	Name	Details
	<i>Object Mode</i>	The default mode, available for all object types, as it is dedicated to <i>Object</i> data-block editing (e.g. position, rotation, size).
	<i>Edit Mode</i>	A mode available for all renderable object types, as it is dedicated to their “shape” <i>Object Data</i> data-block editing (e.g. vertices/edges/faces for meshes, control points for curves/surfaces, strokes/points for Grease Pencil, etc.).
	<i>Sculpt Mode</i>	A mesh-only mode, that enables Blender’s mesh 3D-sculpting tool.
	<i>Vertex Paint Mode</i>	A mesh-only mode, that allows you to set your mesh’s vertices colors (i.e. to “paint” them).
	<i>Weight Paint Mode</i>	A mesh-only mode, dedicated to vertex group weighting.
	<i>Texture Paint Mode</i>	A mesh-only mode, that allows you to paint your mesh’s texture directly on the model, in the 3D Views.
	<i>Particle Edit Mode</i>	A mesh-only mode, dedicated to particle systems, useful with editable systems (hair).
	<i>Pose Mode</i>	An armature only mode, dedicated to armature posing.
	<i>Draw Mode</i>	A Grease Pencil only mode, dedicated to create Grease Pencil strokes.

**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.

## Multi-Object Editing

Edit and Pose Modes support editing of multiple objects at once.

This is convenient if you want to perform the same edits on multiple objects or want to animate multiple characters at once.

- To use edit multiple objects at once, simply select multiple objects and enter the mode.
- The Outliner can also be used to add/remove objects while you are in a mode, by setting or clearing the mode from the context menu, or `Ctrl-LMB` clicking on the objects data icon.
- Only the active object will be used to display properties such as shape keys, UV layers, etc.
- Selecting any element from an object will set this as the active object.
- There are limits to the kinds of operations that can run on multiple objects.

*You can't for example create an edge that has vertices from different objects.*

## Navigating

### Introduction

Navigating in the 3D space is done with the use of both mouse movement and keyboard shortcuts.

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 View* editor, most of the other editors have similar functions. For example, it is possible to pan and zoom in the Image editor.

---

**Tip:** Mouse Buttons and Numpad

If you have a mouse with less than three buttons or a keyboard without a numpad, see the *Keyboard and Mouse* page of the manual to learn how to use them with Blender.

---

## Navigation Gizmo

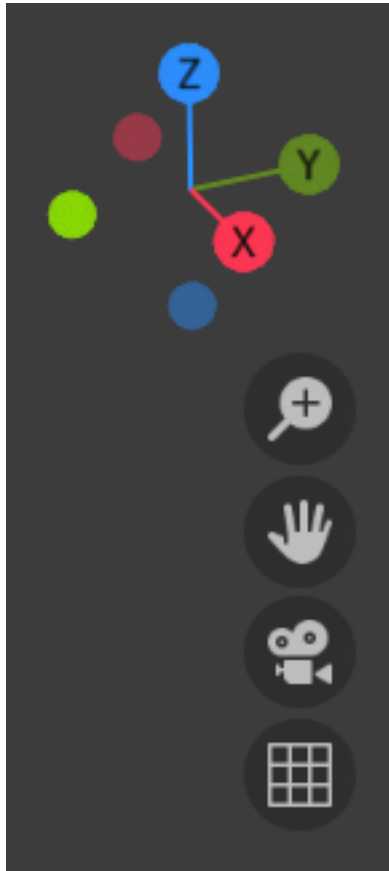


Fig. 84: Navigation Gizmo.

The navigation gizmo can be found in the top right of the editor. The four buttons (listed from top to bottom) do the following:

- *Zooms the 3D Viewport*
- *Pans the 3D Viewport*
- *Toggles the Camera View*
- *Toggles the Projection*

The *Orbit* gizmo at the top can be used to rotate around the 3D Viewport. Hovering over the gizmo and dragging with LMB will orbit the view. Clicking any of the axis labels will *Align* to that view. Clicking the same axis again switches to the opposite side of that same axis.

## Navigation

### Orbit

---

### Reference

**Mode** All modes

**Menu** *View* → *Navigation* → *Orbit*

**Hotkey** MMB, Numpad2, Numpad4, Numpad6, Numpad8, Ctrl-Alt-Wheel, Shift-Alt-Wheel



Rotate the view around the point of interest. Click and drag MMB on the viewport's area. If you start in the middle of the area and move up and down or left and right, the view is rotated around the middle of the area.

To change the viewing angle in discrete steps, use Numpad8 and Numpad2 or use Numpad4 and Numpad6 to rotate the scene around the global Z axis from your current point of view. Finally Numpad9 switches to the opposite side of the view.

Or if the *Emulate 3 button mouse* option is select in the Preferences you can press and hold Alt while dragging LMB in the viewport's area.

---

**Note:** Hotkeys

Remember that most hotkeys affect the **active** area (the one that has focus), so check that the mouse cursor is in the area you want to work in before you use the hotkeys.

---

**See also:**

- *Orbit Style Preference*
- *Auto-Perspective Preference*

## Roll

---

### Reference

**Mode** All modes

**Menu** *View → Navigation → Roll*

**Hotkey** Shift-Numpad4, Shift-Numpad6

---

Rotate the viewport camera around its local Z axis in 15° discrete steps.

## Pan

---

### Reference

**Mode** All modes

**Menu** *View → Navigation → Pan*

**Hotkey** 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 (note: you can replace Ctrl with Shift).

If your input device has no middle button, you can hold Shift-Alt while dragging with LMB.

## Zoom In/Out

---

### Reference

**Mode** All modes

**Menu** *View → Navigation → Zoom In/Out*

---

**Hotkey** Ctrl-MMB, Wheel, NumpadPlus, NumpadMinus

---

Moves the camera forwards and backwards. You can zoom in and out by holding down Ctrl and dragging MMB. To zoom in with discrete steps, use the hotkeys NumpadPlus and NumpadMinus. If you have a wheel mouse, you can zoom by rotating the Wheel.

---

**Hint:** If You Get Lost

If you get lost in 3D space, which is not uncommon, two hotkeys will help you: Home changes the view so that you can see all objects *View → Frame All*, while NumpadPeriod zooms the view to the currently selected objects when in perspective mode *View → Frame Selected*.

---

## Zoom Region

---

### Reference

**Mode** All modes

**Menu** *View → Navigation → Zoom Region...*

**Hotkey** Shift-B

---

The *Zoom Region* tool allows you to specify a rectangular region and zoom in so that the region fills the 3D Viewport.

You can access this through via the shortcut Shift-B, then LMB click and drag a rectangle to zoom into.

Alternatively you can zoom out using the MMB.

## Dolly Zoom

---

### Reference

**Mode** All modes

**Hotkey** Shift-Ctrl-MMB

---

In most cases its sufficient to zoom the view to get a closer look at something, however, you may notice that at a certain point you cannot zoom any closer.

This is because Blender stores a view-point that is used for orbiting and zooming. It works well in many cases, but sometimes you want to move the view-point to a different place. This is what Dolly supports, allowing you to transport the view from one place to another.

You can dolly back and forth by holding down Shift-Ctrl and dragging with MMB.

## Fly/Walk Navigation

There are cases where it's preferable to navigate with first person controls, especially for large environments such as architectural models. In these cases orbiting around the view center is limiting. While zoom, pan and dolly can be used, it's inconvenient.

With walk/fly navigation you can navigate around the scene where view rotation is performed from the cameras location.

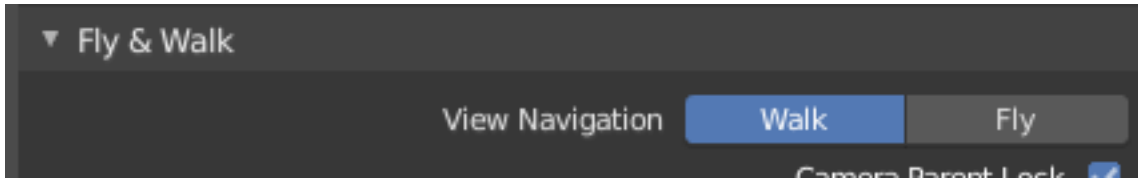


Fig. 85: View Navigation.

In the *Preferences editor* select the navigation method you want to use as default when invoking the View Navigation operator. Alternatively you can call the individual methods from the *View → Navigation* menu.

Common use cases for walk/fly include:

**Navigation** This can be a quick way to navigate a large scene.

**Camera Placement** When activated from a camera view, this will move the camera too.

**Recording Animation** Running from a camera with auto-keyframe and playing animation will record the motion as you make it allowing you to record the walk-through.

## Walk Navigation

---

### Reference

**Mode** All modes

**Hotkey** Shift-AccentGrave

**Menu** *View → Navigation → Walk Navigation*

---

On activation the mouse pointer will move at the center of the view, and a cross marker will appear...

This navigation method behaves similar to the first person navigation system available in most 3D world games. It works with a combination of keyboard arrow keys and mouse movement.

### Shortcuts

- Move the mouse in the direction you want to look.
- Arrow keys or W, A, S, D move forwards/backwards and strafe left/right.
- Teleport Spacebar.

This moves you to the location at the cross-hair (offset by the *Camera Height* value set in the *Preferences*).

- Jump V - only available if *Gravity* is on.
- Move up and down Q, E - only available if *Gravity* off.
- Toggle *Gravity* Tab.
- Change the movement speed:
  - WheelUp or NumpadPlus to increase the movement speed for this open session.
  - WheelDown or NumpadMinus to decrease the movement speed for this open session.
  - Shift (hold) - to speed up the movement temporarily.
  - Alt (hold) - to slow down the movement temporarily.

When you are happy with the new view, click LMB to confirm. In case you want to go back from where you started, press Esc or RMB, as usual.

If the defaults values (speed, mouse sensitivity, ...) need adjustments for your project, in the *Preferences* you can select a few options for the navigation system:

## Fly Navigation

---

### Reference

**Mode** All modes

**Hotkey** Shift-AccentGrave

**Menu** *View* → *Navigation* → *Fly Navigation*

---

On activation the cursor is centered inside a rectangle that defines a safe region. When the cursor is outside this region the view will rotate/pan.

### Shortcuts

- Move the mouse outside the safe region in the direction you want to look.
- Move the view forward/backward:
  - WheelUp or NumpadPlus to speed up the movement forward.
  - WheelDown or NumpadMinus to speed up the movement backward.

So if the view is already moving forward, WheelDown, NumpadMinus will eventually stop it and then move it backward, etc.

Arrow keys or W, A, S, D can also be used to adjust the acceleration and direction of the camera movement.
- MMB drag to pan the view.
 

In this case the view can move laterally on its local axis at the moment you drag the mouse.
- Shift precision (slow the momentum).
- 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.

Click LMB or press Spacebar to keep the current view and exit fly navigation. In case you want to go back from where you started, press Esc or RMB.

## Aligning

---

### Reference

**Menu** *View* → *Align View*

---

These options allow you to align and orient the view.

**Align View to Active** The options in this menu align your view with specified local axes of the selected active object, bone, or, in *Edit Mode* with the normal of the selected face.

Hold down Shift while using the numpad to set the view axis.

**Align Active Camera to View Ctrl-Alt-Numpad0** Moves and rotates the active camera to the current viewpoint.

**Align Active Camera to Selected** Points the active camera toward the selected object; based on the direction of the current viewpoint.

**Center Cursor and Frame All Shift-C** Moves the cursor back to the origin and zooms in/out so that you can see everything in your scene.

**Center View to Cursor** Centers view to 3D cursor.

**View Lock to Active** Centers view to the last selected active object, overriding other view alignment settings.

**View Lock Clear** Returns the view alignment to the view align settings before use of *View Lock to Active*.

## Perspective/Orthographic

### Reference

**Mode** All modes

**Menu** *View* → *Perspective/Orthographic*

**Hotkey** 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.

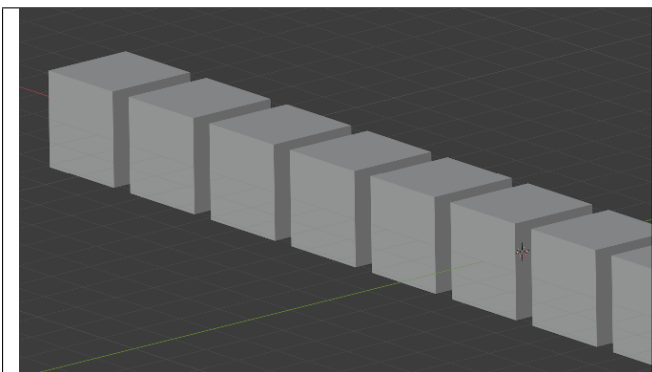


Fig. 86: Orthographic projection.

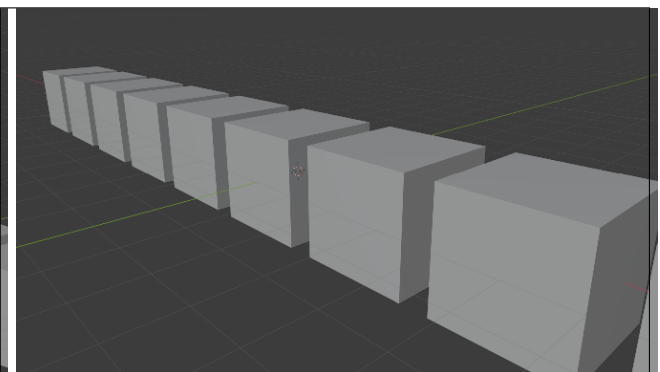


Fig. 87: Perspective projection.

Our eye is used to perspective viewing because 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 is very useful, because it provides a more “technical” insight into the scene, making it easier to model and judge proportions.

### Options

To change 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 and press the *Orthographic* button in the *Lens* panel.

**See also:**

- *Render perspective*
- *Camera Projections*

## Local View

### Toggle Local View

---

#### Reference

**Mode** All modes

**Menu** *View* → *Local View* → *Toggle Local View*

**Hotkey** NumpadSlash, Slash

---

Global view shows all 3D objects in the scene. Local view isolates the selected object or objects, so that they are the only ones visible in the viewport. This is useful for working on objects that are obscured by other ones, 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.

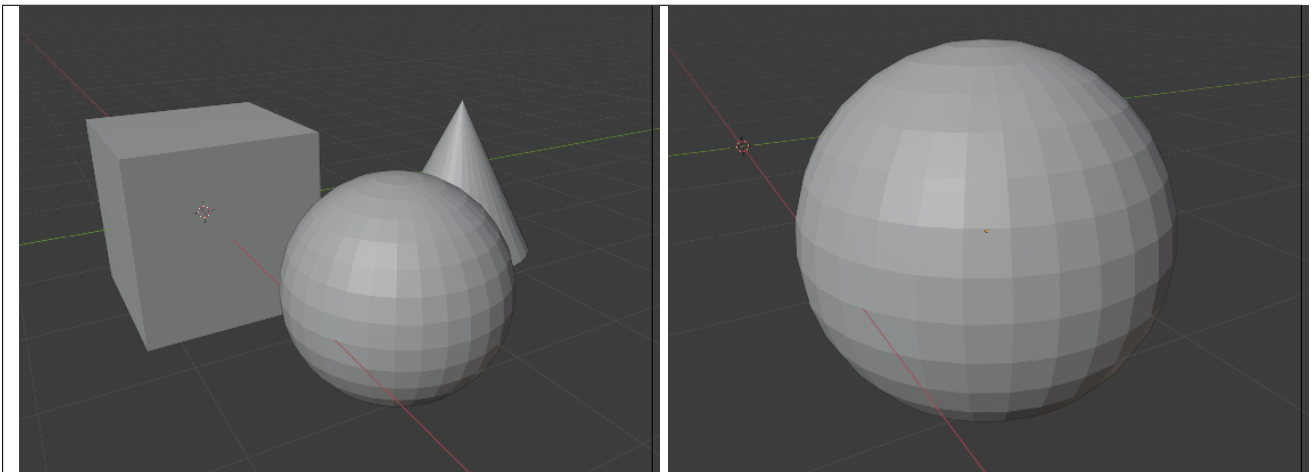


Fig. 88: Global View.

Fig. 89: Local View.

---

**Note:** These notes cover changes in local view which are not immediately obvious.

**3D Cursor** In local view the 3D cursor is not locked to the scene. Instead, each view has an independent cursor location.

**Layers** Local view bypasses layers, using only the selected objects when entering local view. Although new objects can be added while in local view.

---



---

**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 turning off local view.

---

---

## Remove from Local View

### Reference

**Mode** All modes

**Menu** *View* → *Local View* → *Remove from Local View*

**Hotkey** M

---

Objects can be removed from Local View by selecting them and using the *Remove from Local View* operator. This will move the selected object back to global view and all other objects will remain in local view. If the last remaining object is removed, the local view will be left empty and you will have to exit local view to see any objects.

## Camera View

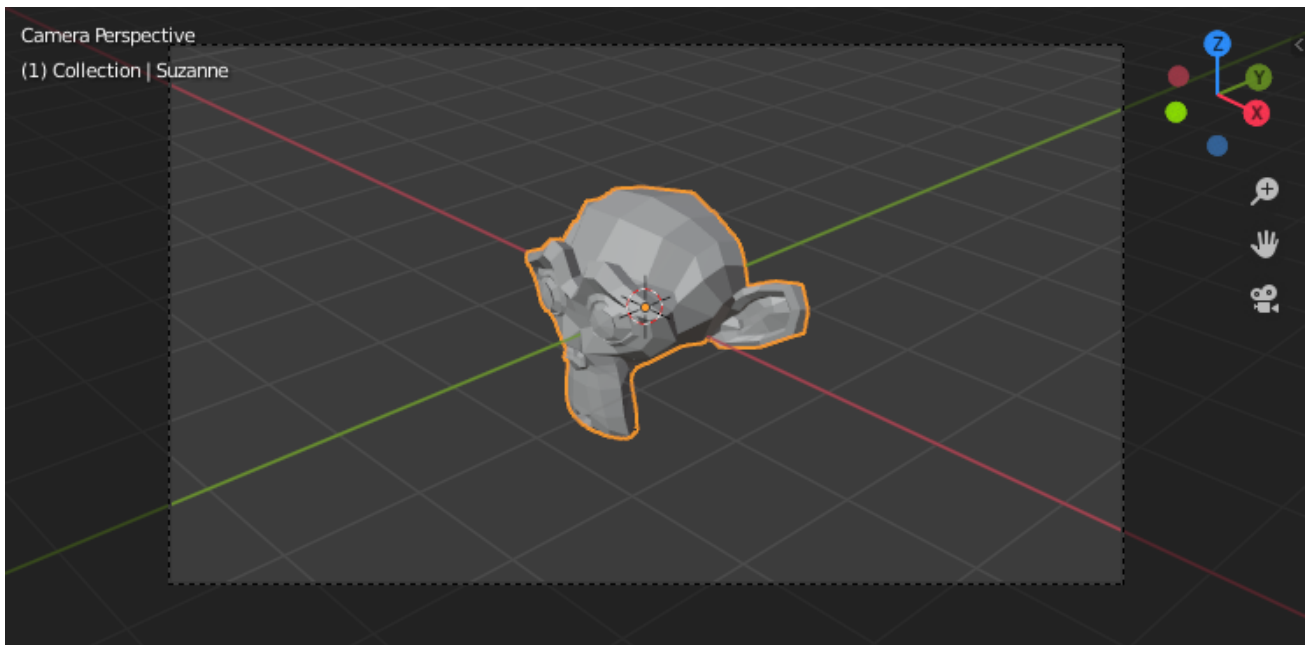


Fig. 90: Demonstration of camera view.

The Camera view shows the current scene as seen from the currently active camera's view point. 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 line.

**See also:**

*Camera Settings* for details how camera settings are used for display & rendering.

---

**Hint:** The active camera can be selected while in camera view using the camera frame (*assuming the object isn't hidden*).

---

## Viewing the Active Camera

### Reference

**Mode** All modes

**Menu** *View* → *Cameras* → *Active Camera*

**Hotkey** Numpad0

---

This switches the view to the active camera. The triangle above the camera will become shaded when active.

### Setting the Active Camera

---

#### Reference

**Mode** Object Mode

**Menu** *View* → *Cameras* → *Set Active Object as Camera*

**Hotkey** Ctrl-Numpad0

---

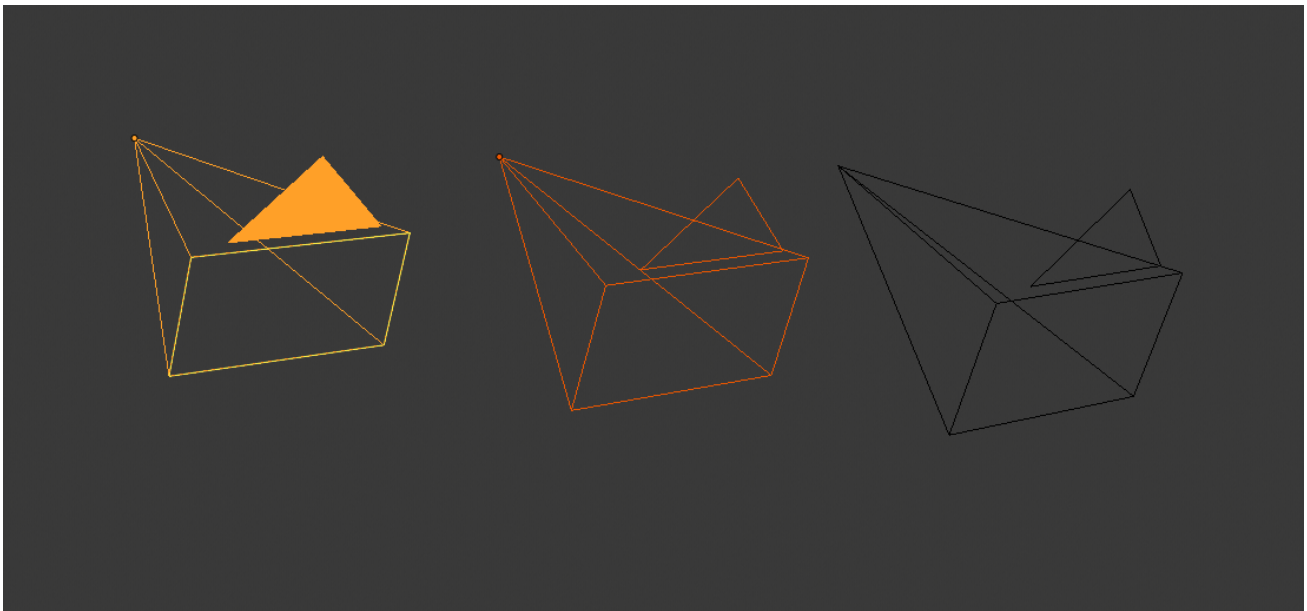


Fig. 91: Active camera (left) displayed with a solid triangle above it. This is the camera currently used for rendering and when viewing from the camera.

This sets the current active object as the active camera & switches to the camera view.

The active camera can also be set in the *Scene* tab of the *Properties*.

---

**Note:** The active camera, as well as the layers, can be specific to a given view, or global (locked) to the whole scene. See *Local Camera*.

---

### Animated Camera Switching

By default a scene contains one camera. However, a scene can contain more than one camera, but only one of them will be used at a time. So you will only need to add a new camera if you are making cuts between them. See *Animating Cameras*.



---

## Frame Camera Bounds

---

### Reference

**Mode** All Modes

**Menu** *View* → *Cameras* → *Frame Camera Bounds*

**Hotkey** Home

---

Centers the camera view inside the 3D Viewport's screen area and resizes the view to fit within the area's bounds.

### Camera Navigation

There are several different ways to navigate and position the camera in your scene, some of them are explained below.

Zooming in and out is possible in this view, but to change the viewpoint, you have to move or rotate the camera.

---

**Hint:** The active "camera" might be any kind of object. So these actions can be used, for example, to position and aim a light.

---

### Move Active Camera to View

---

### Reference

**Mode** Object Mode

**Hotkey** Ctrl-Alt-Numpad0

---

This matches the active camera to a regular (non camera) view, for a convenient method of placing the camera without having to move the object directly.

### Camera View Positioning

By enabling *Lock Camera to View* in the View panel of the Sidebar region, while in camera view, you can navigate the 3D Viewport as usual, while remaining in camera view. Controls are exactly the same as when normally moving in 3D.

**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 in 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, pressing the same letter a second time selects the *local* axis - this works with any axis; see the *axis locking page*).

**Horizontal Pan or Yaw** This corresponds to a rotation around the camera's local Y axis. Press R, and 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 twice or Y to get pure-horizontal or pure-vertical sideways tracking).

## Viewpoint

Blender uses a right-angled "Cartesian" coordinate system with the Z axis pointing upwards.

**X axis** Left / Right

**Y axis** Front / Back

**Z axis** Top / Bottom

You can select the viewing direction for a 3D Viewport with the *View* menu entries, *Navigation Gizmo*, or by pressing the hotkeys.

These operators change the view to be aligned with the specified global axes:

**Top** Numpad7

**Front** Numpad1

**Right** Numpad3

Holding Ctrl shows the other side of the same axis:

**Bottom** Ctrl-Numpad7

**Back** Ctrl-Numpad1

**Left** Ctrl-Numpad3

Holding Shift aligns the view relative to the active selection. So you can for example, view a rotated objects side, or align the view to the active face in mesh Edit Mode.

The view can also be aligned by holding Alt-MMB and moving the mouse towards the view to align to.

## View Regions

### Clipping Region

---

#### Reference

**Mode** All modes

**Menu** *View* → *View Regions* → *Clipping Region...*

**Hotkey** 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, in the wanted 3D Viewport. 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 have 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

Table 9: Region/Volume clipping.

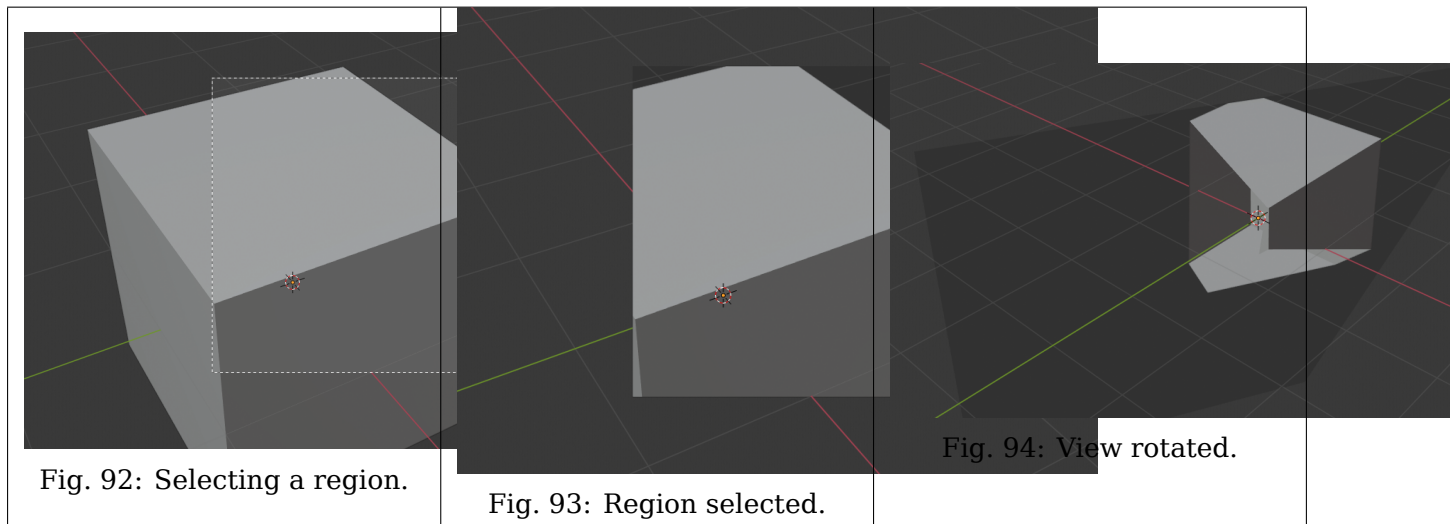


Fig. 92: Selecting a region.

Fig. 93: Region selected.

Fig. 94: 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** (upper left of the image). This will generate a dashed cross-hair cursor. Click with the LMB and drag out a rectangular region shown in the upper right. Now a region is defined and clipping is applied against that region in 3D space. Notice that part of the cube is now invisible or clipped. Use the MMB to rotate the view and you will see that only what is inside the pyramidal volume is visible. All the editing tools still function as normal but only within the pyramidal 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

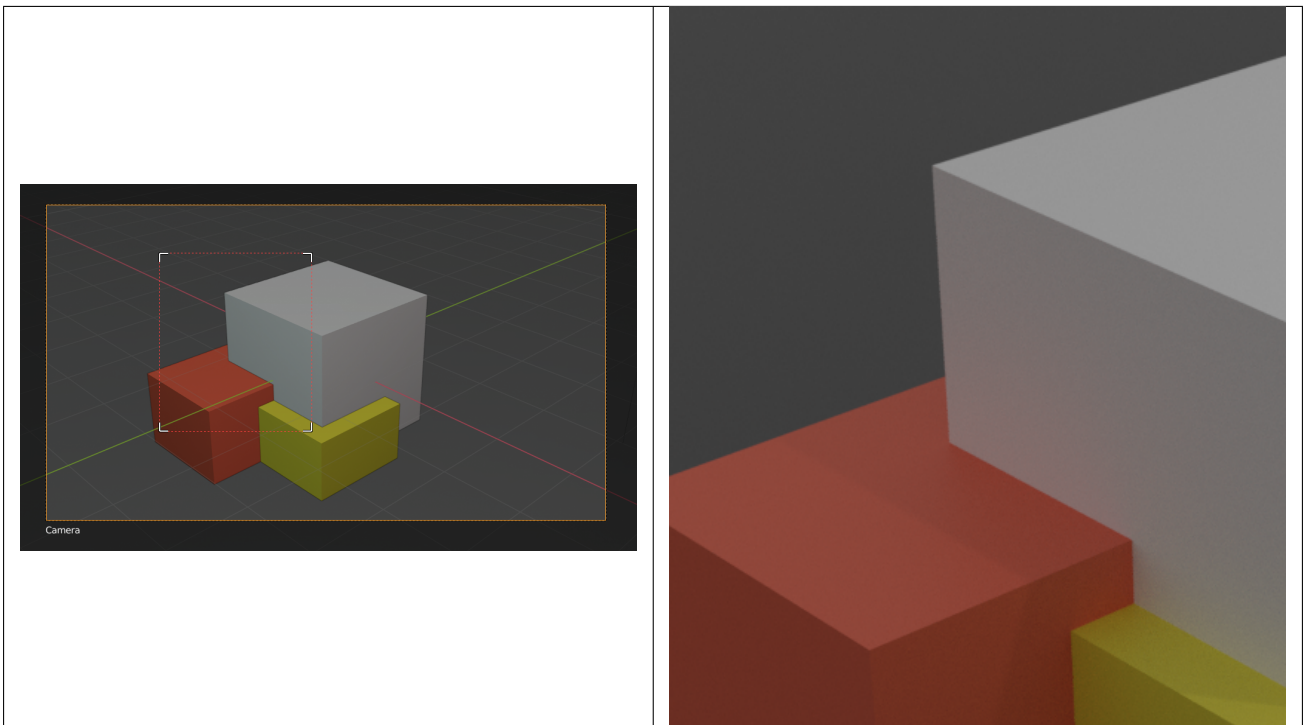
**Mode** All modes

**Menu** *View* → *View Regions* → *Render Region...* *View* → *View Regions* → *Clear Render Region*

**Hotkey** Mark: **Ctrl-B** Clear: **Ctrl-Alt-B**

When using *rendered shading* mode, it can be quite slow to render the entire 3D Viewport. To fix this, you can define a subregion to render just a portion of the viewport instead of the entire viewport. This can be very useful for reducing render times for quick previews on an area of interest.

Table 10: Render region and associated render.



**Tip:** You can also use this region in a final render by setting a render region from within the *Camera View* and enabling *region* in the Dimensions panel.

#### See also:

*Zoom Region.*

### Contextual Views

The 3D Viewport has several “contextual view” modes that can be set for a particular 3D Viewport. These views can change how the overall 3D Viewport looks or how you interact with objects.

#### Quad View

#### Reference

**Mode** All modes

**Menu** *View* → *Area* → *Toggle Quad View*

**Hotkey** Ctrl-Alt-Q

Toggling Quad View will split the 3D Viewport into four views: Three *Orthographic* “side views” and one *Camera/User View*. This view will allow you to instantly see your model from a number of view points. In this arrangement, you can zoom and pan each view independently but you cannot rotate the view.

**Note:** Quad View is different from *splitting the area* and aligning the view manually. In Quad View, the four views are still part of a single 3D Viewport. So they share the same display options

and layers.

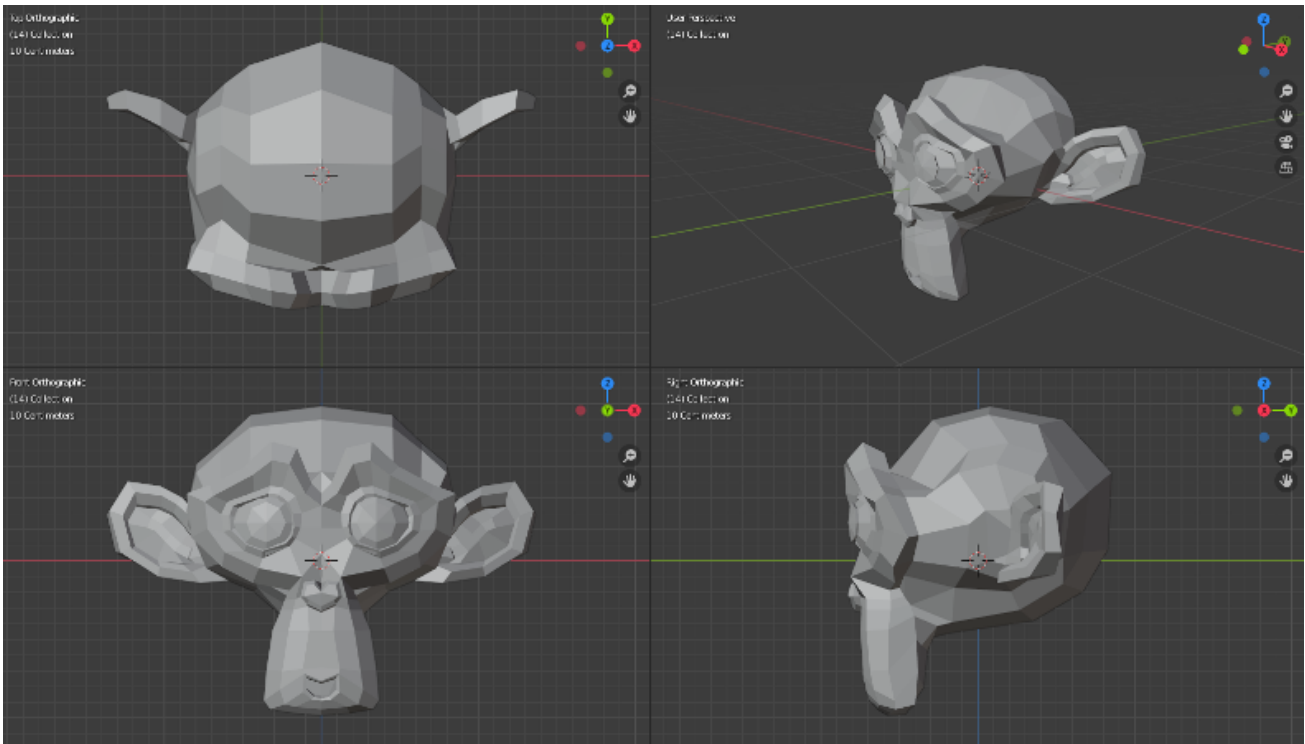


Fig. 95: Quad View.

## Options

### Reference

**Mode** All modes

**Menu** *Sidebar* → *View* → *Quad View*

**Lock** If you want to be able to rotate each view, you can uncheck the *Locked* option.

**Box** Syncs the view position between side views. (Requires *Lock* to be enabled.)

**Clip** Clip objects based on what is visible in other side views. (Requires *Box* to be enabled.)

### 3D Cursor

The 3D Cursor is a point in 3D space which can be used for a number of purposes.

### Placement

There are a few methods to position the 3D cursor.

#### Direct Placement with the Mouse

With the cursor tool enabled, using LMB in the 3D Viewport will place the 3D cursor directly under your mouse pointer.

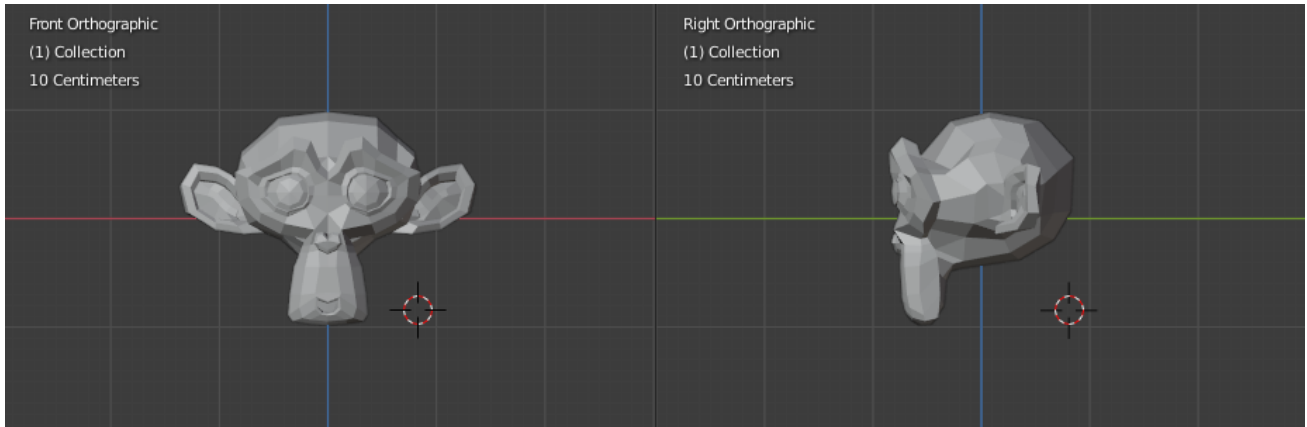


Fig. 96: Positioning the 3D cursor with two orthogonal views.

The view space is used to control the rotation of the 3D Cursor.

For accuracy you should use two perpendicular orthogonal 3D Views, 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 depth in the second view.

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*.

**See also:**

The *Snap Menu* which allows the cursor placement relative to scene objects.

### 3D Cursor Panel

---

#### Reference

**Mode** All Modes

**Panel** *Sidebar region* → *View* → *3D Cursor*

---

The 3D cursor can also be positioned and oriented by editing these values:

**Location** The location of the 3D Cursor.

**Rotation** The rotation of the 3D Cursor.

**Rotation Mode** The Rotation mode of the 3D Cursor.

#### Usage

The 3D Cursor is used as the origin for any added object, can be used and moved with the *snap tool*, and is an option for the *pivot point*.

#### Selecting

This page discusses specific selecting tools for the 3D Viewport. The 3D Viewport also uses the general select tools used which are described in the *interface section*.

**Center Point Ctrl** Selects the object by its center point rather than its contents.

**Menu Alt** If the objects are overlapping in the view, a menu of objects under the cursor can be used, so you can pick the object by its name.

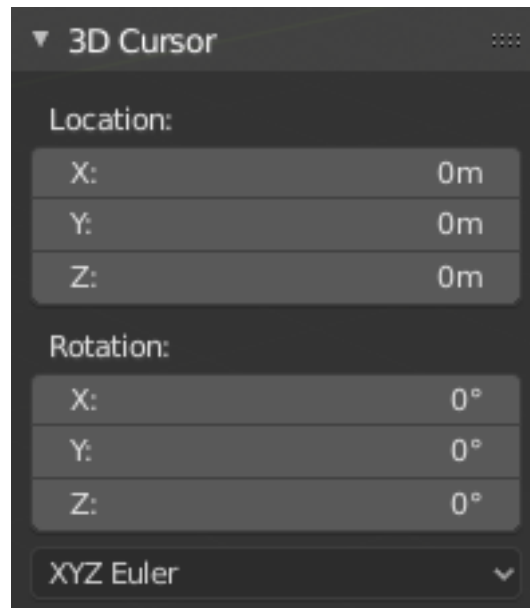


Fig. 97: The 3D Cursor panel of the Sidebar region.

These can be combined so you can for example toggle the selection from an objects center point.

---

**Note:** Right-click-select (see *Select With Mouse Button* option) has the advantage that selection is possible without having to switch to the select tool.

---

## 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*

## Controls

### Transform Orientations

---

#### Reference

**Mode** Object and Edit Modes

**Panel Header** → *Transform Orientations*

**Hotkey** Comma

---

*Transform Orientations* affect the behavior of *Transformations*. You will see an effect on the *Object Gizmo* (the widget in the center of the selection), as well as on transformation constraints, *Axis Locking*.

For example, when you press X, during the execution of the operation, it will constrain the transformation to the *Global* X axis. But if you press X a second time it will constrain to your *Transform Orientation's* X axis.

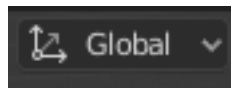


Fig. 98: Transform Orientations selector.

The Orientations options can be set through the *Transform Orientation* selector in a 3D Viewport header.

In addition to the five preset options, you can define your own custom orientation (see *Custom Orientations* below).

## Orientations

**Global** Align the transformation axes to world space.

The *Navigation Gizmo* in the top right corner of the viewport, and the *Grid Floor*, shows the axes of world coordinate system.

**Local** Align the transformation axes to the selected objects' space.

When an object is rotated, the direction of the *Local* gizmo matches to the object's rotation relative to the global axes. While the *Global* gizmo always correspond to world coordinates.

**Normal** Align the transformation axes so that the Z axis of the gizmo will match the average *Normal* of the selected element. If multiple elements are selected, it will orient towards the average of those normals.

In *Object Mode*, this is equivalent to *Local* orientation.

**Gimbal** Align each axis to the Euler rotation axis as used for input. Uses a *Gimbal* behavior that can be changed depending on the current *Rotation Mode*.

**View** Align the transformation axes to the window of the 3D Viewport:

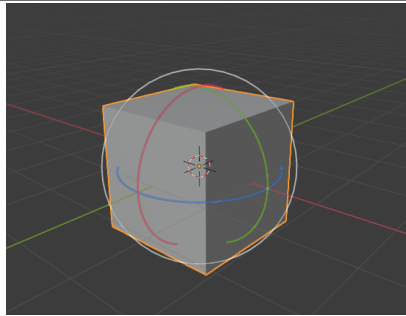
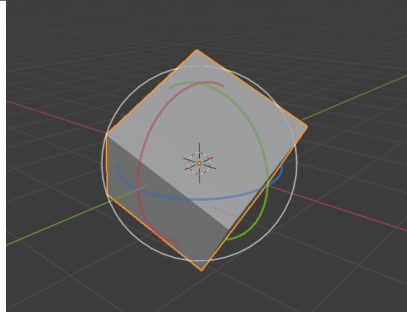
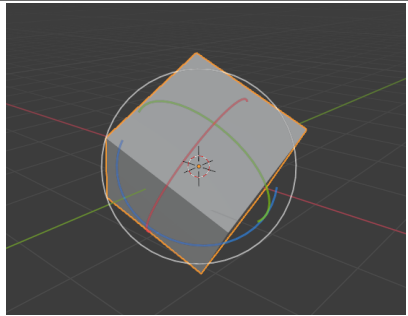
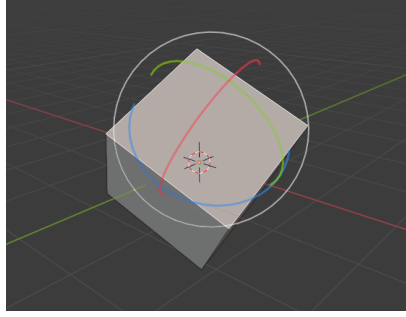
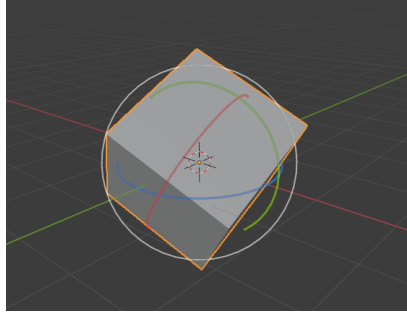
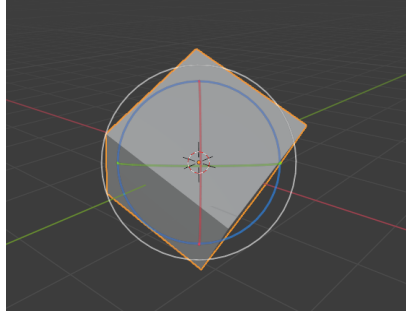
- Y: Up/Down
- X: Left/Right
- Z: Towards/Away from the screen

**Cursor** Align the transformation axes to the 3D cursor.



## Examples

Table 11: Cube with the rotation gizmo active in multiple transform orientations.

		
<p>Fig. 99: Default cube with Global transform orientation selected.</p>	<p>Fig. 100: Rotated cube with Global orientation, gizmo has not changed.</p>	<p>Fig. 101: Local orientation, gizmo matches to the object's rotation.</p>
		
<p>Fig. 102: Normal orientation, in Edit Mode.</p>	<p>Fig. 103: Gimbal transform orientation.</p>	<p>Fig. 104: View transform orientation.</p>

## Custom Orientations

### Reference

**Mode** Object and Edit Modes

**Panel** Header → *Transform Orientations*

You can define custom transform orientations, using object or mesh elements. Custom transform orientations defined from objects use the *Local* orientation of the object whereas those defined from selected mesh elements (vertices, edges, faces) use the *Normal* orientation of the selection.

The *Transform Orientations* panel, found in the header of the 3D Viewport, can be used to manage transform orientations: selecting the active orientation, adding (“+” icon), deleting (“X” icon) and rename custom orientations.

The default name for these orientations is derived from what you have selected. If it's an edge, it will be titled, “Edge”, if it's an object, it will take that object's name, etc.

### Create Orientation

To create a custom orientation, select the object or mesh element(s) and click the “+” button on the *Transform Orientations* panel.

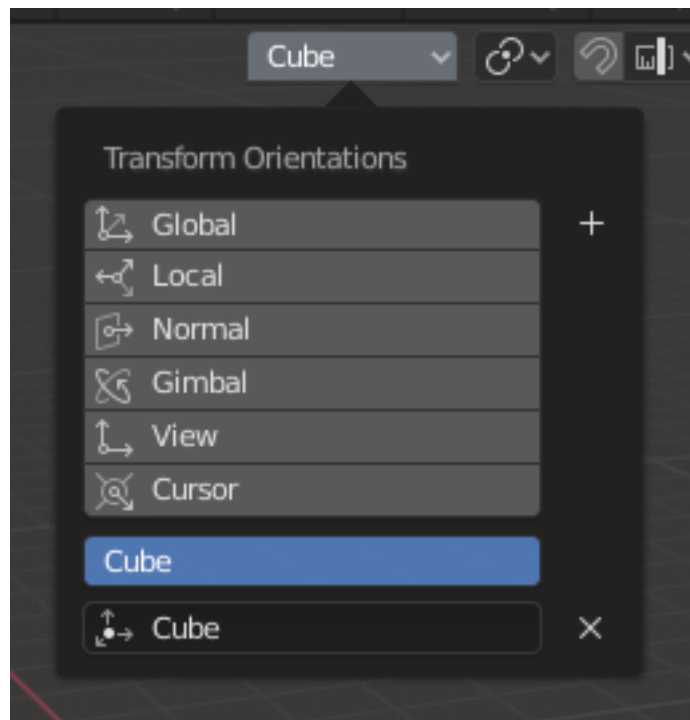


Fig. 105: Transform Orientations panel.

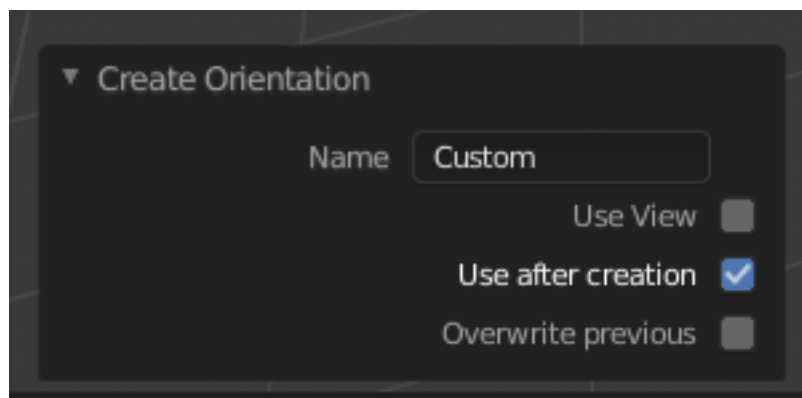


Fig. 106: Create Orientation *Adjust Last Operation* panel.

Just 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** If checked it leaves the newly created orientation active.

**Overwrite Previous** If the new orientation is given an existing name, a suffix will be added to it to avoid overwriting the old one, unless *Overwrite Previous* is checked, in which case it will be overwritten.

## Pivot Point

---

### Reference

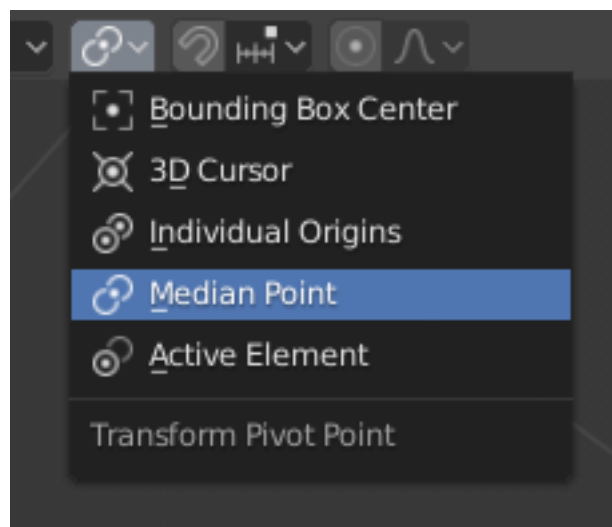
**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point*

**Hotkey** Period

---

When rotating or scaling an object or group of vertices, edges, or faces, you may want to move the *Pivot Point* to make it easier to manipulate an object. Using this selector in the header of any 3D Viewport, you can change the location of the pivot point.



## Pivot Types

### Bounding Box Center

---

### Reference

**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point* → *Bounding Box Center*

**Hotkey** Period

---

The bounding box is a rectangular box that is wrapped as tightly as possible around the selection. It is oriented parallel to the world axes. In this mode the pivot point lies at the center of the bounding box. You can set the pivot point to *Bounding Box* with Comma or via the menu in the editor's header. The image below shows how the object's bounding box size is determined by the size of the object.

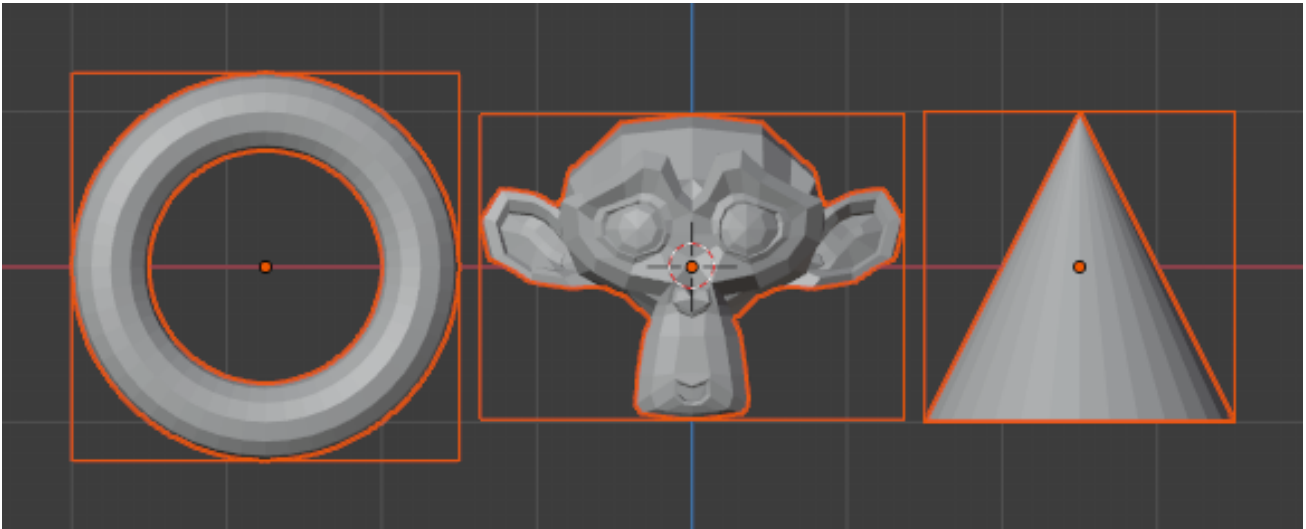


Fig. 107: Relationship between an object and its bounding box.

### In Object Mode

In *Object Mode*, transformation takes place relative to the location of the objects origin (indicated by the yellow circle), and the size of objects is not taken into account. The image below shows the results of using the *Bounding Box* as the pivot point in some situations.

In this example, the orange rectangle has its origin located on the far left of the mesh, while the blue rectangle has its origin located in the center of the mesh.

When a single object is selected, the rotation takes place around its origin.

The image above (left) shows that when multiple objects are selected, the pivot point is calculated based on the location of all the selected objects. More precisely, the centers of objects are taken into account.

### In Edit Mode

This time it is the geometry that is enclosed in the bounding box. The bounding box in *Edit Mode* takes no account of the object(s) origins, only the center of the selected vertices.

### 3D Cursor

#### Reference

**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point* → 3D Cursor

**Hotkey** Period

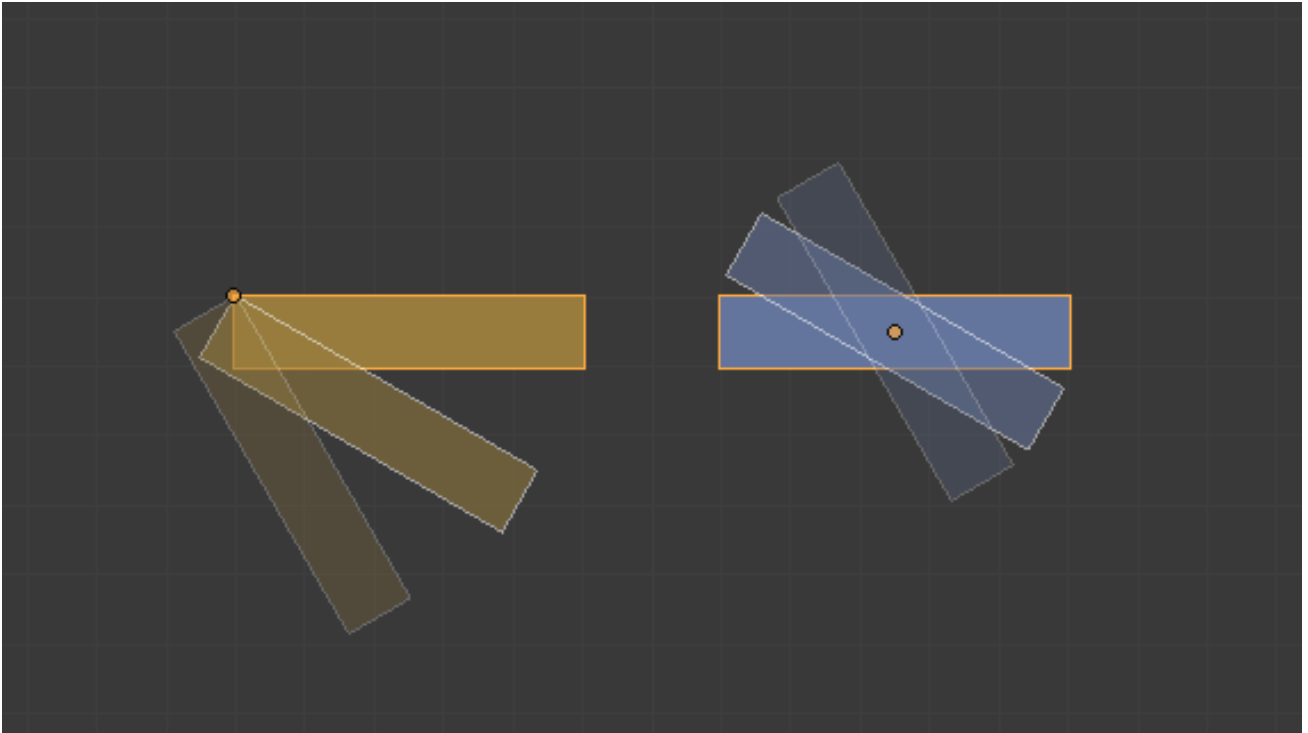


Fig. 108: Single object rotation.

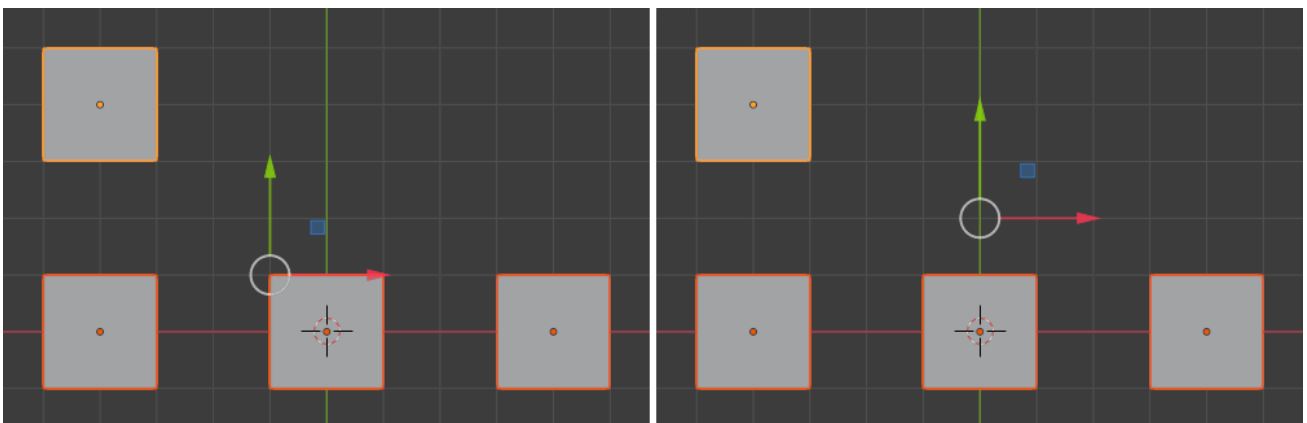


Fig. 109: Shows the location of the bounding box (right) pivot point compared to the median point (left).

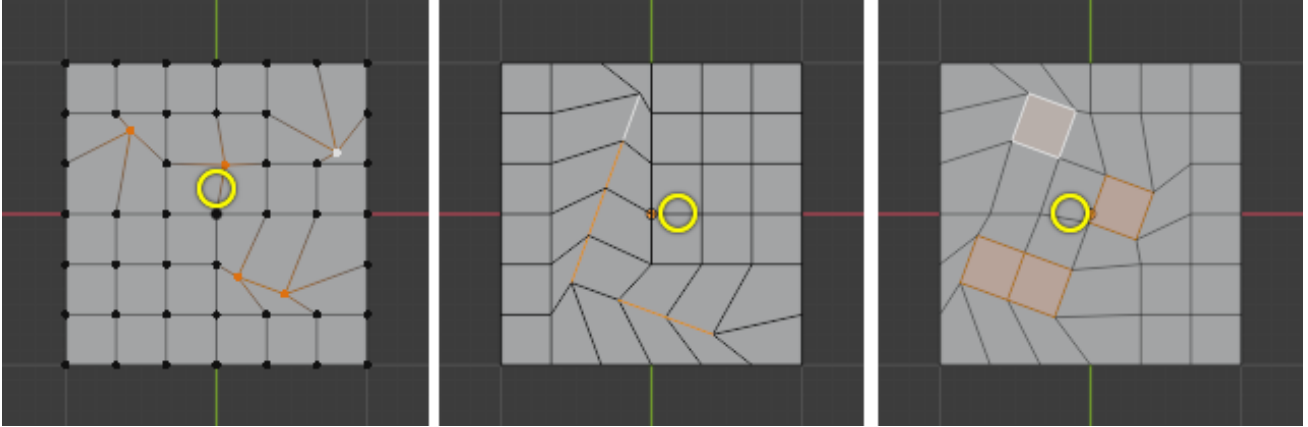


Fig. 110: The effects of rotation in different mesh selection modes when the bounding box is set as the pivot point. The pivot point is shown by a yellow circle.

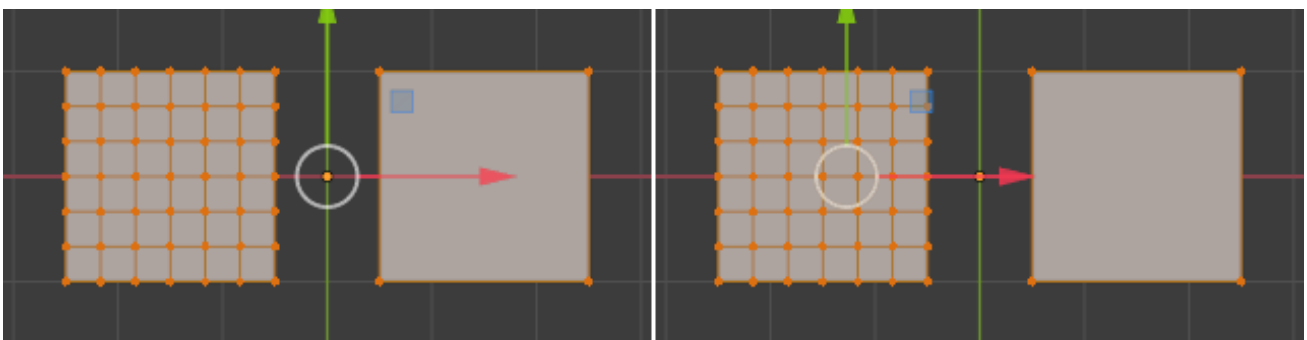


Fig. 111: The bounding box center compared to the median point.

The 3D cursor is the most intuitive of the pivot points. With the 3D cursor selected as the active pivot point (from either the *Editor's Header* or via *Period*), simply position the 3D cursor and then do the required transformation. All rotation and scaling transformations will now be done relative to the location of the 3D cursor.

### Example

The image below shows the difference when rotating an object around the median point (right) and around the 3D cursor (left).

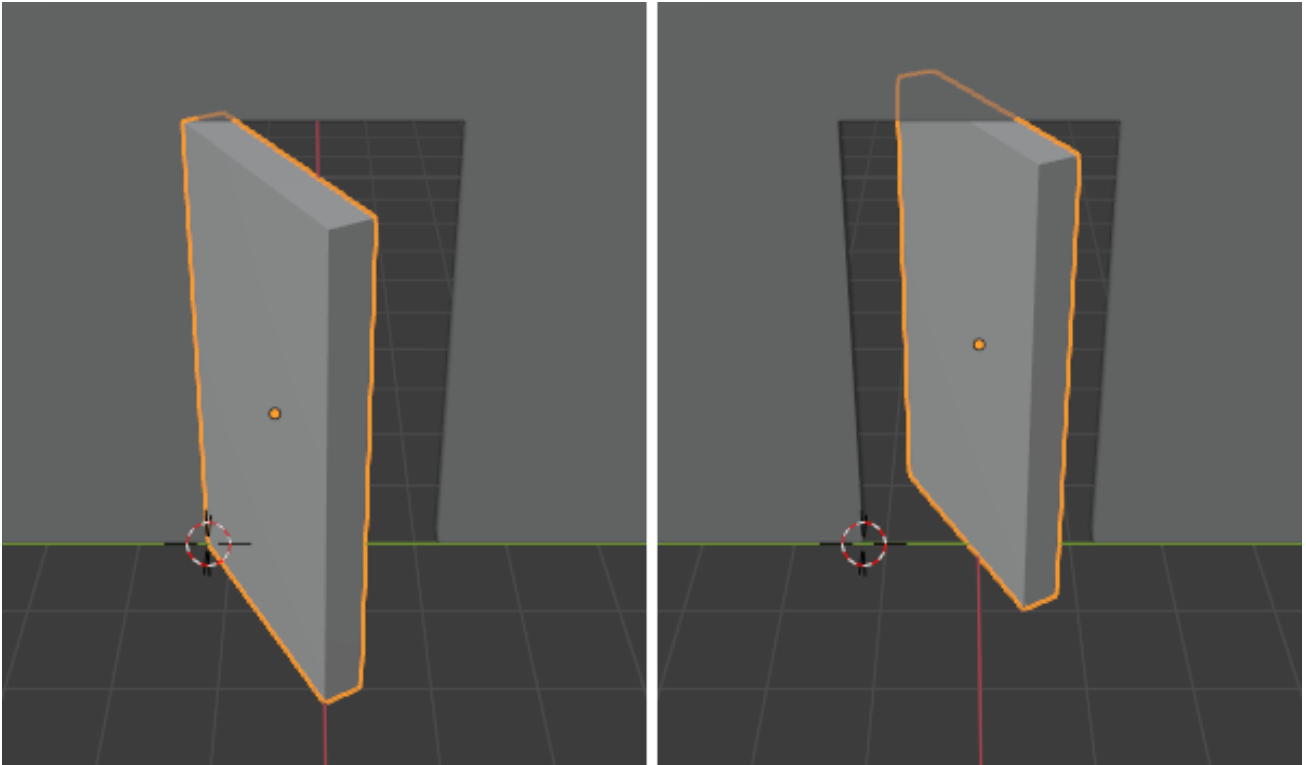


Fig. 112: Rotation around the 3D cursor compared to the median point.

### Individual Origins

#### Reference

**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point* → *Individual Origins*

**Hotkey** *Period*

#### In Object Mode

The origin of an object is shown in the 3D Viewport by a small orange circle. It tells Blender the relative position of that object in 3D space.

The origin does not have to be located in the center of the geometry (e.g. mesh). This means that an object can have its origin located on one end of the mesh or even completely outside the mesh.

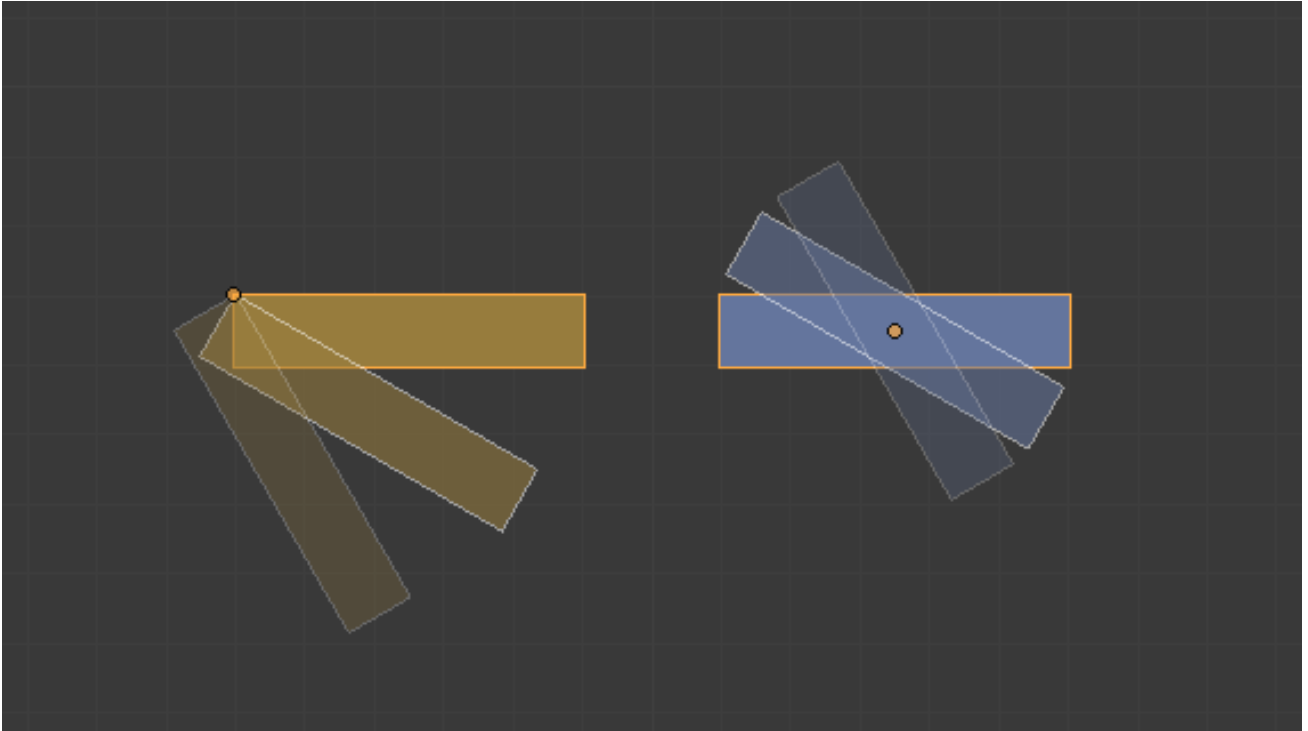


Fig. 113: Rotation around individual origins.

For example, the orange rectangle in the image above has its origin located on the far left of the mesh, while the blue rectangle has its origin located in the center of the mesh.

When the Pivot Point is set to *Individual Origins*, the origin of each object remains in place while the object rotates or scales around it.

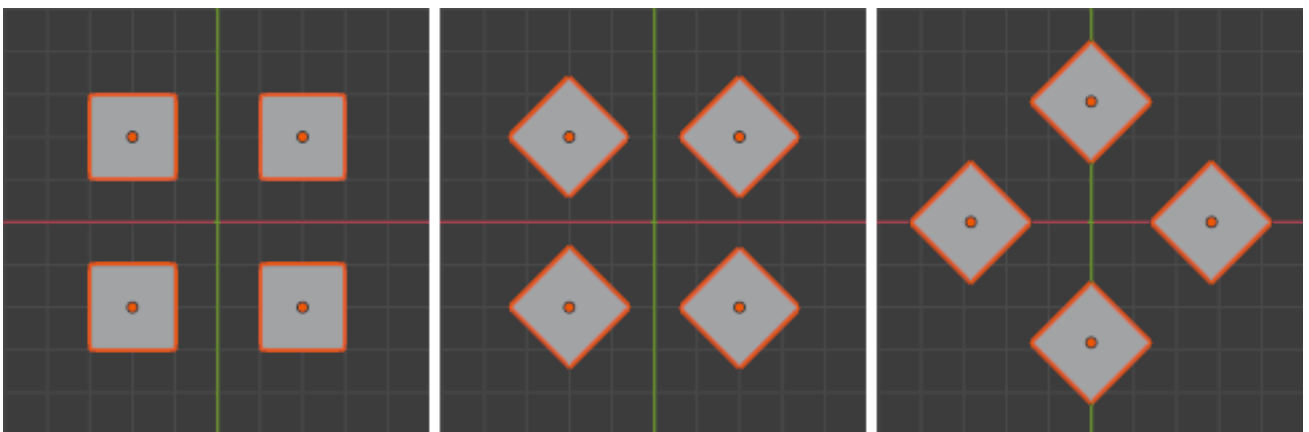


Fig. 114: Rotation around Individual Origins (middle) compared to the Median Point (right).

### In Edit Mode

When you rotate or scale the touching faces/edges, they are treated as a single element, and keep the shape of the group. Each group is transformed independently around its median point.



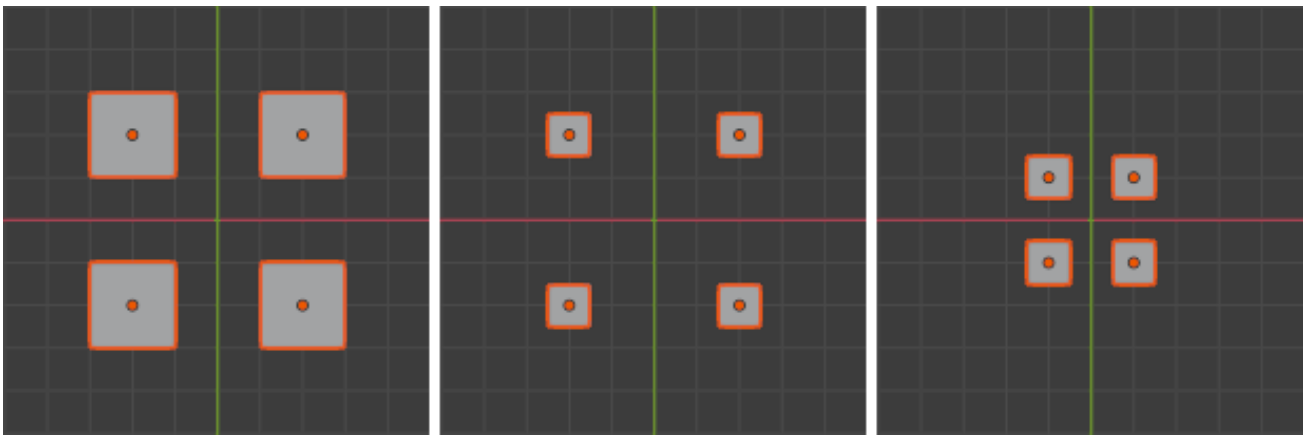


Fig. 115: Scaling around Individual Origins (middle) compared to the Median Point (right).

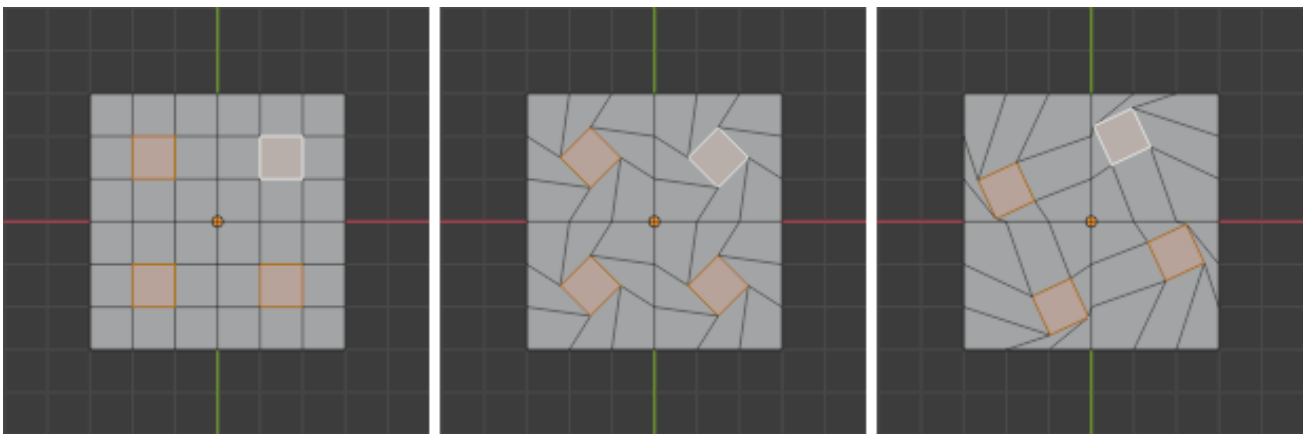


Fig. 116: Rotation of individual faces around Individual Origins (middle) and the Median Point (right).

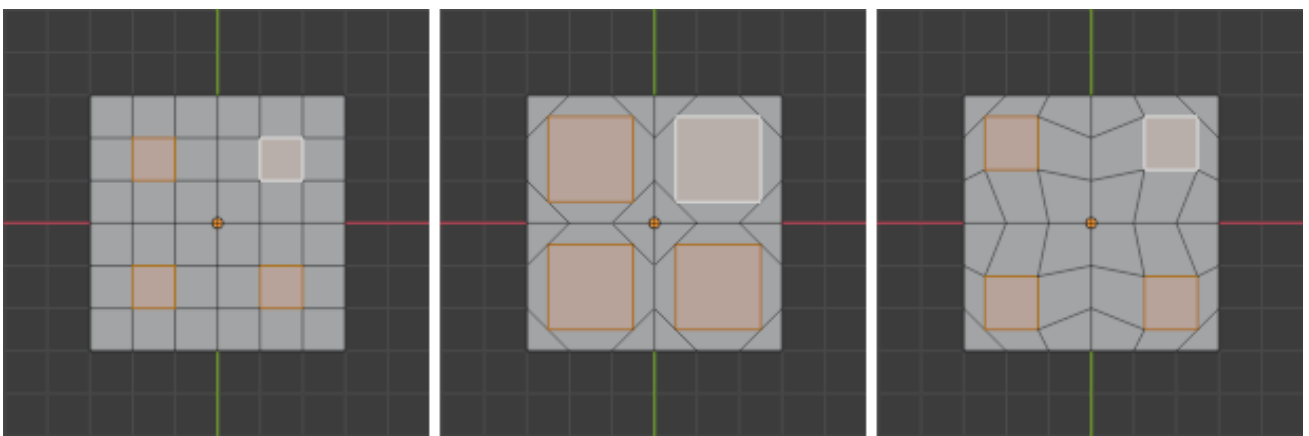


Fig. 117: Scaling with non-touching faces around Individual Origins (middle) and the Median Point (right).

## Median Point

### Reference

**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point* → *Median Point*

**Hotkey** Period

The Median Point can be considered to be broadly similar to the concept of Center of Gravity (COG). If we assume that every element (object, face, vertex, etc.) of the selection has the same mass, the median point would sit at the point of equilibrium for the selection (the COG).

### In Object Mode

In Object Mode, Blender only considers the object origins when determining the median point. This can lead to some counter-intuitive results. In the Fig. *Median points in Object Mode.* below, you can see that the median point is between the object origins and can be nowhere near the objects' mesh (geometric center).

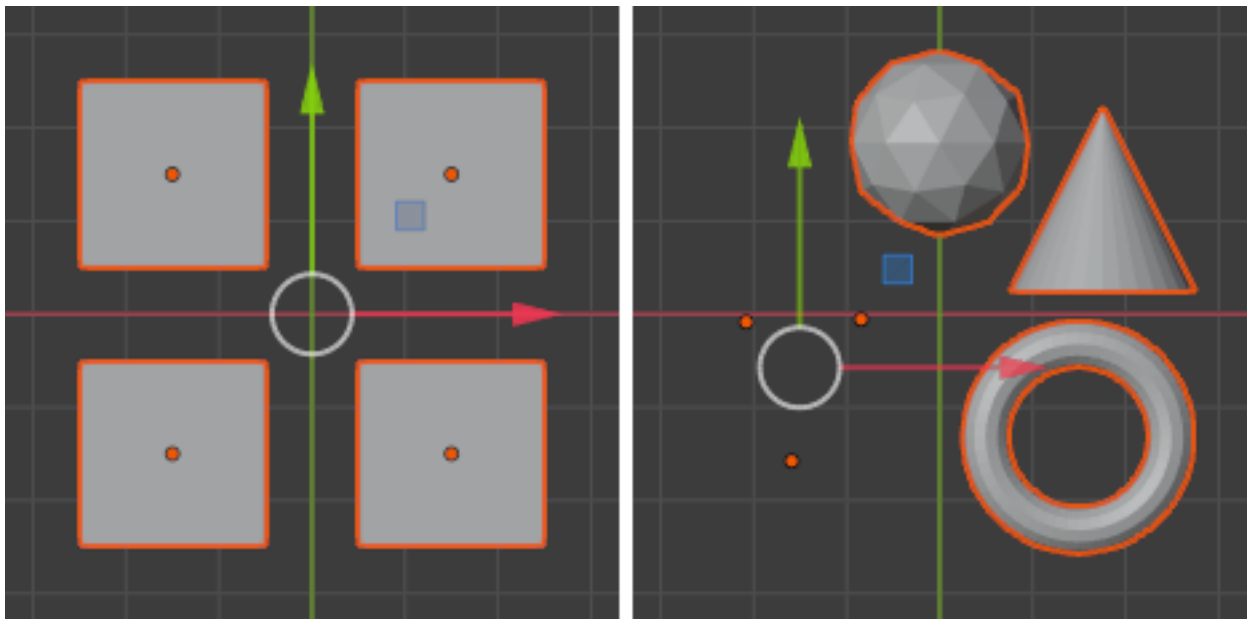


Fig. 118: Median points in Object Mode.

### In Edit Mode

In Edit Mode, the median point is determined via the part of the selection that has the most elements. For example, in the Fig. *Median points in Edit Mode.*, when there are two cubes with an equal number of vertices, the median point lies directly between the two cubes. But if you subdivide one cube multiple times so that it has many more vertices, you can see that the median point has shifted to the region with the most vertices.

### Active Element

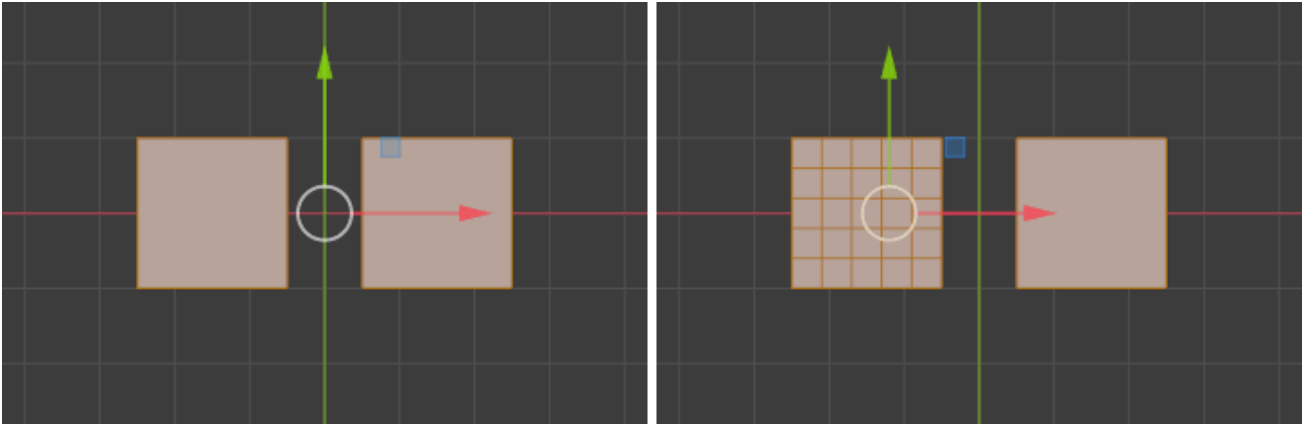


Fig. 119: Median points in Edit Mode.

---

## Reference

**Mode** Object Mode and Edit Mode

**Header**  *Pivot Point* → *Active Element*

**Hotkey** Alt-Period

---

The *active element* can be an object, vertex, edge or a face. The active element is the last one to be selected and will be shown in a lighter orange color when in *Object Mode* and white when in *Edit Mode*. With *Active Element* as *Pivot* set to active, all transformations will occur relative to the active element.

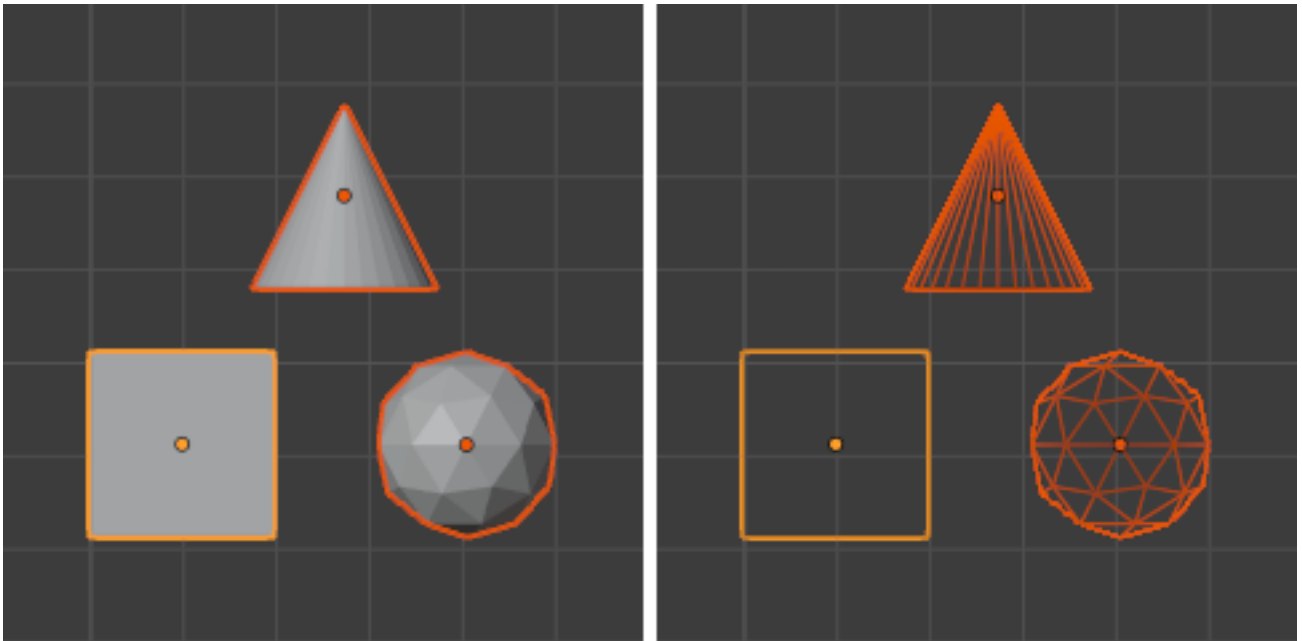


Fig. 120: Display of active elements in Object Mode where the active element (cube) is a lighter orange.

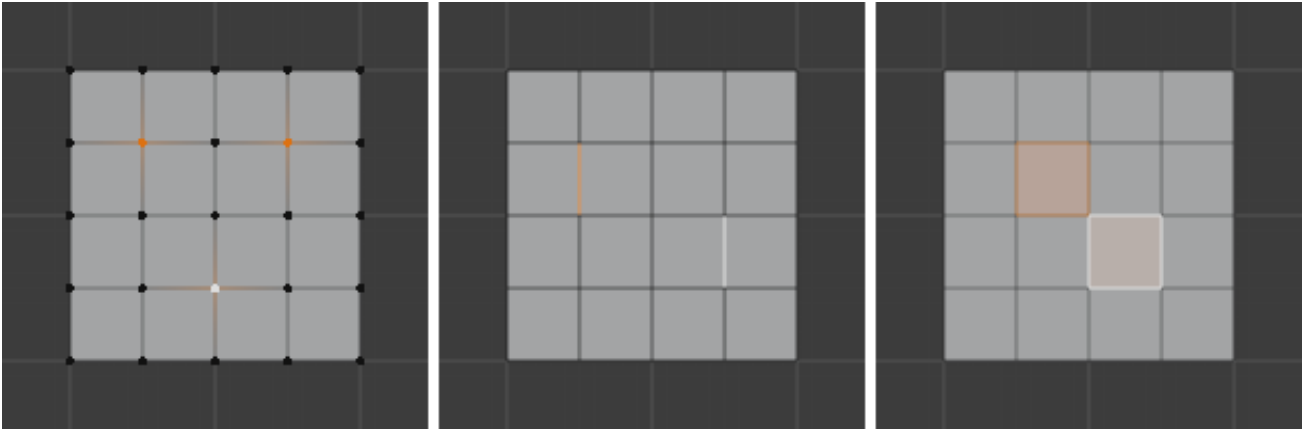


Fig. 121: Active elements for vertices, edges and faces in Edit Mode are displayed in white.

### In Object Mode

When in *Object Mode*, rotation and scaling happen around the origin of the active object. This is shown by the figure to the below where the active object (the cube) remains in the same location (note its position relative to the background grid) while the other objects rotate and scale in relation to the active element.

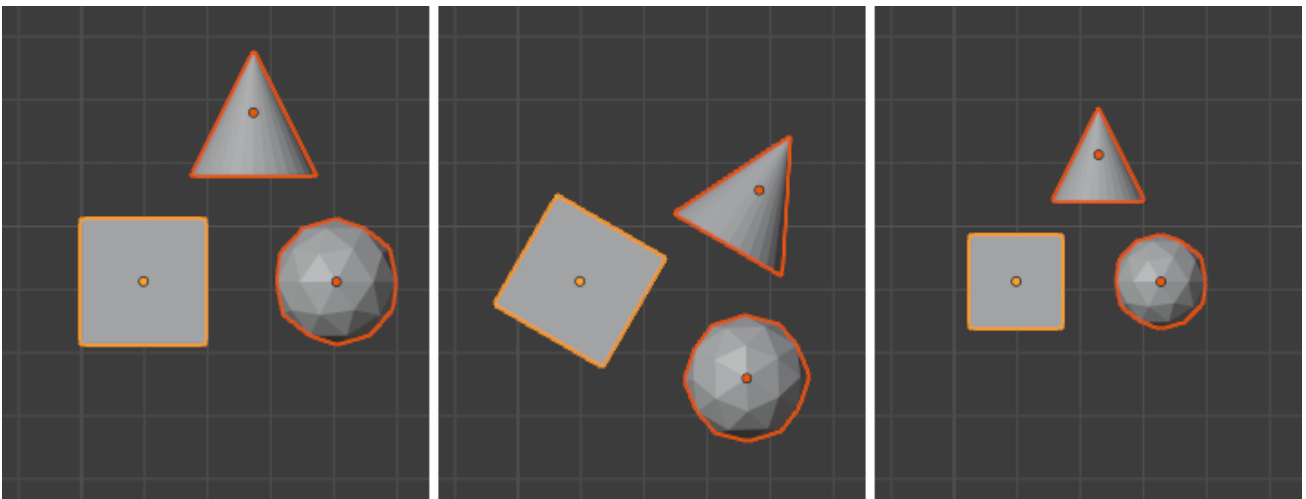


Fig. 122: Rotation and scaling with the cube as the active element.

### In Edit Mode

Using the active element as a pivot point in *Edit Mode* may seem complex but all the possible transformations follow a few rules:

- The pivot point is always at the median of the active element.
- The transformations occur by transformation of the *vertices* of the selected element(s). If an unselected element shares one or more vertices with a selected element then the unselected one will get some degree of transformation also.

Let us examine the following examples: in each case we will see that the two rules apply.

## Single Selection

When one single element is selected it becomes automatically active. In the image below, you can see that when it is transformed its vertices move, with the consequence that any adjacent element which shares one or more vertices with the active element is also transformed.

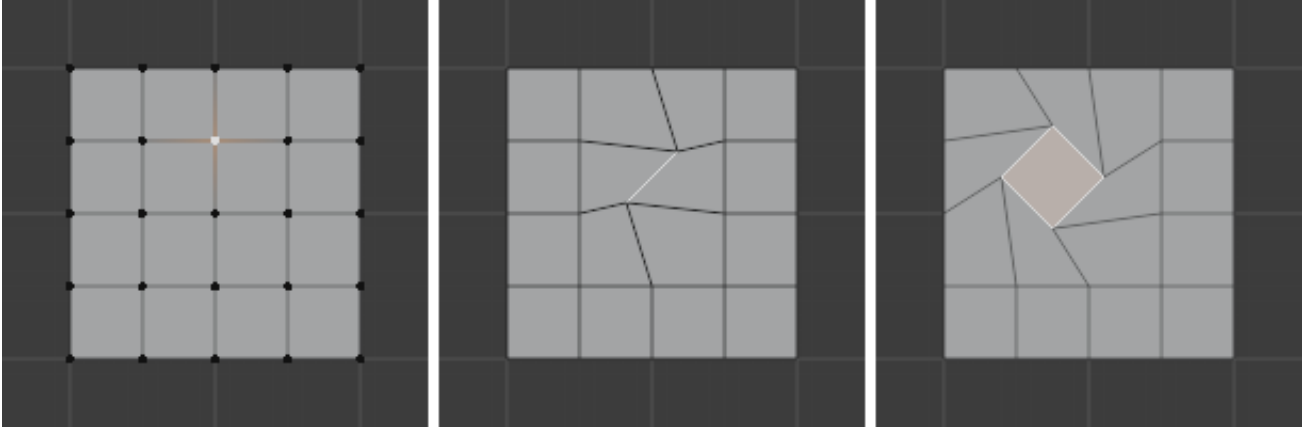


Fig. 123: Edit Mode and only one element selected.

Let us review each case:

- *Faces* have their pivot point where the median of their vertices is.
- *Edges* have their pivot point on their middle since this is always where the median of an edge is.
- A single *vertex* has no dimensions at all so it cannot show any transformation (except translation, which is not affected by the pivot point).

## Multiple Selection

When multiple elements are selected they all transform. The pivot points stay in the same place as what we have described above. In the image below, the selected elements have been rotated.

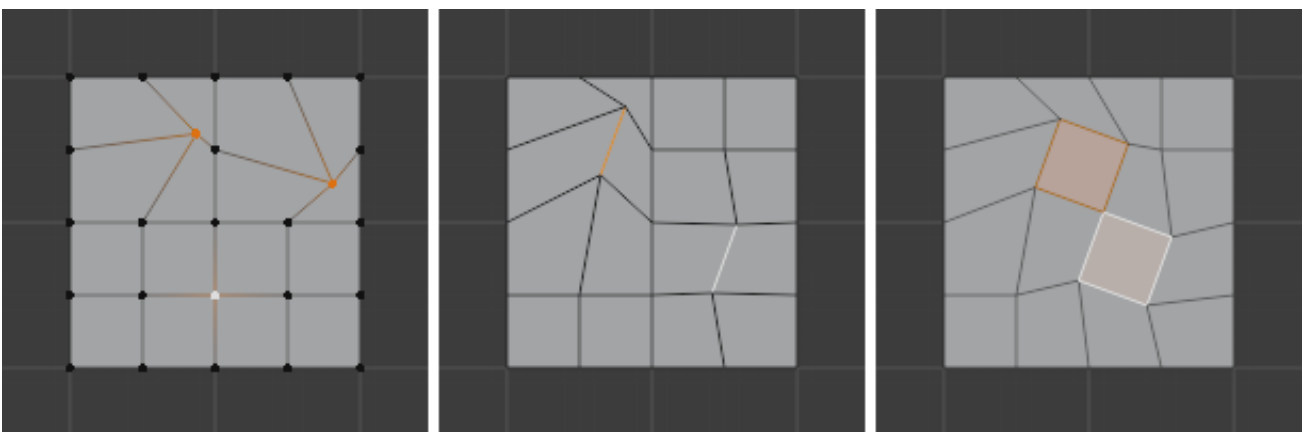


Fig. 124: Edit Mode and multiple selections.

- For *faces* the transformation occurs around the median of the vertices of the selected face.
- *Edges* also keep the same behavior with their pivot point at their median.
- There is a case for *vertices* this time: the active vertex is where the pivot point resides. All other vertices are transformed relative to it.

## Snapping

### Reference

**Mode** Object, Edit, and Pose Mode

**Header** *Snap*

**Hotkey** Shift-Tab

The ability to snap objects and mesh element to various types of scene elements during a transformation is available by toggling the magnet icon in the 3D Viewport's header buttons.

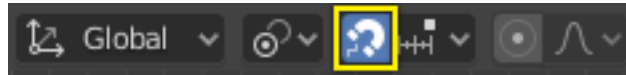


Fig. 125: Magnet icon in the 3D Viewport header (blue when enabled).

## Snap Element

### Reference

**Mode** Object, Edit, and Pose Mode

**Header** *Snapping* → *Snap to*

**Hotkey** Shift-Ctrl-Tab

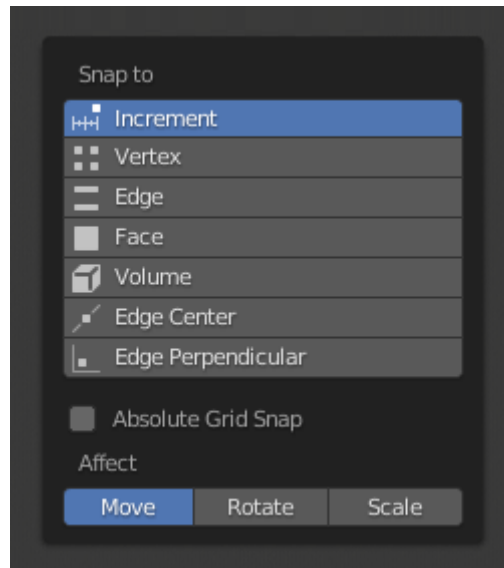


Fig. 126: Snap Element menu.

**Increment** Snap to grid points. When in Orthographic view, the snapping increment changes depending on zoom level.

**Note:** In this context the grid does not mean the visual grid cue displayed. Snapping will use the resolution of the displayed grid, but all transformations are relative to the initial position (before the snap operation).

**Vertex** Snap to vertices of mesh objects.

**Edge** Snap to edges of mesh objects.

**Face** Snaps to the surfaces of faces in mesh objects; This is useful for retopologizing.

**Volume** Snaps to regions within the volume of the first object found below the mouse cursor. Unlike the other options, this one controls the depth (i.e. Z coordinates in current view space) of the transformed element. By toggling the button that appears to the right of the snap target menu (see below), target objects will be considered as a whole when determining the volume center.

**Edge Center** Snaps to the middle of an edge. This snap element only pertains to mesh objects.

**Edge Perpendicular** Snaps to the nearest vertex in an edge that makes a perpendicular angle. This snap element only pertains to mesh objects.

**Tip:** Multiple snapping modes can be enabled at once by Shift-LMB the different snapping elements.

## Snap Target

### Reference

**Mode** Object, Edit, and Pose Mode

**Header** *Snapping* → *Snap with*

**Hotkey** Shift-Ctrl-Tab

Snap target options become active when either *Vertex*, *Edge*, *Face*, or *Volume* is selected as the snap element. These determine what part of the selection snaps to the target objects.

**Active** Moves the active element (vertex in Edit Mode, object in Object Mode) to the target.

**Median** Moves the median of the selection to the target.

**Center** Moves the current transformation center to the target. Can be used with 3D cursor to snap with an offset.

**Closest** Moves the closest point of the selection to the target.

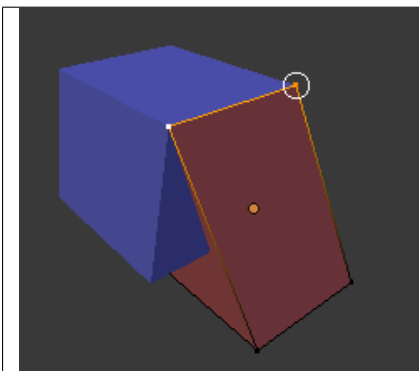


Fig. 127: Closest.

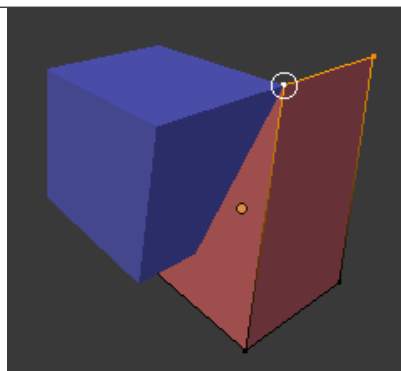


Fig. 128: Active.

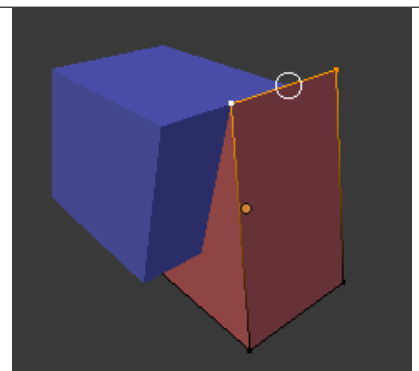
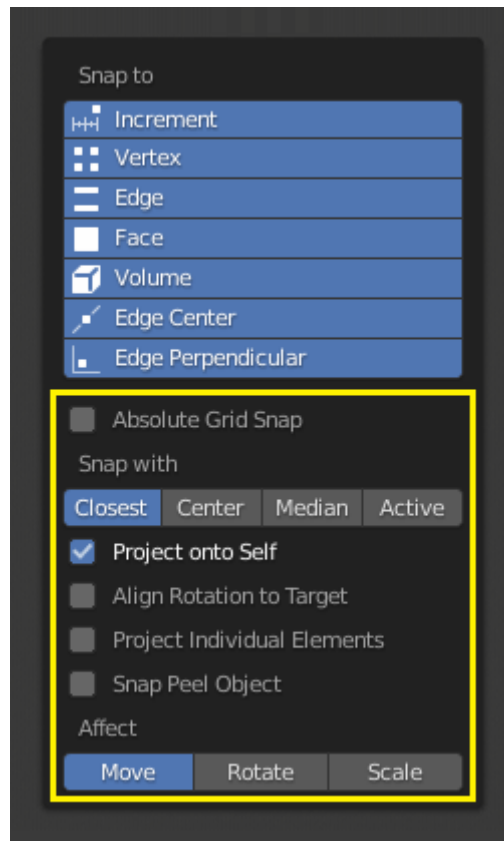


Fig. 129: Median.

## Additional Snap Options

As seen by the yellow highlighted areas in the image above, besides the snap target, additional controls are available to alter snap behavior. These options vary between mode (Object and Edit)



as well as Snap Element. The four options available are:

**Absolute Grid Snap** Available only for the increase option. Snap to grid, instead of snapping in increments relative to the current location.

**Backface Culling** Exclude back facing geometry from snapping.

**Project Onto Self** Available only in editing mode for Vertices, Edges, Faces and Volume. Snaps elements to its own mesh.

**Align Rotation to Target** Available for Vertices, Edges, Faces and Volume. When the Snap Affects Rotation, this align rotation with the snapping target.

**Project Individual Elements** Available for snap to Faces. Project individual elements on the surface of other objects.

**Snap Peel Object** Available for snap to Volume. Consider objects as whole when finding volume center.

**Affect** Limits the effect of the snap to the transformation type.

### Multiple Snap Targets

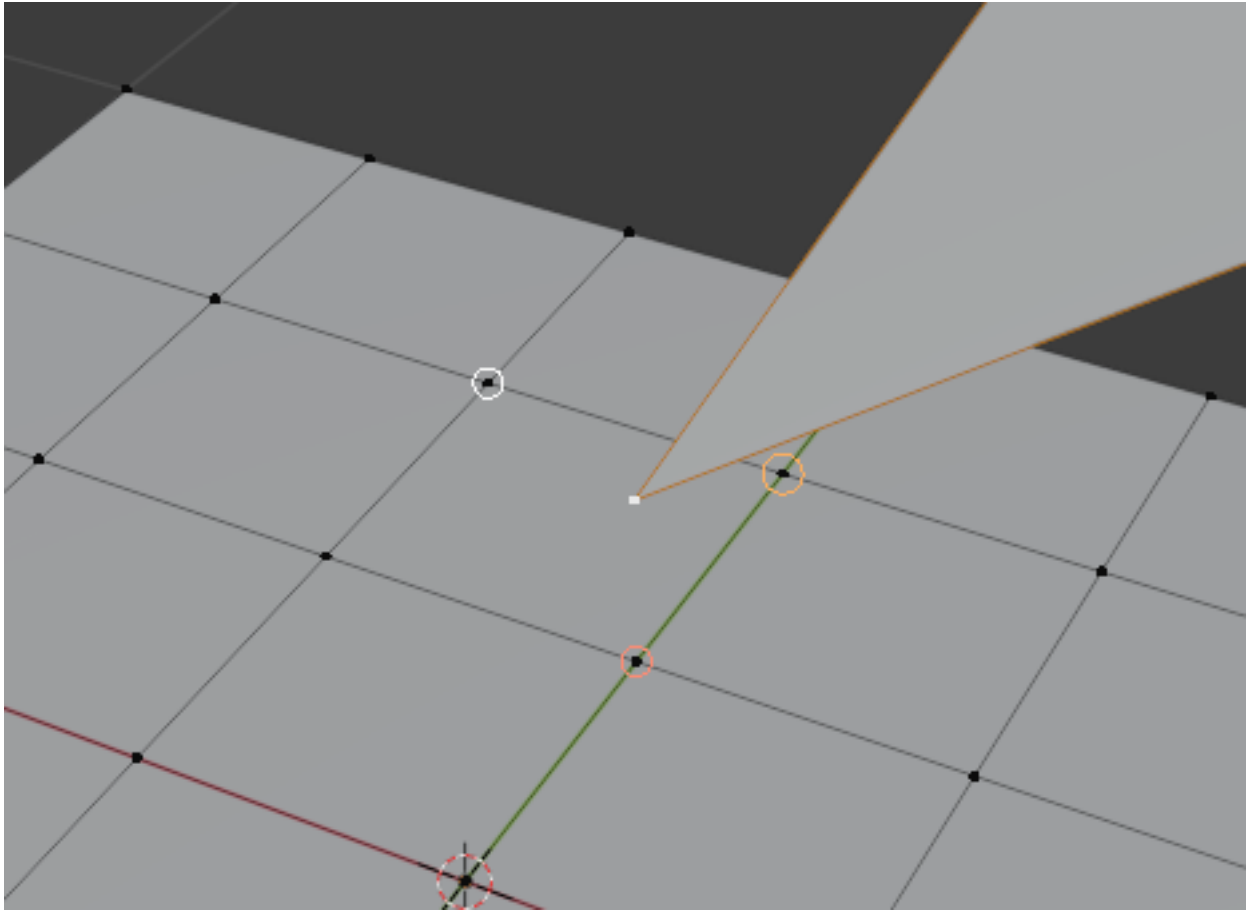
Once transforming a selection with Snapping on (not just when holding Ctrl), you can press A to mark the current snapping point, then proceed to mark as many other snapping points as you wish and the selection will be snapped to the average location of all the marked points.

Marking a point more than once will give it more weight in the averaged location.

Multiple snapping targets.

### Proportional Editing

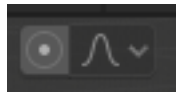




---

**Reference**

**Mode** Object and Edit Mode



**Header** Via the icon in the header.

**Hotkey** 0

---

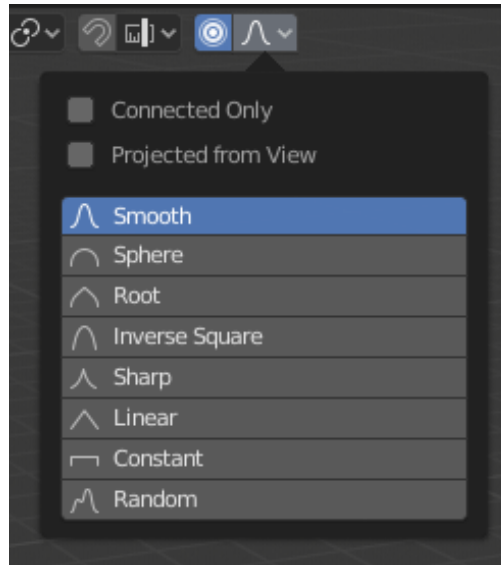


Fig. 130: Proportional Editing popover.

Proportional Edit is a way of transforming selected elements (such as vertices) while having that transformation affect other nearby elements. For example, having the movement of a single vertex cause the movement of unselected vertices within a given range. Unselected vertices that are closer to the selected vertex will move more than those farther from it (i.e. they will move proportionally relative to the location of the selected element). Since Proportional Editing affects the nearby geometry, it is very useful when you need to smoothly deform the surface of a dense mesh.

---

**Note:** Blender also has *Sculpting* that contains brushes and tools for proportionally editing a mesh without seeing the individual vertices.

---

### Controls

**Disable 0, Alt-0** Proportional Editing is off, only selected vertices will be affected.

**Enable 0, Alt-0** Vertices other than the selected vertex are affected, within a defined radius.

### Influence

You can increase or decrease the radius of the tool's influence during a transform operation with WheelUp, WheelDown or PageUp, PageDown respectively. As you change the radius, the points surrounding your selection will adjust their positions accordingly.

### Falloff

While editing, you can change the curve profile used by either using the header icon *Falloff* menu, or by pressing Shift-0 to toggle between the various options.

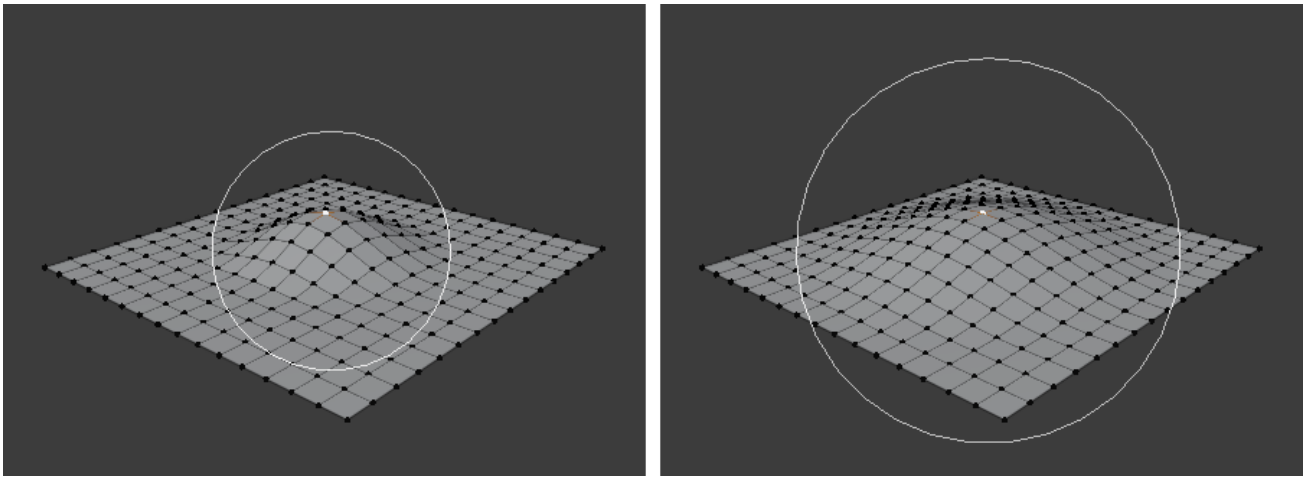
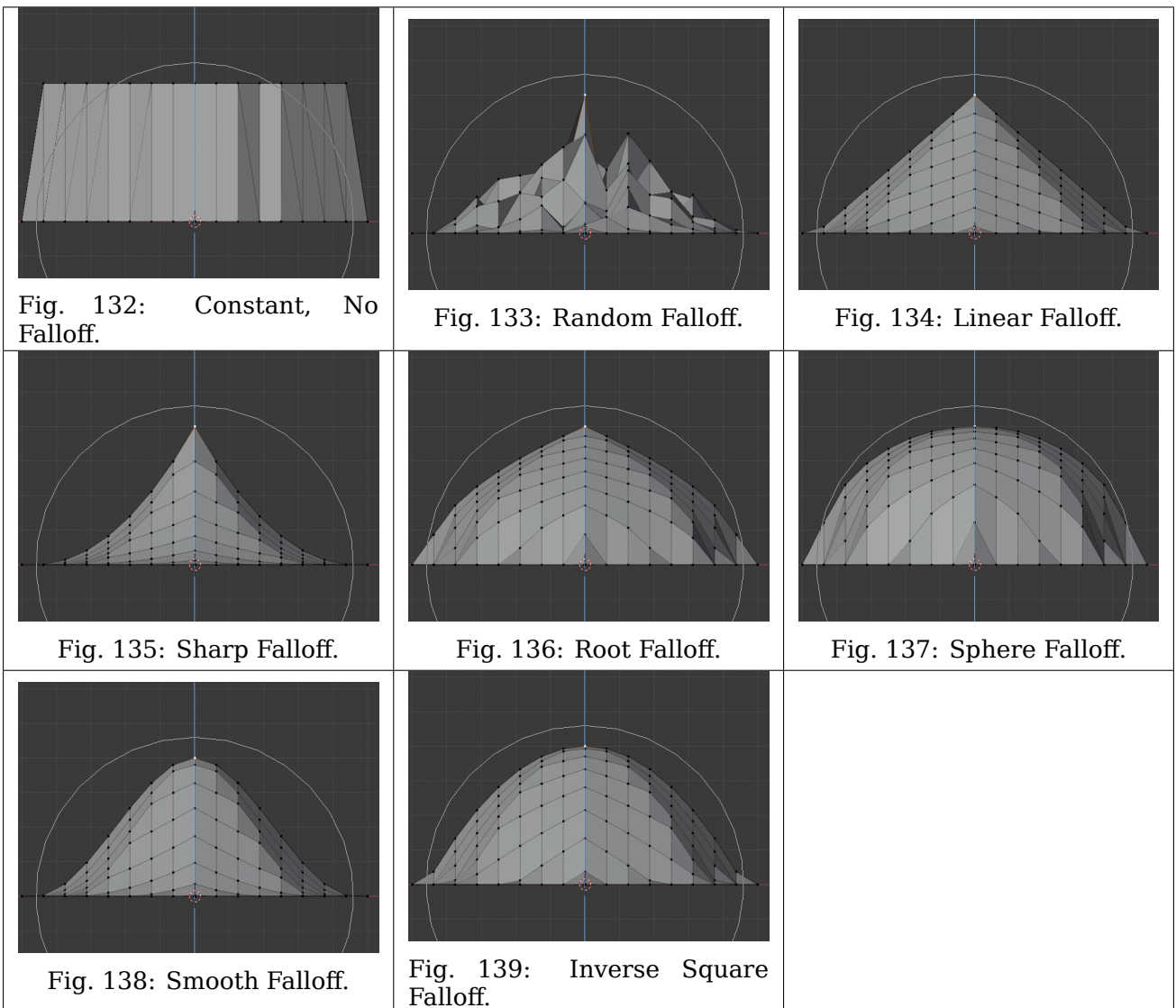


Fig. 131: Influence circle.



## Object Mode

Proportional Editing is typically used in *Edit Mode*, however, it can also be used in *Object Mode*. In *Object Mode* the tool works on entire objects rather than individual mesh components.

In the image below, the right cylinder is scaled along the Z axis. When the *Proportional Editing* is enabled, the adjacent cylinders are also within the tool's radius of influence.

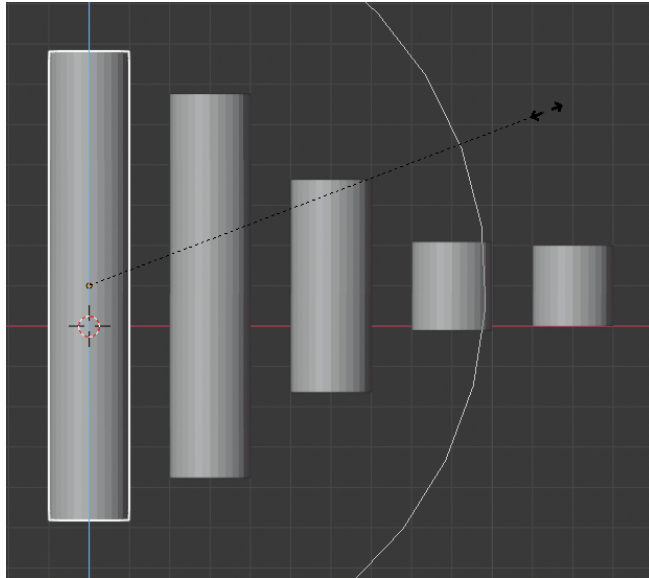


Fig. 140: Proportional Editing in Object Mode.

## Edit Mode

When working with dense geometry, it can become difficult to make subtle adjustments to the vertices without causing visible lumps and creases in the model's surface. When you face situations like this the Proportional Editing tool can be used to smoothly deform the surface of the model. This is done by the tool's automatic modification of unselected vertices within a given range.

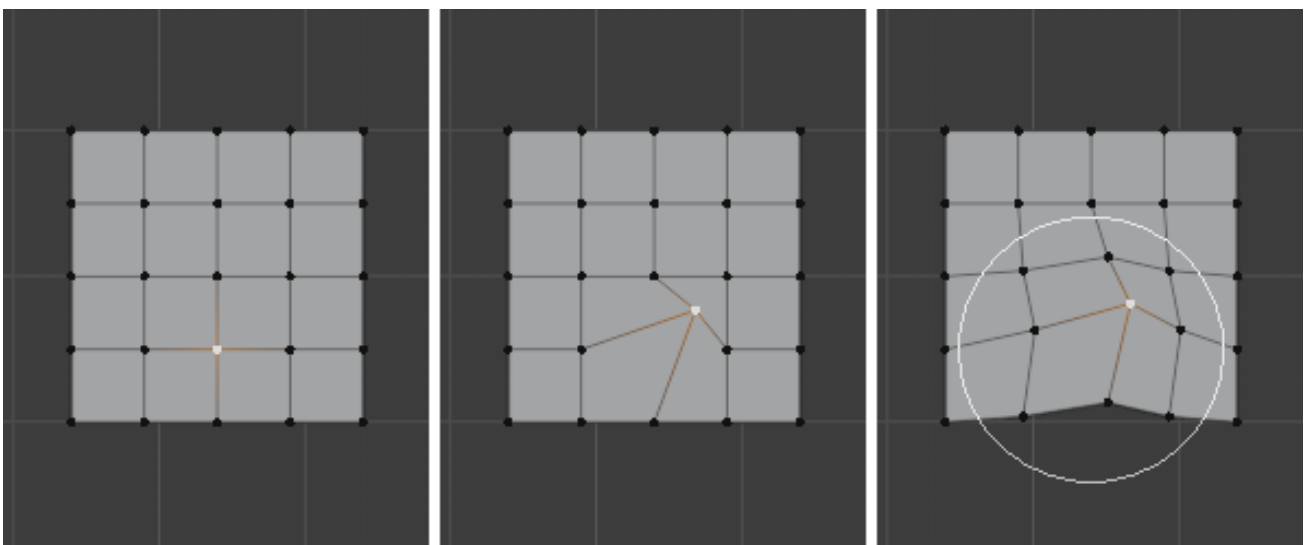


Fig. 141: Proportional Editing in Edit Mode.

## Options

**Connected Only Alt-0** 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.

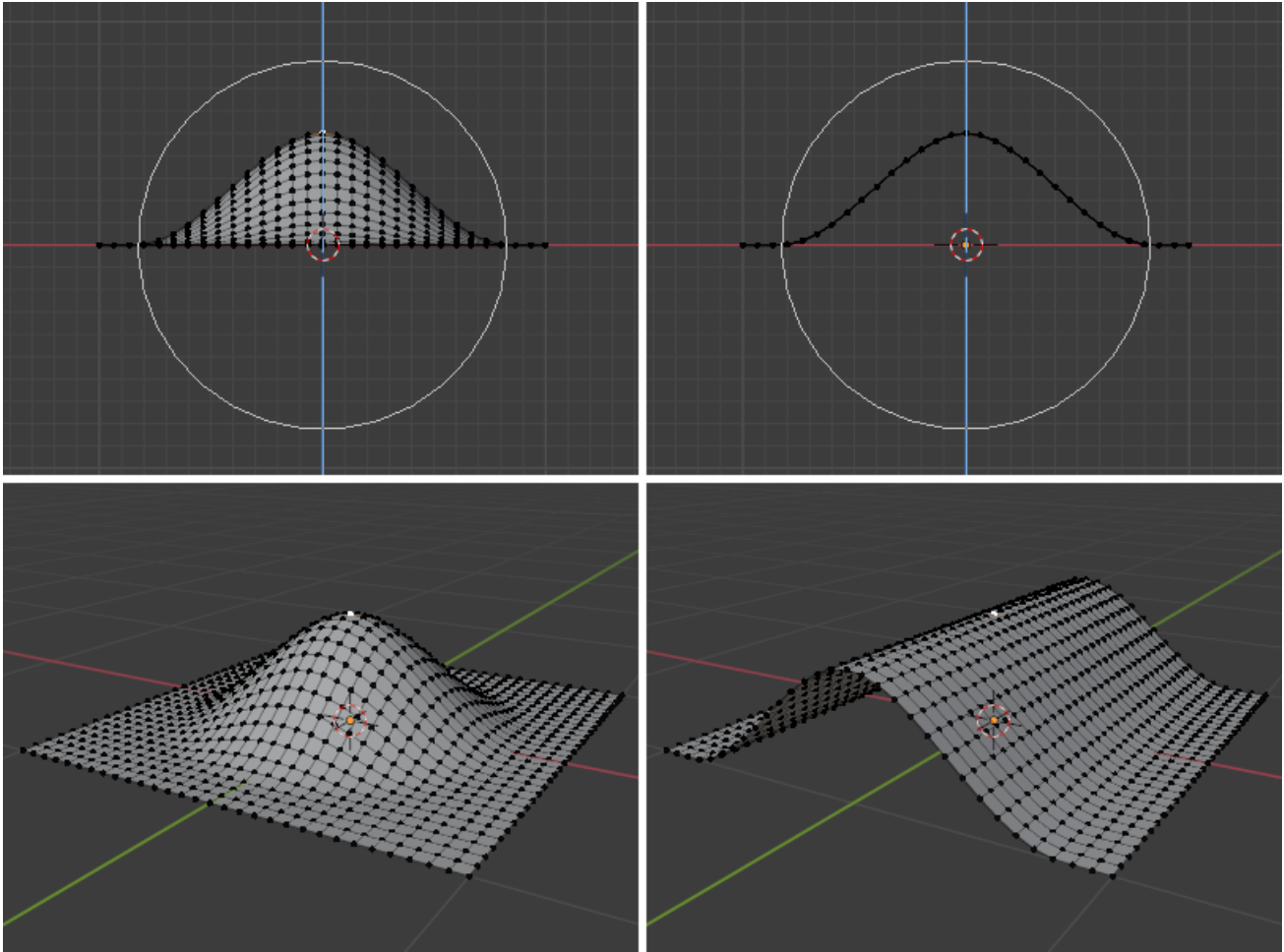


Fig. 142: The difference between regular and Projected (2D) proportional option (right).

## Example

The image below shows the final render of the low-poly landscape obtained by moving up the vertices of the triangulated grid with enabled *Proportional Editing*.

## Display

### Object Type Visibility

In the Object Type Visibility pop-over the visibility and selectability per type of object can be specified. This will limit the visibility and selectability per 3D Viewport.

The object types that can be changed are:

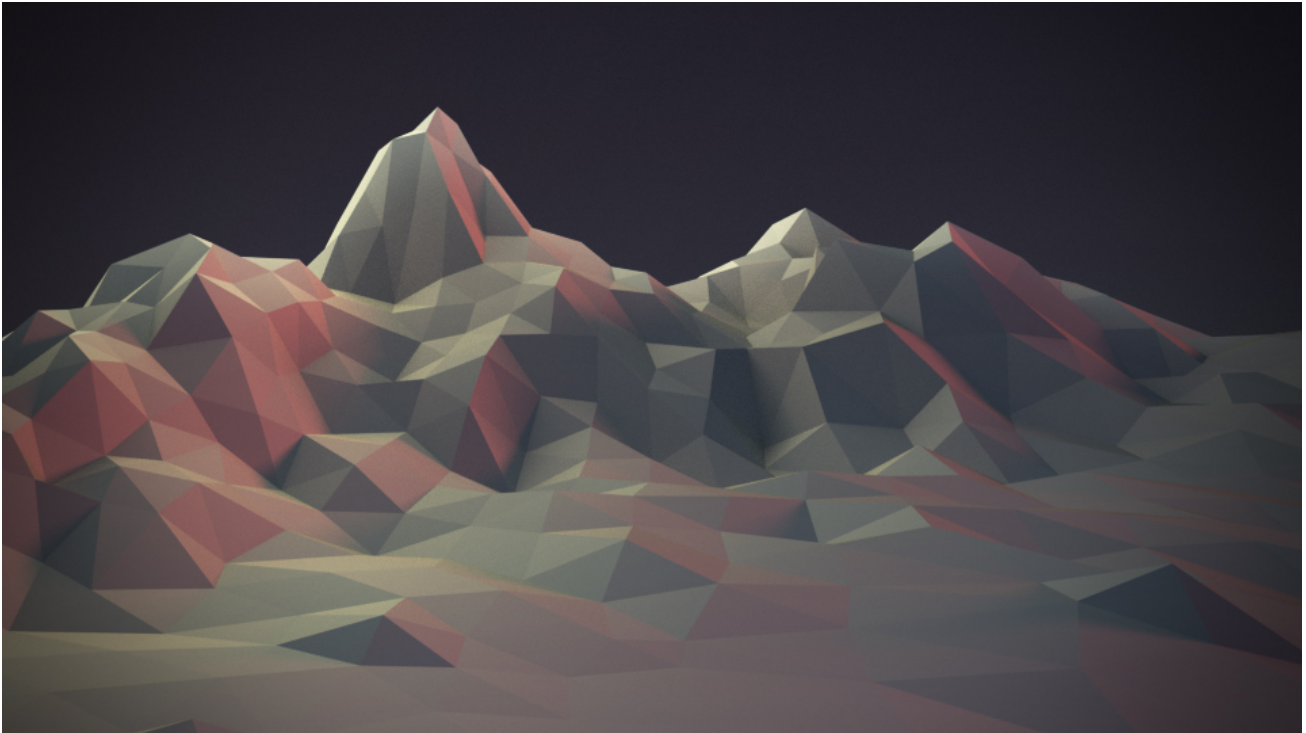


Fig. 143: A landscape obtained via Proportional Editing.

- Mesh
- Curve
- Surface
- Meta
- Text
- Grease Pencil
- Armature
- Lattice
- Empty
- Light
- Light Probe
- Camera
- Speaker

When visibility is turned off any object of this kind will not be rendered in the 3D Viewport. When selectability is turned off any object of this kind will not be selectable via the 3D Viewport.

## Viewport Gizmos

### Reference

**Mode** Object and Edit Modes

**Header**  *Viewport Gizmos* → *Object Gizmos*

The way how gizmos are displayed in the 3D Viewport can be changed in the Viewport Gizmos pop-over. There is a switch to turn off all gizmos for the 3D Viewport.

## Viewport Gizmos

**Navigate** Enable/disable the navigation gizmo.

**Active Tool** Enable/disable the gizmo of the active tool.

**Active Object** Enable/disable the gizmo for the active object.

## Object Gizmos

The Object Gizmos allows mouse controlled translation, rotation and scaling in the 3D Viewport. There is a separate gizmos for each operation. Each gizmo can be used separately or in combination with the others. You can use the gizmos by dragging one of the three colored axes with LMB. The transformation will be locked to the clicked axis.

Holding down **Shift** *after* you LMB the gizmo handle will constrain the action to smaller increments. Holding down **Shift** *before* you LMB click on one of the handles will cause the gizmo action to be performed relative to the other two axes. See *Plane Locking*.

**Orientation** The orientation to use for the gizmo. The orientations can be configured in the viewport orientation *Orientations menu*.

**Move** Show the gizmo to control the location. Dragging the small white circle allows free transformation.

**Rotate** Show the gizmo to control the rotation. When you hover your mouse over the gizmo a highlighted circle will appear, clicking this will activate *trackball rotation*.

**Scale** Show the gizmo to control the scaling.

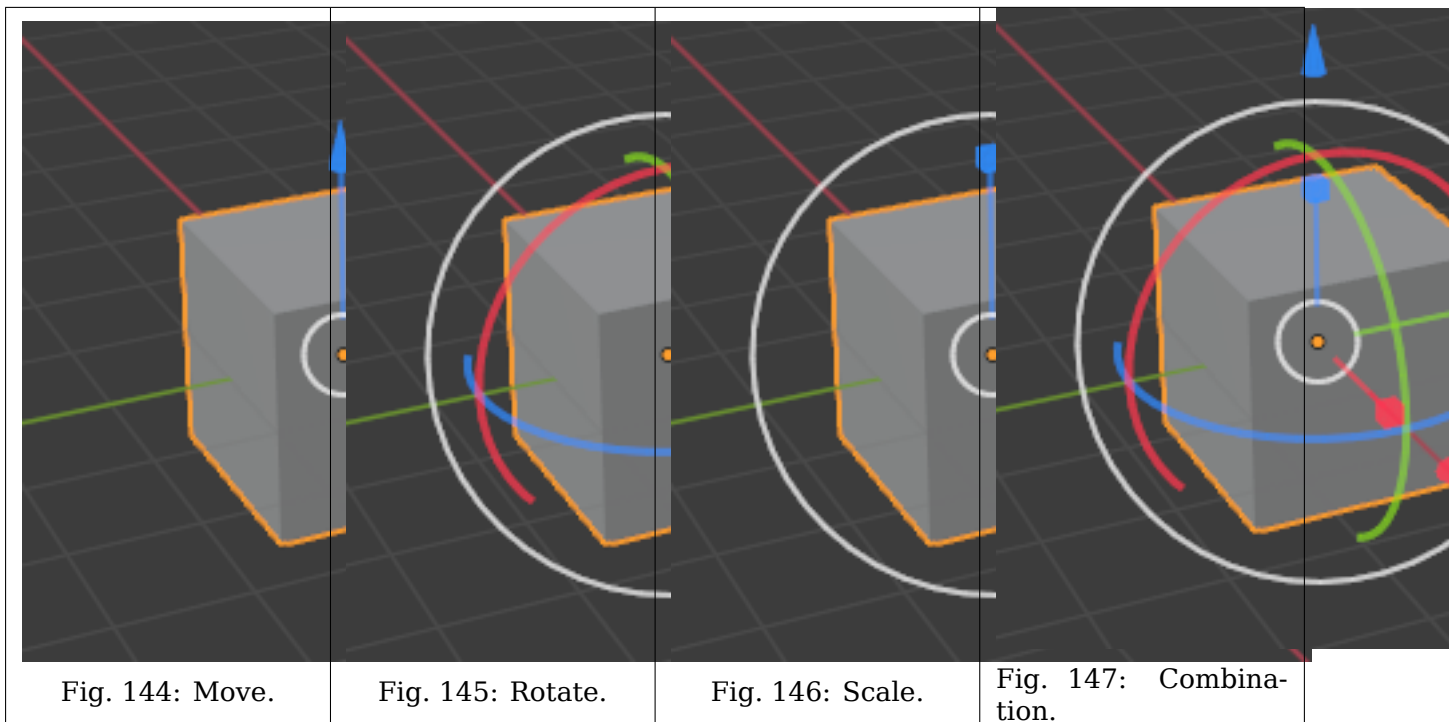


Fig. 144: Move.

Fig. 145: Rotate.

Fig. 146: Scale.

Fig. 147: Combination.

### See also:

The *Gizmo Preferences*.



## Empty

Gizmo settings for empties.

**Image** Show the gizmo to adjust the image size and position of empties.

**Force Field** Show the gizmo to adjust the force field.

## Light

Gizmo settings for lights.

**Size** Show the gizmo to adjust the size of lights.

**Look At** Show the gizmo to adjust the direction of the light.

## Camera

Gizmo settings for cameras.

**Lens** Show the gizmo to adjust the lens and orthographic size.

**Focus Distance** Show to gizmo to adjust the focus distance.

## Viewport Overlays

Using the Viewport Overlays pop-over settings for the overlays can be configured. There is a switch to turn off all overlays for the 3D Viewport.

The options that are visible in the pop-over depend on the mode that the 3D Viewport is in.

## Object Mode

The next options are always present, independent the current mode.

### Guides

**Grid** Show grid in orthographic side view.

**Floor** Show the ground plane.

**Axis** Show the X and/or Y and/or Z axis line.

**Scale** The distance between lines in the grid/floor.

**Subdivision** The number of subdivisions between grid lines.

**Text Info** Shows information such as the *View Perspective*, playback FPS (Frames Per Second), current frame number, and the name of the active *Collection* and Object.

**Statistics** Show information about the amount of objects and geometry.

**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.

**HDRI Preview** Show two spheres, one glossy and one diffuse, to preview HDRIs used in *Material Preview* and *Rendered* shading modes.

**3D Cursor** Show the 3D Cursor overlay.

**Annotations** Show the annotation overlay.



## Objects

**Extra** Show details of objects including empty wires, cameras and other visual guides.

**Relationship Lines** Show dashed lines indicating parents or constraint relationships.

**Outline Selected** Show an outline highlight around selected objects.

**Bones** Show Bones. Disable to only show their motion path.

**Motion Paths** Show the motion path overlay.

**Origin** Show the object origin of the active object.

**Origin (All)** Show the object origin of all objects.

## Geometry

**Wireframe** Displays the mesh's face edges, similar to *Wireframe Shading* but displays edges on top of existing shading. The value slider adjusts which edges to display by only displaying wires on prominent edges. Lower values hide edges with angles close to 180 degrees while a value of 1 shows all wires.

**Face Orientation** Show the face orientation overlay. In the face orientation overlay all faces where the face normal points towards the camera are colored blue. All faces where the face normal points away from the camera are colored red. With this overlay, it is easy to detect the orientation of the face normals.

**Fade Inactive Geometry** Fade inactive geometry using the viewport background color. The value slider controls the factor of the objects are blended with the background.

## Motion Tracking

Show the motion tracking overlay.

**Camera Path** Show the reconstruction camera path.

**Marker Names** Show the names for reconstructed track objects.

**Tracks** Change the display of the reconstructed tracks.

- Plain Axes
- Arrows
- Single Arrow
- Circle
- Cube
- Sphere
- Cone

**Size** Change the display size of the reconstructed tracks.

## Mesh Edit Mode

The next options are available when in Edit Mesh Mode.

**Edges** Highlighted selected and partially selected edges.

*Only affects vertex and face select mode (as edges are always highlighted in edge select mode).*

**Faces** Highlight faces using a face overlay that applies to both selected and unselected faces.

*Affects all selection modes.*

**Center** Show face center points in solid shading modes.

*Only affects face select 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.

**Edge Marks and Face Marks** Used by Freestyle.

## Shading

**Hidden Wire** Show only front-facing wireframes. This is useful for a retopology workflow.

---

**Tip:** Optimally this could be combined with the *X-Ray* display setting.

---

**Vertex Groups Weights** Display weights in Edit Mode.

**Zero Weights** To 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** Show in black vertices with no weights in the active group.

**All** The vertex is shown in black if it has zero weight in all groups.

## Mesh Analysis

Show the mesh analysis overlay.

See: *Mesh Analysis*.

## Measurement

Numerical measures of the selected elements on screen as part of the text info overlay. 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 *Global/Local*.

Use *Global* if you want the Object's scale to be applied to the measurements.

---

## Normals

- Display vertex normals
- Display face normals at vertices (split normals)
- Display face normals

**Size** The size to show the selected normals.

## Developer

**Indices** Display the indices of selected vertices, edges and faces.

## Freestyle

**Edge Marks** Display Freestyle edge marks, used with the Freestyle renderer.

**Face Marks** Display Freestyle face marks, used with the Freestyle renderer.

## Sculpt Mode

**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

**Stencil Mask Opacity** Opacity of the stencil mask overlay in Vertex Paint Mode.

**Show Wire** Use wireframe display in paint modes.

## Weight Paint

**Opacity** The opacity of the overlay.

**Zero Weights** To 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** Show in black vertices with no weights in the active group.

**All** The vertex is shown in black if it has zero weight in all groups.

**Show Weight Contours** Show contour lines formed by points with the same interpolated weight.

**Show Wire** Use wireframe display in paint modes.

## Texture Paint

**Stencil Mask Opacity** The opacity of the stencil mask overlay in Texture Paint Mode.

## Pose Mode

**Fade Geometry** Show the bones on top and face other geometry to the back. The opacity can be controlled with the slider.

## Grease Pencil

**Onion Skin** Show ghosts of the keyframes before and after the current frame.

**Canvas** Display a grid over Grease Pencil drawing plane. The opacity of the grid can be controlled with the slider. When using the *Canvas X-Ray* option objects are drawn behind the canvas grid.

**Fade 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 Objects** Cover all 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** Show edit lines when editing strokes.

**Only in Multiframe** Show edit lines only when using multiframe edition.

**Stroke Direction** Toggles the display of the strokes start point (green) and end point (red) to visualize the line direction.

**Material Name** Show material name next to the linked stroke.

**Vertex Opacity** Opacity for edit vertices (points).

**Vertex Paint Opacity** The opacity of the overlay.

## Viewport Shading

The shading of the 3D Viewport can be adjusted to match the task at hand. There are several modes to choose from.

---

**Note:** The Material Preview option is not available when the render engine of the scene is set to Workbench.

---

## Wireframe

Shows the full scene by only displaying the edges of the objects (wireframes).

### Color

**Single** Render the whole scene using a single color.

**Object** Use the color that can be set per object in the Viewport Display *Object* panel.

**Random** A random color will be selected for every object in the scene.

**Background** How the background is displayed in the 3D Viewport.

**Theme** Use the background of the theme.

**World** Use the world viewport display options.

**Viewport** Select a custom color for the background of the 3D Viewport.

### Options

**X-Ray** Render the scene transparent. With the slider you can control how transparent the scene should appear. In wireframe mode the opacity of the back wires can be adjusted.

**Outline** Render the outline of objects in the viewport. The color of the outline can be adjusted.

## Solid

Show the scene using in solid mode. This mode utilized the Workbench engine to render the 3D Viewport. The *lighting*, *color* and *options* can be found at Workbench render engine section.

**Background** The way the background is displayed in the 3D Viewport.

**Theme** Use the background of the theme.

**World** Use the world viewport display options.

**Viewport** Select a custom color for the background of the 3D Viewport.

## Material Preview

Render the 3D Viewport with *Eevee* and a HDRI environment. This mode is particularly suited for previewing materials and texture painting. You can select different lighting conditions to test your materials.

### Lighting

**Scene Lights** Use the lights in the scene when rendering the scene.

**Scene World** Use the world of the scene when rendering the scene. When turned off a world will be constructed with the next 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. Disabling this setting makes the lighting changes as the view is change allowing materials to be used as matCaps for sculpting and painting.

**Strength** Light intensity of the environment.

**World Opacity** The opacity level of the HDRI will be rendered as background in the 3D Viewport.

**Blur** Factor to unfocus the *HDRI Environment*, note that this does not change the diffusion of the lighting, only the appearance of the colors.

**Render Pass** Instead of the combined render, show another render pass. Useful to analyze and debug geometry, materials and lighting.

## Rendered

Render the 3D Viewport with the scene *Render Engine*, for interactive rendering. By default the scene lights are used for lighting. An HDRI environment can be used as well, with the same options as Material Preview mode.

**Render Pass** Instead of the combined render, show another render pass. Useful to analyze and debug geometry, materials and lighting.

## Toolbar

### 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.

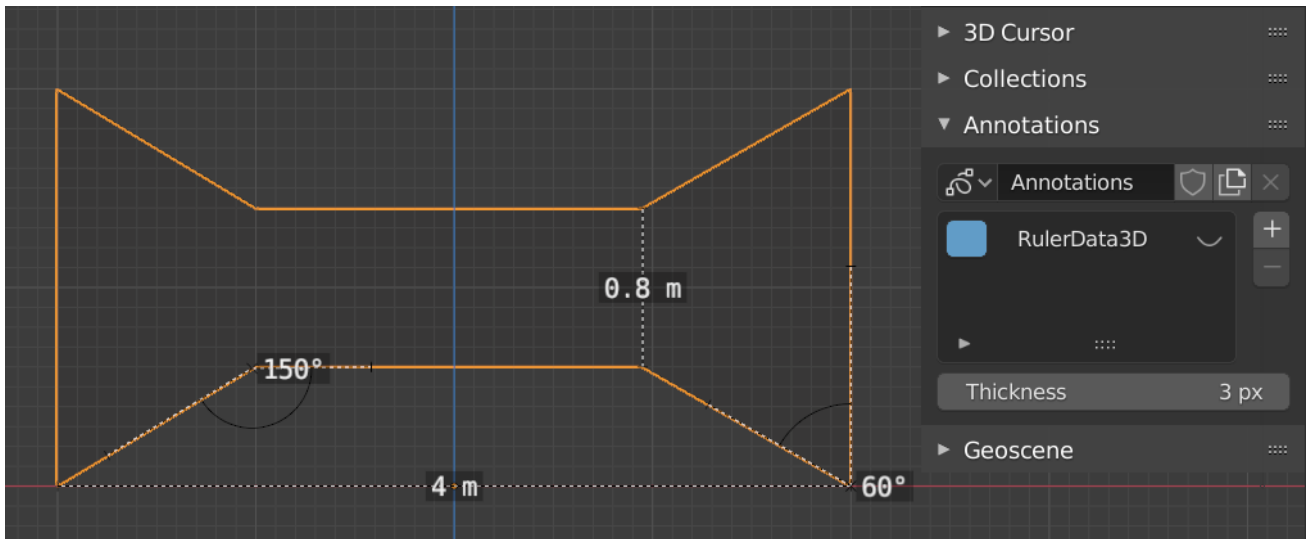


Fig. 148: Examples of the Measure tool.

## 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. Click on either end of the ruler to select it and move the endpoints.
  - Holding **Ctrl** while moving enables snap to edges and vertices. A small circle appears when the end point is snapped to a vertex or edge. This way you can place the endpoints more accurately.
  - Holding **Shift** while moving lets you measure the distance between faces. This works well only 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. Move this midpoint to set the vertex of the angle. Holding down **Ctrl** enables snap to edges and vertices. Move the endpoints to change the angle size.
5. A selected ruler can be deleted with **Delete** or **X**. To delete all measurements, make the *Sidebar* → *View* → *Annotations* panel visible. Delete the “RulerData3D” layer (see image above).
6. You can copy the measurement value to the clipboard with **Ctrl-C**.

All measurements are hidden when another tool is selected. They are shown when the *Measure* tool is selected again. Yet you can do editing operations while the ruler is active. For example, you can edit the rotation or dimension of the selected object in the *Sidebar*. The measurement values do not appear in the Render output.

Unit settings and scale from the scene are used for displaying dimensions. Changing the units 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* setting in the *Viewport Overlays* popover. Edge length, edge angle, face area and face angle can be displayed through this setting.

---

The Toolbar contains a list of tools. Links to each modes 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*

## **Sidebar**

### **Item**

Shows *Transform* settings of the active objects.

### **Tool**

Show settings of the active tool and Workspace.

## View

### View Panel

The *View* panel lets you set other settings regarding the 3D Viewport. You can show it with the *View* → *View Properties...* menu entry.

**Focal Length** Control the focal length of the 3D Viewport camera in millimeters, unlike a *rendering camera*.

**Clip Start/End** Adjust the minimum and maximum distances range to limit the visible range to the area between two planes that are orthogonal to the viewing direction of the viewport camera. Objects outside the range will not be shown.

---

**Note:** The definition of the two planes depends on the kind of view:

- Perspective view: The planes with distance of start and end from viewport camera.
  - Orthographic view: The planes with distance of negative end and positive end from the focus point, in this case the *Start* is ignored.
- 

**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** Use a local camera in this view, selected from the object selector, rather than the scene's (global) active camera.

**Render Region** Use a Render Region when not looking through a camera. Using `Ctrl-B` to draw a region will automatically enable this option.

### View Lock

**Lock to Object** A *Data ID* that defines an object as the center of the view. In this case, the view can be rotated around or zoomed towards that central object, but not while you move the object itself (this option is not available in a camera view).

### Lock

**To 3D Cursor** Lock the center of the view to the position of the 3D cursor. It is only available when *Lock to Object* is not active.

**Camera to View** When in camera view, all changes in the view (pans, rotations, zooms) will affect the active camera. The camera frame will be outlined with a red dashed line.

---

**Hint:** This will move the camera's parent which can be useful with camera rigs. Use the *Camera Parent Lock* preference to change this behavior.

---

### 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 the visibility of collections in the viewport. If a collection contains objects, there is a circle to the left of the collection name. If a collection is empty, there is no circle to the left of the collection name.

**Local Collections** Allows the list of visible collections to be controlled per viewport rather than globally.

**Hide in Viewport (eye icon)** Collections can be hidden in the viewport by clicking on the eye icon.

By clicking directly on the collection names, it “isolates” the collection by hiding all other collections, and showing the direct parents and all the children of the selected collection.

### See also:

Read more about *Collections*.

## Annotations

See *Annotations* for more information.

## Viewport Render

Viewport rendering uses the 3D Viewport rendering for quick *preview* renders.

This allows you to inspect your animatic (for object movements, alternate angles, etc.).

This can also be used to preview your animations - in the event your scene is too complex for your system to play back in real-time in the 3D Viewport.

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.

Table 12: Model by © 2016 pokedstudio.com



**Tip:** Disable overlays to render the viewport without any additional overlays.

While this option is not specific to Viewport rendering, it’s often useful to enable, since it removes data such as rigs and empties that can be a distraction.

## Settings

For the most part, *Viewport Render* uses the current viewport settings. Some settings are located in the render panel 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.

Sampling and Alpha Transparency Mode options can be set in *Properties* → *Render* → *Sampling*. Make sure the Workbench or Eevee render engine is selected to see the appropriate values.

Additionally, some render settings are used too:

- Render Dimensions
- Render Aspect
- File Format & Output (file path, format, compression settings, etc.)

## Rendering

Activating *Viewport Render* will render from the current active view. This means that if you are not in an active camera view then 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*.

---

## 2.3.2 Image Editor

### Introduction

The Image Editor is where you can view/edit 2D assets like images or textures.

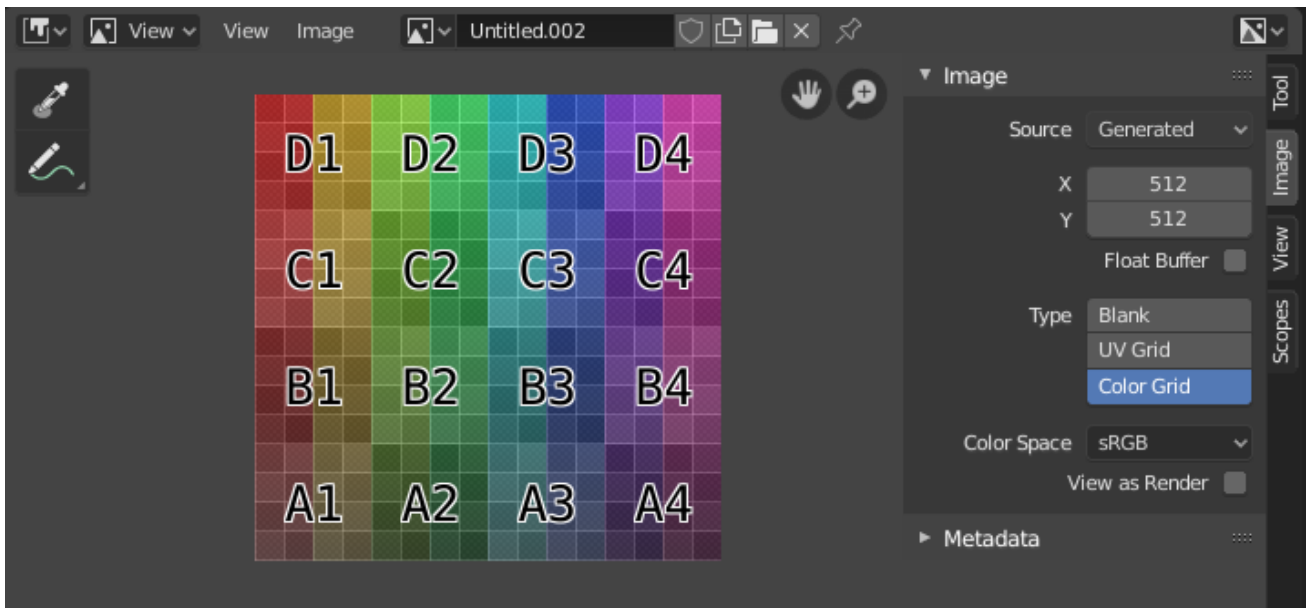


Fig. 152: Image Editor with a test grid texture.

## Toolbar

**Sample Tool** Used to sample a pixel's color from anywhere within Blender.

**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. See *Editing*.

**Image** A *data-block menu* used for selecting images. When an image has been loaded or created in the Image editor, the Image panel appears in the *Sidebar region*. See *Image Settings*.

- Render Result
- Viewer Node

**Image Pin** *Todo*.

**Slot** You can save successive renders into the render buffer by selecting a new slot before rendering. If an image has been rendered to a slot, it can be viewed by selecting that slot. Empty slots appear as blank grids in the Image editor. Use the J and Alt-J to cycle forwards and backwards through saved renders. Alternatively you can use the number keys 1, 2, 3, etc, to select the slot with the corresponding number. A slot can be renamed by double clicking its name in the Image panel in the Sidebar.

**View Layer** If you are using *View Layers*, use this menu to select which layer is displayed.

**Render Pass** If you are using *Render Passes*, use this menu to select which pass is displayed.

**Display Channels** Select what color channels are displayed.

**Color and Alpha** Replaces transparent pixels with background checkerboard, denoting the alpha channel.

**Color** Display the colored image, without alpha channel.

**Alpha** Displays the Alpha channel a grayscale image. White areas are opaque, black areas have an alpha of 0.

**Z-Buffer** Display 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.

## Main View

When LMB / RMB dragging mouse the color under the cursor is shown in the footer as well the cursor position and the color values in the RGBA, HSV and Luminance *Color Space*.

## Navigating

Panning can be done by clicking the MMB and dragging.

Zooming can be done by scrolling Wheel up or down. Also, as in the 3D Viewport, you can use NumpadPlus or NumpadMinus to zoom.

## 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

**Region Controls** Adjust which regions are visible in the Image editor.

**Update Automatically** Update the view in multiple areas.

**Show Metadata** Displays the metadata if they were set in the render tab's *Metadata* panel.

**Display Texture Paint UVs** Toggles UVs in Paint Mode.

**Zoom In/Out Wheel** Adjusts the zoom level.

### Fractional Zoom

- Zoom 1:8 Numpad8
- Zoom 1:4 Numpad4
- Zoom 1:2 Numpad2
- Zoom 1:1 Numpad1
- Zoom 2:1 Shift-Numpad2
- Zoom 4:1 Shift-Numpad4
- Zoom 8:1 Shift-Numpad8

**Frame All Home** Center the view to the entire image.

**Frame All Fit Shift-Home** Fit the view to the image dimensions.

**Center View to Cursor** When the 2D cursor is visible, move the view so that it is at the center of the editor.

**Render Region Ctrl-B** See *Render Region*.

**Clear Render Region Ctrl-Alt-B** See *Render Region*.

**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 Tab

#### Display

You can set the editors display options in this panel.

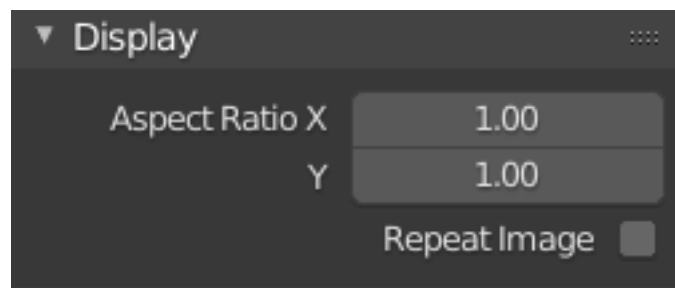


Fig. 153: Display panel.

**Aspect Ratio** Display Aspect for this image. Does not affect rendering.

**Repeat Image** Duplicate the image until it is repeated to fill the main view.

## Annotations

Options for the annotation tool. See *Annotate Tool*.

## Scopes

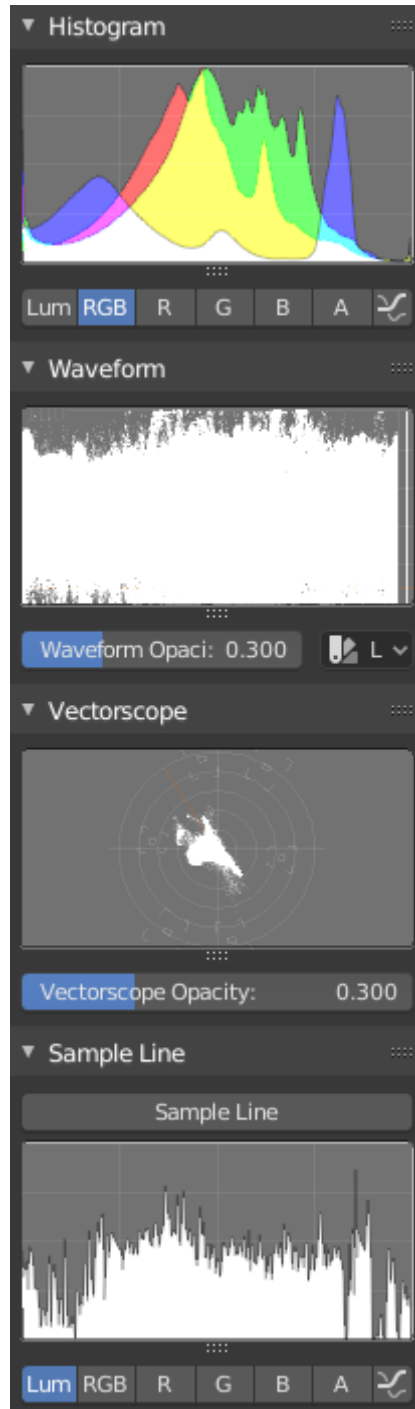


Fig. 154: Scopes in the Image editor.

### Histogram

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information 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.

**Luma** Shows the luminosity of an image.

**RGB** Shows the RGB (Red, Green, Blue) channels stacked on top of each other.

**R/G/B/A** Depending on the channel you choose the scope will show the appropriate channel.

**Show Line** Displays lines rather than filled shapes.

## Waveform

**Waveform Opacity** Opacity of the points.

### Waveform Mode

**Luma** ToDo.

**YCbCr** ToDo.

**Parade** The RGB channels are shown side-by-side.

**Red Green Blue** Shows the RGB channels overlaid as a “Full color” waveform. It is useful for color grading.

## Vectorscope

**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 use to read the sample data from.

## Scope Samples

**Full Sample** Sample every pixel.

**Accuracy** Proportion of original image source pixel lines to sample.

## 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 *UDIMs* tile information. The options that are visible in the pop-over depend on the Image Editor mode.

## Geometry

**Display Texture Paint UVs** Display overlay of texture paint UV layer.

## Image

**Show Metadata** Displays the metadata if they were set in the render tab’s *Metadata* panel.

## Image Settings

### Image Panel

**Image** Data-block menu.

**New +** The *New Image* button opens a pop-up to configure a *Generated* image.

### Source

See about supported *Supported Graphics Formats*.

### Single Image

Still image or a single frame.

### Image Sequence

Each frame is stored in a separate file. How to *Opening an Image Sequence*.

**Frame** A label showing the current frame.

**Further options** See *Movie* below.

### Movie

Frames packed into a container.

**Deinterlace** Removes fields in a video file. For example, if it is an analog video and it has even or odd interlacing fields.

### Frame

**Frames** Sets the range of frames to use.

**Start** Global starting frame of the sequence, when the playback should start. This is a global setting which means it affects all clip users such as the Movie Clip editor itself, motion tracking constraints and Compositor nodes.

**Offset** Offsets the first frame of the clip. It adds an extra offset to the frame number when converting a scene frame to the frame number in the file name. This option does not affect tracking data or any other associated data.

**Match Movie Length** This button sets the movies *frames* to the length of the selected movie.

**Auto Refresh** Automatically refresh images on frame changes.

**Cyclic** Start over and repeats after the last frame to create a continuous loop.

### Generated

Image generated in Blender.



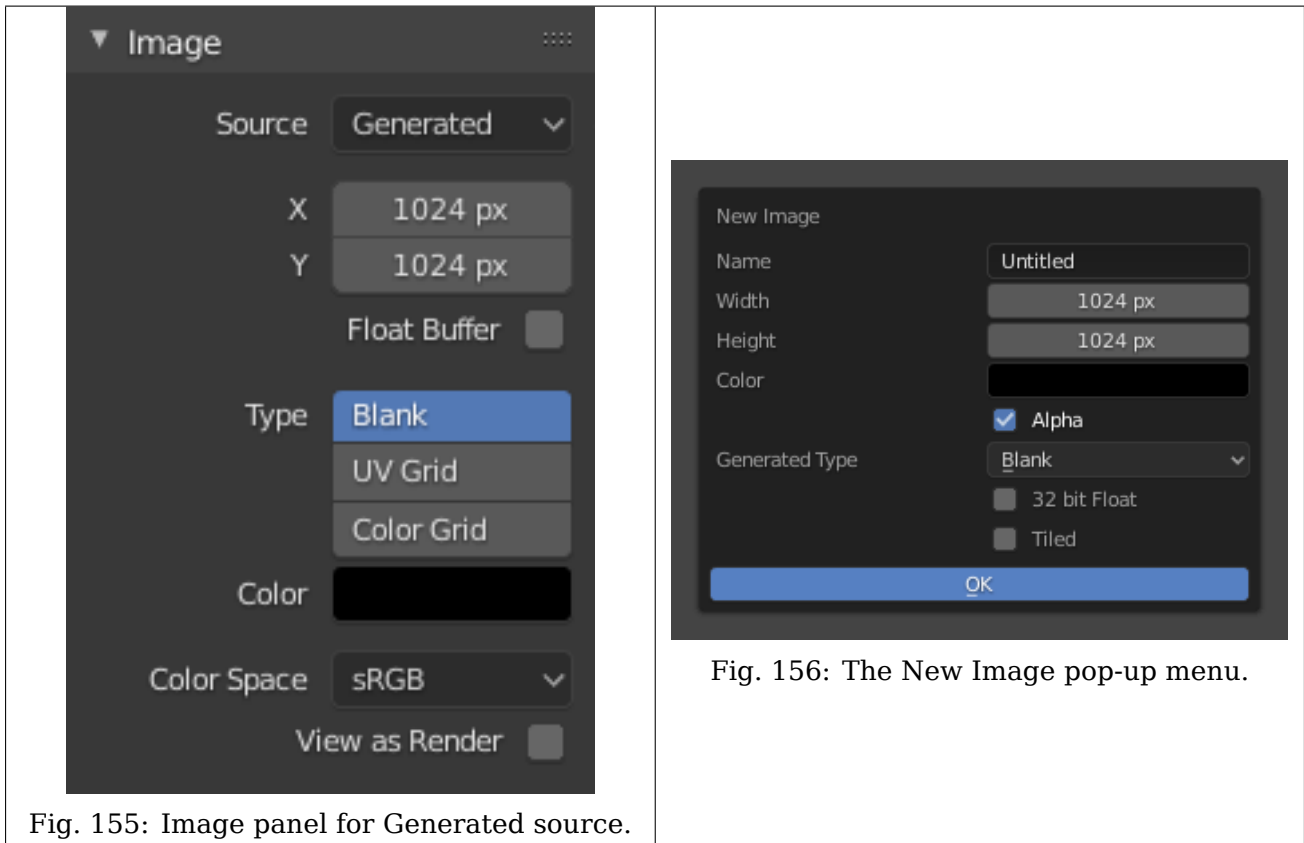


Fig. 155: Image panel for Generated source.

Fig. 156: The New Image pop-up menu.

**Width, Height** The size of image in pixels.

**Color** Sets the fill color if creating a blank 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 in the grid. It could be used for testing how the UVs have been mapped and to reduce stretching or distortion.

**32 bit Float** Creates a 32 bit image. This is 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.

**Tiled** Creates an image with support for *UDIMs*. This option creates the first 1001 tile; more tiles can be added later in the *UDIM Tiles* panel.

### Common Options

**File** Use for replacing or packing files.

**Pack** Embed the resource into the current blend-file.

**Path** Path to the linked file.

**Open** Opens the *File Browser* to select a file from a drive.

**Reload** Reloads the file. Useful when a file has been reworked in an external application.

**Color Space** *Color Space*.

**sRGB** Standard RGB display space.

**Linear** Linear 709 (full range). Blender native linear space.

**Linear ACES** ACES linear space.

**XYZ** Standard linear XYZ space.

**Non-Color** Color space used for images which contains non-color data (e.g. normal maps).

**Raw** Same as Non-Color.

**Filmic Log** Intermediate log color space of Filmic view transform.

**Half Float Precision** Load the image as having only a *Bit Depth* of 16 bit per channel instead of 32 bit which saves memory.

**View as Render** Applies *color transform* when displaying this image on the screen.

**Use Multi-View** See *Multi-View*.

**Alpha** Representation of alpha in the image file, to convert to and from when saving and loading the image. See *Alpha Channel*.

**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.

## Editing

---

### Reference

**Mode** All Modes

**Menu** *Image*

---

**New** Creates a new *Generated* Image.

**Open** Load image from a file.

**Open Cache Render** Load the current scene's render layers from disk cache, if available. This can be used to save RAM while rendering because the render layers do not have to be saved in RAM. This can also be used to recover some information from a fail render. For this to work, *Save Buffers* must be enabled.

**Replace** Replaces the current image throughout the blend-file with another image.

**Reload** Reload the image from the file on drive.

**Edit Externally** Using the *Edit Externally* tool Blender will open an external image editor, as specified in the *Preferences* and load in the image to be edited.

**Save** Save the image, if the image is already a file Alt-S.

**Save As** Save the (rendered) image in a separate file Shift-Alt-S or you want to save it under a different name.

**Save a Copy** Using *Save as Copy* will save the file to a specified name, but will keep the old one open in the Image editor.

**Save All Images** Save all modified images. Packed images will be repacked.

## Invert

**Invert Image Colors** Invert the colors of an image.

**Invert Channel** Red, Green, Blue, Alpha

**Resize** Adjust the image size in pixels.

**Pack** Packs the image into the blend-file. See *Packed Data*.

**Unpack** Unpack the image to a drive.

**Extract Palette** Extracts a *Color Palette* from the image for use by other tools.

**Generate Grease Pencil** Creates a *Grease Pencil* object using the currently selected image as a source.

---

**Important:** Rendered images are not automatically saved, they have to be saved to drive manually.

---

## 2.3.3 UV Editor

### Introduction

The UV Editor is used to map 2D assets like images/textures onto 3D objects and edit what are called UVs.

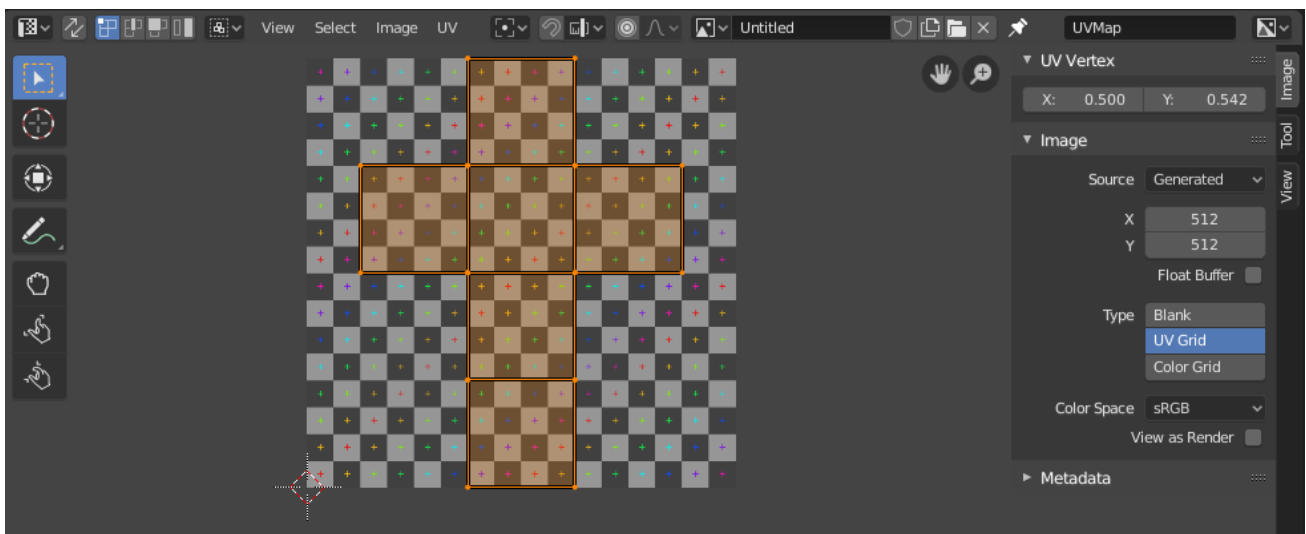


Fig. 157: UV Editor with a UV map and a test grid texture.

The most flexible way of mapping a 2D texture over a 3D object is a process called “UV mapping”. In this process, you take your three-dimensional (X, Y & Z) mesh and unwrap it to a flat two-dimensional (X & Y ... or rather, as we shall soon see, “U & V”) image. Colors in the image are thus mapped to your mesh, and show up as the color of the faces of the mesh. Use UV texturing to provide realism to your objects that procedural materials and textures cannot do, and better details than Vertex Painting can provide.

### UVs Explained

The best analogy to understanding UV mapping is cutting up a cardboard box. The box is a three-dimensional (3D) object, just like the mesh cube you add to your scene.

If you were to take a pair of scissors and cut a seam or fold of the box, you would be able to lay it flat on a tabletop. As you are looking down at the box on the table, we could say that U is the left-right direction, and V is the up-down direction. This image is thus in two dimensions (2D). We use U and V to refer to these “texture-space coordinates” instead of the normal X and Y, which are always used (along with Z) to refer to the three-dimensional space (3D).

When the box is reassembled, a certain UV location on the paper is transferred to an (X, Y, Z) location on the box. This is what the computer does with a 2D image in wrapping it around a 3D object.

During the UV unwrapping process, you tell Blender exactly how to map the faces of your object (in this case, a box) to a flat image in the UV Editor. You have complete freedom in how to do this. (Continuing our previous example, imagine that, having initially laid the box flat on the tabletop, you now cut it into smaller pieces, somehow stretch and/or shrink those pieces, and then arrange them in some way upon a photograph that is also lying on that tabletop.)

### Example

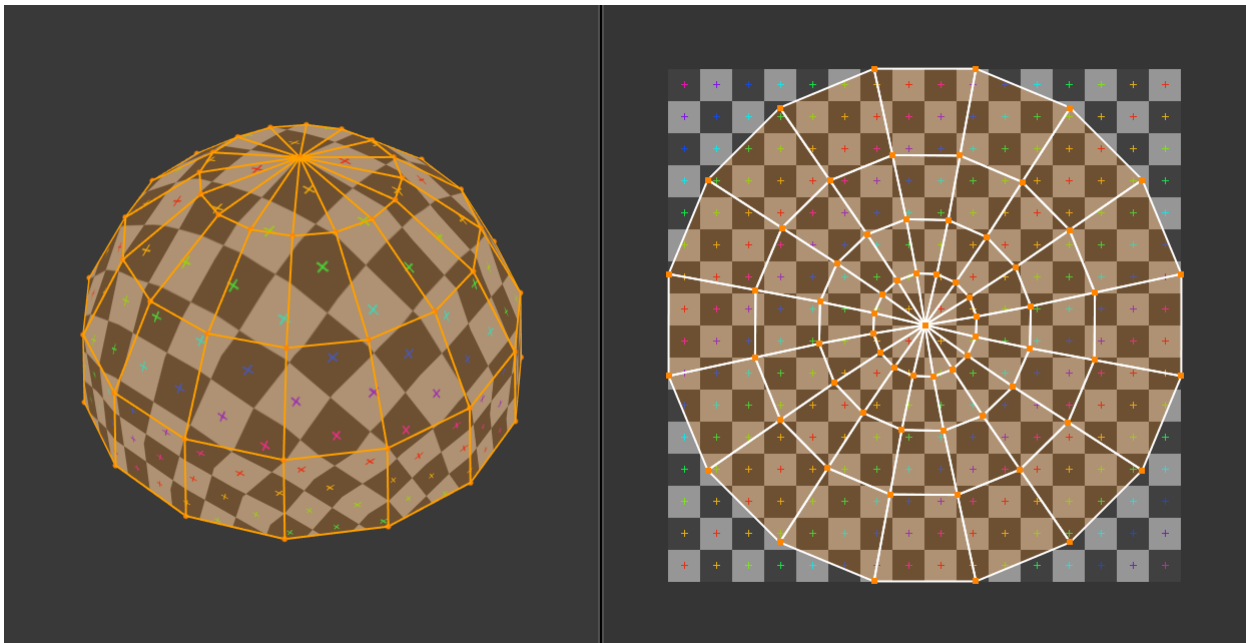


Fig. 158: 3D space (XYZ) versus UV space.

In this image you can easily see that the shape and size of the marked face in 3D space is different in UV space. This difference is caused by the “stretching” (technically called mapping) of the 3D part (XYZ) onto a 2D plane (i.e. the UV map).

If a 3D object has a UV map, then, in addition to the 3D coordinates X, Y, and Z, each point on the object will have corresponding U and V coordinates.

---

**Note:** On more complex models (like seen in the sphere above) there pops up an issue where the faces cannot be cut, but instead they are stretched in order to make them flat. This helps making easier UV maps, but sometimes adds distortion to the final mapped texture.

---

### Advantages of UVs

While procedural textures are useful – they never repeat themselves and always “fit” 3D objects – they are not sufficient for more complex or natural objects. For instance, the skin on a human



## Tool Settings

**Pivot** Similar to working with pivot points in the 3D Viewport.

**UV Snapping** Similar to Snapping in the 3D Viewport.

**Proportional Editing** See *Proportional Editing*.

## Navigating

The UV Editor has a 2D cursor. Its position can be changed by LMB clicking in the UV editor while the cursor tool is active. You can also manually adjust its position in the Sidebar region. The range by default is from 0.0 to 1.0 starting from the lower left corner. By enabling *Pixel Coordinates*, the coordinates match the pixels in your image with XY(0, 0) located in the lower left corner.

## 2D Viewport

Panning can be done by clicking the MMB and dragging.

Zooming can be done by scrolling Wheel up or down. Also, as in the 3D Viewport, you can use NumpadPlus or NumpadMinus to zoom.

## View Menu

Also see *Navigating* in the Image editor.

**Frame Selected NumpadPeriod** Change view so that all selected UV vertices are visible.

## 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 UV Editor. This option also toggles the visibility of *UDIMs* tile information. The options that are visible in the pop-over depend on the UV Editor mode.

## UV Editing

**Display Stretch** Shows how much of a difference there is between UV coordinates and 3D coordinates. Blue means low distortion, while Red means high distortion. Choose to display the distortion of *Angles* or the *Area*.

## Geometry

**UV Opacity** Opacity of the above UV overlays.

**Display As** Controls how edges are shown.

**Outline** Display white edges with black outline.

**Dash** Display dashed black-white edges.

**Black** Display black edges.

**White** Display white edges.

**Modified Edges** Show results of modifiers in the UV display.

**Faces** Display faces over the image.

### **Image**

**Show Metadata** Displays the metadata if they were set in the render tab's *Metadata* panel.

### **Sidebar**

#### **Image Tab**

#### **UV Vertex**

Transform Properties *Selecting UVs.*

### **Image**

See *Image Settings.*

### **UDIM Grid**

Allows you to control the grid size of *UDIM* tiles.

### **Tool Tab**

Shows the settings for the active tool.

### **View Tab**

#### **Display**

You can set the editors display options in this panel.

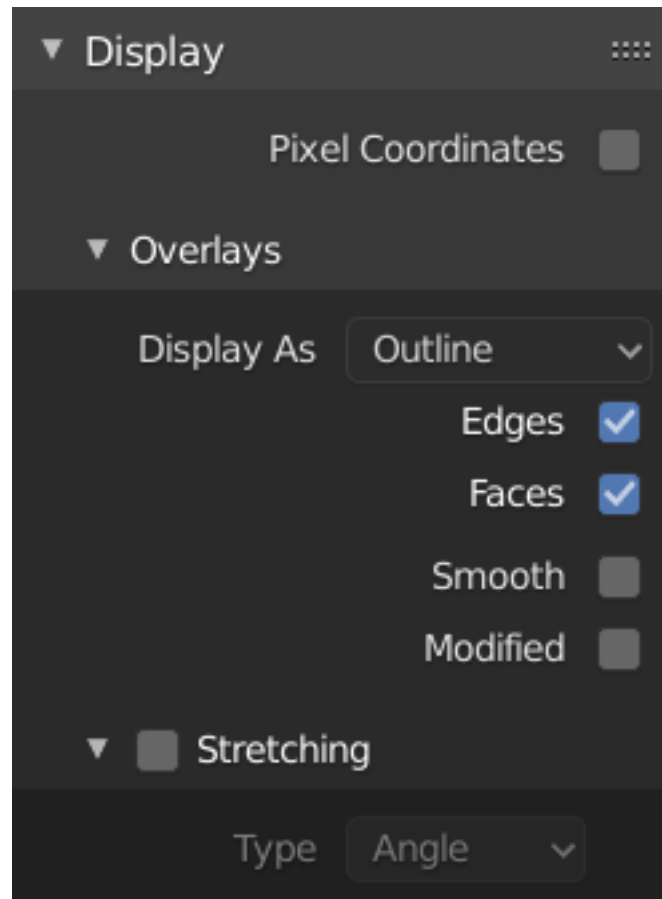


Fig. 160: Display panel: With both an image and UVs selected.

**Aspect Ratio** Display Aspect for this image. Does not affect rendering.

**Repeat Image** Duplicate the image until it is repeated to fill the main view.

**Pixel Coordinates** Display UV coordinates in pixels rather than from 0.0 to 1.0

### 2D Cursor

**Location X, Y** Control 2D cursor location.

### Annotations

Options for the *annotation tool*.



## Scopes

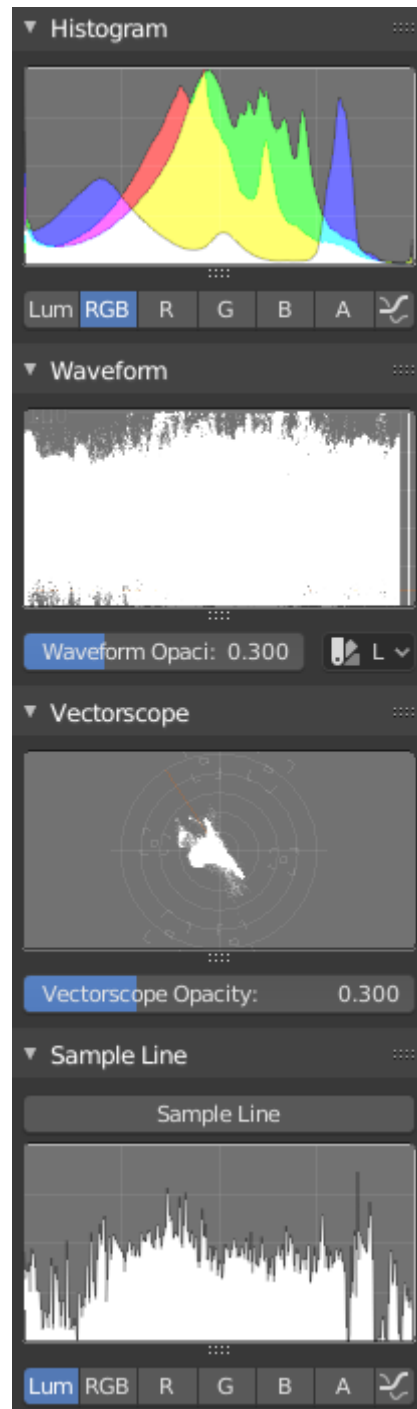


Fig. 161: Scopes in the Image editor.

### Histogram

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information 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.

**Luma** Shows the luminosity of an image.

**RGB** Shows the RGB channels stacked on top of each other.

**R/G/B/A** Depending on the channel you choose the scope will show the appropriate channel.

**Show Line** Displays lines rather than filled shapes.

## Waveform

**Waveform Opacity** Opacity of the points.

### Waveform Mode

**Luma** ToDo.

**YCbCr** ToDo.

**Parade** The RGB channels are shown side-by-side.

**Red Green Blue** Shows the RGB channels overlaid as a “Full color” waveform. It is useful for color grading.

## Vectorscope

**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 use to read the sample data from.

## Scope Samples

**Full Sample** Sample every pixel.

**Accuracy** Proportion of original image source pixel lines to sample.

## Selecting

Selection tools are available in the *Select Menu* in the header, and the shortcuts listed below.

## Sync Selection

Turning on the *Sync Selection* button causes selection of components in the 3D Viewport to sync with their corresponding elements in the UV editor. If off only the selected faces are displayed in the UV editor. These two modes have very different results when transforming components in the UV editor.

### See also:

*Selecting in the 3D Viewport.*

## Selection Modes

Select Modes dependent on the Sync Selection.

### Sync Selection Off

**Vertex** Select individual vertices.

**Edge** Select edges.

**Face** Select faces.

**Island** Select contiguous groups of faces.

### Sync Selection On

When selecting UVs or Edges, it behaves like the *Shared Vertex* option of the *Sticky Selection Mode*. When selecting Faces, it behaves like the *Disabled* option of the *Sticky Selection Mode*.

**Vertex** Select individual vertices.

**Edge** Select edges.

**Face** Select faces.

### Sticky Selection Mode

This selector lets you enable automatic additional selection.

**Shared Vertex** Selects UVs that share a mesh vertex, even if they are in different UV locations.

**Shared Location** Selects UVs that are in the same UV location and share a mesh vertex.

**Disabled** Disables Sticky Selection. When you move a UV in this mode, each face owns its own UVs, allowing them to be separated.

## Menu

**Box Select B** Click and drag to box select UV coordinates. Alternatively, use B to start *box selection*.

**Box Select Pinned Ctrl-B** Use the box lasso to select only pinned UV coordinates.

**Circle Select** See *Circle Select*.

**Select All A** Selects all UV coordinates.

**Select None Alt-A** Deselects all UV coordinate.

**Inverse Ctrl-I** Inverts the current selection.

**More/Less Ctrl-NumpadPlus, Ctrl-NumpadMinus** Expands/Contracts the selection to/from the adjacent elements of the selection type.

**Select Pinned Shift-P** Selects all *pinned* UVs.

### Select Linked

**Linked Ctrl-L** This operator selects all UVs that are connected to currently selected UVs. This works similarly to the tools in 3D Viewport.

**Shortest Path** Path between two selected elements.

**Select Split Y** Cuts apart the selected UVs from the map. Only those UVs which belong to fully selected faces remain selected. As the name implies, this is particularly useful to unlink faces and move them elsewhere. The hotkey is analogous to the mesh Split tool.

**Select Overlap** Selects any UVs that are extended over other UVs while also selecting any underlying UVs.

## Shortest Path

---

### Reference

**Mode** Edit Mode

**Menu** *Select* → *Select Linked* → *Shortest Path*

**Hotkey** Ctrl-LMB

---

Selects all UV components along the shortest path from the active component to the one which was selected.

**Face Stepping** Supports diagonal paths for vertices and faces, and selects edge rings with edges.

**Topological 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.

**See also:**

Mesh edit *Select Shortest Path*.

## Select Edge Loops

---

### Reference

**Mode** Edit Mode

**Hotkey** Alt-LMB, or Shift-Alt-LMB for modifying existing selection.

---

Holding Alt while selecting a UV component 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.

**See also:**

Mesh edit *Select Edge Loops*.

## 2.3.4 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*.

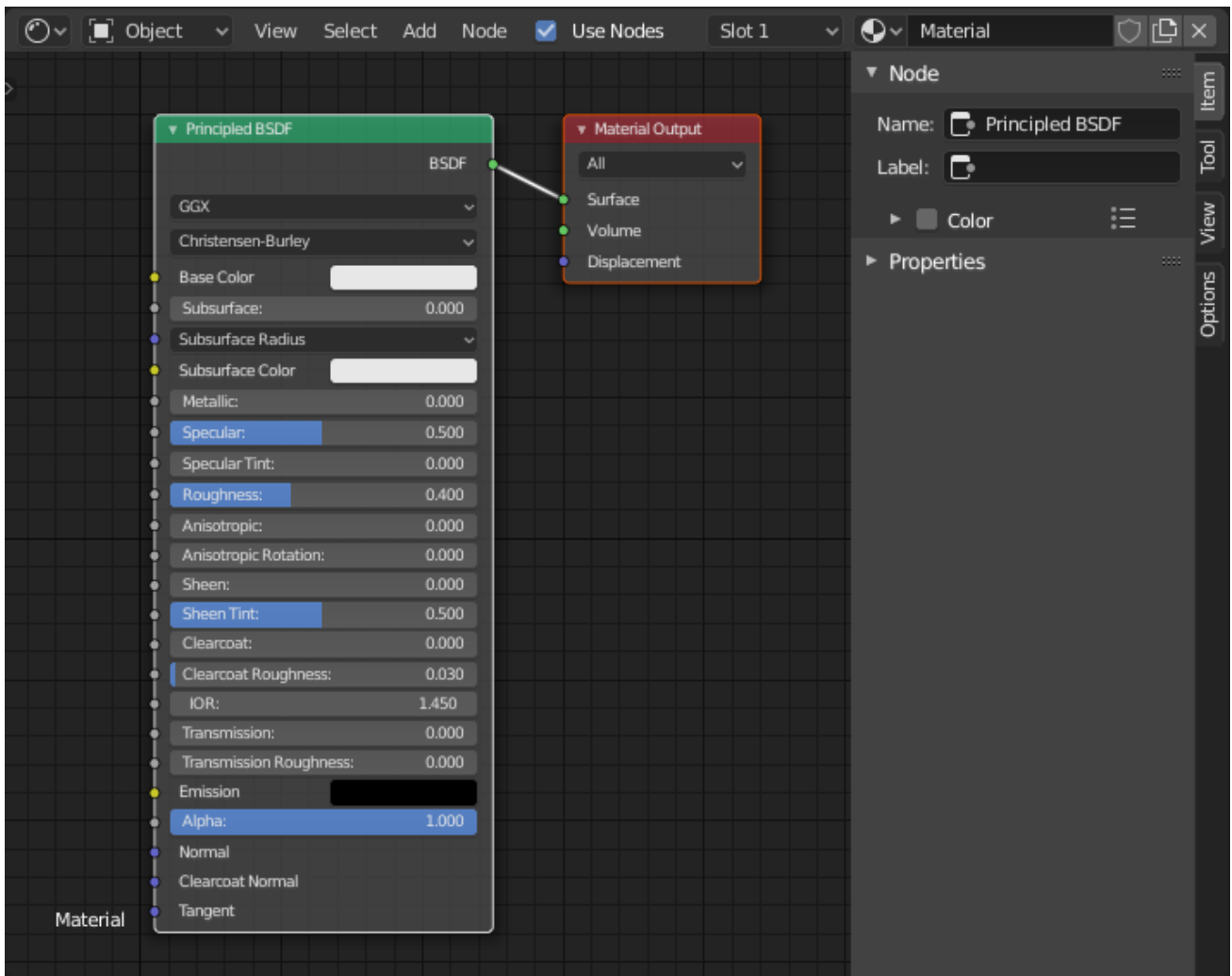


Fig. 162: Shader Editor with the default material node tree.

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.

## 2.3.5 Compositing

### 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.

### 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 Panel*.

---

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).

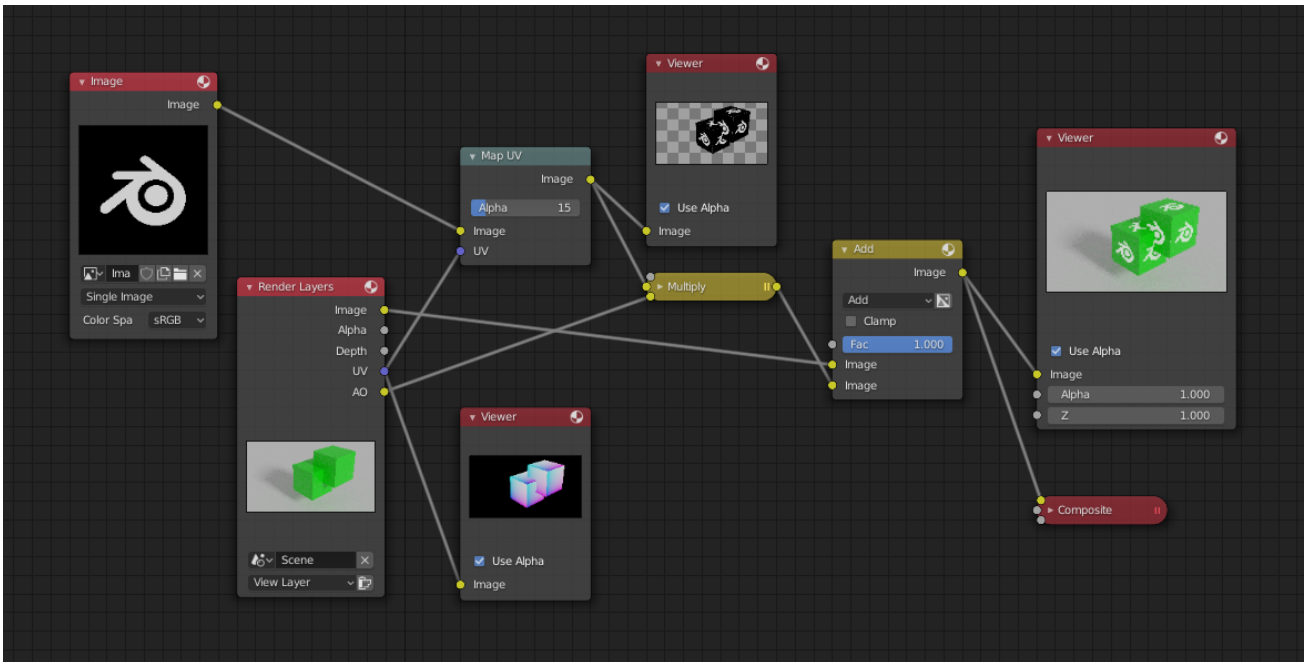


Fig. 163: An example of a composition.



Fig. 164: An example of color correction.

---

**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 settings in Properties *Render* → *Dimensions*. 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 *Image* → *Save Image* or Alt-S. 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

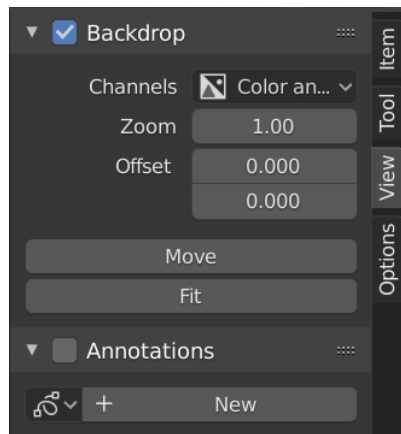


Fig. 165: Backdrop panel.

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.

**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

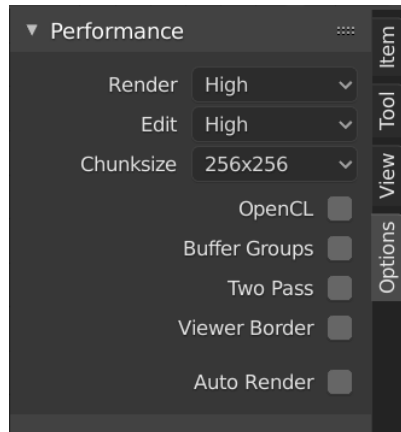


Fig. 166: Performance panel.

This panel helps you tweak the performance of the Compositor.

**Render** Sets the quality when doing the final render.

**Edit** Sets the quality when making edits.

**Chunk Size** Max size of a tile (smaller values give a better distribution of multiple threads, but more overhead).

**OpenCL** This allows the use of an OpenCL platform to aid in rendering. Generally, this should be enabled unless your hardware does not have good OpenCL support.

**Buffer Groups** Enables buffering of group nodes to increase the speed at the cost of more memory.

**Two Pass** Use two pass execution during editing: the first pass calculates fast nodes, the second pass calculates all nodes.

**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.

**Auto Render** Re-render and composite changed layer when edits to the 3D scene are made.

## Node Types

### 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.

## Bokeh Image Node

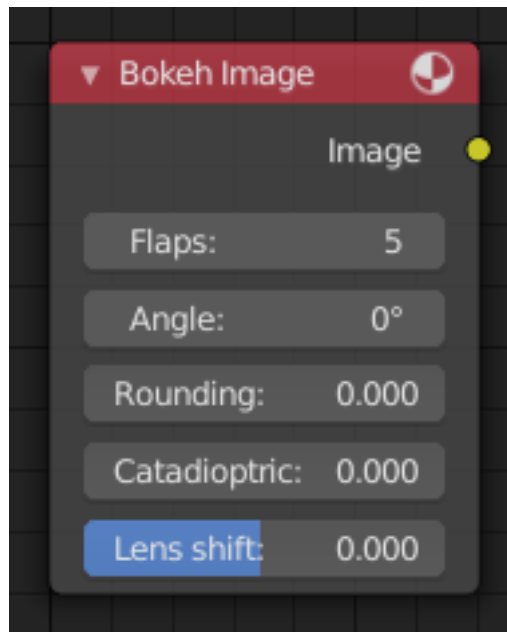


Fig. 167: Bokeh Image Node.

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.

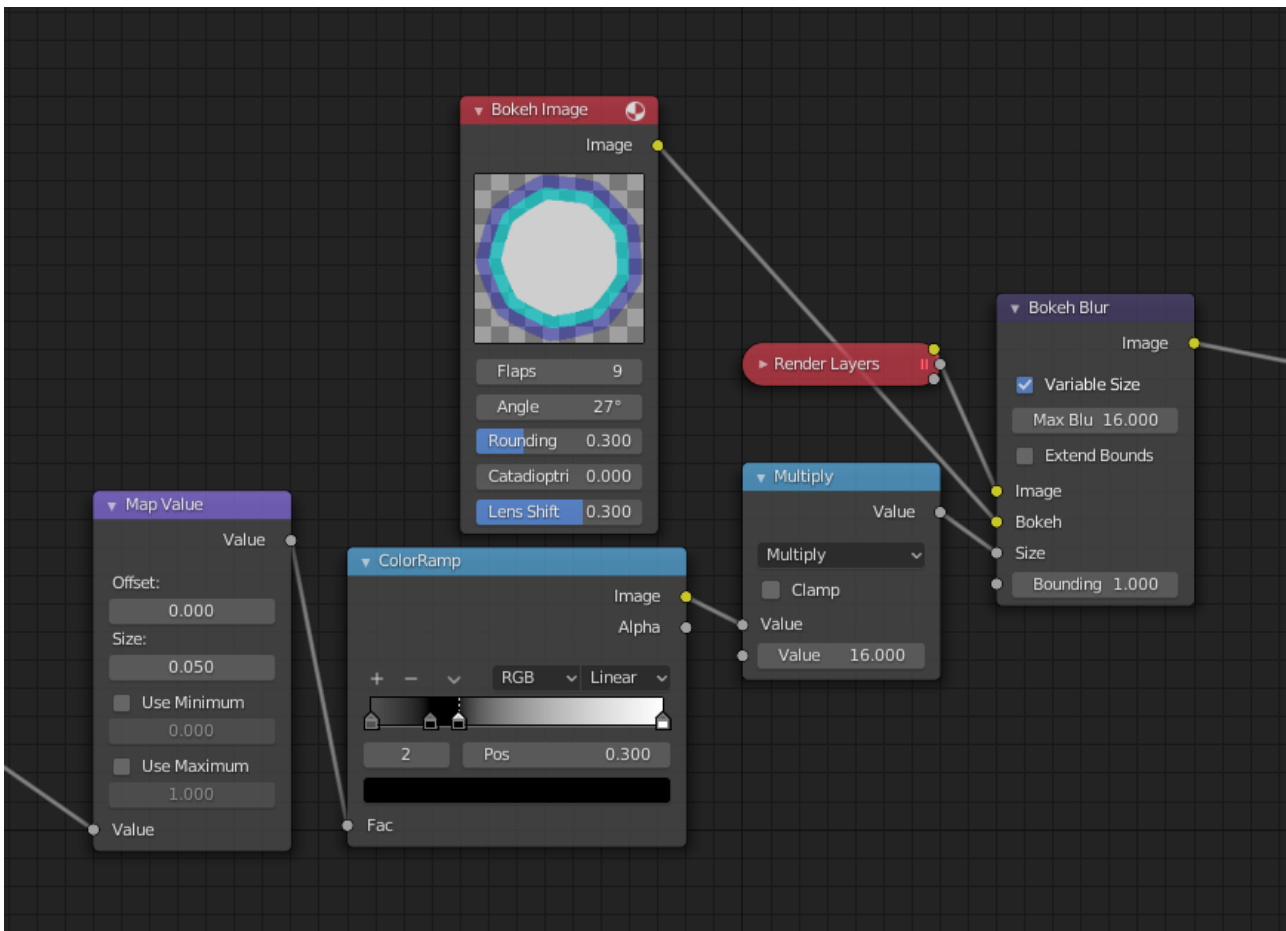


Fig. 168: Example of *Bokeh Image* node.

## Image Node

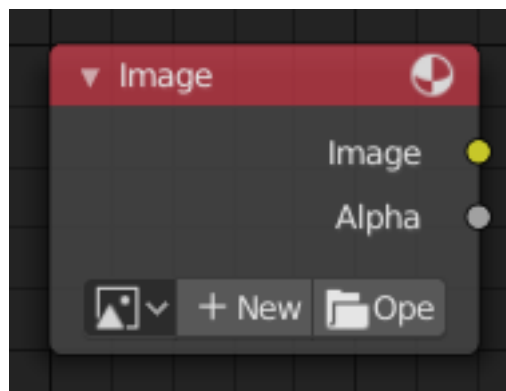


Fig. 169: Image Node.

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 image output.

**Alpha** Separate Alpha value.

**Z** Z depth layer.

---

**Note:** Multi-Layer Format

When a multi-layer file format, like EXR, is loaded, each layer is made available as a socket.

---

## Mask Node

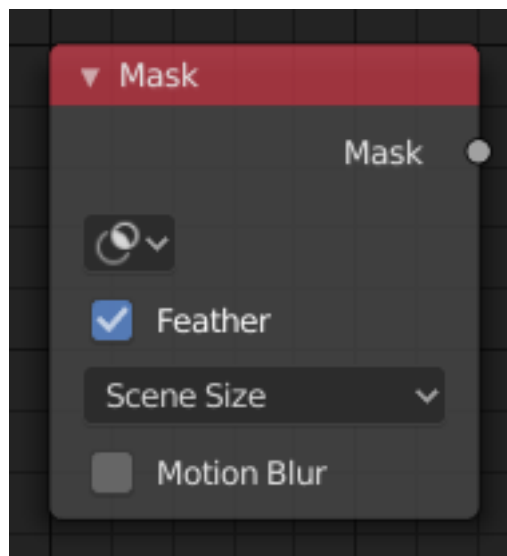


Fig. 170: Mask Node.

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

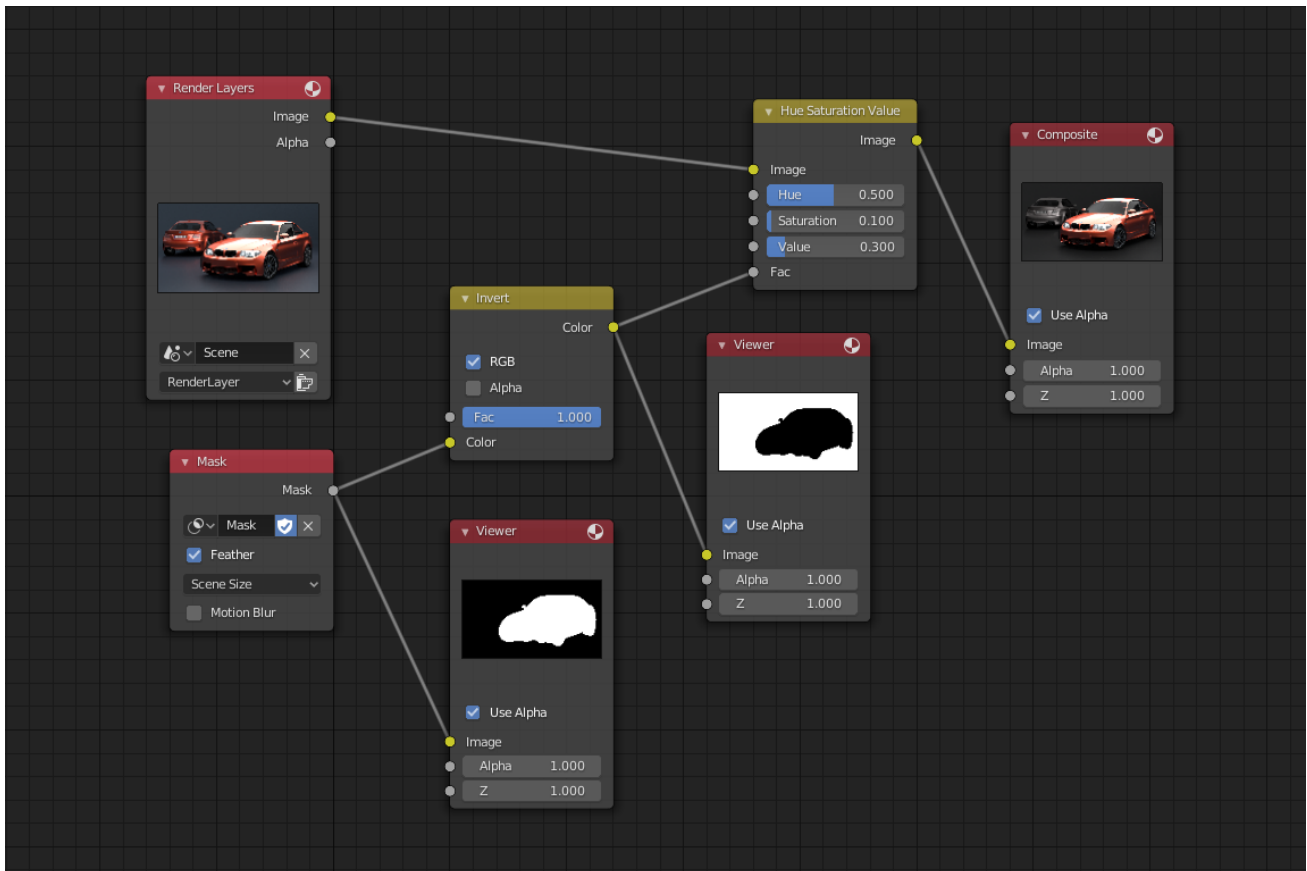


Fig. 171: Example of the Mask node.

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

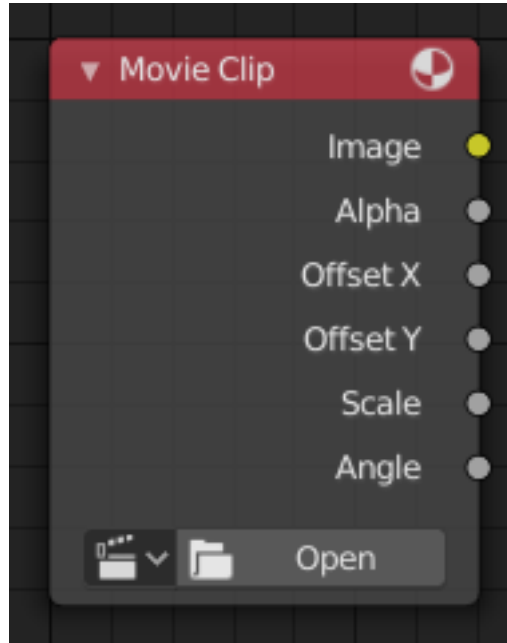


Fig. 172: Movie Clip node.

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.

## Render Layers Node

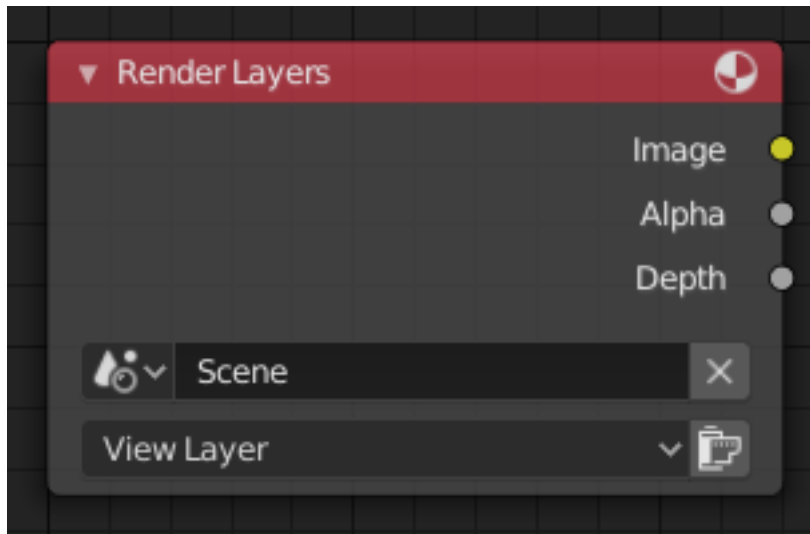


Fig. 173: Render Layers Node.

This node is the starting place for getting a picture of your scene into the compositing node tree.

### 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 short hand 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*.

**Z** By default the Z depth pass is enabled.



## RGB Node

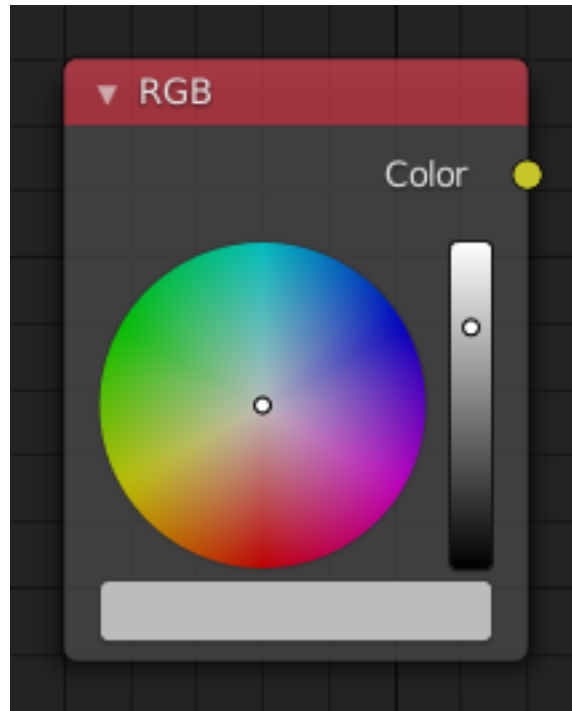


Fig. 174: RGB Node.

### Inputs

This node has no input sockets.

### Properties

The RGB node uses the *color picker widget*.

### Outputs

**Color / RGBA** A single RGBA color value.

## Texture Node

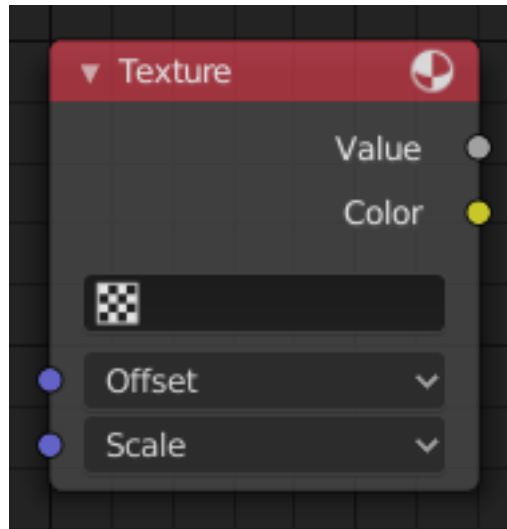


Fig. 175: Texture Node.

The Texture node makes 3D textures available to the 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.

## Time Node

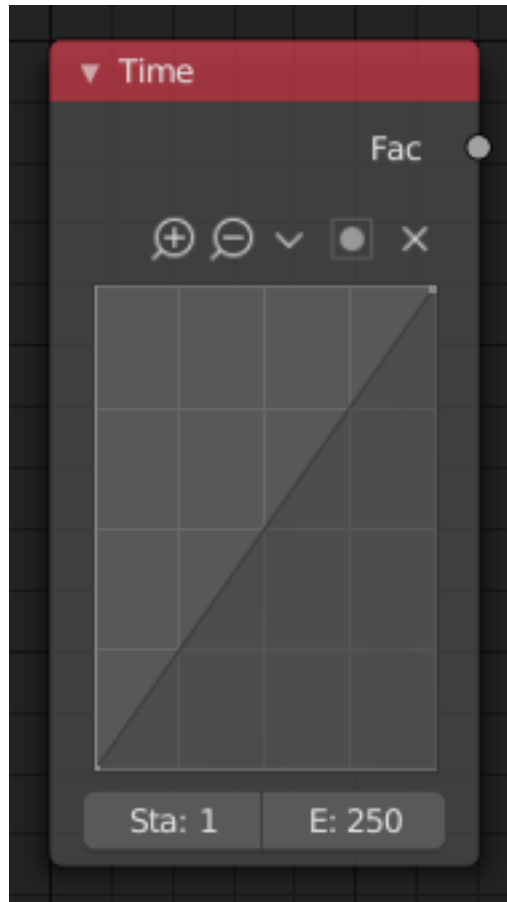


Fig. 176: Time Node.

The *Time node* generates a factor value (from 0.0 to 1.0) that changes according to the curve as time progresses through the *Timeline*.

### 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** A speed of time factor (from 0.0 to 1.0) relative to the frame rate defined in the *Render Dimensions Panel*. The factor changes according to the defined curve.

**Hint:** Output values

The *Map Value* node can be used to map the output to a more appropriate value. With sometimes curves, it is possible that the Time 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

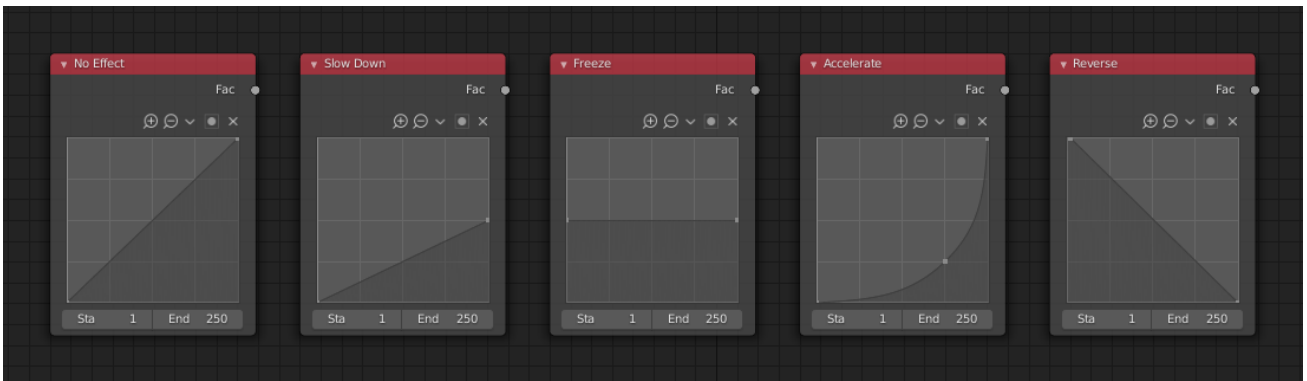


Fig. 177: Time controls from left to right: no effect, slow down, freeze, accelerate, reverse.

## Track Position Node

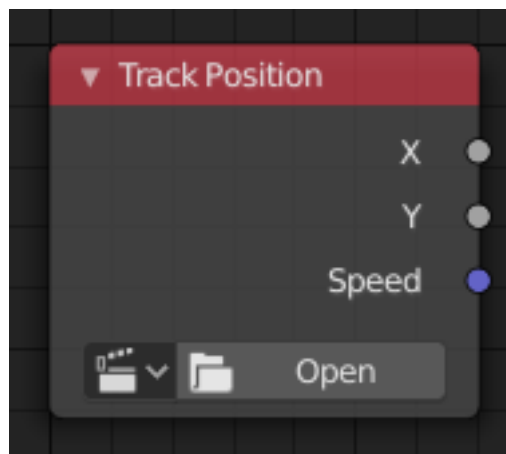


Fig. 178: Track Position Node.

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.

## Value Node

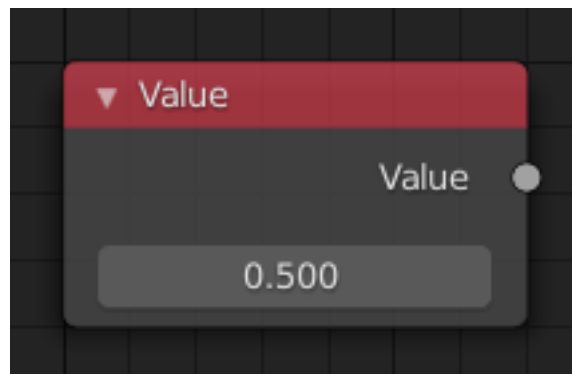


Fig. 179: Value Node.

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 example below the *Value Node* is used to control multiple values at once, this make the node a useful organizational tool.

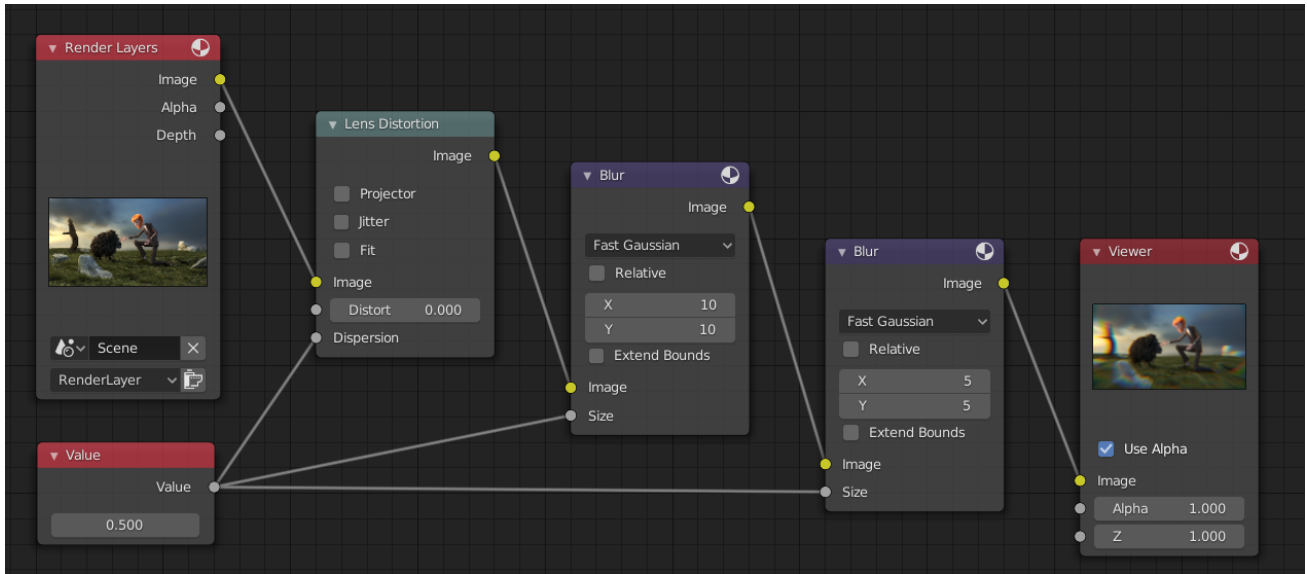


Fig. 180: Example of the *Value* node.

**Tip:** From this you can also make different values proportional to each other by adding a *Math Node* in between the different links.

## Output Nodes

These nodes are used to output the composited result in some way.

## Composite Node

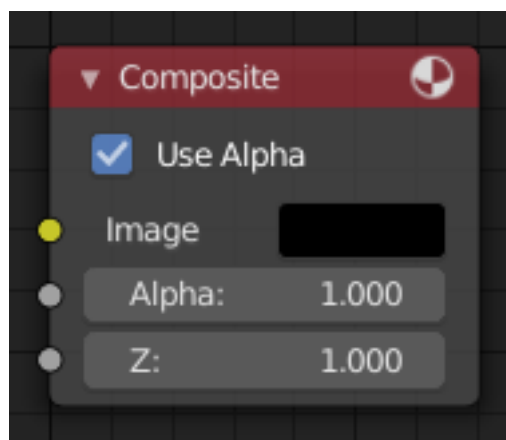


Fig. 181: Composite Node.

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.

**Z** Z depth.

### 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.

---

### File Output Node

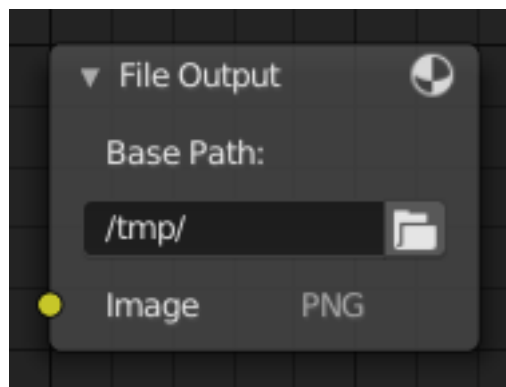


Fig. 182: File Output Node.

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.

---

**Note:** To support subsequent arrangement and layering of images, the node can supply a Z-depth map. However, please note that only the OpenEXR image formats save the Z information.

---

## 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.

## Levels Node

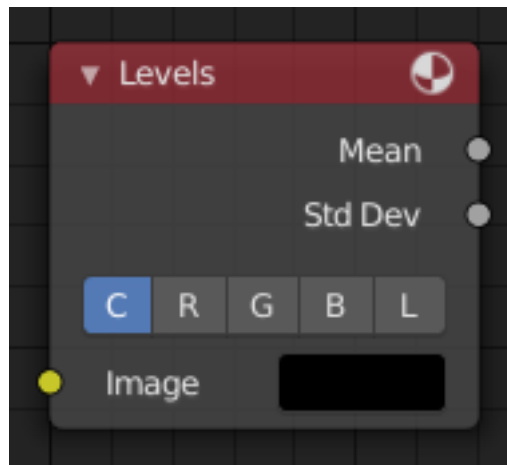


Fig. 183: Levels Node.

The Levels Node read the input color channels and outputs analytical values.

## Inputs

**Image** Standard image input.



## Properties

**Channel** C (Combined RGB), R (Red), G (Green), B (Blue), L (Luminance)

## Outputs

1D values based on the levels of an image.

**Mean** The mean is the average value of all image pixels in specified channel (combined, red, green, blue, luminance). 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.

The visualization of such data is just a gray rectangle.

## Split Viewer Node

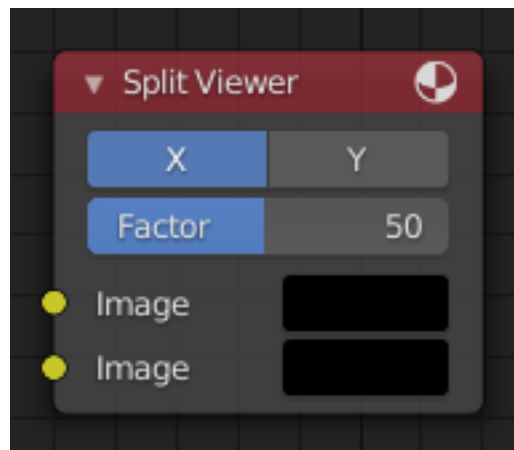


Fig. 184: Split Viewer Node.

The *Split Viewer* node takes two images and displays these side-by-side as backdrop or as a Viewer Node output.

## 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



Fig. 185: Example of a Split Viewer node.

## Viewer Node

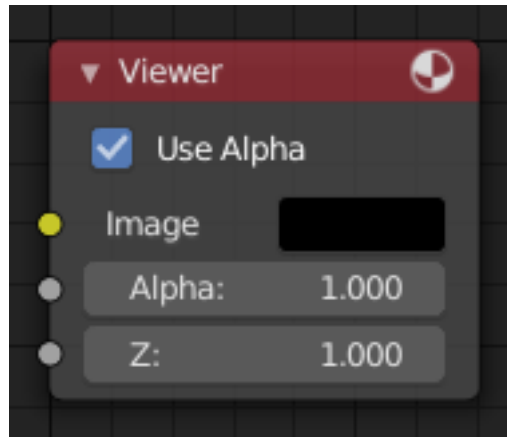


Fig. 186: Viewer Node.

The *Viewer* node is a temporary, in-process viewer. It can be plug 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

See *Composite Node*.

### Properties

**Tile Order** The tile order can be defined for the backdrop image, using the *Tile order* field in the properties of the Viewer node (*Properties* panel in Sidebar region, with the Viewer node selected):

**Rule of thirds** Calculates tiles around each of the nine zones defined by the *rule of thirds*.

**Bottom up** Tiles are calculated from the bottom up.

**Random** Calculates tiles in a non-specific order.

**Center** Calculates the tiles around a specific center, defined by X and Y fields.

X, Y

### 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.

## Color Nodes

These nodes adjust the image's colors, for example increasing the contrast, making it warmer, overlaying another image, etc.

### Alpha Over Node

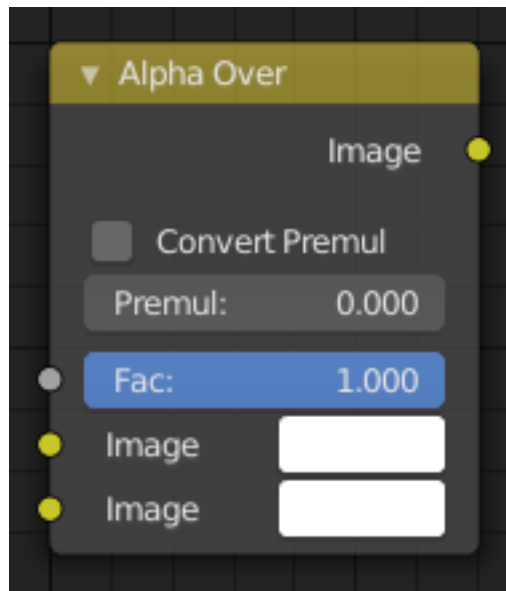


Fig. 187: Alpha Over Node.

The *Alpha Over* node is used to layer images on top of one another. Where the foreground image pixels have an alpha greater than 0, it will be overlaid over the background image.

### Inputs

**Factor** Controls the transparency of the foreground image. A factor less than 1 will make the foreground more transparent.

**Image** Input for the *background* image.

**Image** Input for the *foreground* image.

## Properties

**Convert Premultiplied** Converts foreground image to *premultiplied alpha* format.

The *Alpha Over* node is designed to work with *premultiplied alpha* color format. Use *Convert Premul* when you know that your image has *straight alpha* color values, to perform the correct over operation. Result will still be *premultiplied alpha*.

See *Alpha Channel*.

**Premultiply** The *Premul* slider allows to mix between the using *premultiplied* or *non premultiplied alpha*.

When set to 1, the foreground color values will be multiplied by alpha, i.e. *premultiplied*. This is equivalent to enabling the *Convert Premul* option. When set to 0, color values does not change.

If *Premultiply* is not zero, *Convert Premul* will be ignored.

---

**Note:** This is a legacy option.

---

## Outputs

**Image** Standard image output.

## Examples

### Overlay

In the node tree below, *Color Ramp* node is used to add an alpha channel to the black-and-white swirl image. Then *Alpha Over* node is used to overlay it on top of another image.

### Fade In

In the next example, the *Factor* is used to make a “Fade In” effect. This effect can be animated by adding a *Time* node to feed the *Factor* socket as shown below. Over the course of 30 frames, the *Alpha Over* node outputs a image that starts with the pure background image, and the title slowly appearing.

Note the *Convert Premul* checkbox is enabled, since as the foreground used a PNG image that has *straight alpha*.

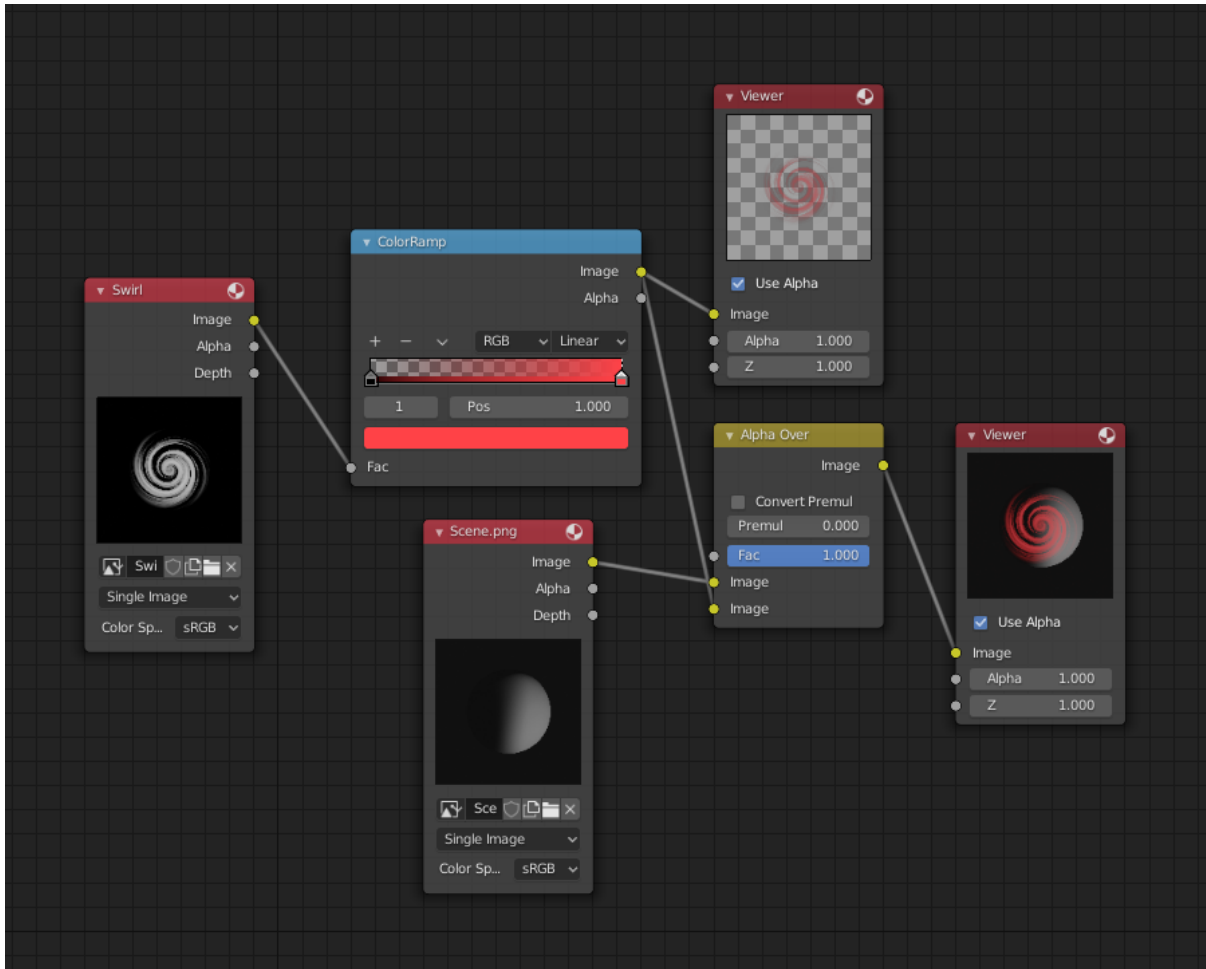


Fig. 188: Assembling a composite image using Alpha Over.

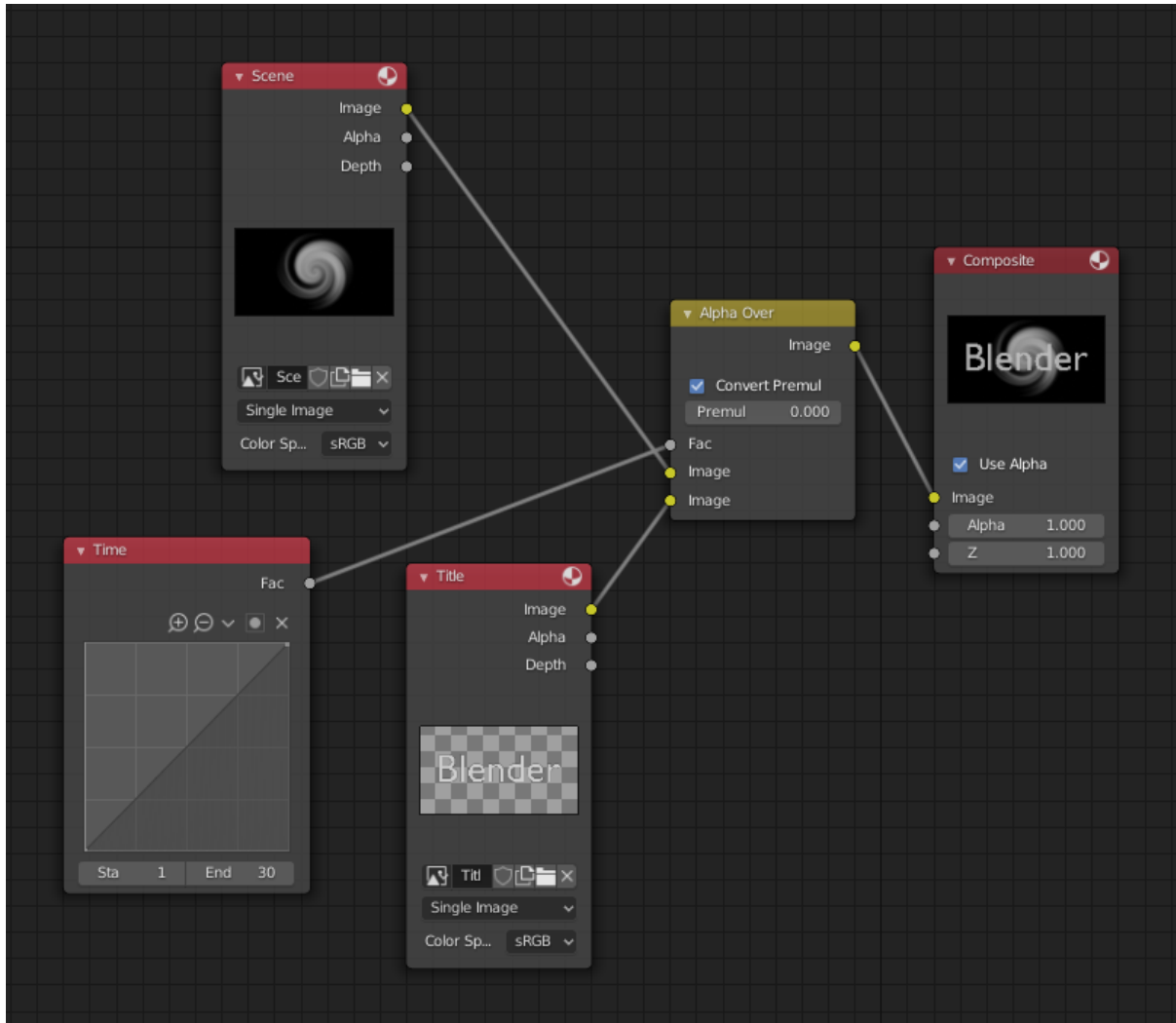


Fig. 189: Animated fade in effect using Alpha Over.

## Bright/Contrast Node

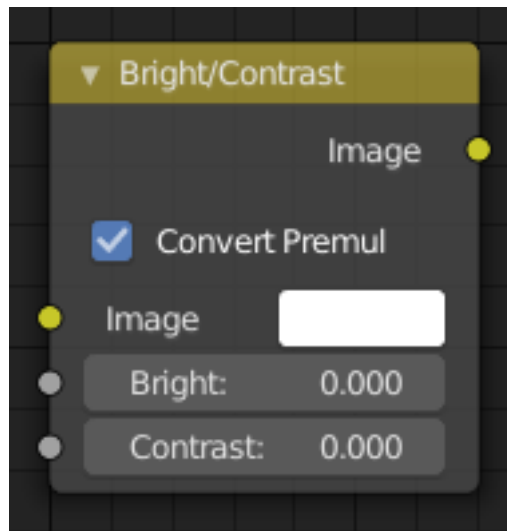


Fig. 190: Brightness/Contrast Node.

### Inputs

**Image** Standard image 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 image 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).

Either of these nodes will scale the values back to normal range. In the example image, we want to amp up the specular pass. The bottom thread shows what happens if we do not clamp the values; the specular pass has valued 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.



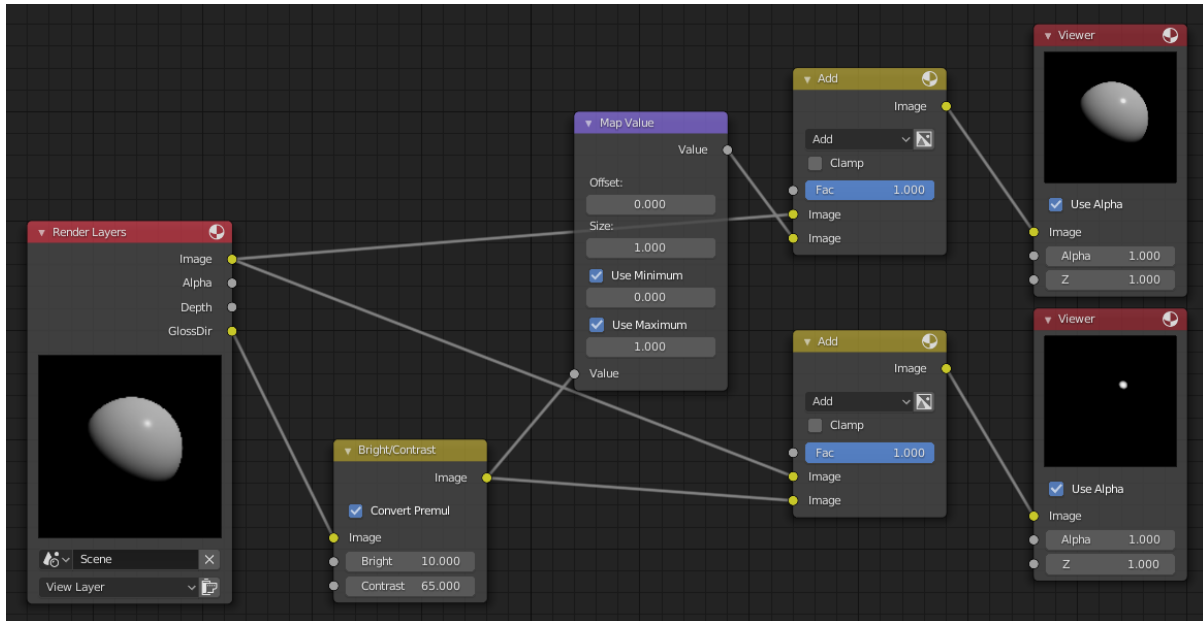


Fig. 191: Clamp the values to normal range.

## Example

### Color Balance Node

The Color Balance node can adjust the color and values of an image.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Color** Standard image input.

### Properties

Two different correction formulas could be selected.

#### Lift/Gamma/Gain

**Lift** Increases the value of dark colors.

**Gamma** Adjusts midtones.

**Gain** Adjusts highlights.

#### Offset/Power/Slope (ASC-CDL)

**Offset** Summand. (Adjusts the overall brightness.)

**Basis** Additional offset, allows to specify a negative Offset value.

**Power** Over-all exponent. (Adjusts the midtones.)

**Slope** Multiplier. (Adjusts the highlights.)

### Outputs

**Color** Standard output image.

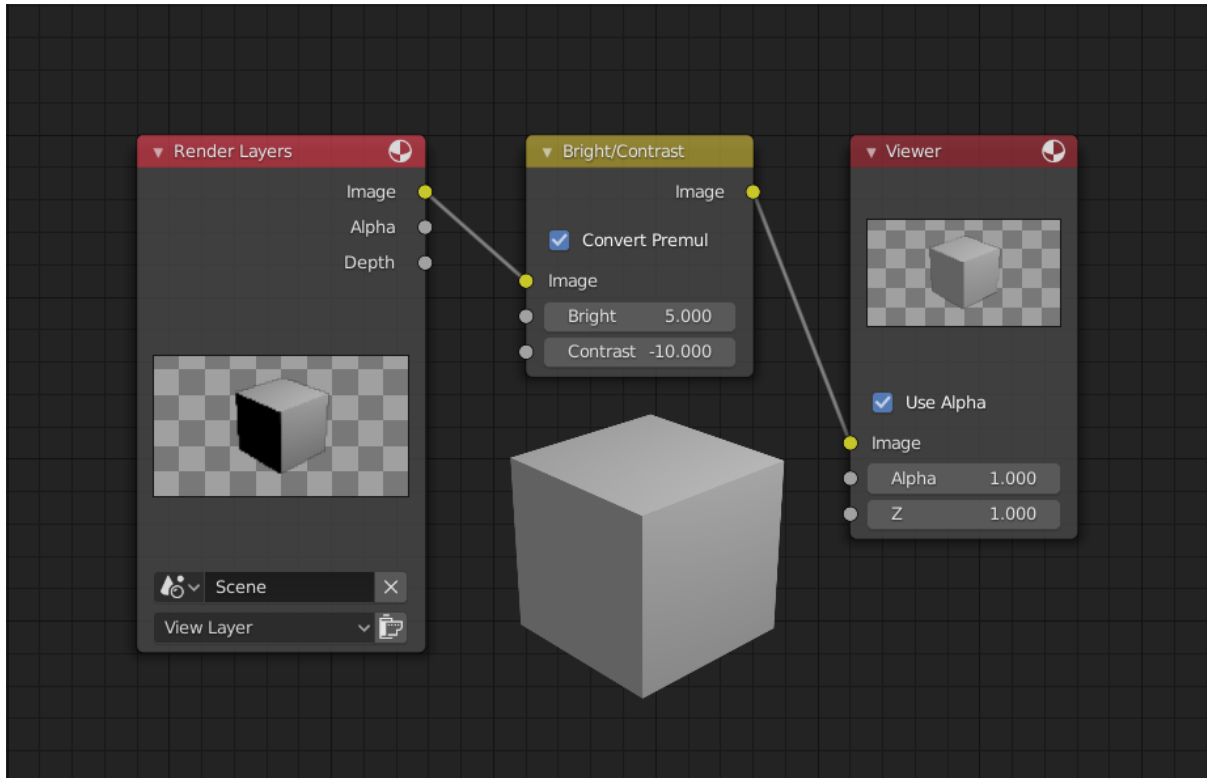


Fig. 192: A basic example.

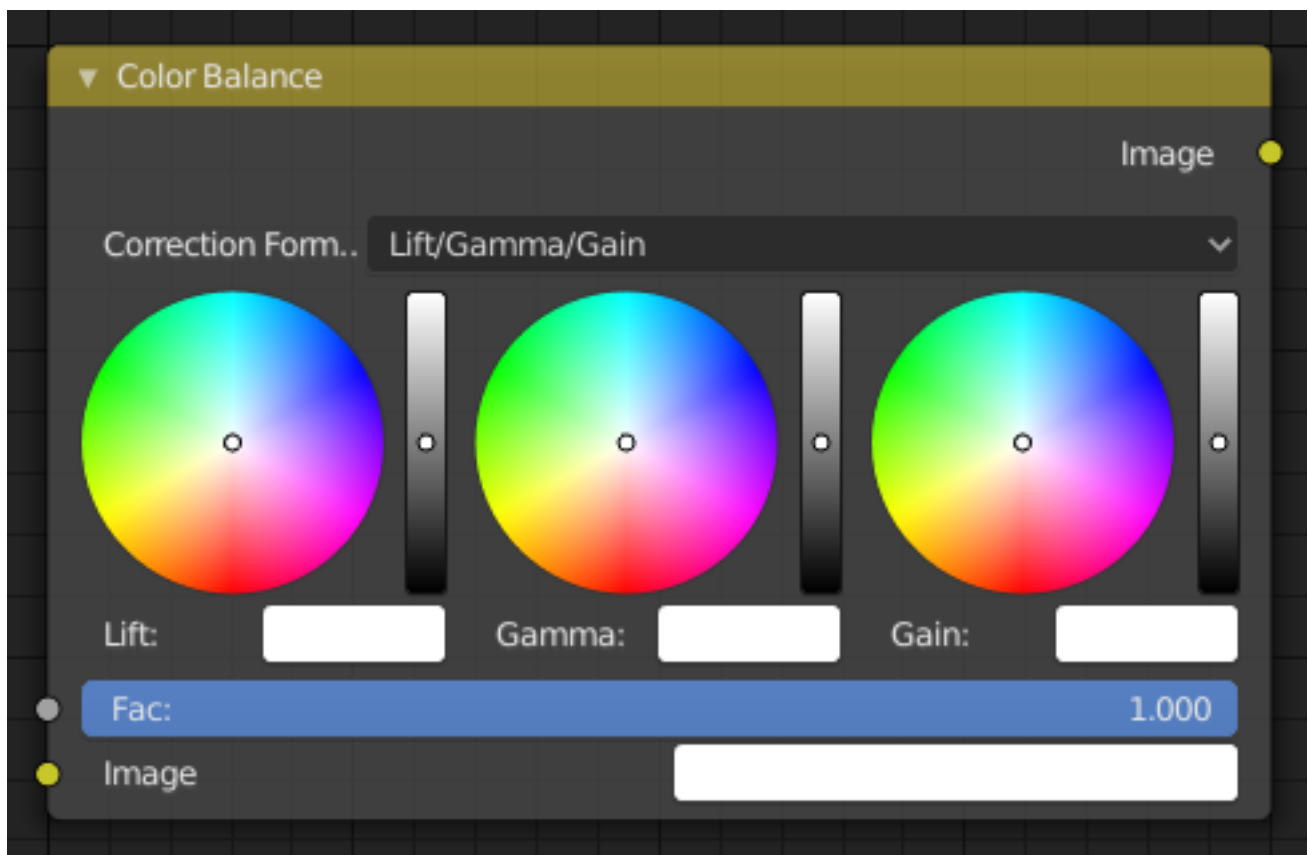


Fig. 193: Color Balance Node.

## Advanced

### The Offset/Power/Slope Formula

$$out = (i \times s + o)^p$$

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

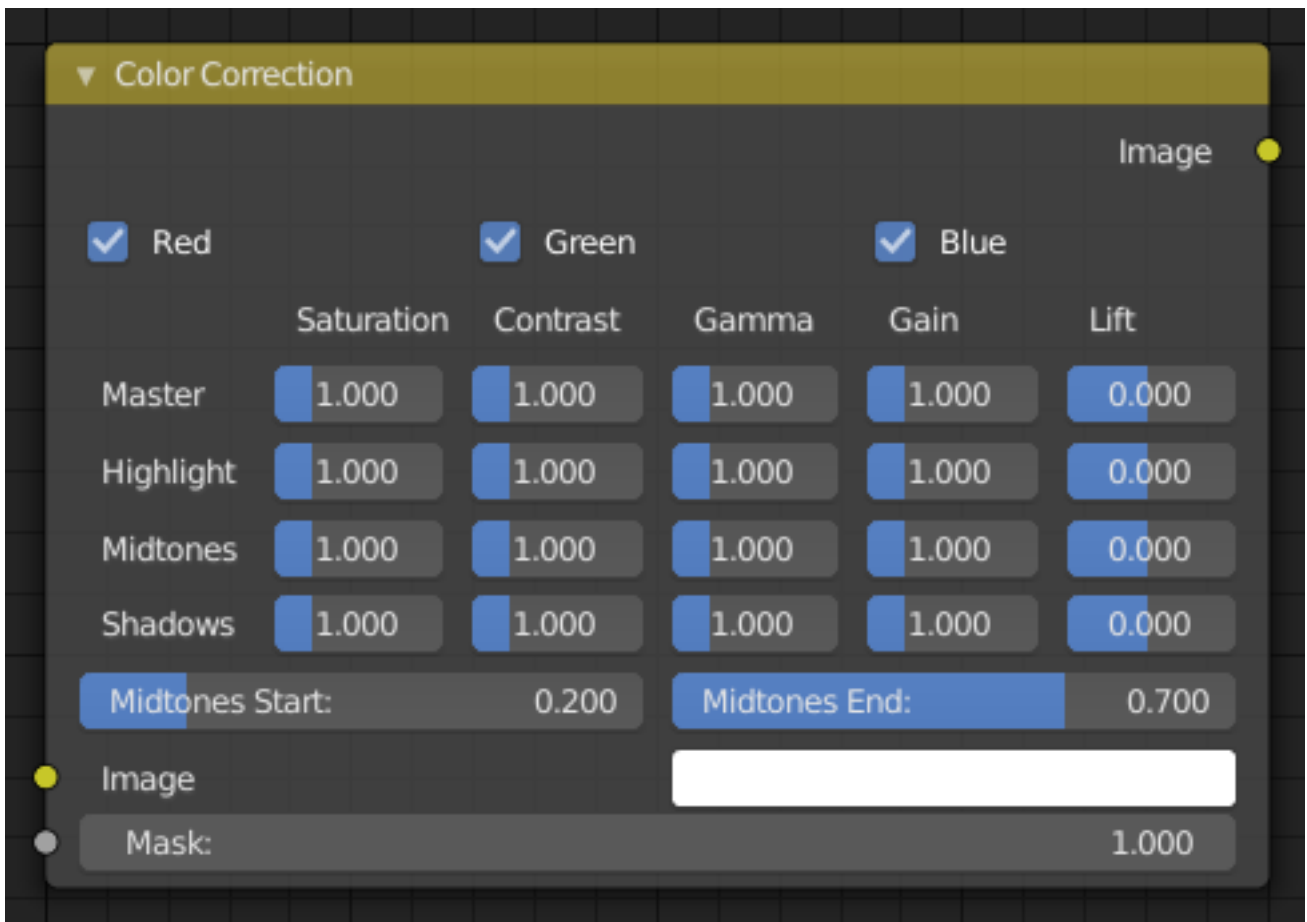


Fig. 194: Color Balance Node.

The Color Correction node can adjust the color of an image, separately in several tonal ranges (highlights, midtones and shadows) and only affect the necessary RGB channels.

### 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 image input.

**Mask** Controls the amount of influence the node exerts on the output image.

## Outputs

**Color** Standard image output.

## Gamma Node

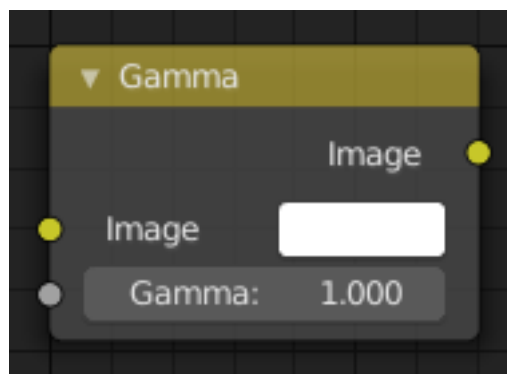


Fig. 195: Gamma Node.

Use this node to apply a gamma correction.

## Inputs

**Image** Standard image input.

**Gamma** An exponential brightness factor.

## Properties

This node has no properties.

## Outputs

**Image** Standard image output.

## Examples



Fig. 196: Example of a Gamma node.

## Hue Correct Node

The *Hue Correct Node* is able to adjust the Hue, Saturation, and Value of an image, with an input curve.

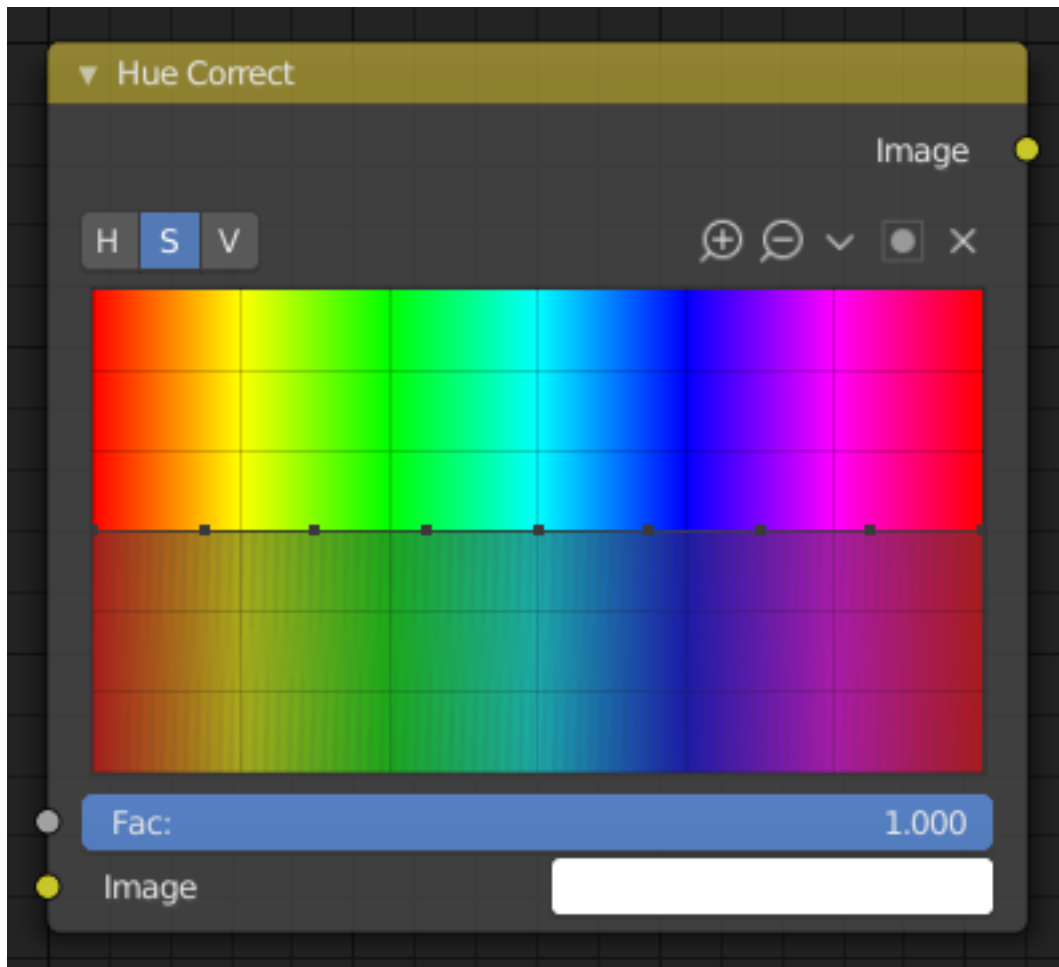


Fig. 197: Hue Correct Node.

## Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** Standard image 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 image output.

## Hue Saturation Value Node

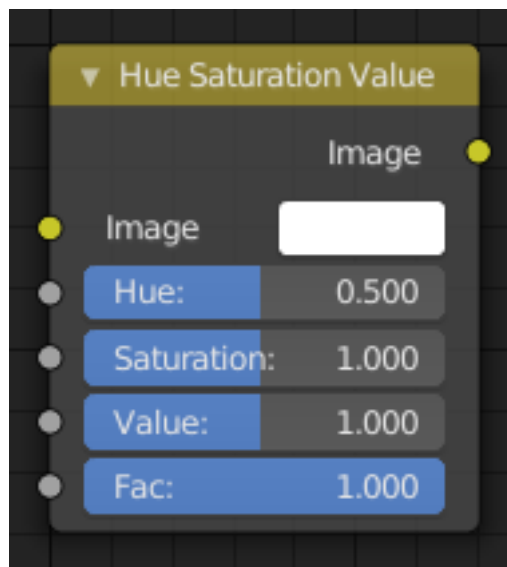


Fig. 198: Hue Saturation Node.

The *Hue Saturation Value Node* applies a color transformation in the HSV color space.

## Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** Standard image input.

## Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

**Hue** Specifies the hue rotation of the image.  $360^\circ$  are mapped to (0 to 1). The hue shifts of 0 ( $-180^\circ$ ) and 1 ( $+180^\circ$ ) 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.

## Outputs

**Image** Standard image output.

## Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

**Hues are vice versa** A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

**Hue and Saturation work together.** So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

**Gray & White are neutral hues** A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

**Changing the effect over time** The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

---

**Note:** Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a Mix node to add in a static color from an RGB input node with your image.

---





Fig. 199: A basic example.

## HSV Example

### Invert Node

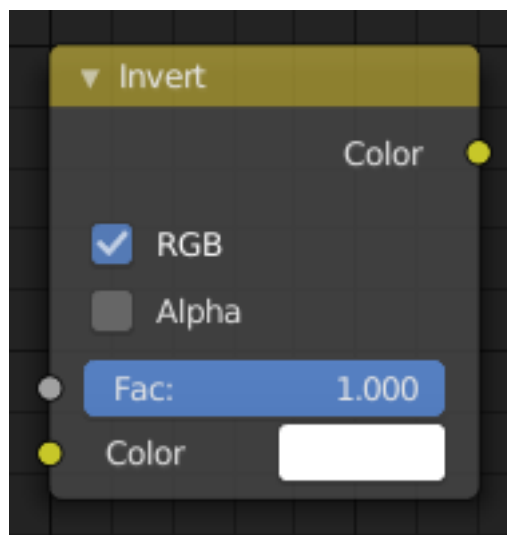


Fig. 201: Invert Node.

The *Invert Node* inverts the colors in the input image, producing a negative.

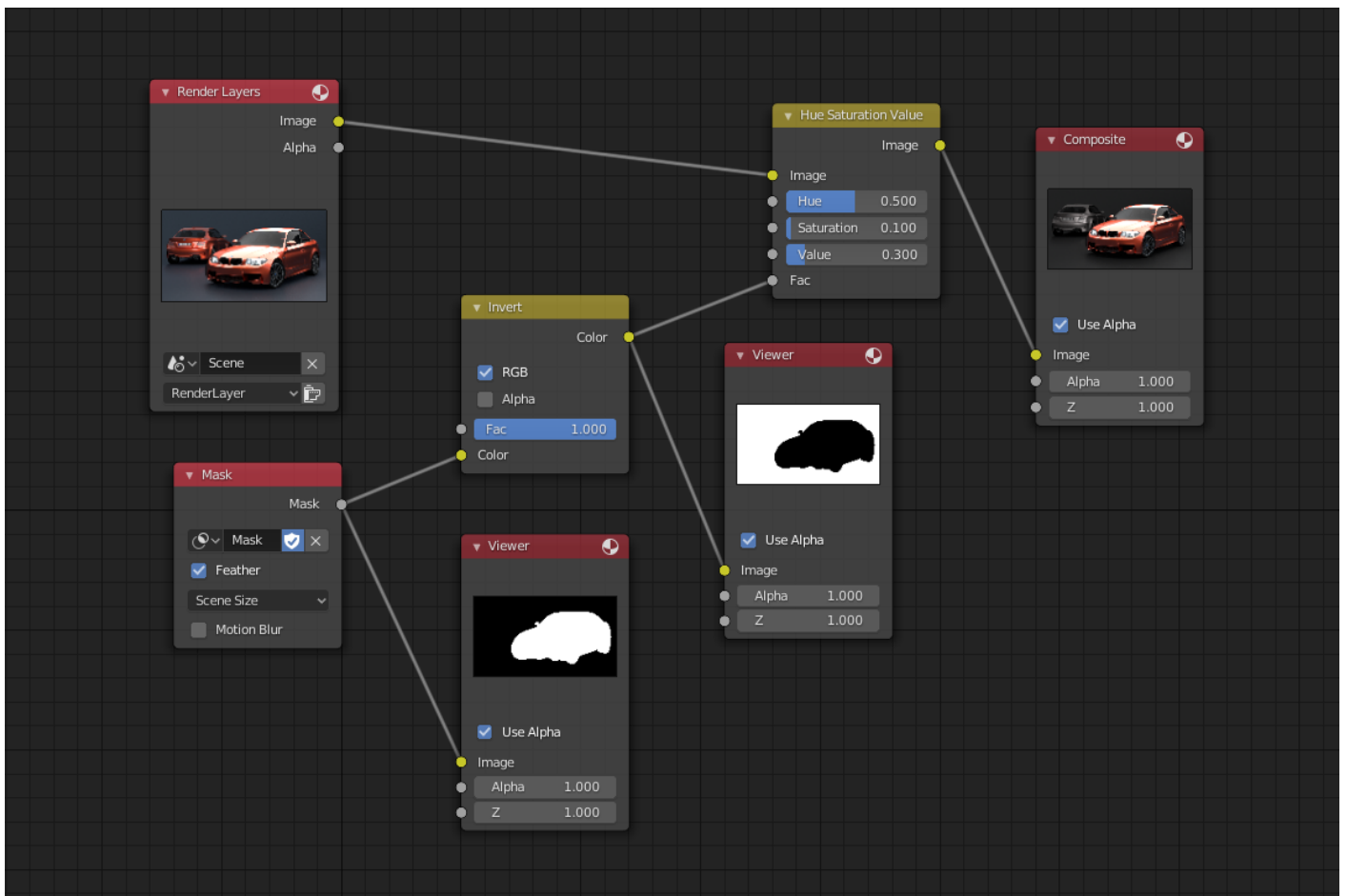


Fig. 200: An example of using the Factor input for masking.

## Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Color** Standard image input.

## Properties

In the compositing context this node has the following properties.

**RGB** De/activation of the color channel inversion.

**Alpha** De/activation of the alpha channel inversion.

## Outputs

**Color** Standard image output.

## Example

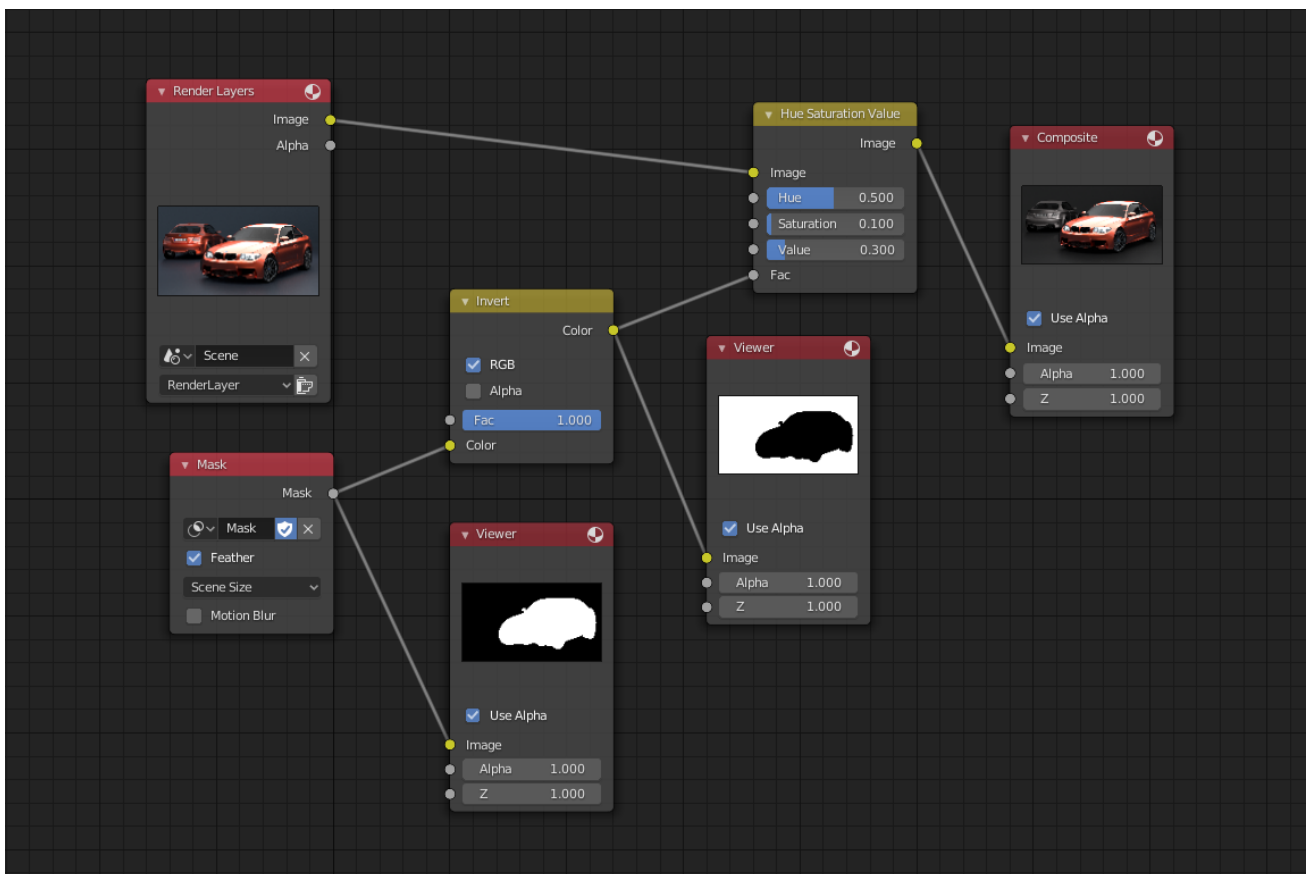


Fig. 202: The Invert node is used to invert the mask.

## Mix Node

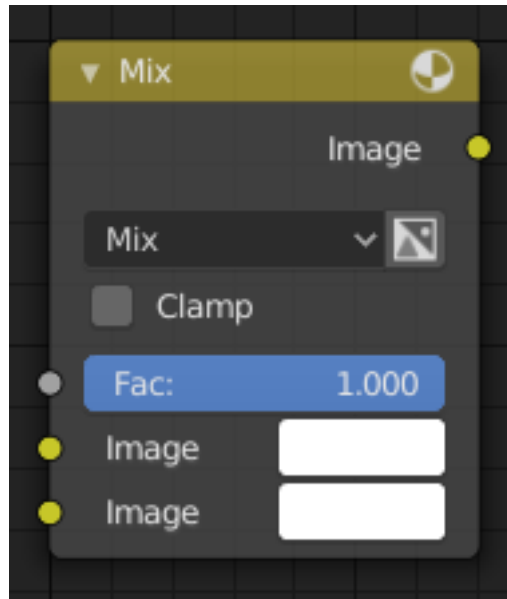


Fig. 203: Mix Node.

The *Mix Node* mixes images by working on the individual and corresponding pixels of the two input images. Called “MixRGB” in the shader and texture context.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** The background image. The image size and resolution sets the dimensions of the output image.

**Image** The foreground image.

### Properties

**Mix** 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

**Use Alpha** If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

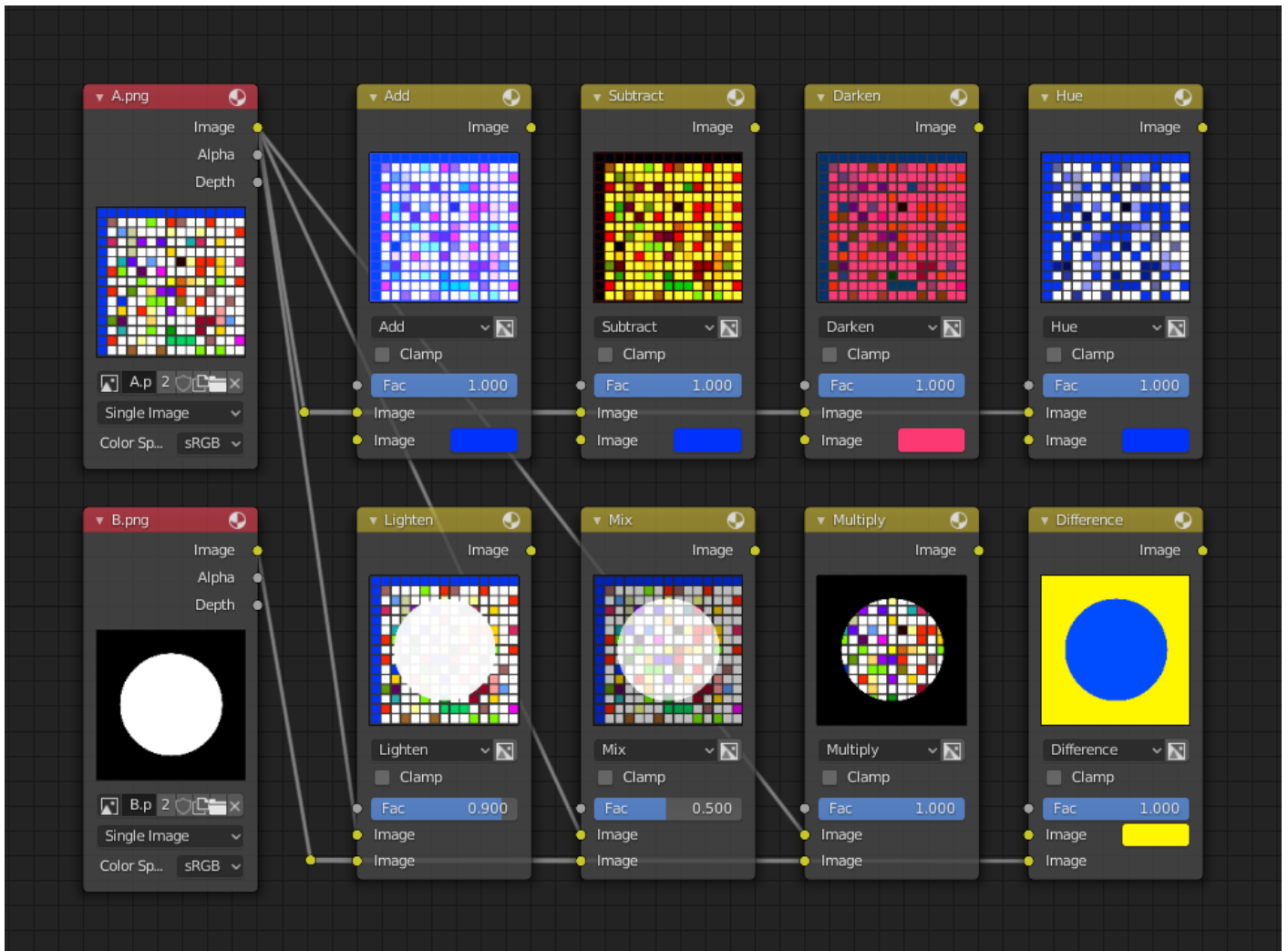
**Clamp** Limit the output value between 0.0 and 1.0.

### Outputs

**Image** Standard image output.

## Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.



Some explanation of the mixing methods above might help you use the Mix node effectively:

**Add** Adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

**Subtract** Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

**Multiply** Black (0.0) times anything leaves black. Anything times White (1.0) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

**Hue** Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

**Mix** Combines the two images, averaging the two.

**Lighten** Like bleach makes your whites whiter. Used with a mask to lighten up a little.

**Difference** It takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two very similar images to see what had been done to one to make it the other; sort of like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

**Darken** With the colors set here, it's like looking at the world through rose-colored glasses.

**Note:** Only add, subtract, multiply and divide are suitable for *Scene Referenced* images.

### Contrast Enhancement

Here is a small node tree showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

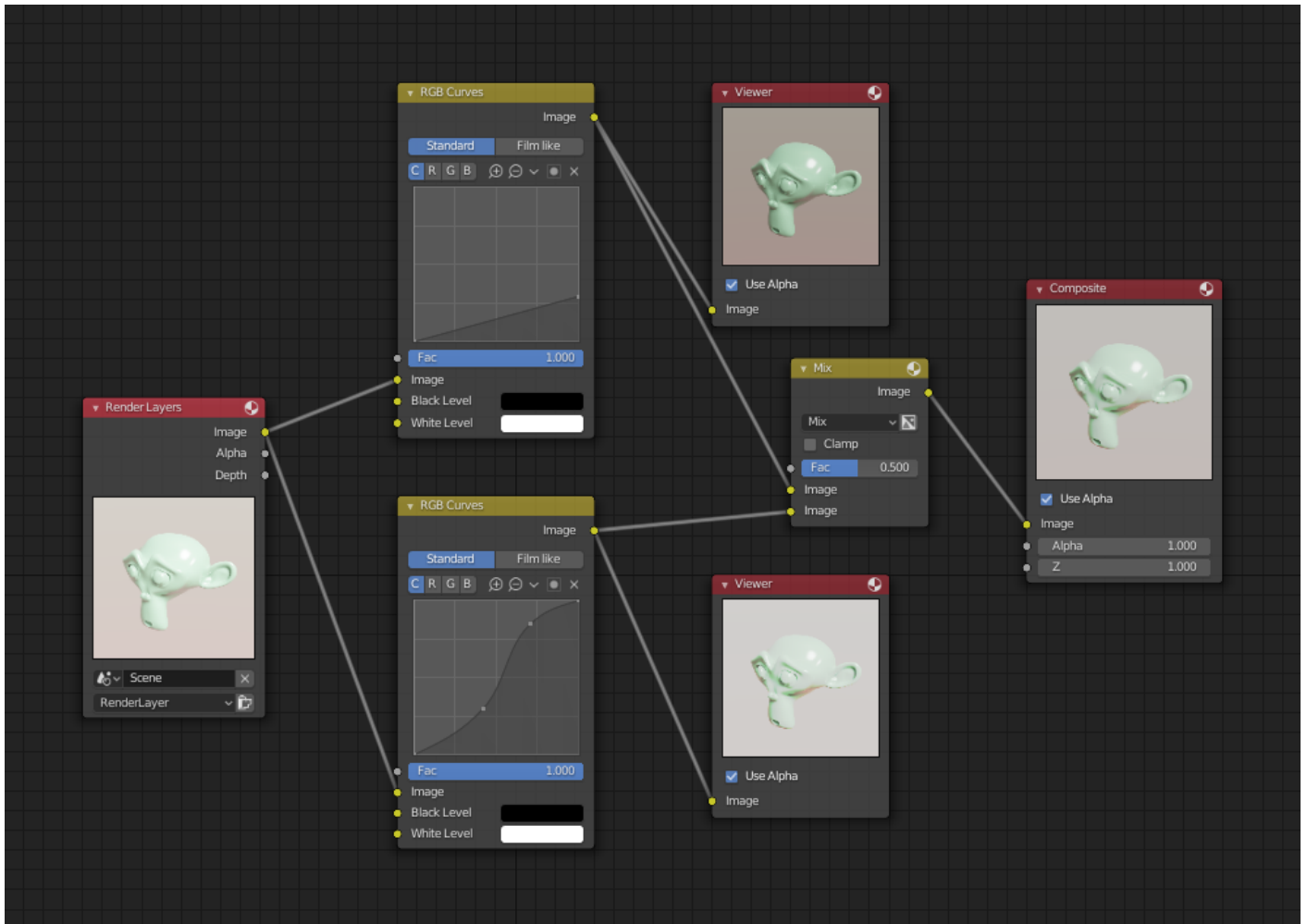


Fig. 204: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S-shaped) curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

## 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 image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, making the ones containing the mark ever-so-slightly brighter.

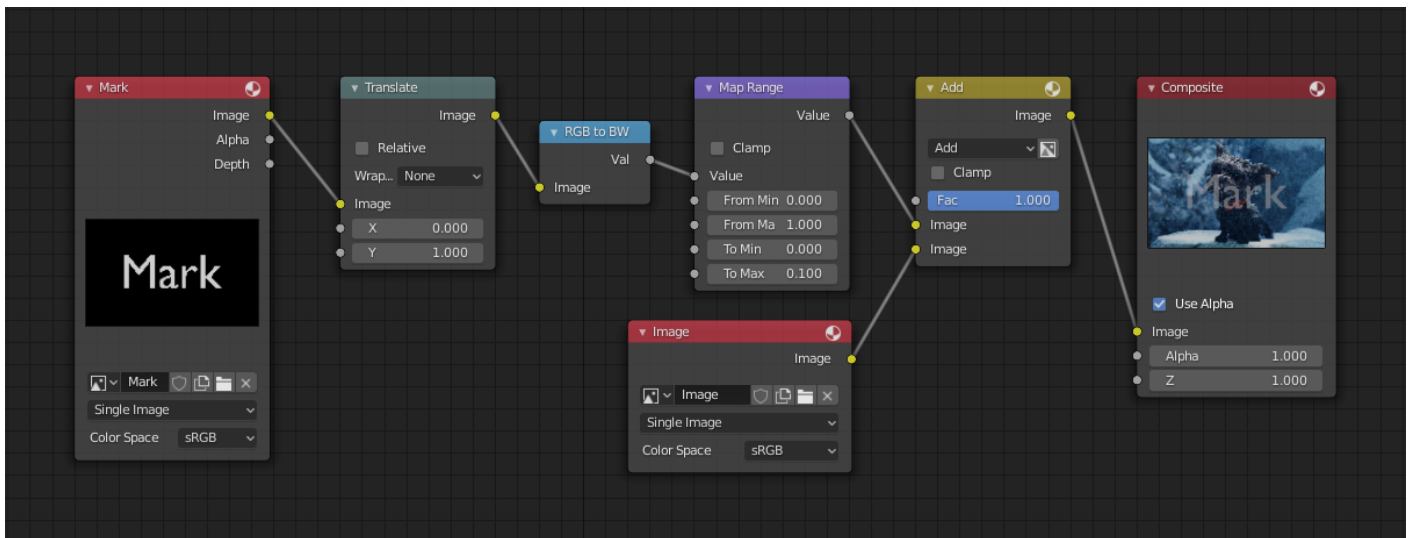


Fig. 205: Embedding your mark in an image using a mark and specific position.

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!

---

#### Hint: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm.

---

### 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. The result is routed to a viewer, and you can see how the original mark clearly stands out.

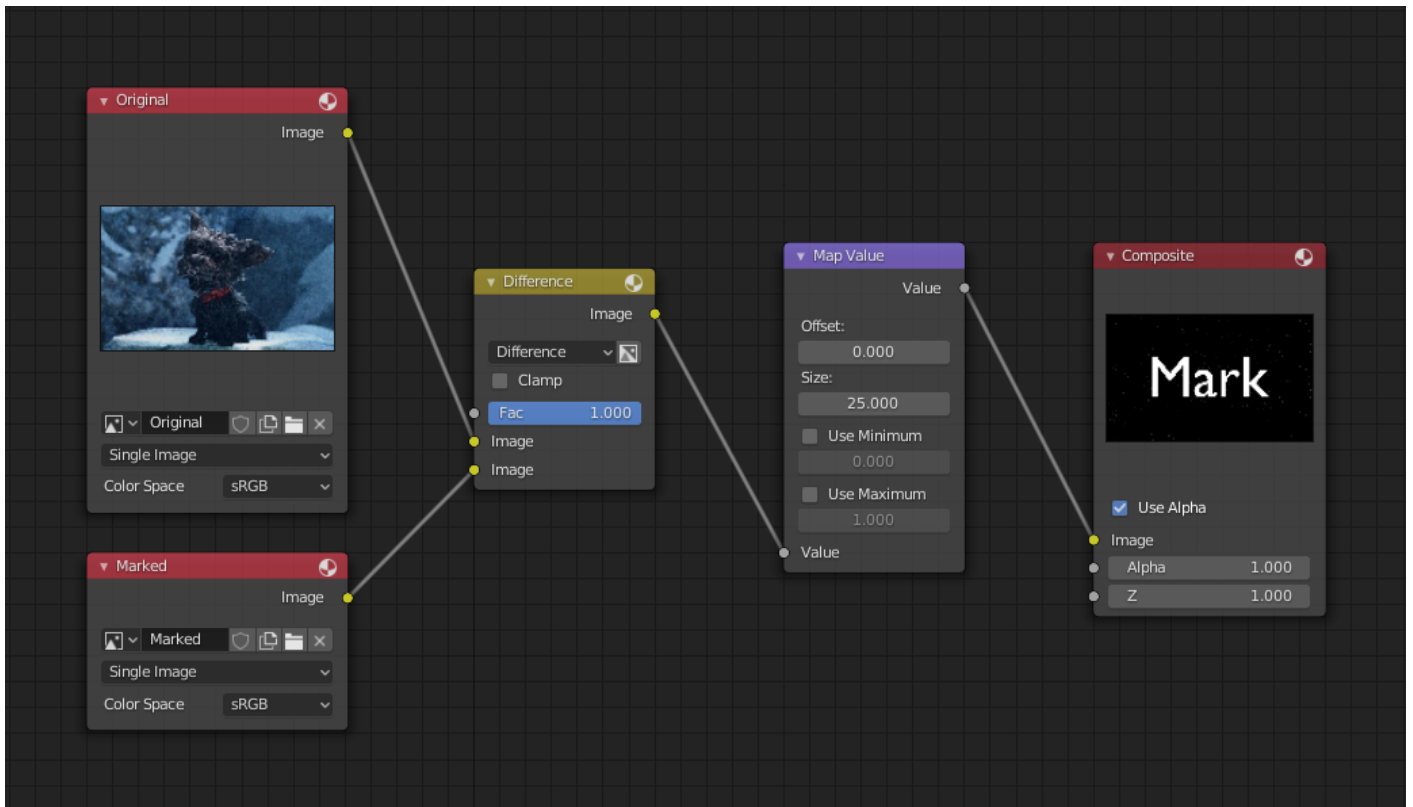


Fig. 206: Checking an image for your watermark.

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding node group in one scene, and the decoding group in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.



## RGB Curves Node

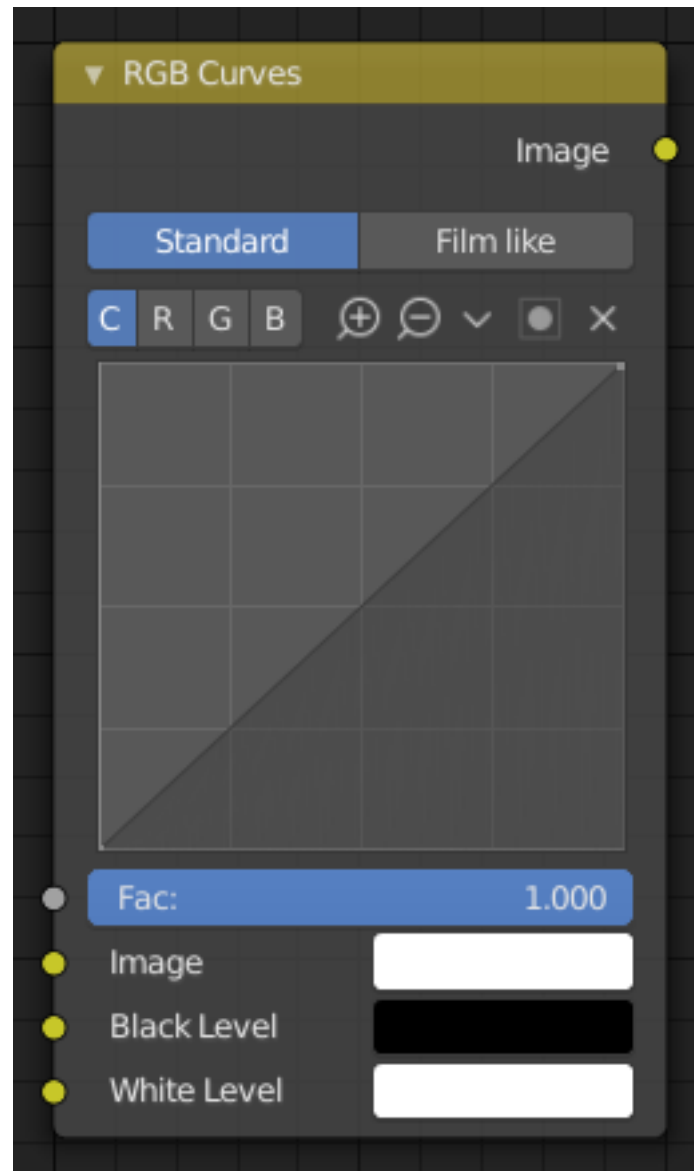


Fig. 207: RGB Curves Node.

The *RGB Curves Node* allows color corrections for each color channel and levels adjustments in the compositing context.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** Standard image input.

**Black Level *Compositor Only*** Defines the input color that is (linear) mapped to black.

**White Level *Compositor Only*** Defines the input color that is (linear) mapped to white.

---

**Tip:** To define the levels, use the *eyedropper* to select a color sample of a displayed image.

---

## Properties

### Tone

**Standard** TODO 2.8

**Film like** TODO 2.8

**Channel** Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue)

**Curve** A Bézier curve that varies the input levels (X axis) to produce an output level (Y axis). For the curve controls see: *Curve widget*.

### Outputs

**Image** Standard image output.

### Examples

Below are some common curves you can use to achieve desired effects.

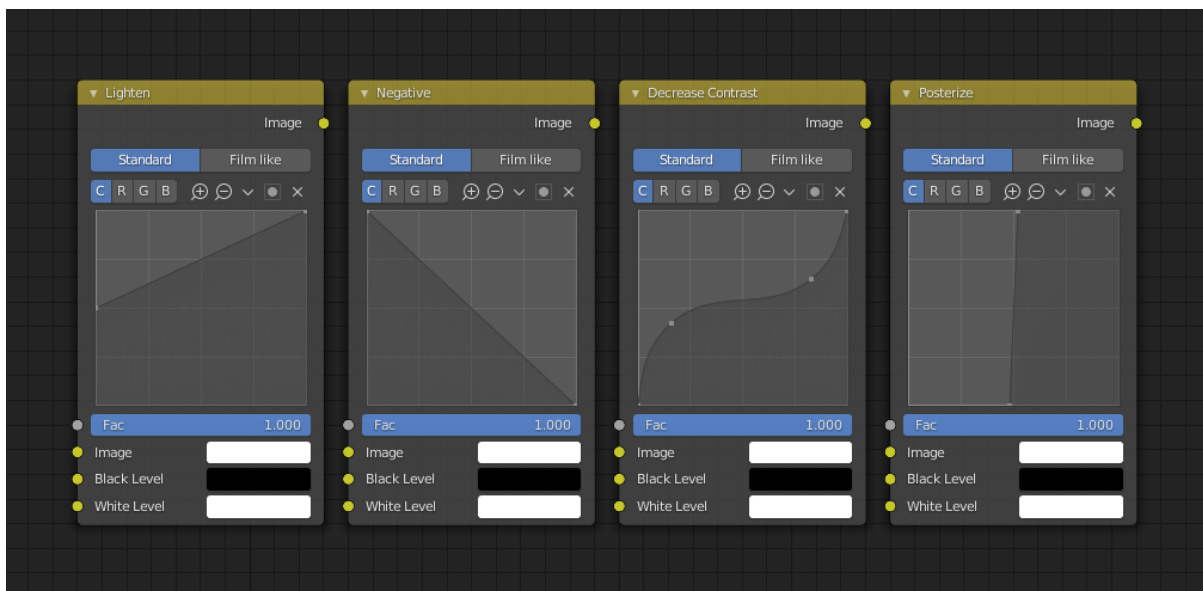


Fig. 208: From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

### Color Correction using 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.

Also, read on for examples of the Darken and Contrast Enhancement curves, [here](#).

### Color Correction using 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.

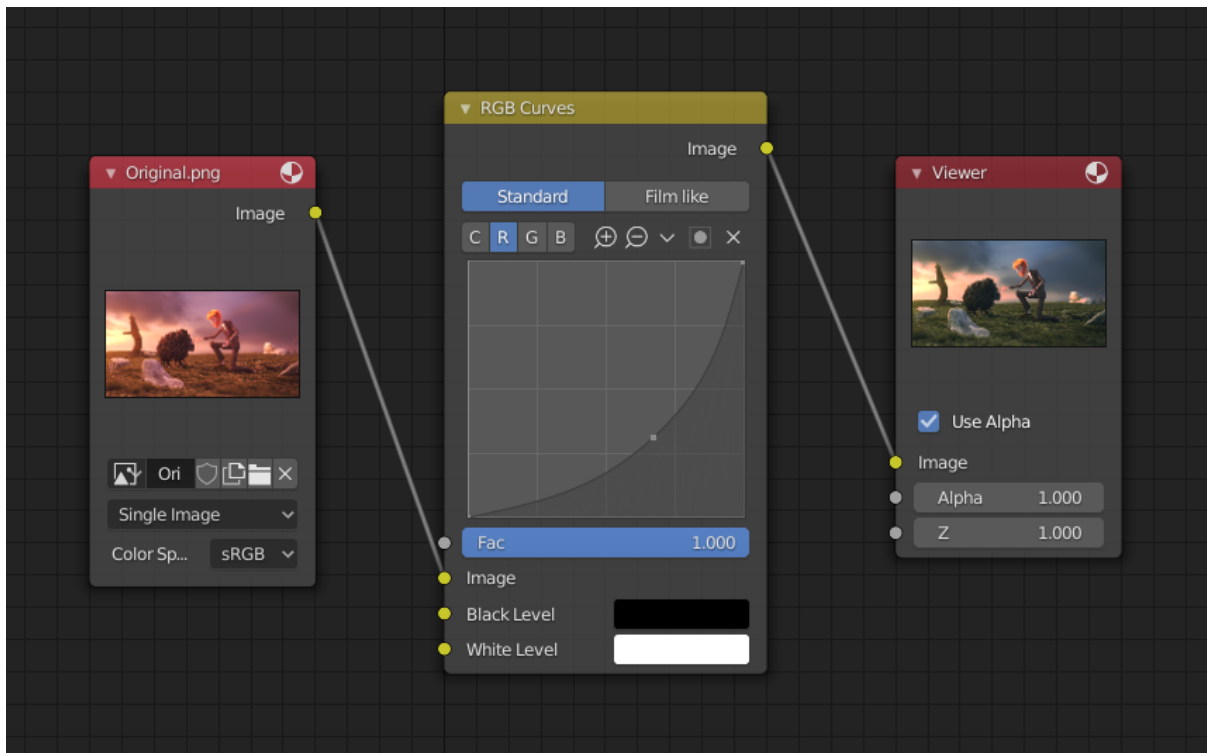


Fig. 209: Color correction with curves.

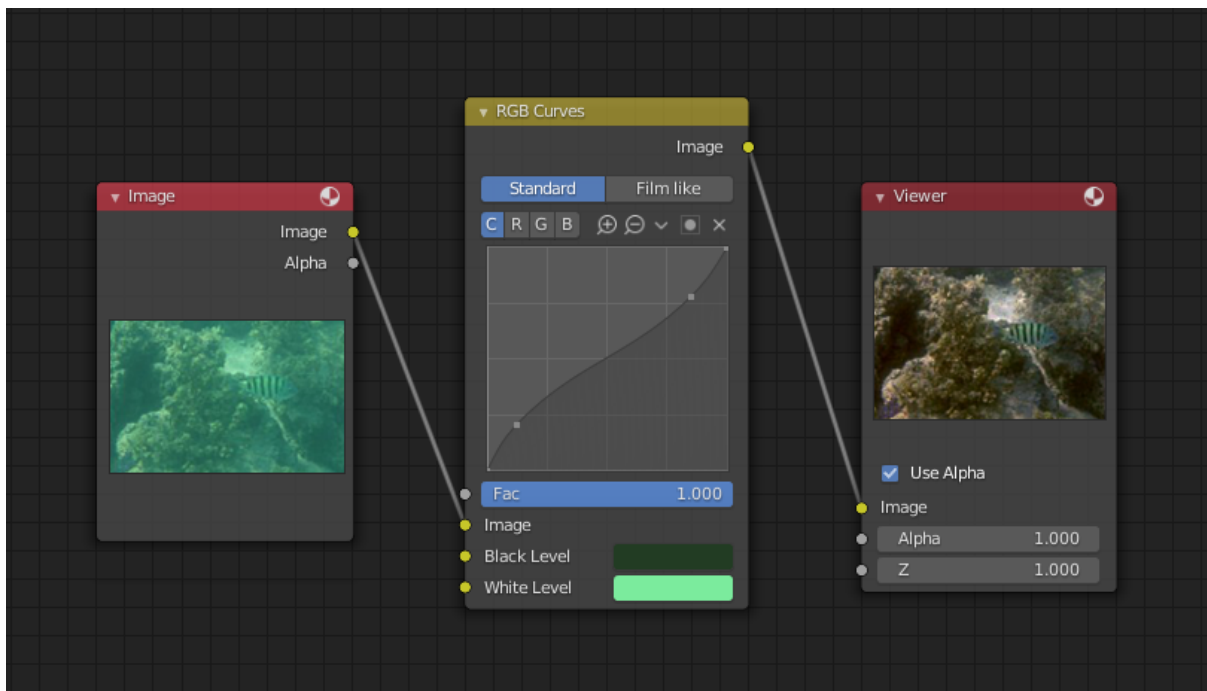


Fig. 210: Color correction with Black/White Levels.

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

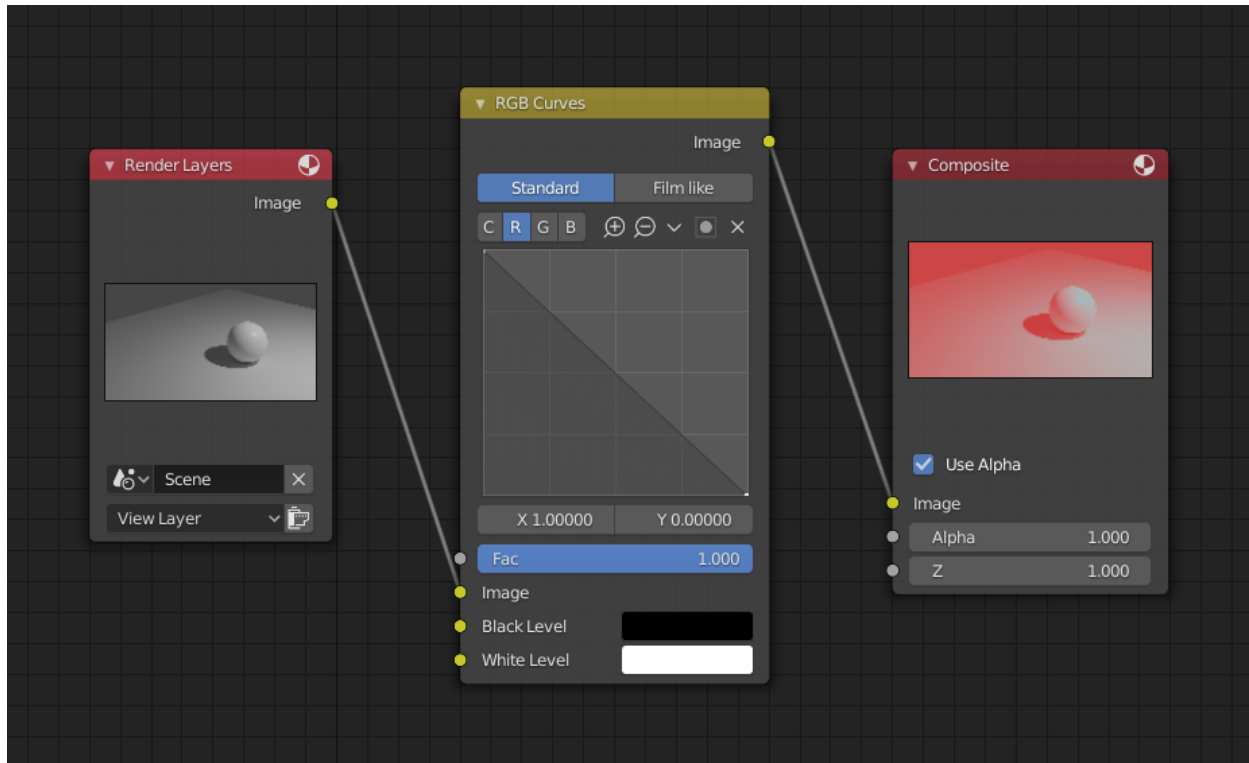


Fig. 211: Changing colors by inverting the red channel.

## Tone Map Node

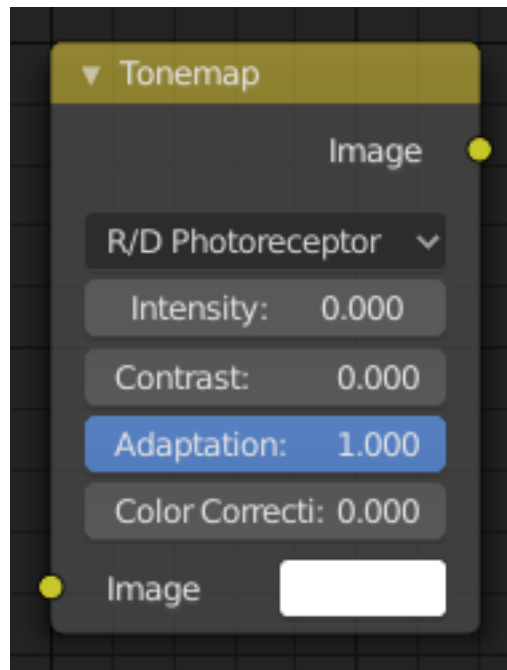


Fig. 212: Tone Map Node.

Tone mapping is a technique used in image processing and computer graphics 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.

Essentially, tone mapping addresses the problem of strong contrast reduction from the scene values (radiance) to the displayable range, while preserving the image details and color appearance. This is important to appreciate the original scene content.

### Inputs

**Image** HDR (High Dynamic Range) 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 (Low Dynamic Range) image.

## Z Combine Node

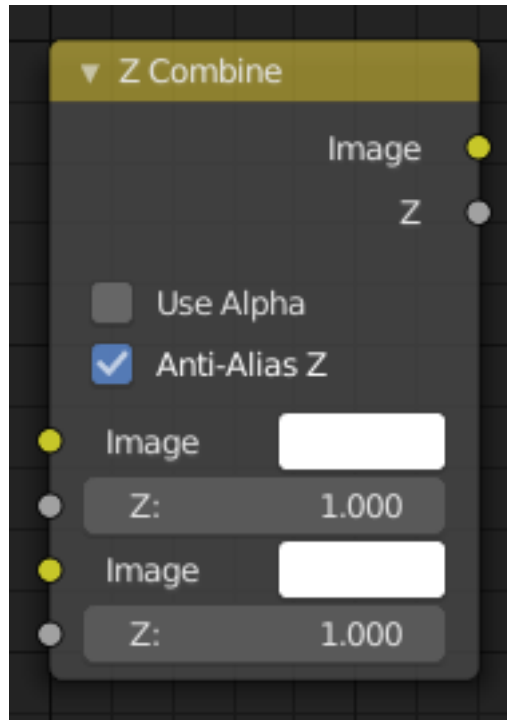


Fig. 213: Z Combine Node.

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

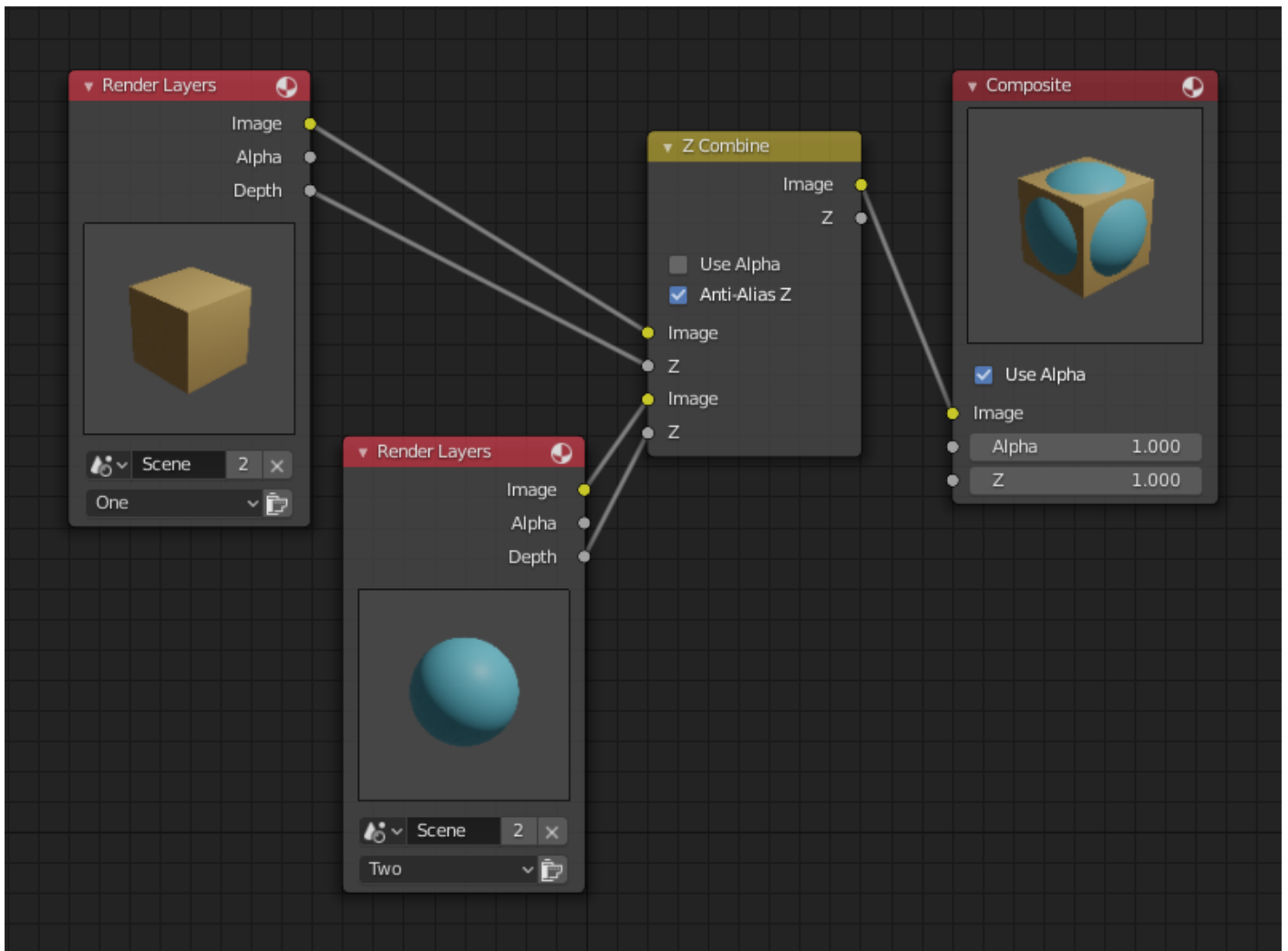


Fig. 214: Choosing closest pixels.

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.

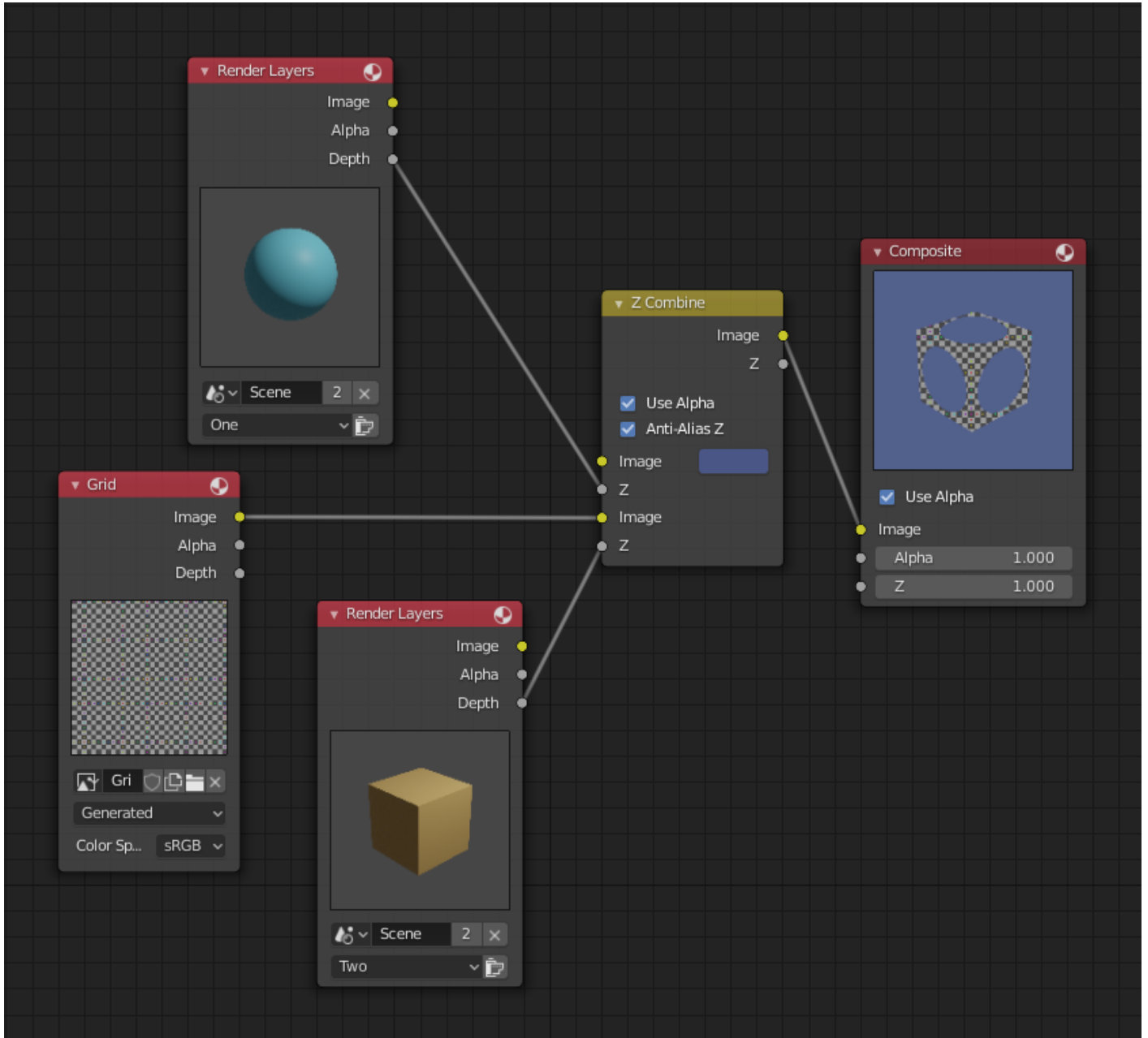


Fig. 215: Mix and match images.

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.

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



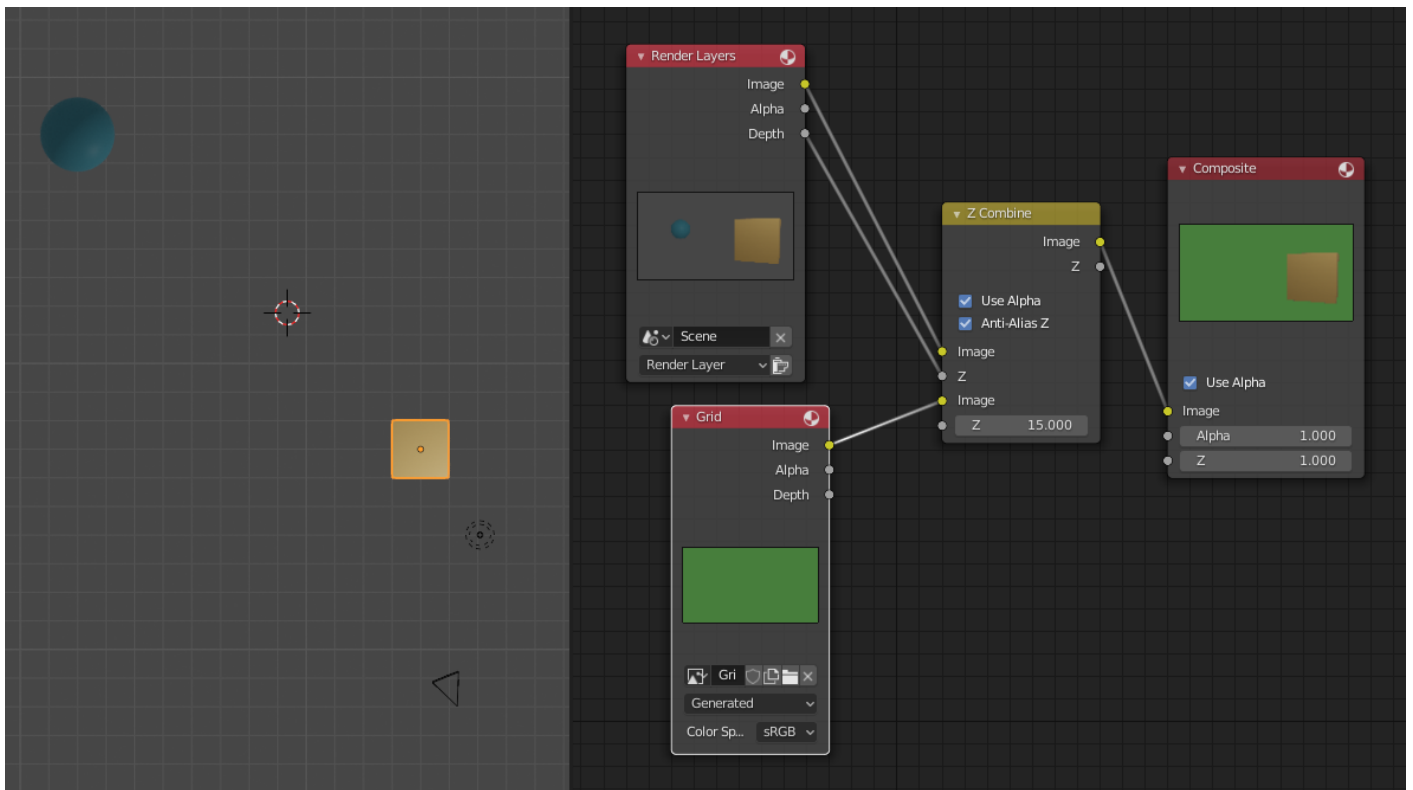


Fig. 216: Z Combine in action.

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.

### Converter Nodes

As the name implies, these nodes convert the colors or other properties of various data (e.g. transparency) in some way.

They also split out or re-combine the different color channels that make up an image, allowing you to work on each channel independently. Various color channel arrangements are supported, including traditional RGB, HSV and HDMI (High Definition Media Interface) formats.

## Alpha Convert Node

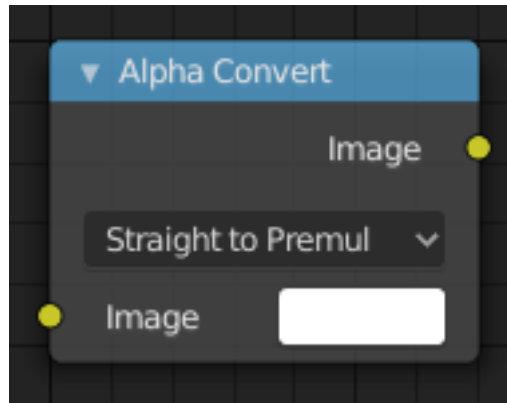


Fig. 217: Alpha Convert Node.

The *Alpha Convert Node* converts the alpha channel interpretation 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 image input.

### Properties

#### Mapping

**Straight to Premul, Premul to Straight** Conversion in both directions. Premul. stands for Premultiplied. For details on the difference between both ways to store alpha values see *Alpha Channel*.

### Outputs

**Image** Standard image output.

## Color Ramp Node

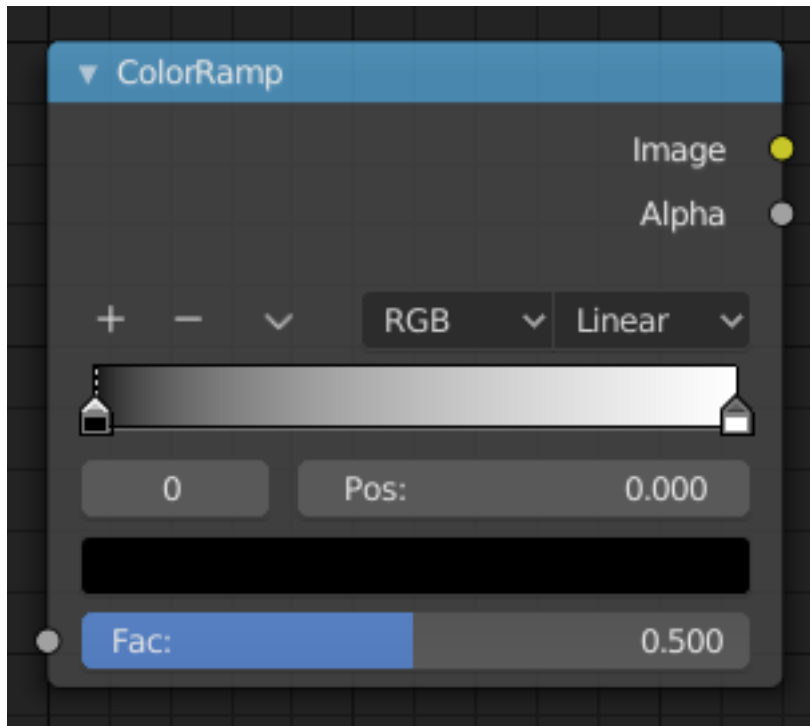


Fig. 218: Color Ramp Node.

The Color Ramp Node is used for mapping values to colors with the use of a gradient.

### Inputs

**Factor** The Factor input is used as an index for the color ramp.

### Properties

**Color Ramp** For controls see *Color Ramp Widget*.

### Outputs

**Image** Standard image output.

**Alpha** Standard alpha output.

### Examples

#### Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to create an alpha mask, or a mask that is overlaid on top of another image. Such a mask allows you to select parts of the background to be show through.

In the map above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

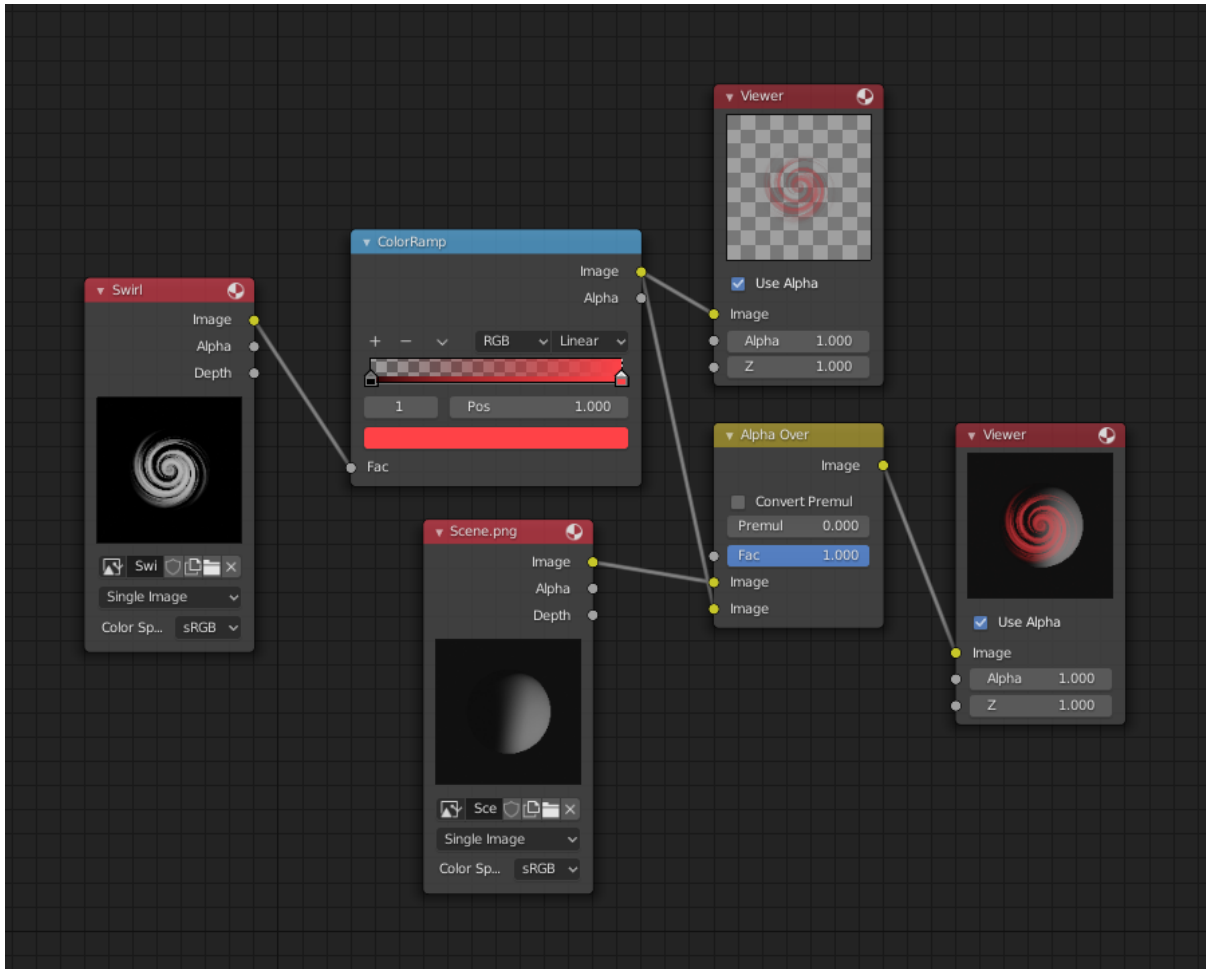


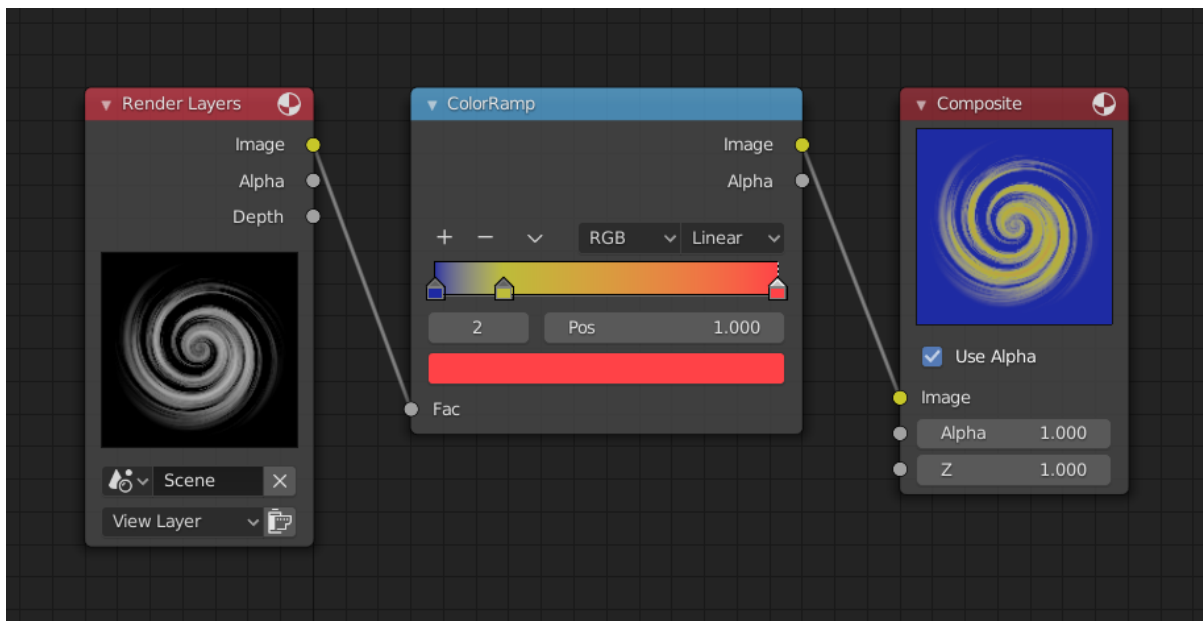
Fig. 219: Using the Color Ramp node to create an alpha mask.

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 puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp outputs the color at the left end of the gradient, which is set to transparent. The Color Ramp image is fully red and opaque where the image is white (which is 1).

You can verify that the output image mask is indeed transparent by overlaying it on top of another image.

### 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 (alpha of 1). Where the image is black, Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, 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/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from its composite color channels.

These nodes can be used to manipulate each color channel independently. Each type is differentiated in the applied *Color Space*.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of Cycles combine and separate nodes are added for HSV and vectors (XYZ).

The Combine nodes can also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operations could easier be executed with the *Color Nodes*.

## Separate/Combine RGBA Nodes



Fig. 220: Combine RGBA Node.

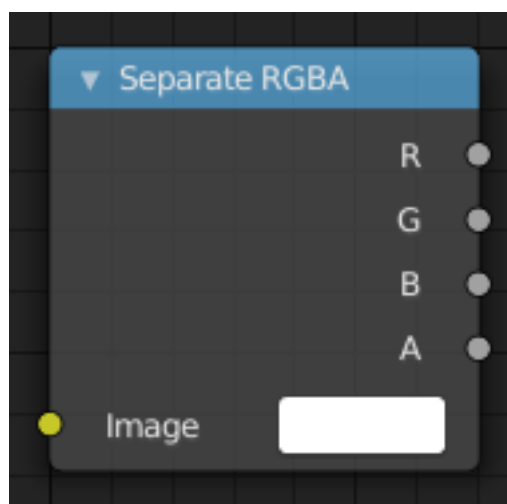


Fig. 221: Separate RGBA Node.

### Input/Output

**Image** Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)
- A (Alpha)

### Properties

This node has no properties.

## Examples

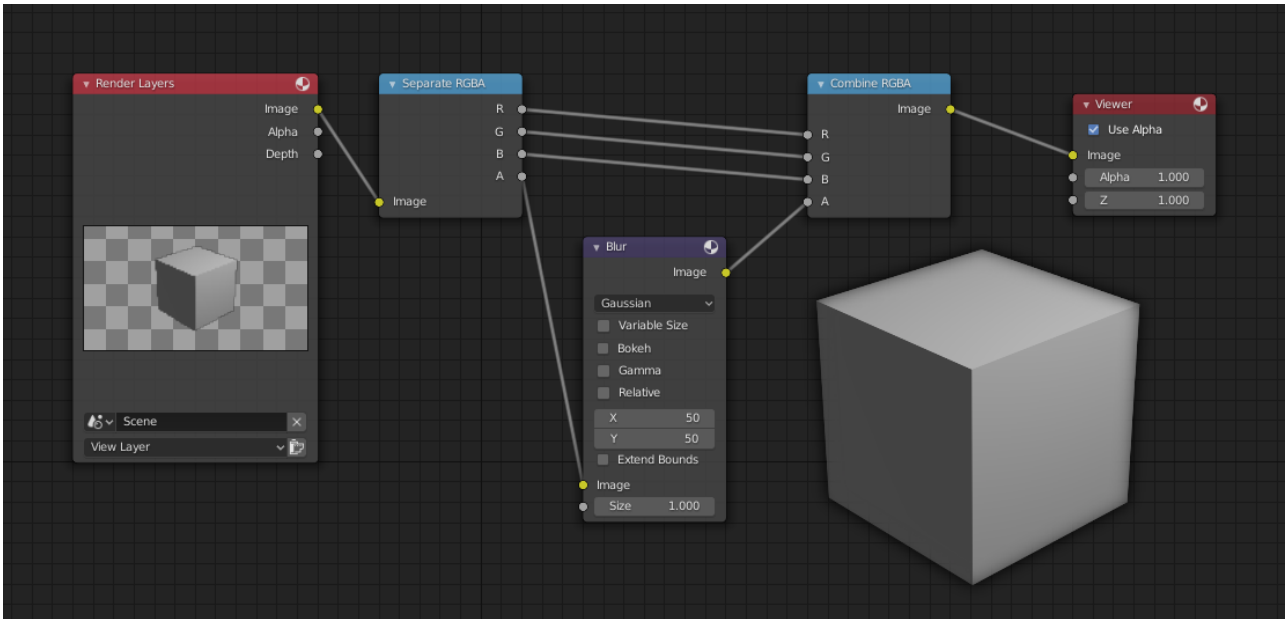


Fig. 222: An example of blurring the alpha channel.

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.

## Separate/Combine HSVA Nodes

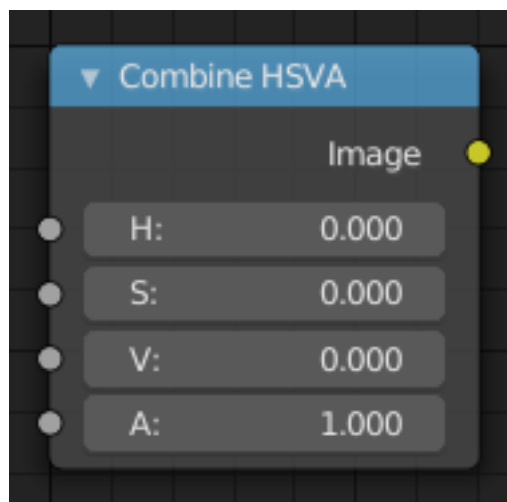


Fig. 223: Combine HSVA Node.

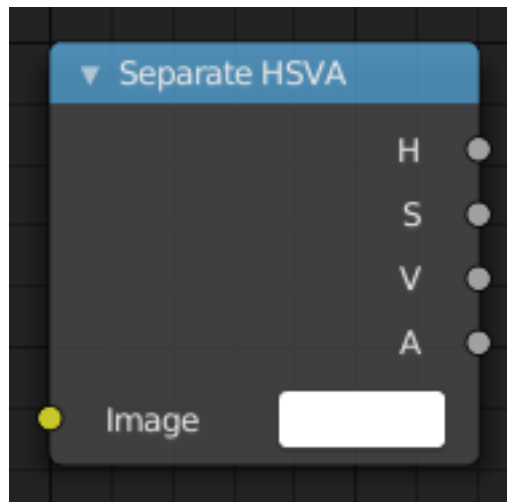


Fig. 224: Separate HSVA Node.

### Input/Output

**Image** Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

### Properties

This node has no properties.

### Separate/Combine YUVA Nodes

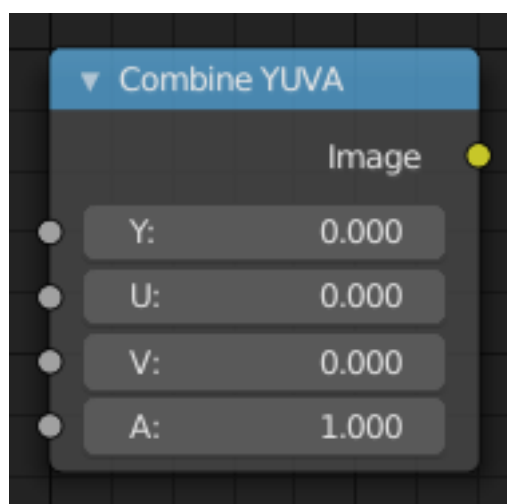


Fig. 225: Combine YUVA Node.





Fig. 226: Separate YUVA Node.

### Input/Output

**Image** Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

### Properties

This node has no properties.

### Separate/Combine YCbCrA Node

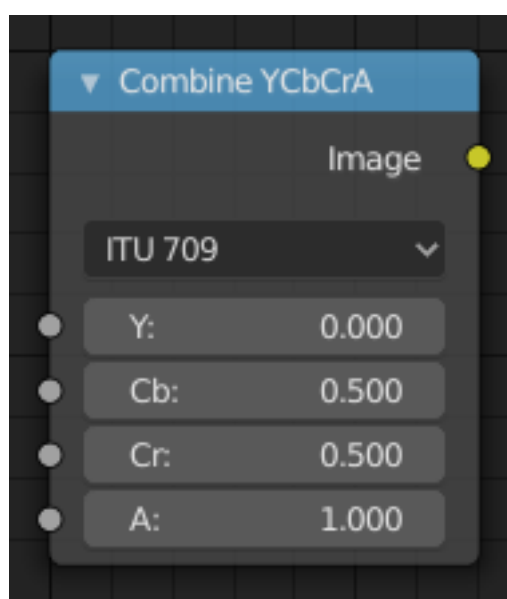


Fig. 227: Combine YCbCrA Node.

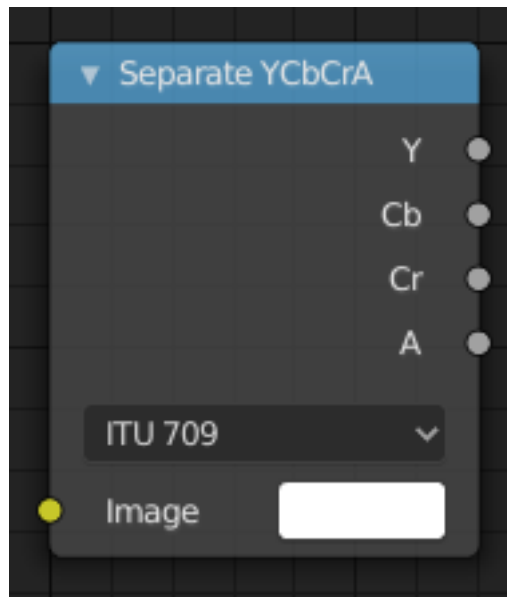


Fig. 228: Separate YCbCrA Node.

### Input/Output

**Image** Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

### Properties

**Mode** ITU 601, ITU 709, Jpeg

### Examples

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.

---

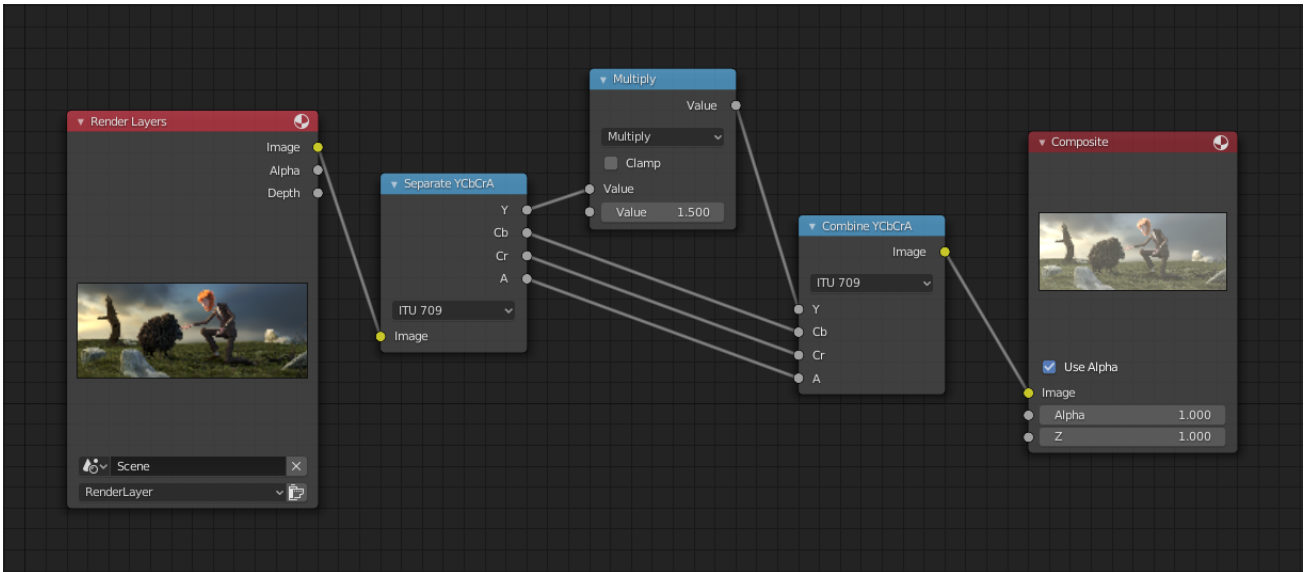


Fig. 229: An example of the scaling the Luminance channel.

## ID Mask Node

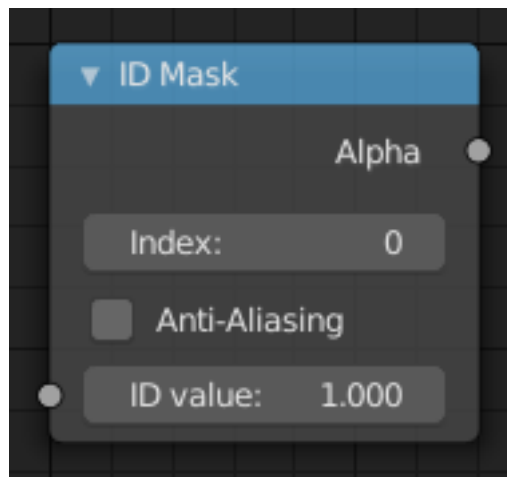


Fig. 230: ID Mask Node.

The *ID Mask Node* can be used to access an alpha mask per object or per material.

### 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, *Object Index* or *Material Index* render pass has to be enabled.

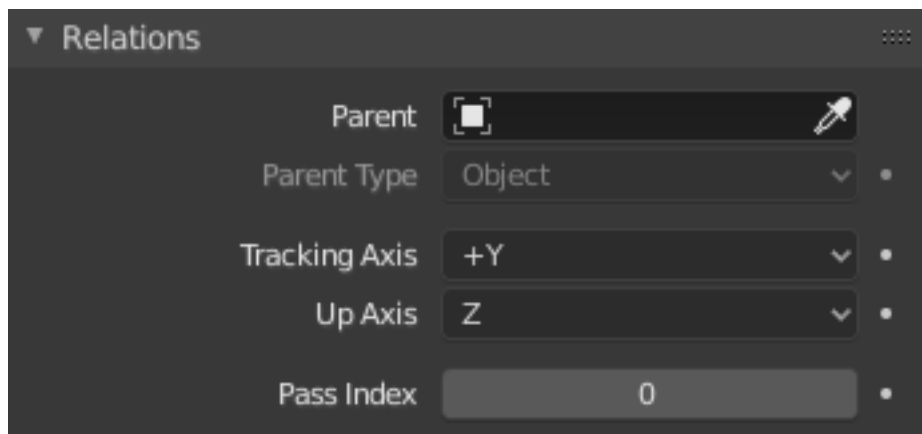


Fig. 231: Object Pass Index.

## 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.

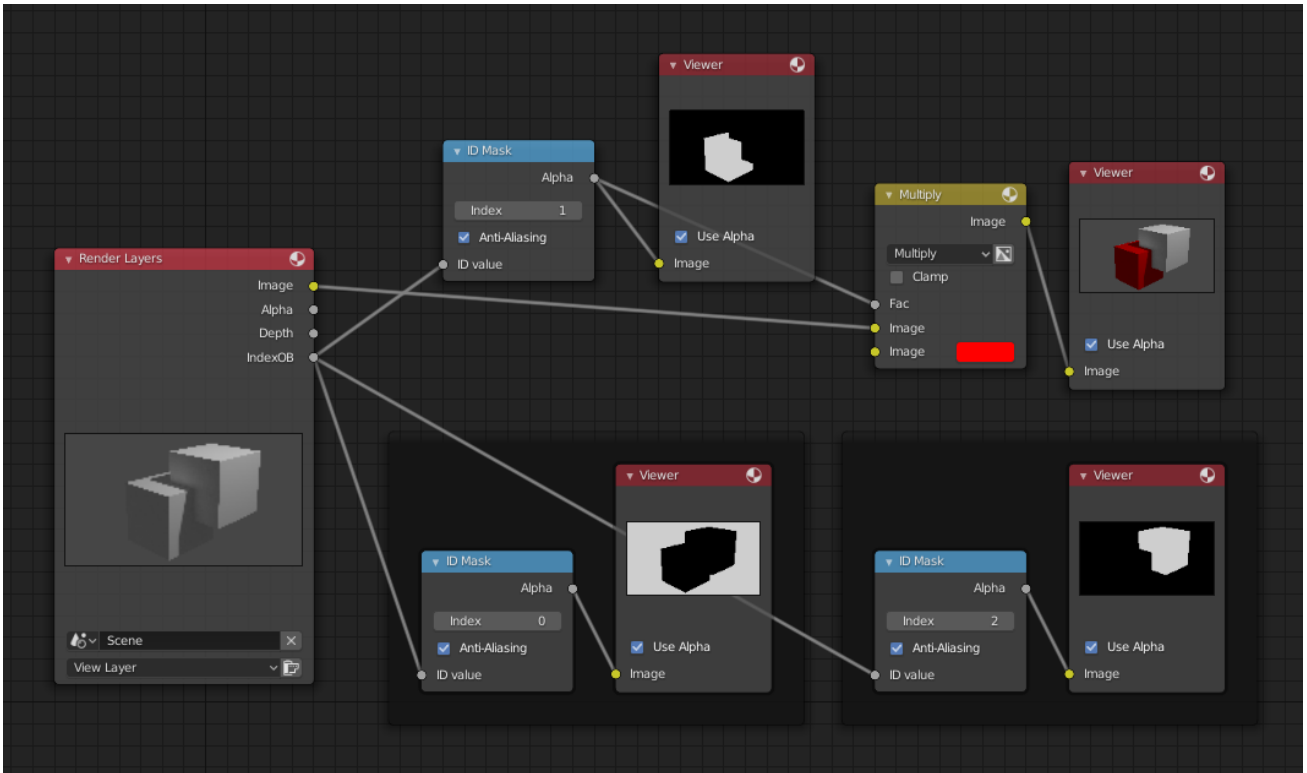


Fig. 232: ID Mask node example.

## Math Node

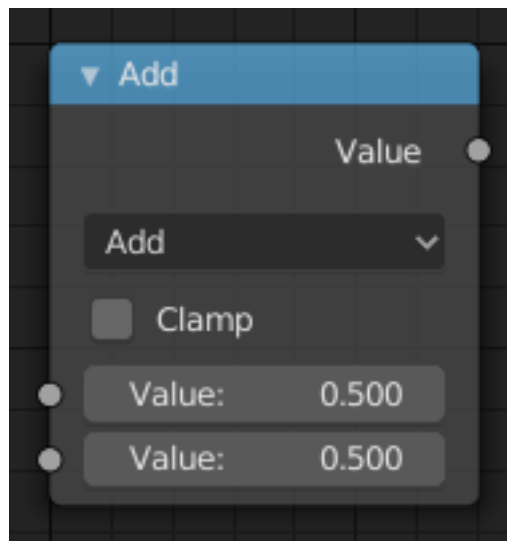


Fig. 233: Math node.

The *Math Node* performs math operations.

## Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Addend* input is only available in 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 with 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** Round 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** *Fraction*.

**Modulo** Outputs the remainder once the first value is divided by the second value.

**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** Round the input value to down to the nearest integer multiple of *Increment*.

**Ping-pong** The output value is moved between 0.0 and the *Scale* based on the input value.

### 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

This example has one scene input by the top *Render Layers* node, which has a cube that is about 10 units from the camera. The bottom *Render Layers* node inputs a scene with 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 divide the Z-buffer by 20 (multiply by 0.05, as shown in the Size field) and clamped to be a min/max of 0.0/1.0 respectively.

For the minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane *Render Layers* Z are infinite (mapped to 1.0) for the right side, so they are chosen.

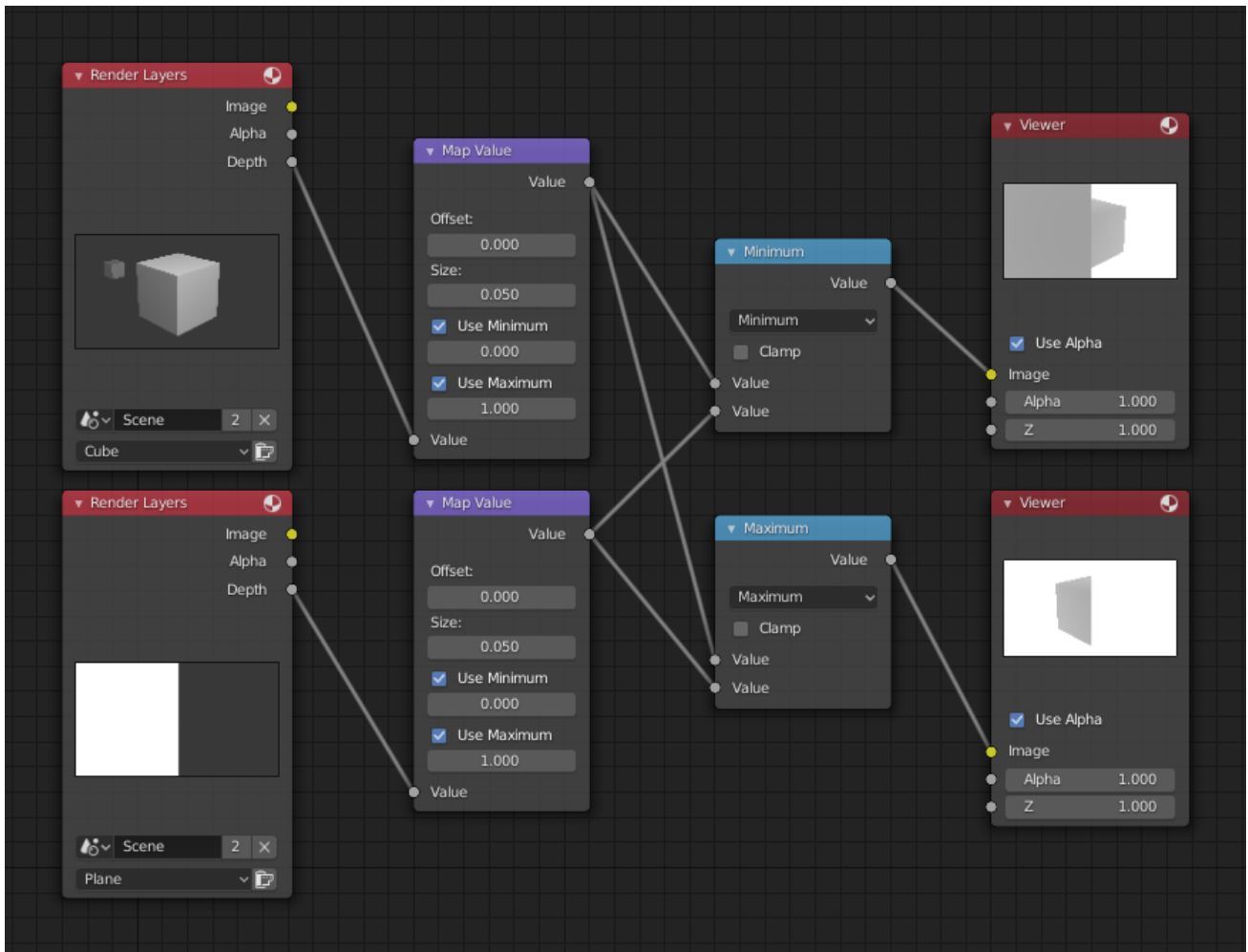


Fig. 234: Minimum and maximum function example.



## Using Sine Function to Pulsate

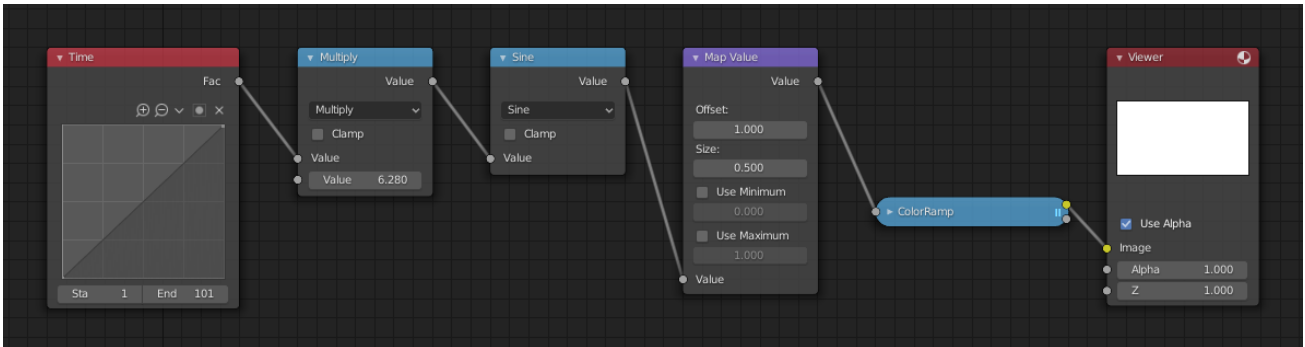


Fig. 235: Using sine function example.

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  $2 \times \pi$  (6.28) and converted to 1.0 by the Sine function, since  $\sin(2 \times \pi / 4) = \sin(\pi / 2) = +1.0$ .

Since the sine function can put out 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 one-half (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

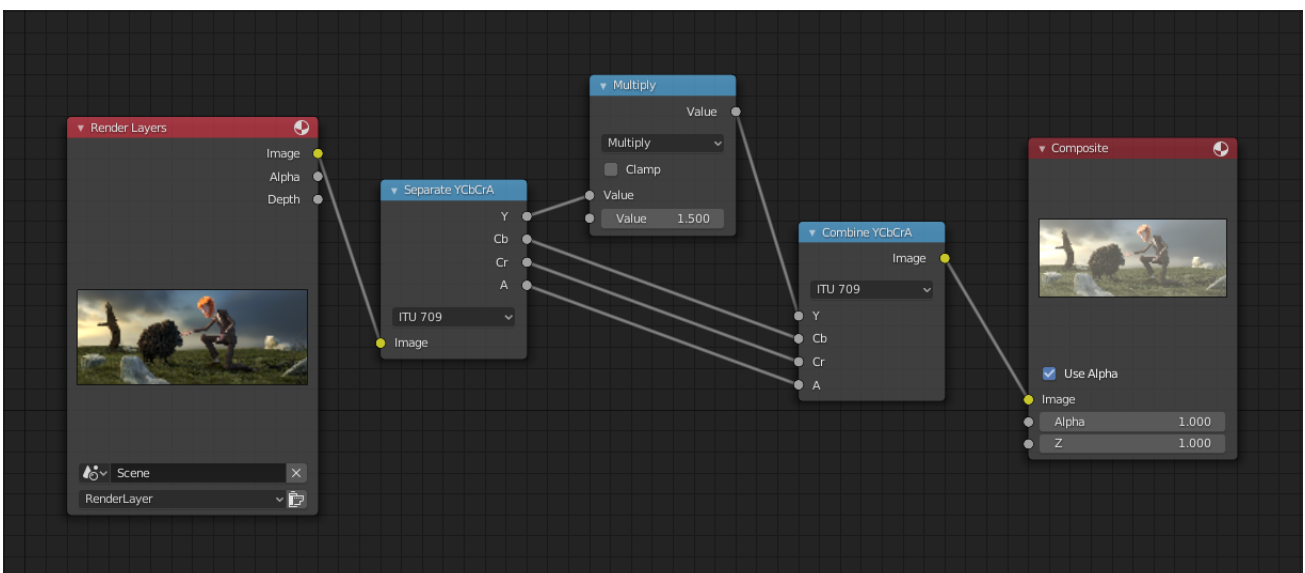


Fig. 236: Scaling a channel example.

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 *Bright/Contrast* node that might give simpler control over brightness.

### Restrict Color Selection (Posterization)

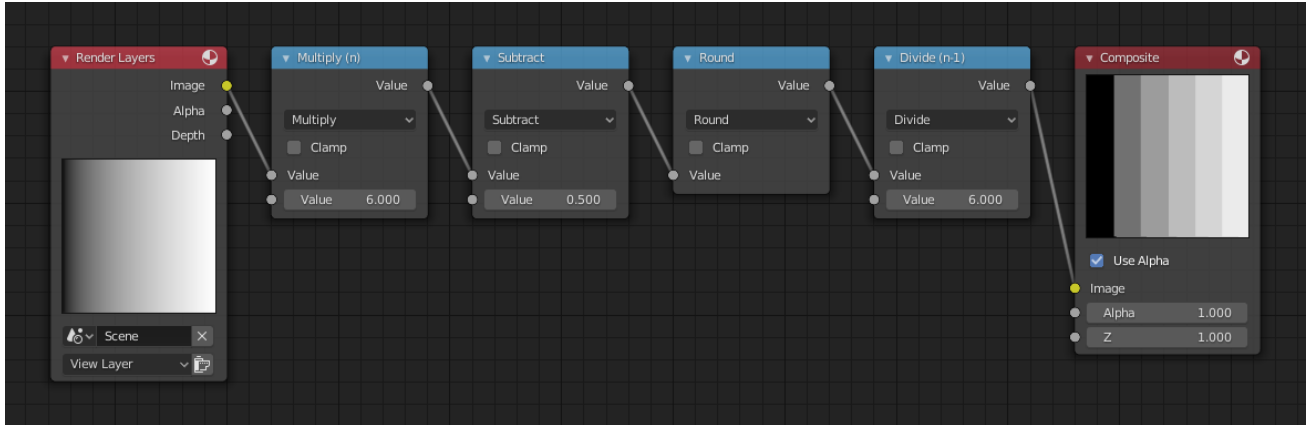


Fig. 237: Posterization example.

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:  $round(x \times n - 0.5) / (n - 1)$ , 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 split it into separate RGB channels using *Separate/Combine RGBA* nodes and perform this operation on each channel independently.

### RGB to BW Node

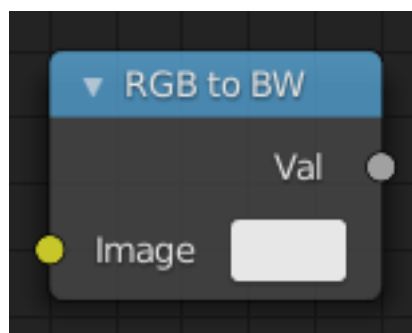


Fig. 238: RGB to BW Node.

The *RGB to BW Node* maps an RGB color image to a gray-scale by the luminance.

## Inputs

**Image** Color image input.

## Properties

This node has no properties.

## Outputs

**Value** Gray-scale value output.

## Set Alpha Node

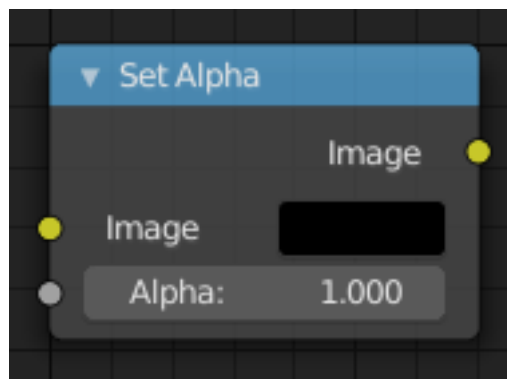


Fig. 239: Set Alpha Node.

The *Set Alpha Node* adds an alpha channel to an image.

## Inputs

**Image** Standard image 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

This node has no properties.

## Outputs

**Image** Standard image 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.

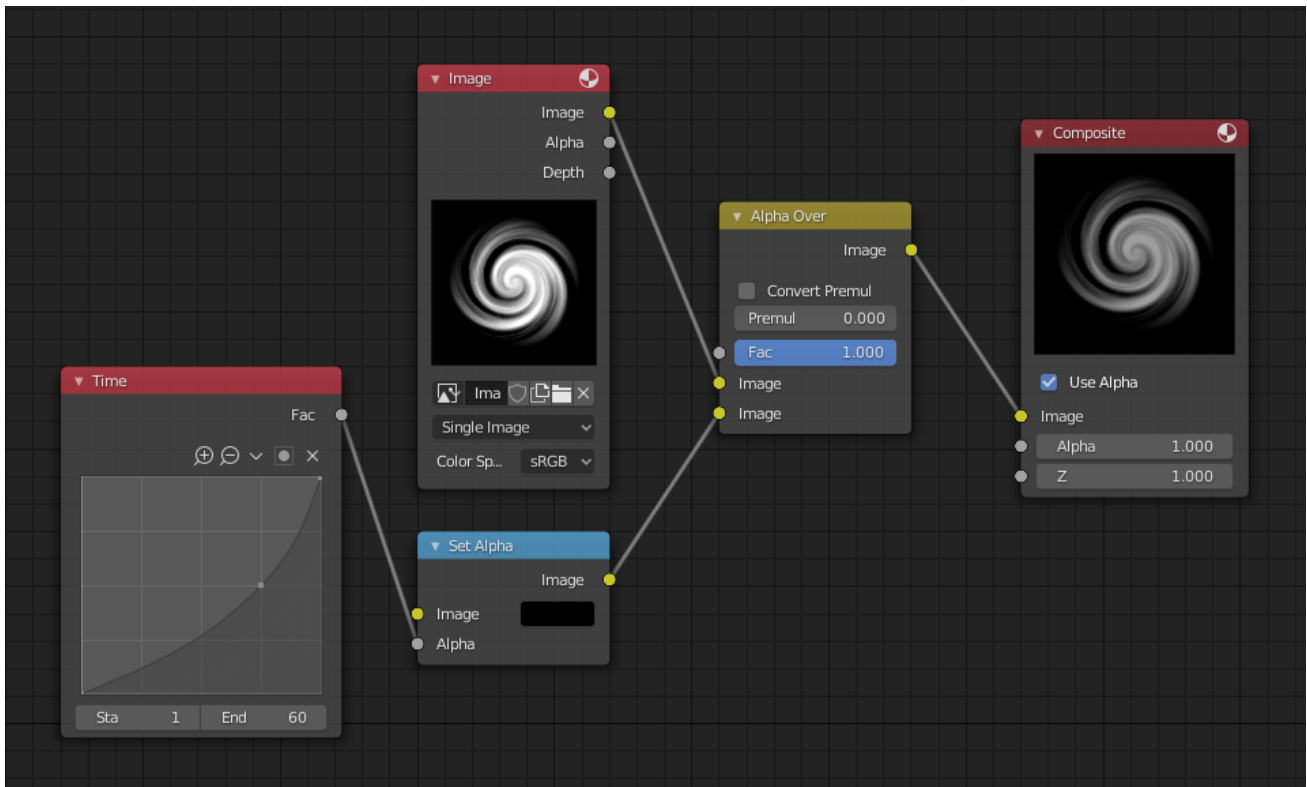


Fig. 240: Fade to black.

In the example above, the alpha channel of the swirl image is ignored. Instead, a *Time 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 node as shown below.

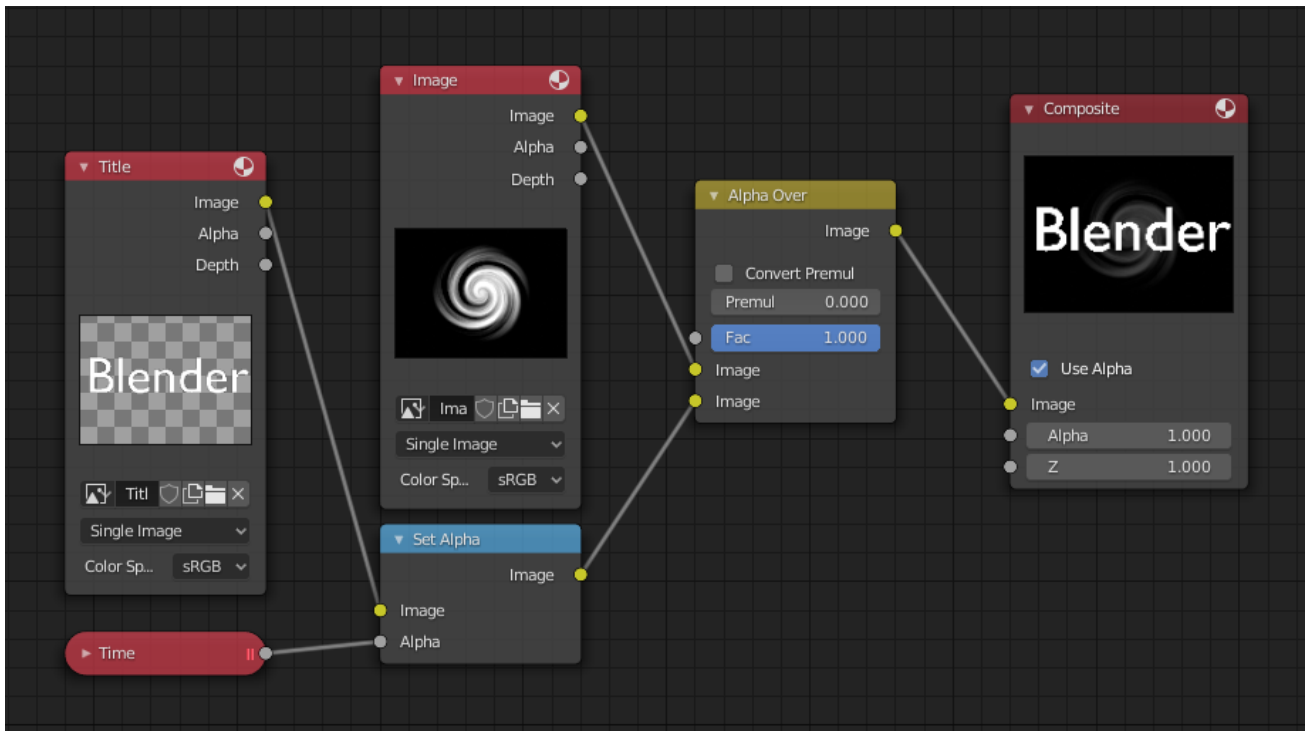


Fig. 241: Using Set Alpha to fade in a title.

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

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.

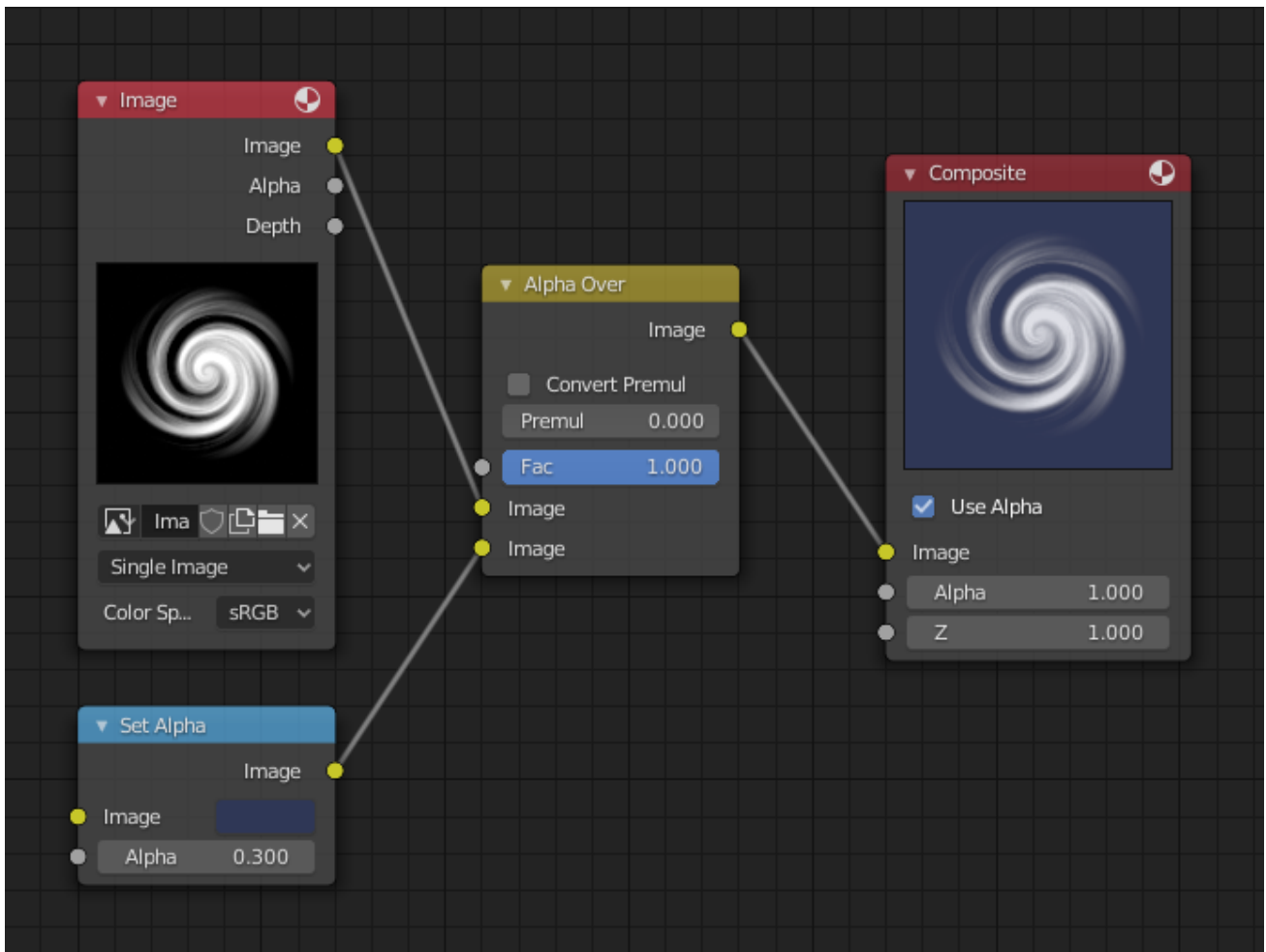


Fig. 242: Using Set Alpha to colorize an image.

## Switch View Node

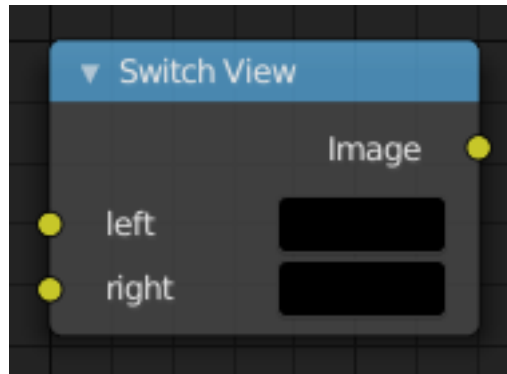


Fig. 243: Switch View Node.

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

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.

### Filter Nodes

Filters process the pixels of an image to highlight additional details or perform some sort of post-processing effect on the image.



Fig. 244: Compositor, Backdrop and Split Viewer Node.

### Bilateral Blur Node

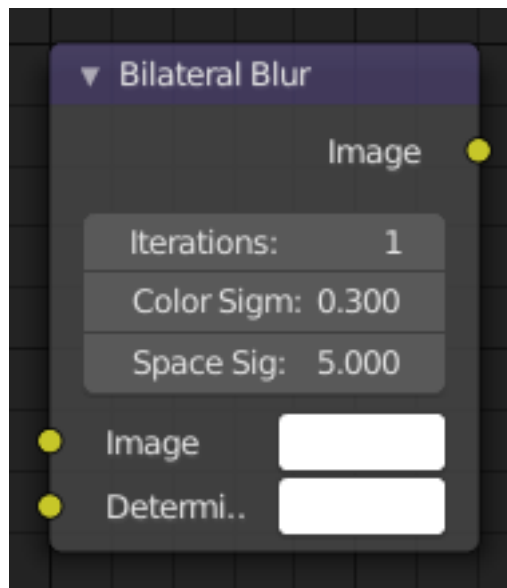


Fig. 245: Bilateral Blur Node.

The Bilateral Blur node performs a high-quality adaptive blur on the source image, allowing to blur images while retaining their 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 image 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 image output.

## Example

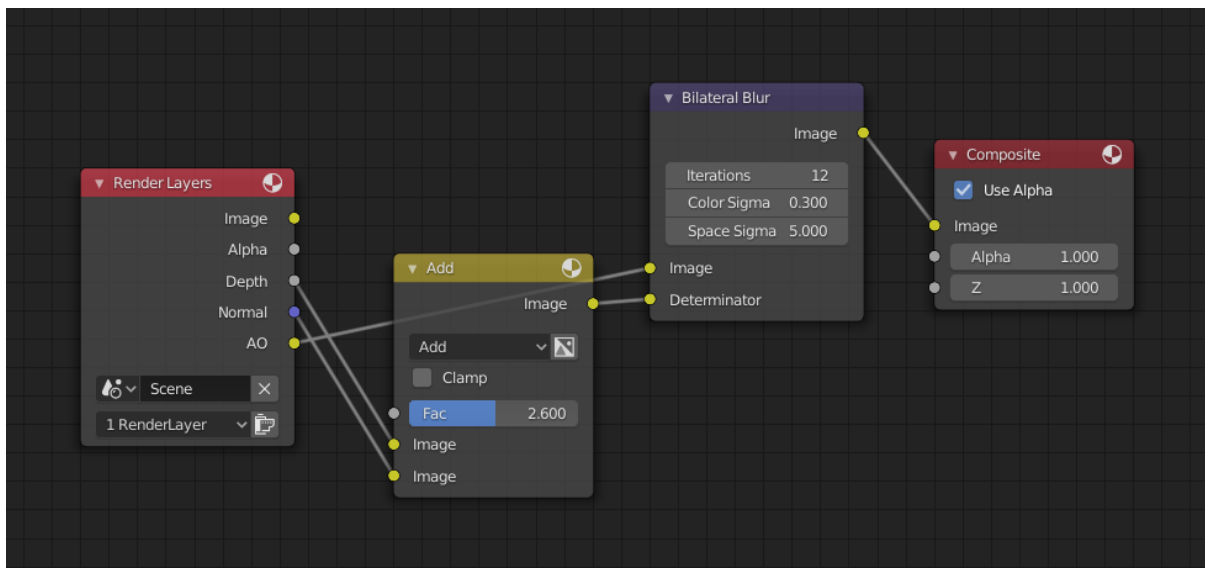
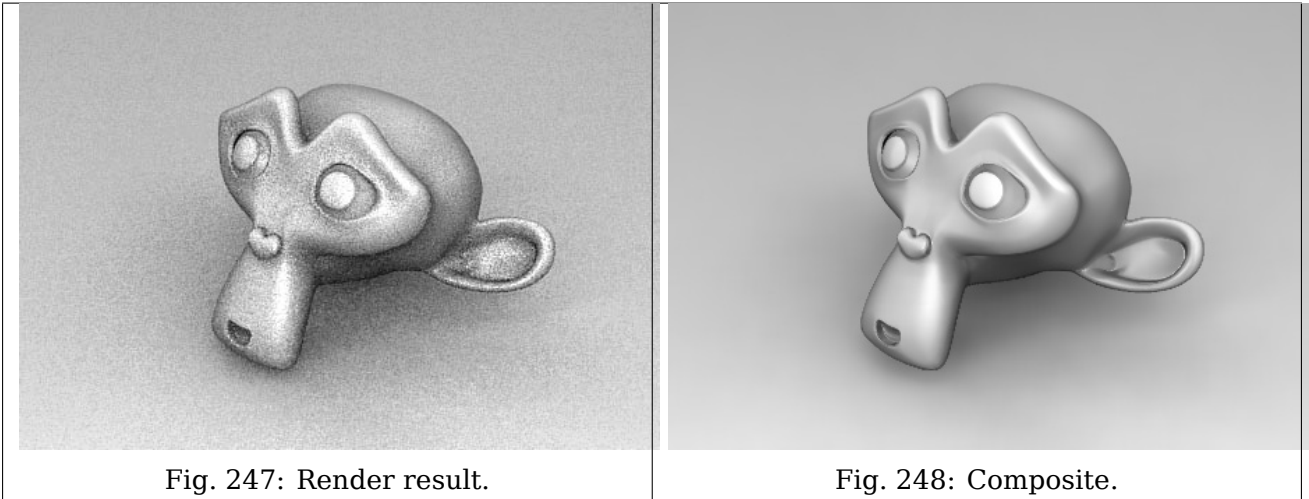


Fig. 246: Bilateral smoothed Ambient Occlusion. [blend-file example](#)



### Blur Node

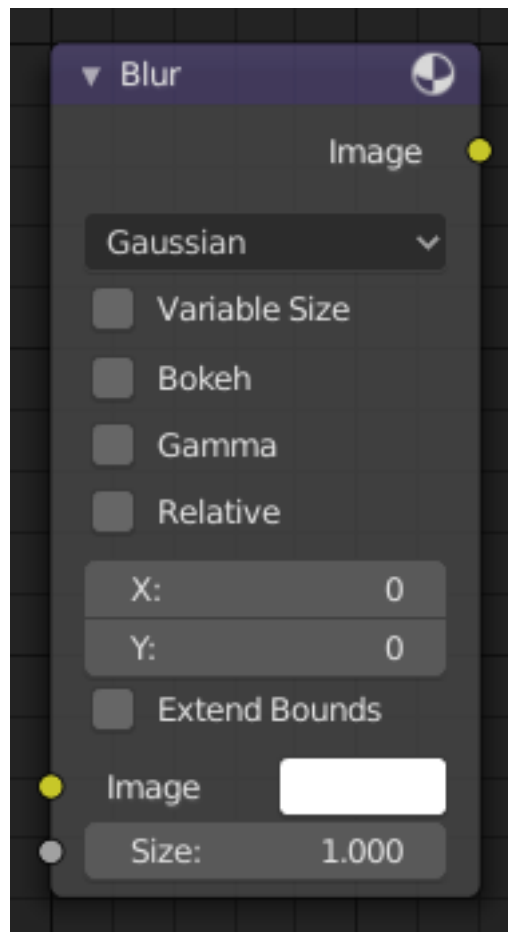


Fig. 249: Blur Node.

The Blur node blurs an image, providing several blur modes.

## Inputs

**Image** Standard image 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.

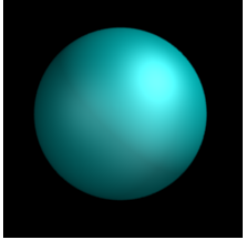
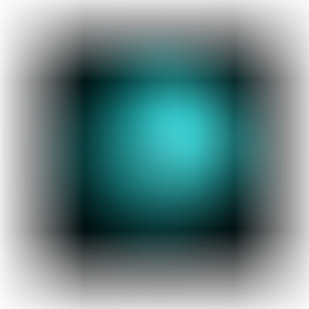







**Extend Bounds** Allows the image, that is being blurred, to extend past its original dimension.

## Outputs

**Image** Standard image output.

## Example

Table 13: Blur node blur modes using 20% of image size as XY, no Bokeh/Gamma.

 <p>Fig. 250: Original image.</p>	 <p>Fig. 251: Flat.</p>	 <p>Fig. 252: Tent.</p>
 <p>Fig. 253: Quadratic.</p>	 <p>Fig. 254: Cubic.</p>	 <p>Fig. 255: Gaussian.</p>
 <p>Fig. 256: Fast Gaussian.</p>	 <p>Fig. 257: Catmull-Rom.</p>	 <p>Fig. 258: Mitch.</p>

## Bokeh Blur Node

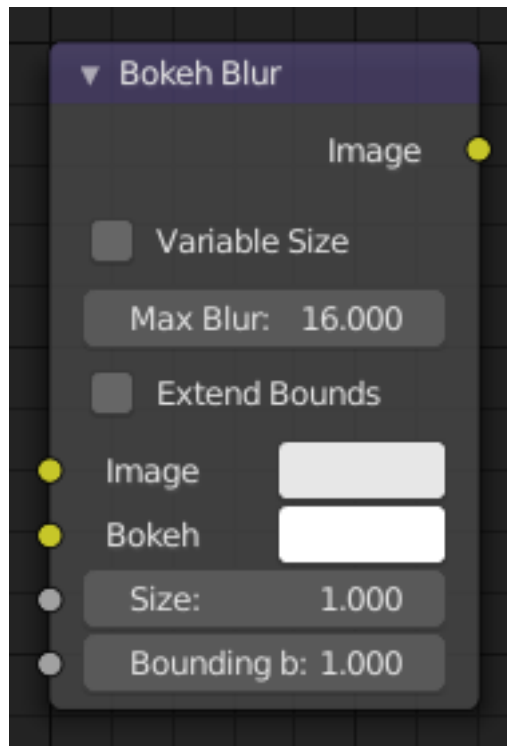


Fig. 259: Bokeh Blur Node.

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 as OpenCL support, calculation area restriction and masking.

### Inputs

**Image** Standard image 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 image 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.

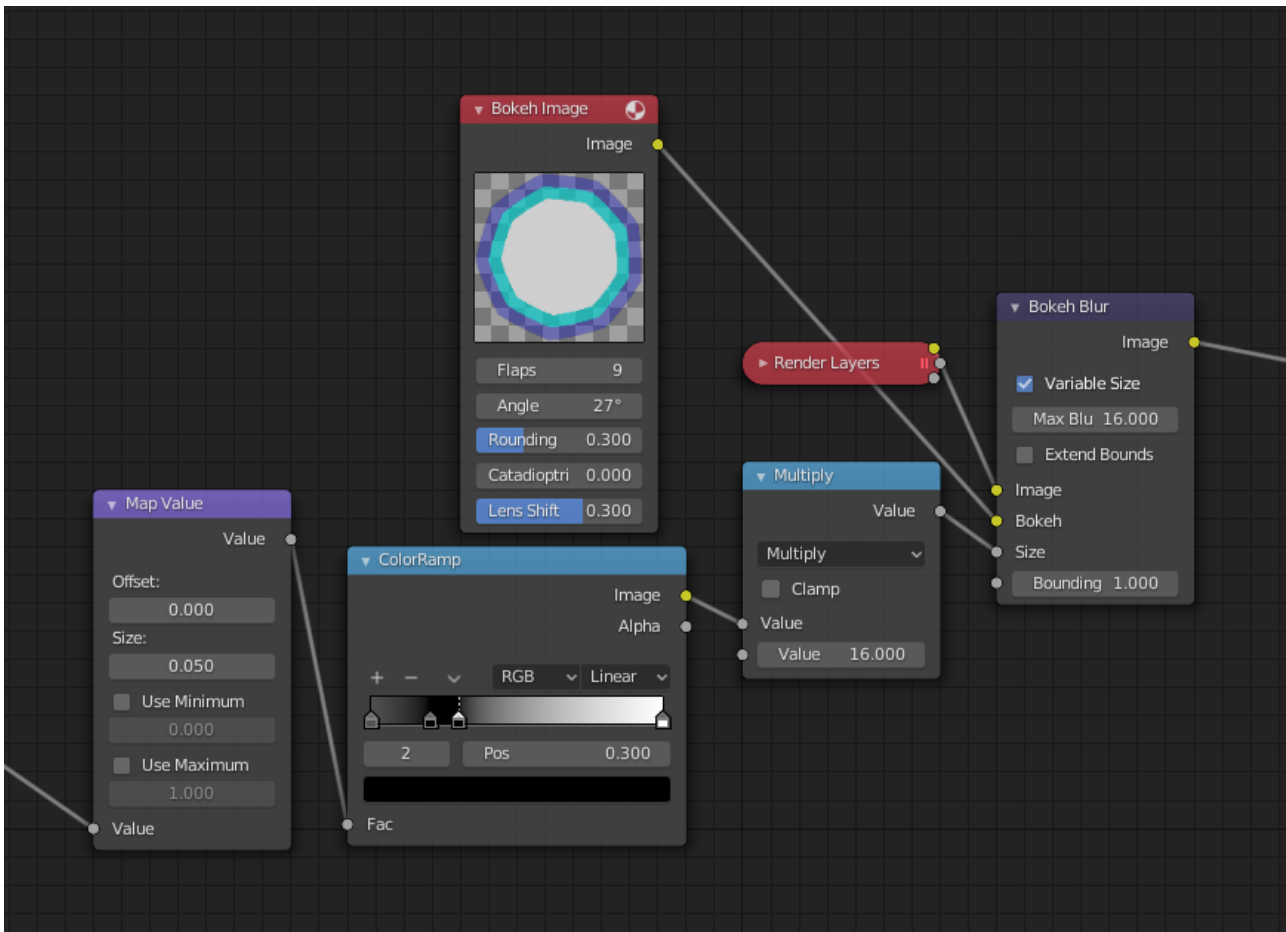


Fig. 260: Z pass used.

A manually created grayscale image can be used to define the sharp and blurry areas of a pre-existing image. Again, a *Multiply Node* can be used so that a blur value greater than one is used.

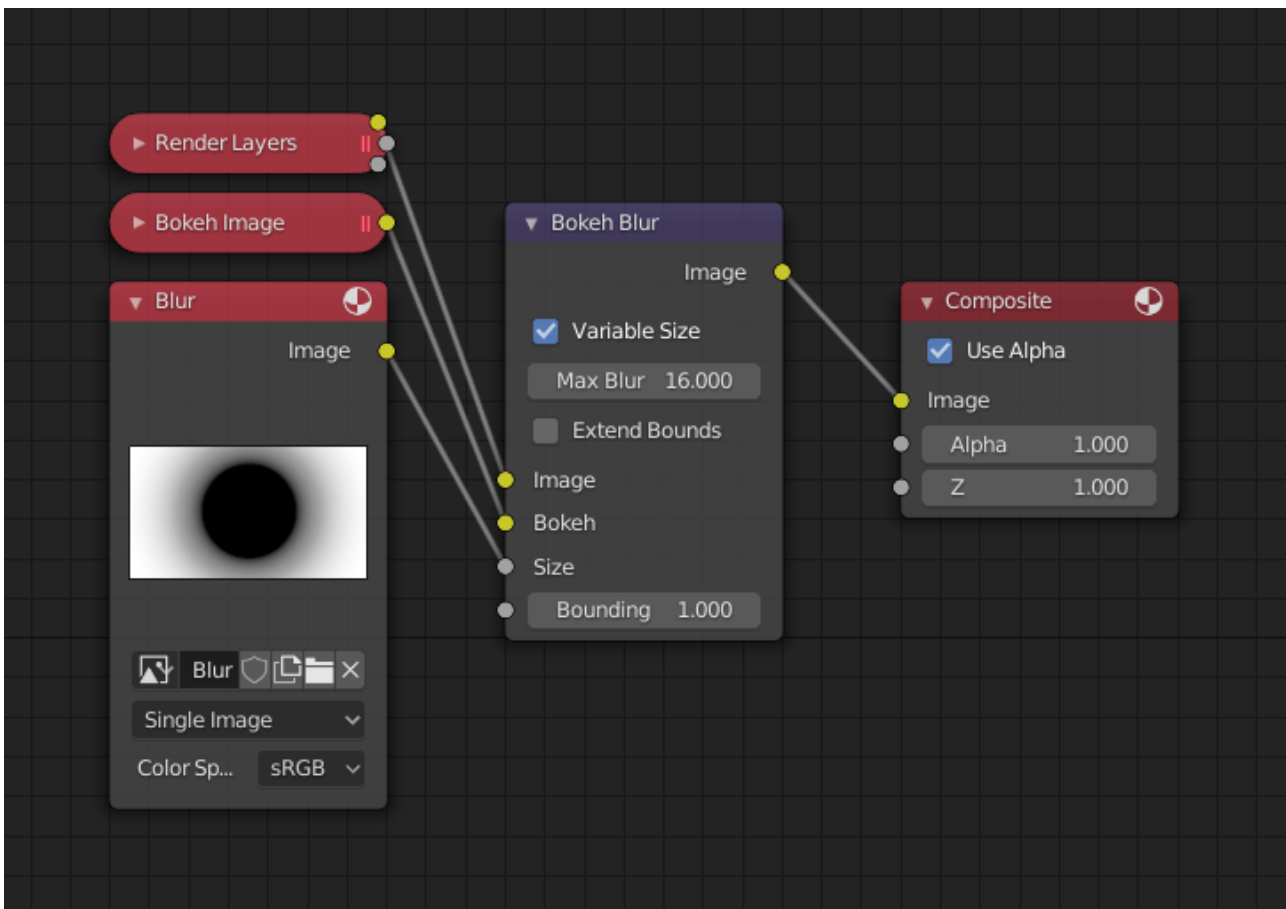


Fig. 261: Image used.



Fig. 262: Z pass used.

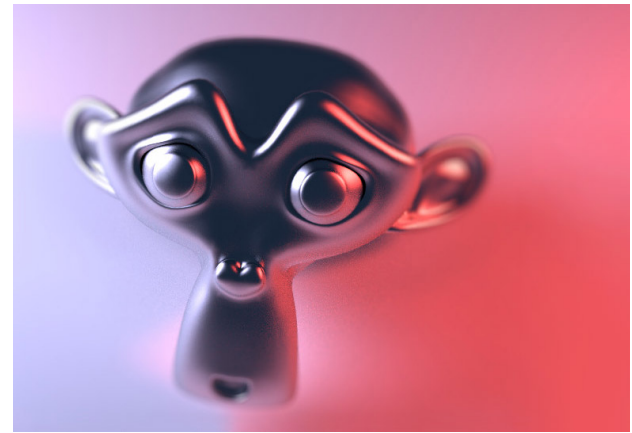


Fig. 263: Image used.



## Defocus Node

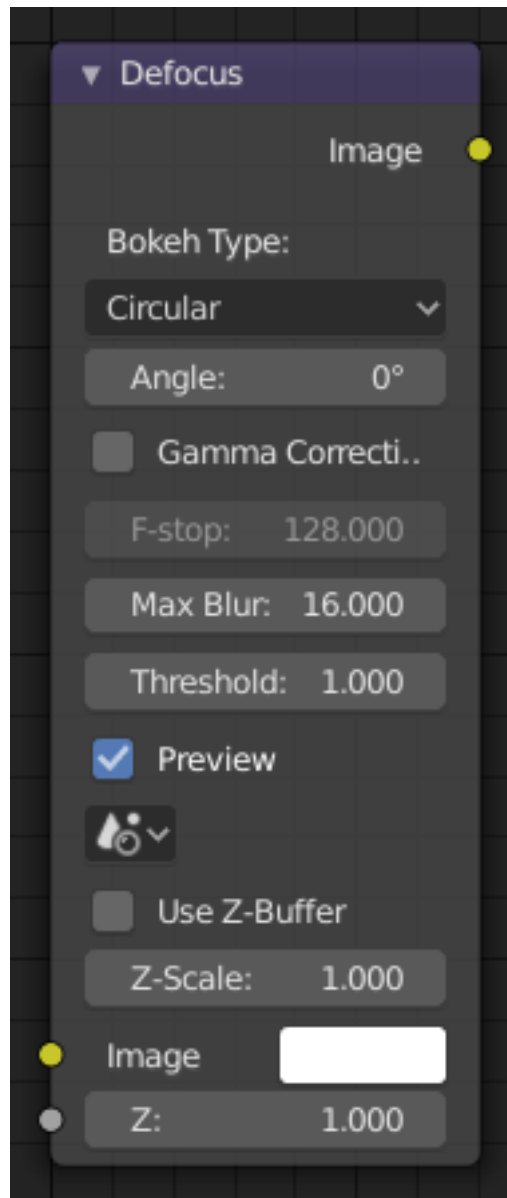


Fig. 264: Defocus Node.

The *Defocus Node* blurs areas of an image based on a map/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 image 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** 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** 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 image 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

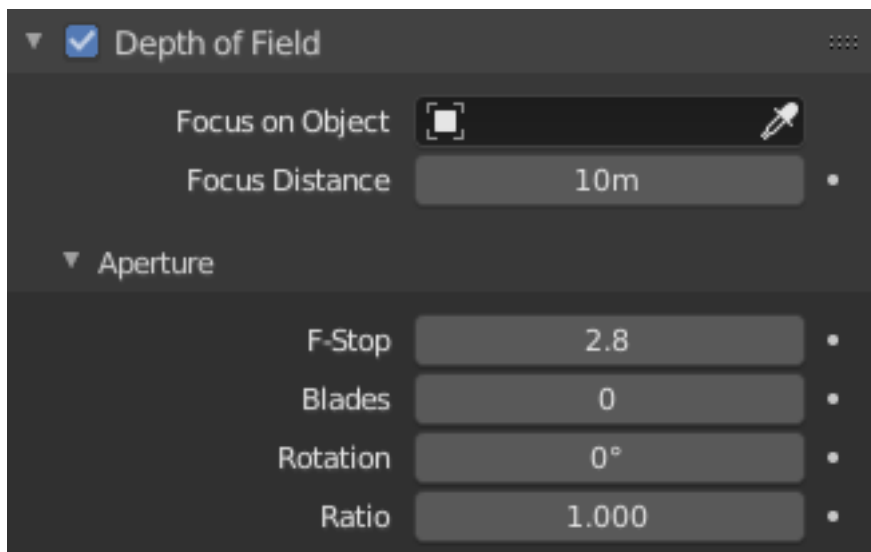


Fig. 265: Distance setting in the Camera Depth of Field panel.

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.

## Denoise Node

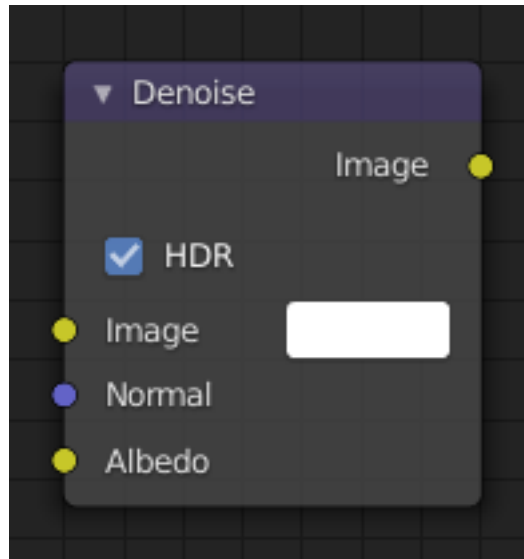


Fig. 266: Denoise Node.

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

**HDR** Preserve colors outside the 0 to 1 range.

### Outputs

**Image** Denoised image output.



Fig. 267: 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.

## Examples

### Despeckle Node

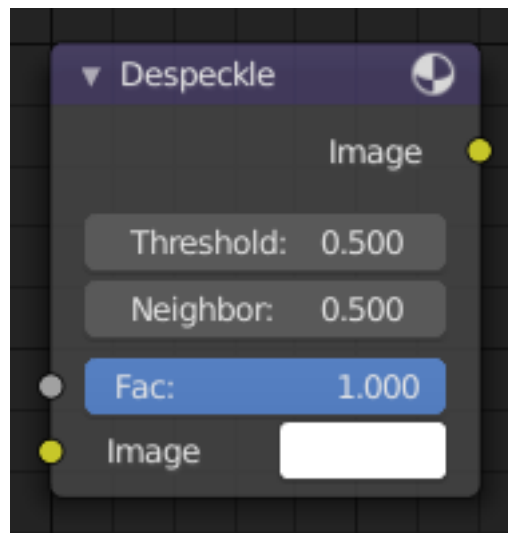


Fig. 268: Despeckle Node.

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 image 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 image output.

## Dilate/Erode Node

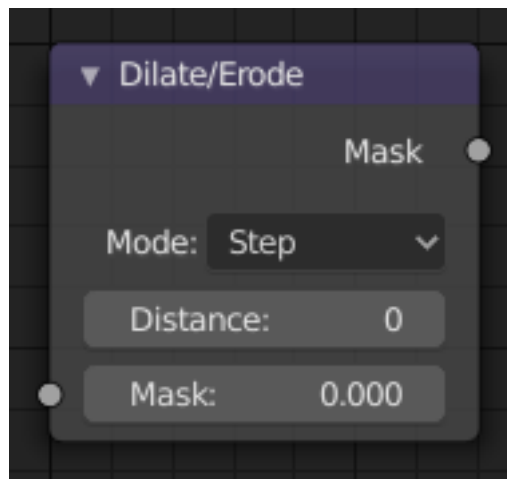


Fig. 269: Dilate/Erode Node.

The *Dilate/Erode node* provides a morphology (mathematical shape analysis) filter.

## Inputs

**Mask** Single color channel (or a black-and-white image) input.

## Properties

**Mode** Step, Threshold, Distance, Feather

**Distance** The Distance is the filter radius. A *positive* value of Distance dilates (expands) the influence of a pixel on its surrounding pixels. A *negative* value erodes (shrinks) its influence.

**Edge** Edge to inset. .. TODO2.8 Explain.

**Falloff** Falloff type the feather. .. TODO2.8 Explain.

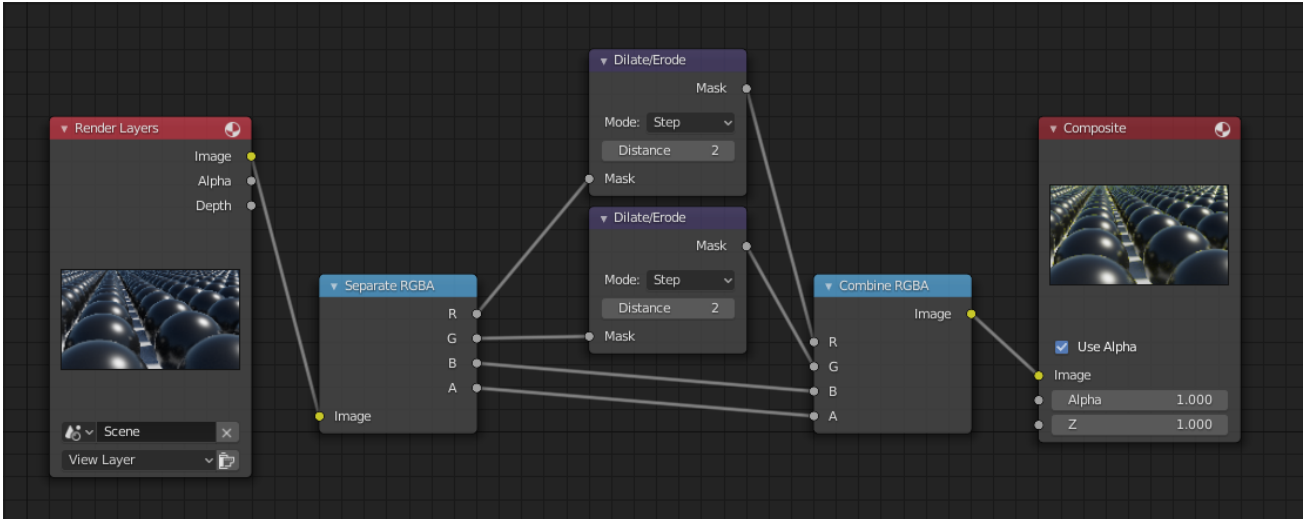


## Outputs

**Mask** The filtered mask output.

## Example

In this example, we wanted to take the rather boring array of ball bearings and add some variation to it. So, we dilated the red and eroded the green, leaving the blue alone. If we had dilated both red and green... (hint: red and green make yellow). The amount of influence is increased by increasing the *Distance* values. [Blend-file available here.](#)





## Directional Blur Node

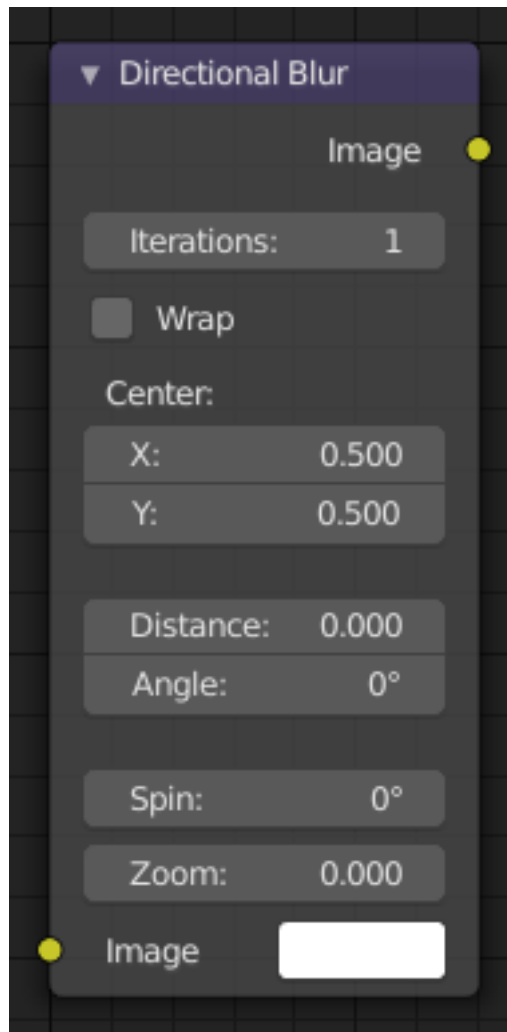


Fig. 270: Directional Blur Node.

Blurs an image in a specified direction and magnitude. Can be used to fake motion blur.

### Inputs

**Image** Standard image input.

### Properties

**Iterations** Controls how many times the image is duplicated to create the blur effect. Higher values give smoother results.

**Wrap** Wraps the image on the X and Y axis to fill in areas, that become transparent from the blur effect.

**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 image output.

## Filter Node

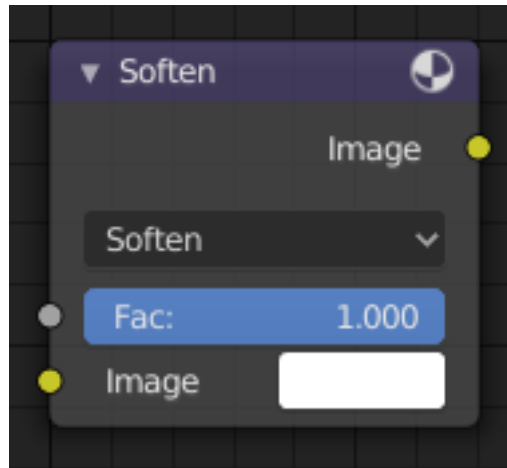


Fig. 271: Filter Node.

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 image 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.

**Sharpen** Increases the contrast, especially at edges.

**Laplace** Softens around edges.

**Sobel** Creates a negative image that highlights edges.

**Prewitt** Tries to do Sobel one better.

**Kirsch** Giving a better blending than Sobel or Prewitt, when approaching an edge.

**Shadow** Performs a relief, emboss effect, darkening outside edges.

## Outputs

**Image** Standard image output.

**Example**

Table 14: The Filter node has seven modes, shown here.

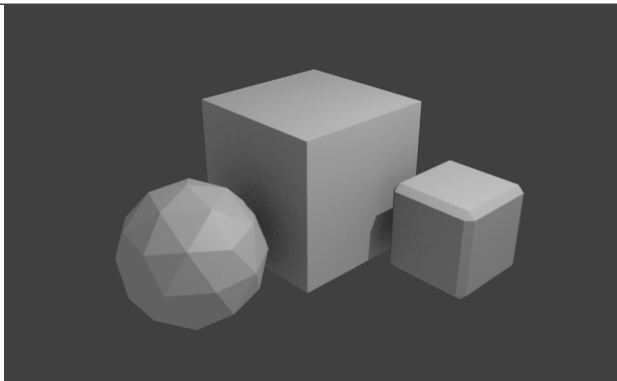


Fig. 272: Original image.

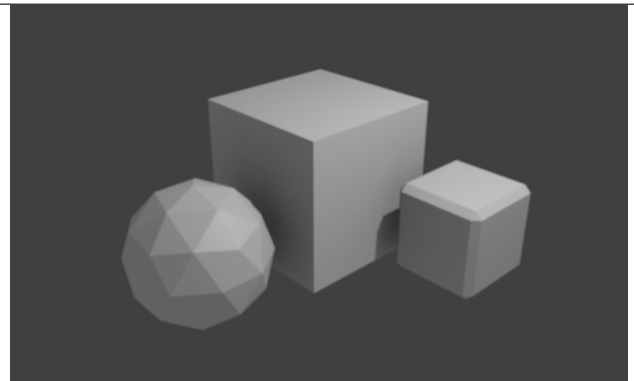


Fig. 273: Soften.

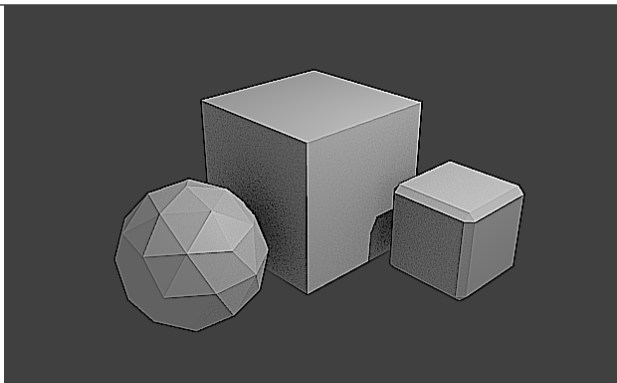


Fig. 274: Sharpen.

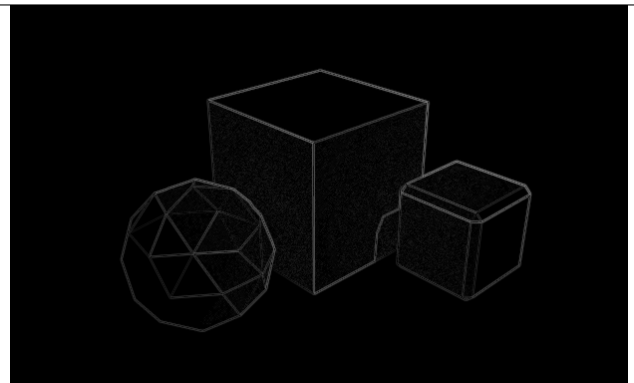


Fig. 275: Laplace.

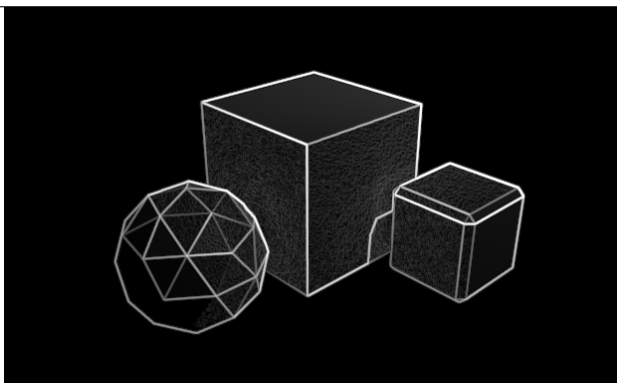


Fig. 276: Sobel.

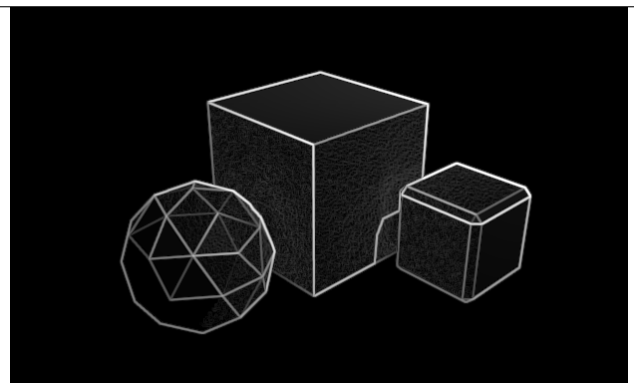


Fig. 277: Prewitt.

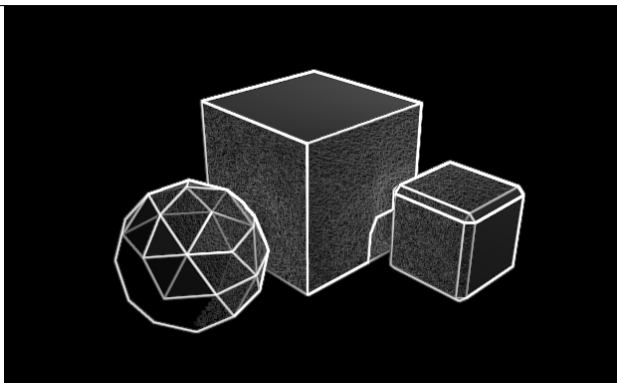


Fig. 278: Kirsch.

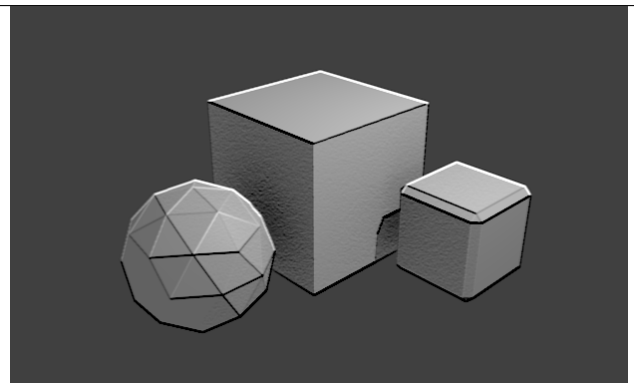


Fig. 279: Shadow.

## Glare Node

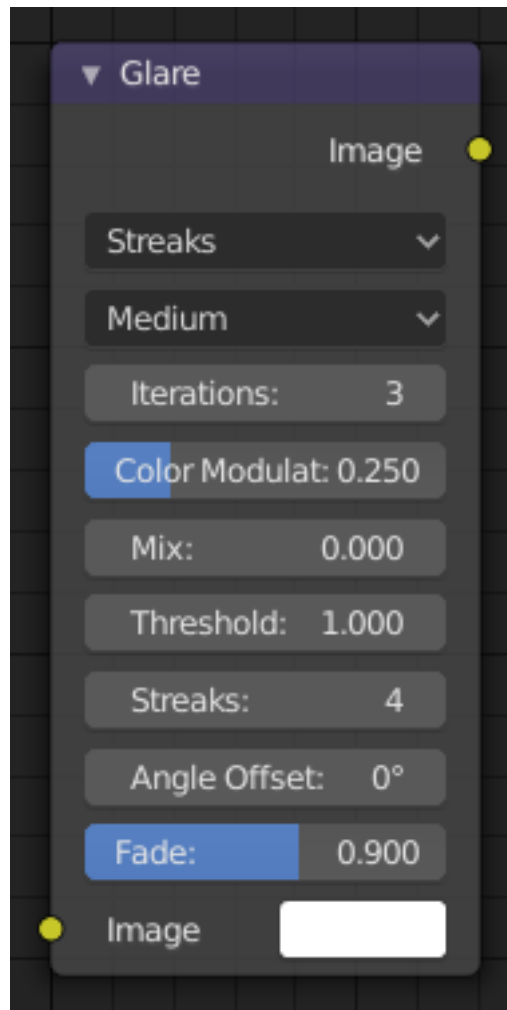


Fig. 280: Glare Node.

The *Glare node* is used to add lens flares, fog, glows around exposed parts of an image and much more.

### Inputs

**Image** Standard image input.

### Properties

#### Glare Type

**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** Looks similar to *Ghost*. However, it is much smaller in size and gives more of an atmospheric haze or “glow” around the image.

**Size** Scale of the glow relative to the size of the original bright pixels.

**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°.

### Common Options

**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 image output.

### Inpaint Node

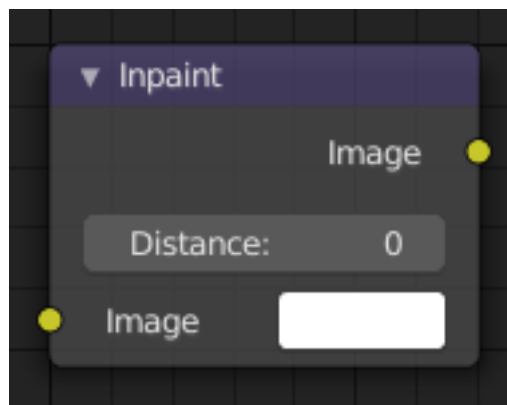


Fig. 281: Inpaint Node.

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 image input.

## Properties

**Distance** The number of times to extend the image.

## Outputs

**Image** Standard image output.

## Examples

The left image shows the “wire” in place and after chroma keying has been applied. You will see you’re left with a blank space – it’s shown as a black line here but it will be alpha in your Blender output.



Fig. 282: 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.

## Pixelate Node

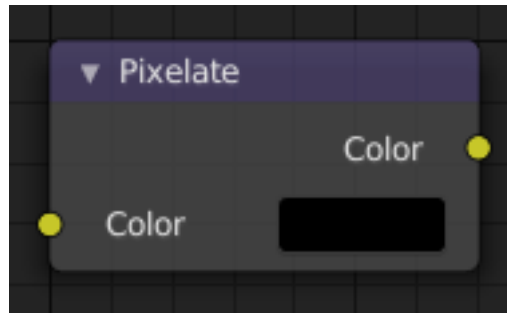


Fig. 283: Pixelate Node.

Add this node in front of a *Scale node* to get a pixelated (non-smoothed) image from the resultant upscaled image.

### Inputs

**Color** Standard image input.

### Properties

This node has no properties.

### Outputs

**Color** Standard image output.

### Example

Open an image you would like to pixelate using an Image node. Add two Scale nodes between the input and output *Add* → *Distort* → *Scale*. Change the values of X and Y to 0.2 in the first scale box and to 5 in the second. The composited image will appear unchanged. Now add a Pixelate node between the two Scale nodes. The result will be a pixelated image.





## Sun Beams Node

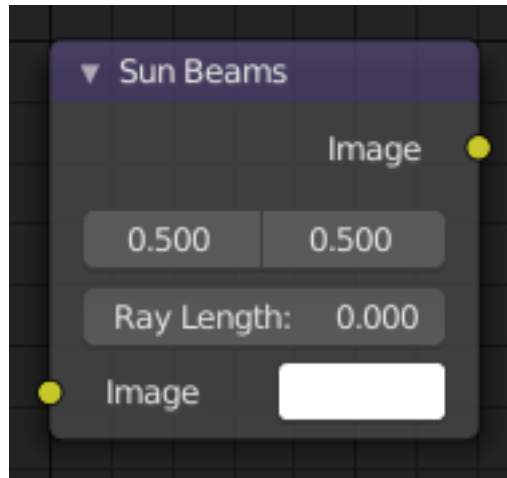


Fig. 284: Sun Beams Node.

The Sun Beams node provides a computationally cheap way of creating the name giving effect based on the 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 image 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 image 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.



## Vector (Motion) Blur Node

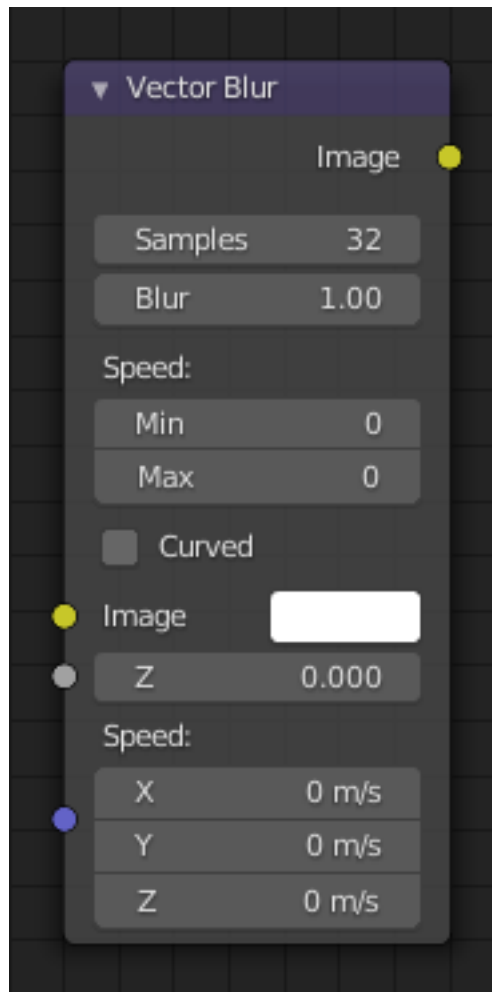


Fig. 285: Vector Blur Node.

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).

**Speed** The vector blur could produce artifacts like streaks, lines and other. To tackle these problems, the filter applies clamping, which can be used to limit which pixels get blurred. The speed is set in pixel units.

**Maximum Speed** The maximum threshold. The majority of artifacts are caused by pixels moving too fast.

**Minimum Speed** The minimum threshold for moving pixels can separate the hardly moving pixels from the moving ones. Especially when the camera itself moves, the vector mask can become the entire image.

**Curved** Interpolates motion between frames using a quadratic Bézier function rather than a linear function.

## 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.

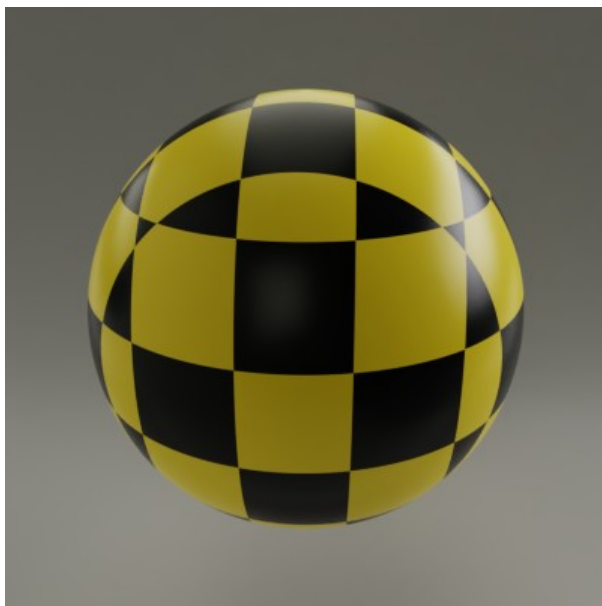


Fig. 286: Render result, no post-processing.

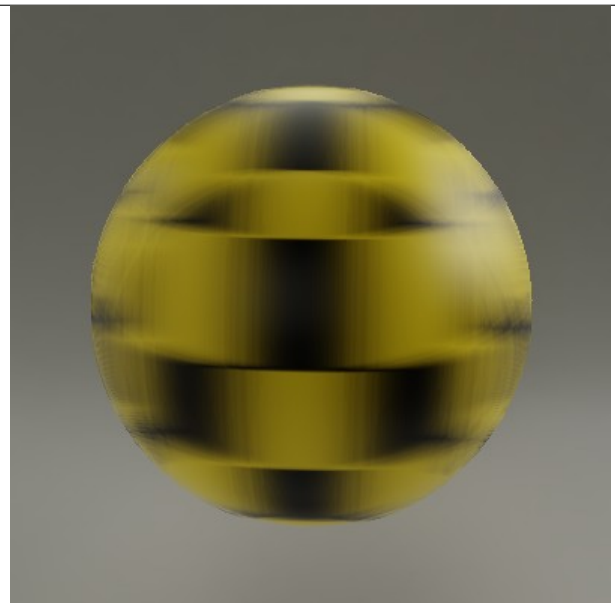


Fig. 287: Composite, with Samples set to 32 and Blur set to 1.0.

## Vector Nodes

These nodes can be used to manipulate various types of vectors, such as surface normals and speed vectors.

### Map Range Node

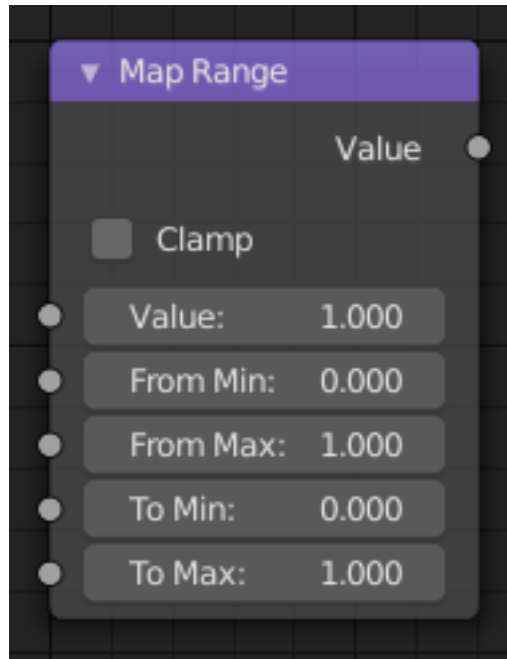


Fig. 288: Map Range Node.

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** Standard value input.

**From Min/Max** Start/End of the input value range.

**To Min/Max** Start/End of the destination 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

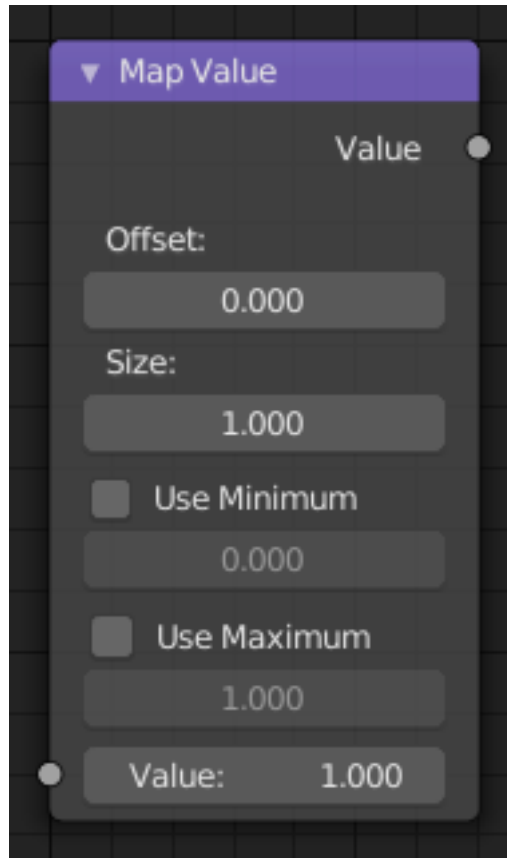


Fig. 289: Map Value Node.

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.

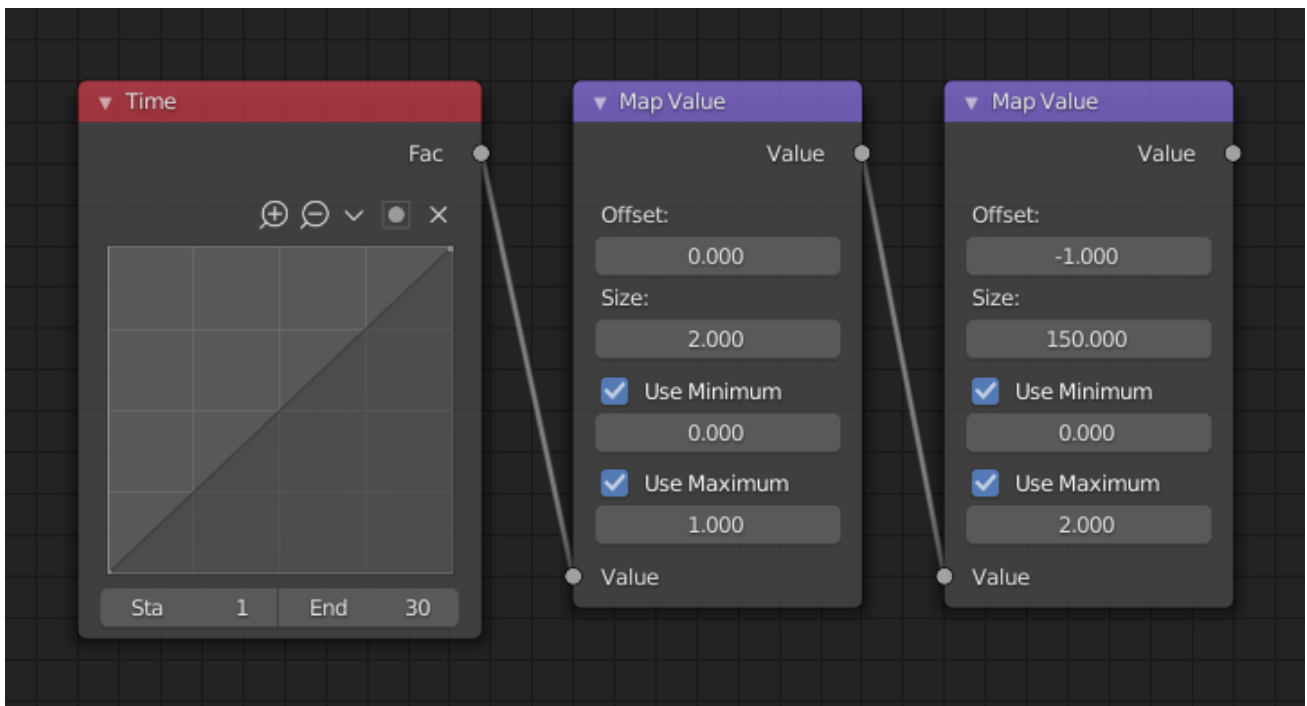


Fig. 290: Using Map Value to multiply.



## Normal Node

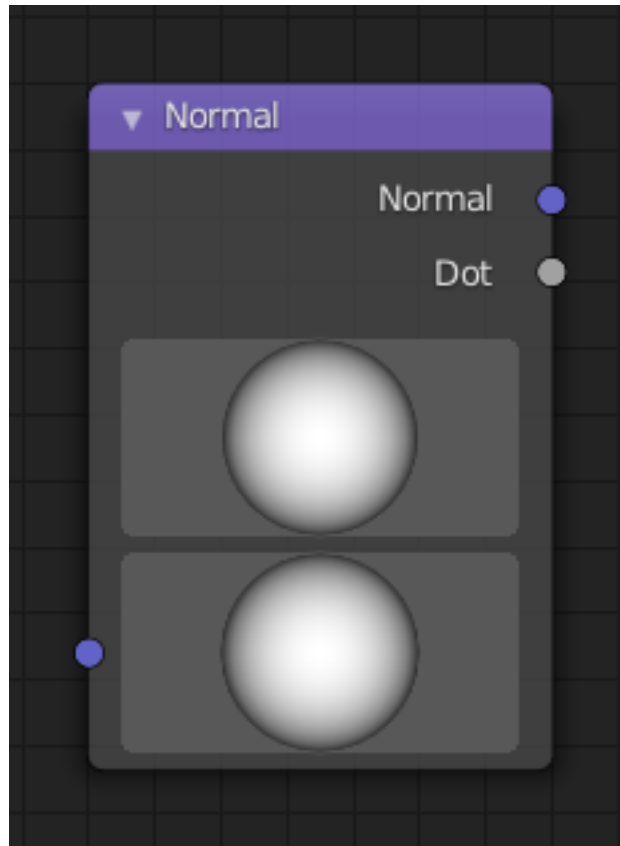


Fig. 291: Normal Node.

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.

## Normalize Node

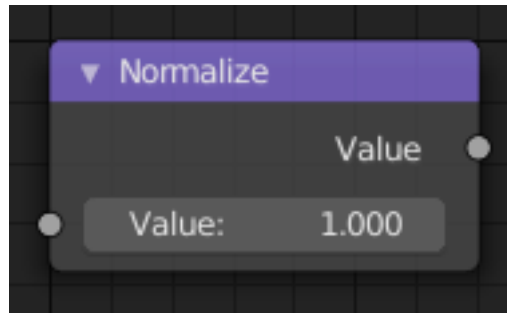


Fig. 292: Normalize Node.

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.

## Vector Curves Node

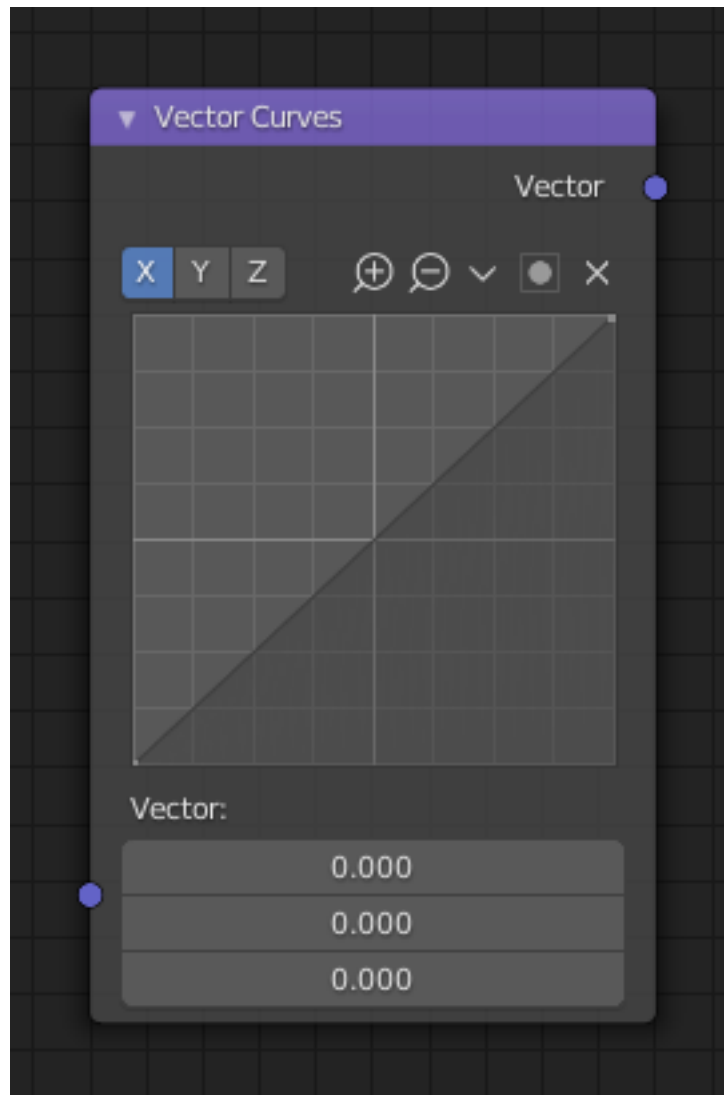


Fig. 293: Vector Curves Node.

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.

## Matte 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 blueish/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*.

---

## Box Mask Node

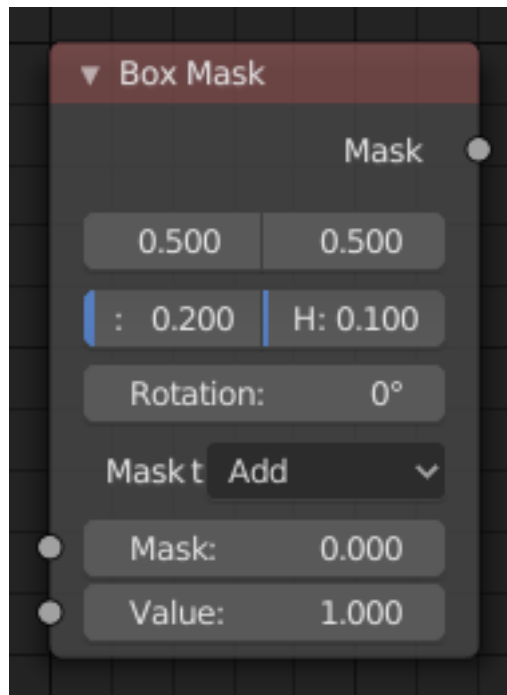


Fig. 294: Box Mask Node.

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*.

## Channel Key Node

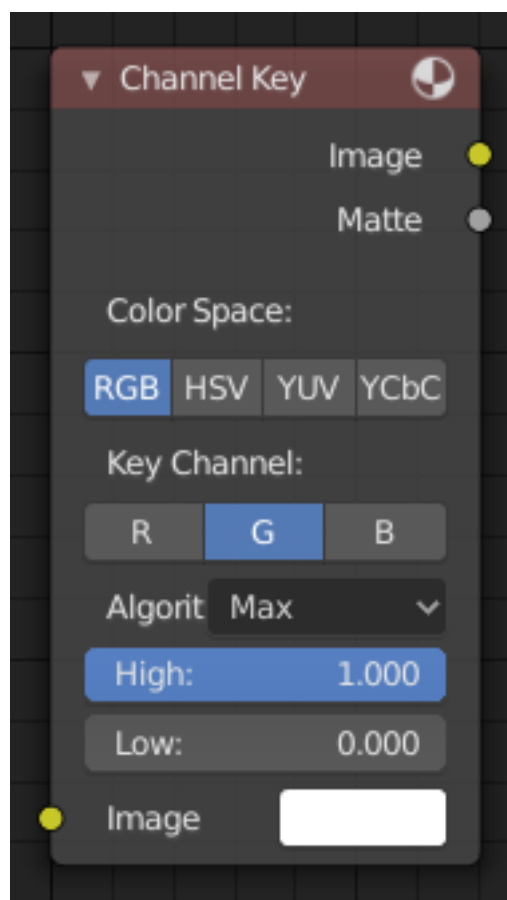


Fig. 295: Channel Key Node.

The *Channel Key* node determines background objects from foreground objects by the difference in the selected channel's levels.

For example in YUV color space, this is useful when compositing stock footage of explosions (very bright) which are normally shot against a solid, dark background.

## Inputs

**Image** Standard image input.

## Properties

**Color Space** This button selects what color space the channels will represent.

RGB, HSV, YUV, YCbCr

**Channel** This button selects the channel, defined by the Color Space, to use to determine the matte.

**Algorithm** Max, Single

**Limit** It is possible to have a separation between the two values to allow for a gradient of transparency between foreground and background objects.

**High** Determines the lowest values that are considered foreground. (Which is supposed to be - relatively - height 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.)

## 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

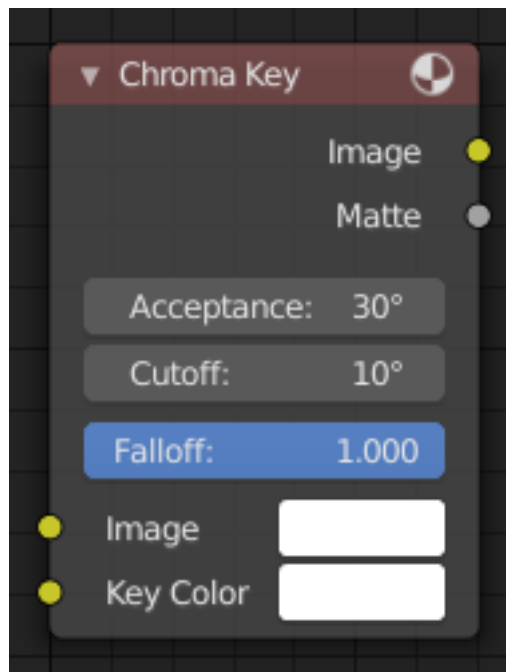


Fig. 296: Chroma Key Node.

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 image 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

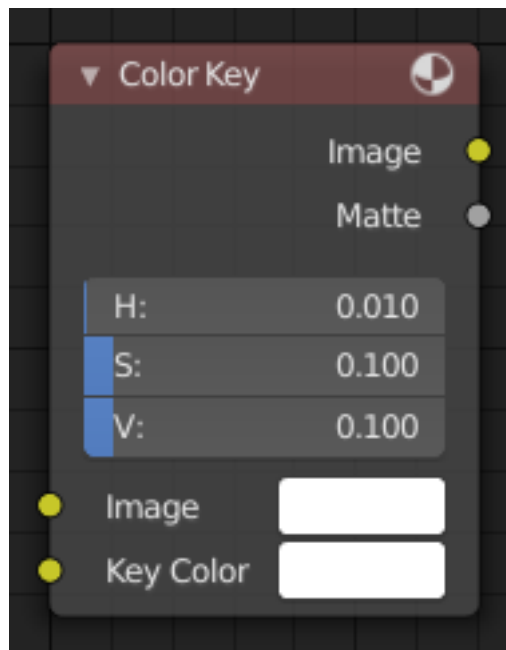


Fig. 297: Color Key node.

The Color Key node creates a matte based on a specified color of the input image.

## Inputs

**Image** Standard image 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

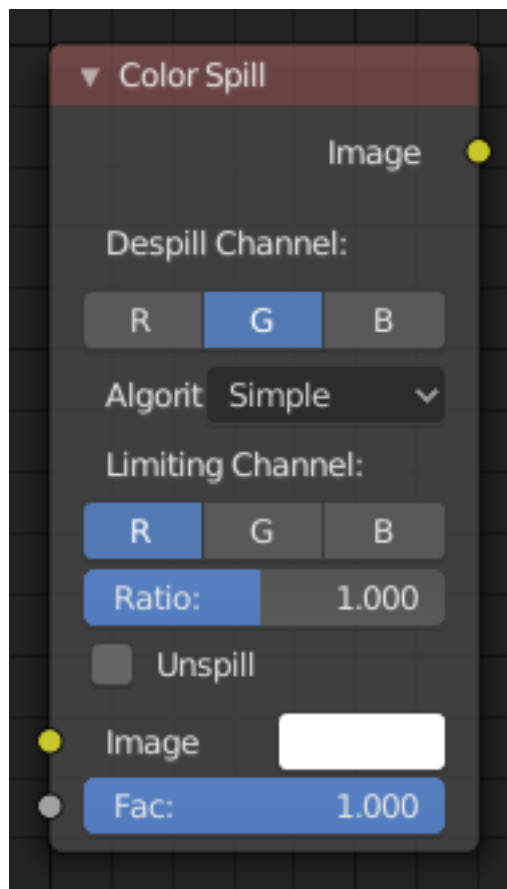


Fig. 298: Color Spill Node.

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 image 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](#).



Fig. 299: Before: green border and green reflections.

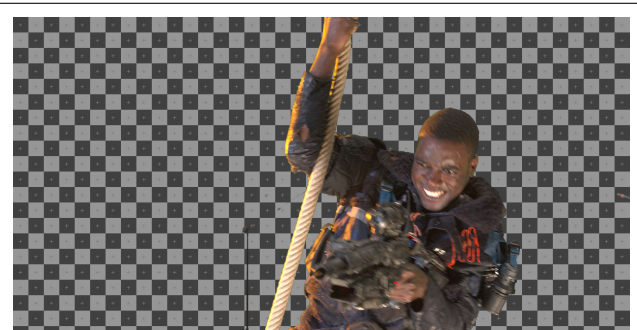


Fig. 300: After: no unwanted green.

## Cryptomatte Node

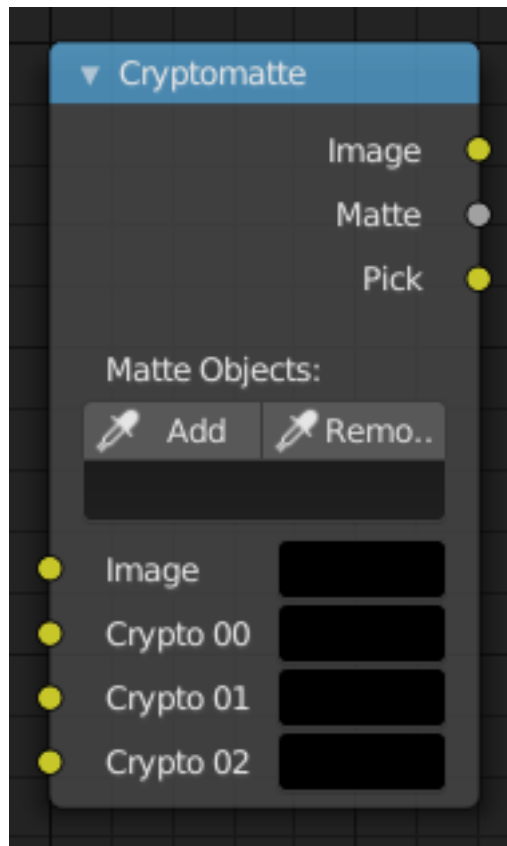


Fig. 301: Cryptomatte Node.

The Cryptomatte node uses the Cryptomatte standard to efficiently create mattes for compositing. Cycles outputs 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 image 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

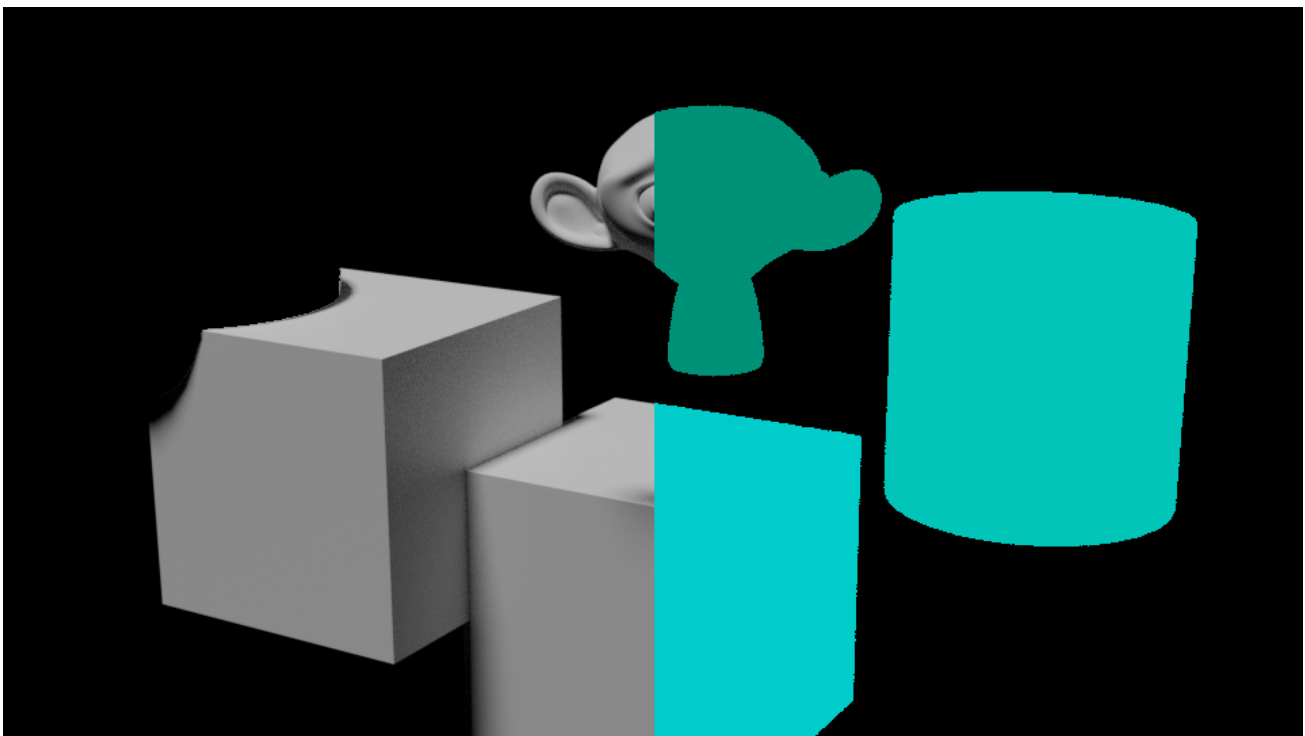
1. Enable Cryptomatte Object render pass in the Passes panel, and render.
2. In the compositing nodes, create a Cryptomatte node and link the Render Layer matching Image and Cryptomatte passes to it.
3. Attach a Viewer node to the Pick output of the Cryptomatte node.
4. Use the Cryptomatte Add/Remove 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.



## Difference Key Node

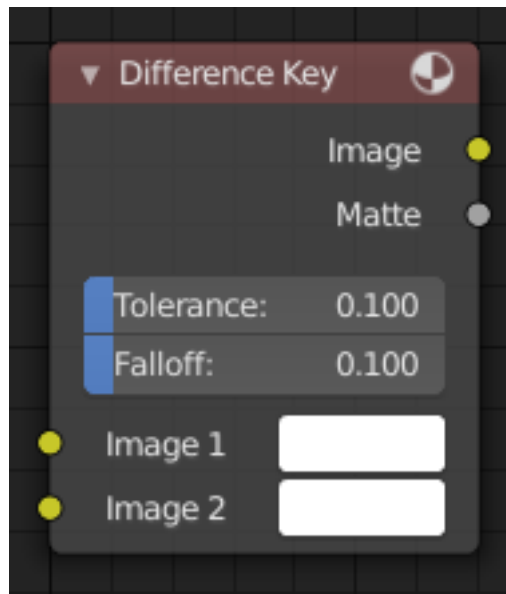


Fig. 302: Difference Key Node.

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

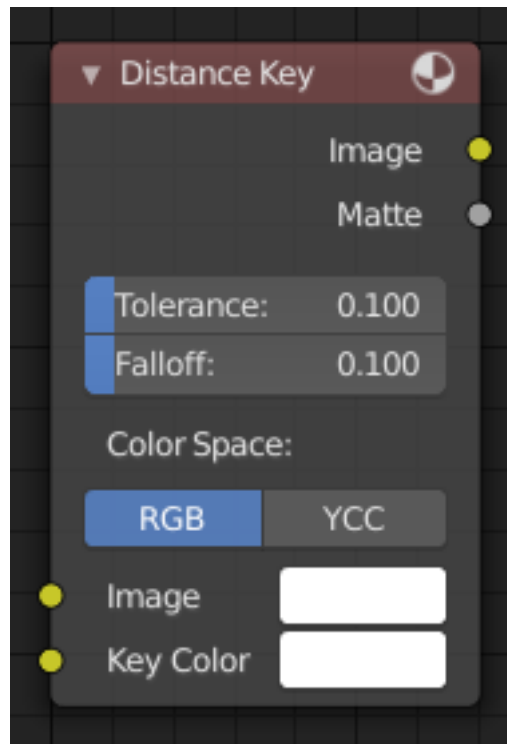


Fig. 303: Distance Key Node.

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 image 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.

## Double Edge Mask Node

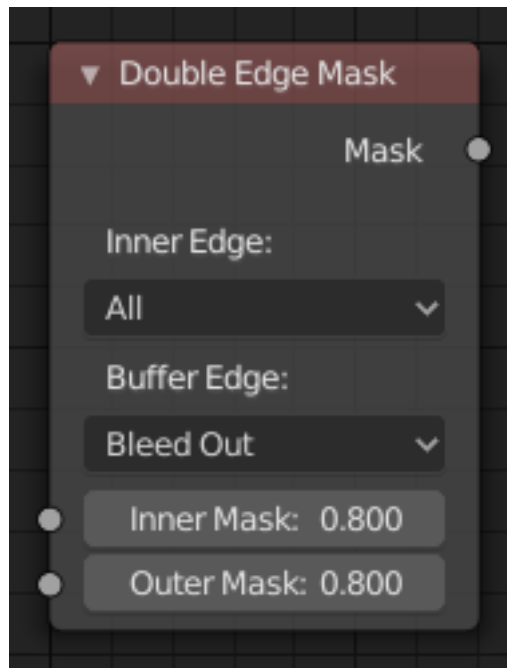


Fig. 304: Double Edge Mask Node.

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.

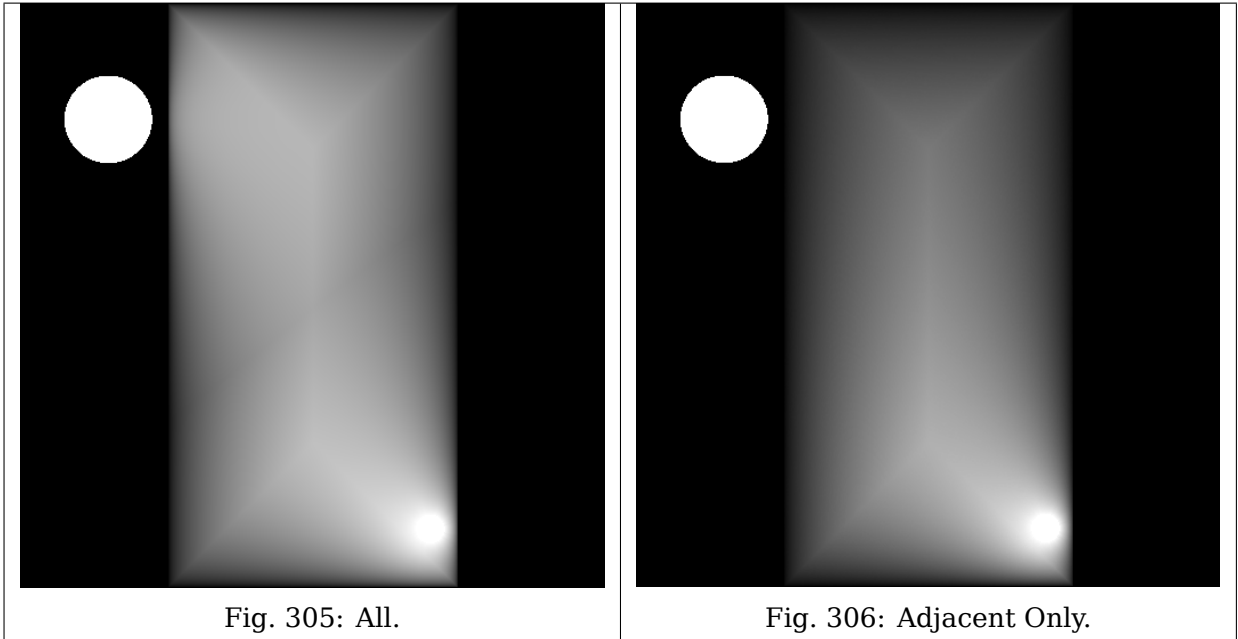
**Outer Mask** A mask representing the outside shape, which will fade from black at its edges to white at the *Inner Mask*.

## Properties

### Inner Edge

**All** All shapes in the *Inner Mask* contribute to the gradient, even ones that do not touch the *Outer Mask* shape.

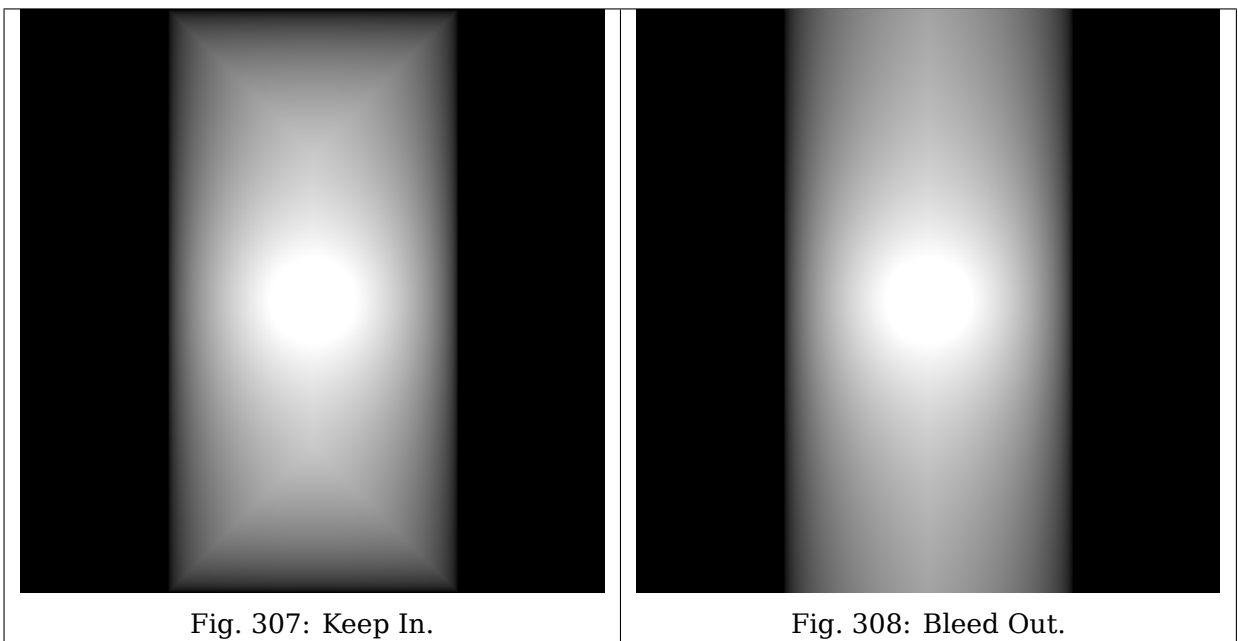
**Adjacent Only** Only shapes in the *Inner Mask* that overlap with the *Outer Mask* contribute to the gradient.



### 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.



### Outputs

**Mask** Standard mask output.



## Example

A video can be found at <https://www.youtube.com/watch?v=VcjEfoNIH2s>

## Ellipse Mask Node

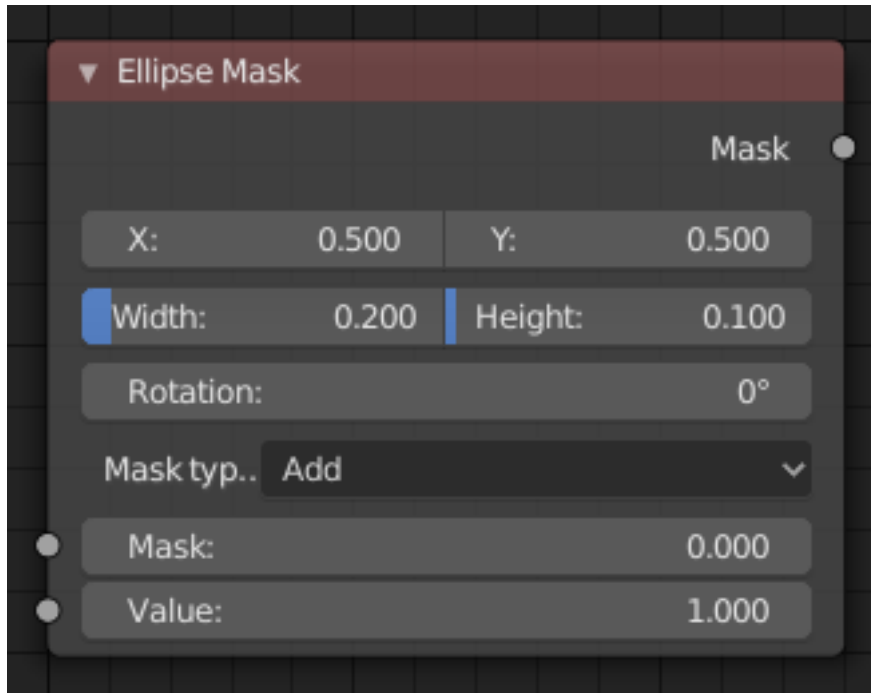


Fig. 309: Ellipse Mask Node.

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 will 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 mask 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.

---

## Keying Node

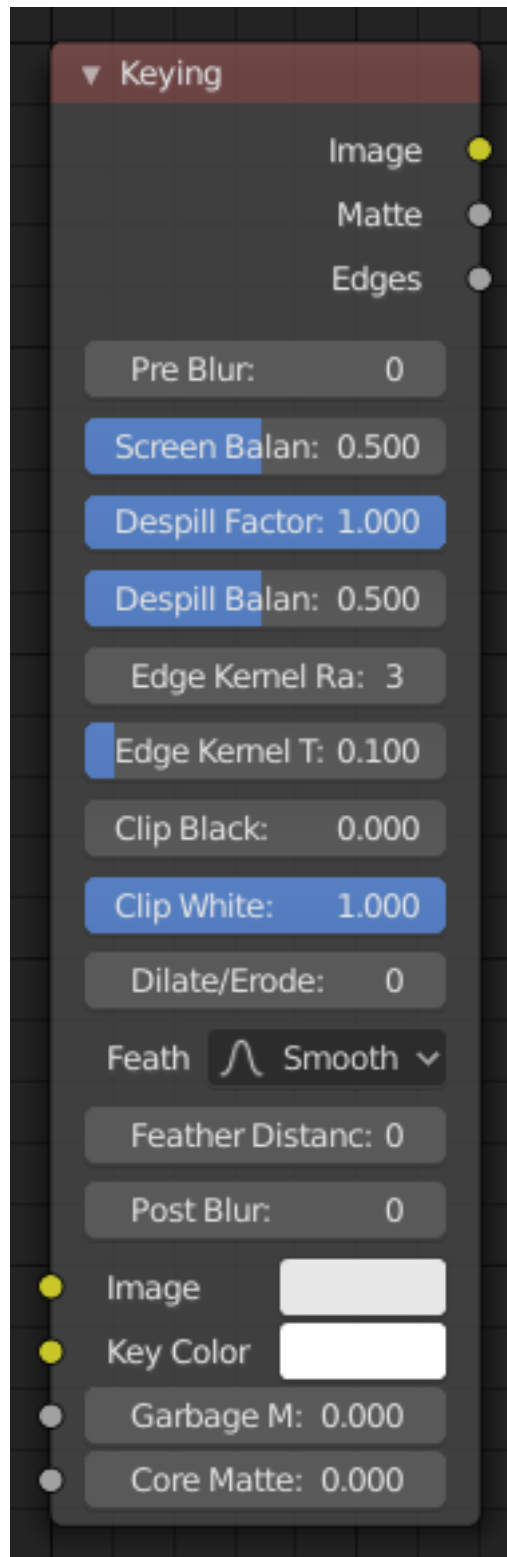


Fig. 310: Keying Node.

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 image 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 despill, 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 lose 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

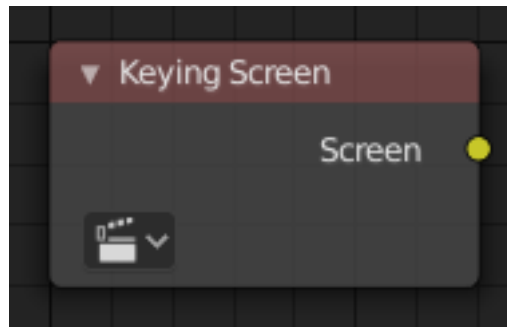


Fig. 311: Keying Screen node.

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

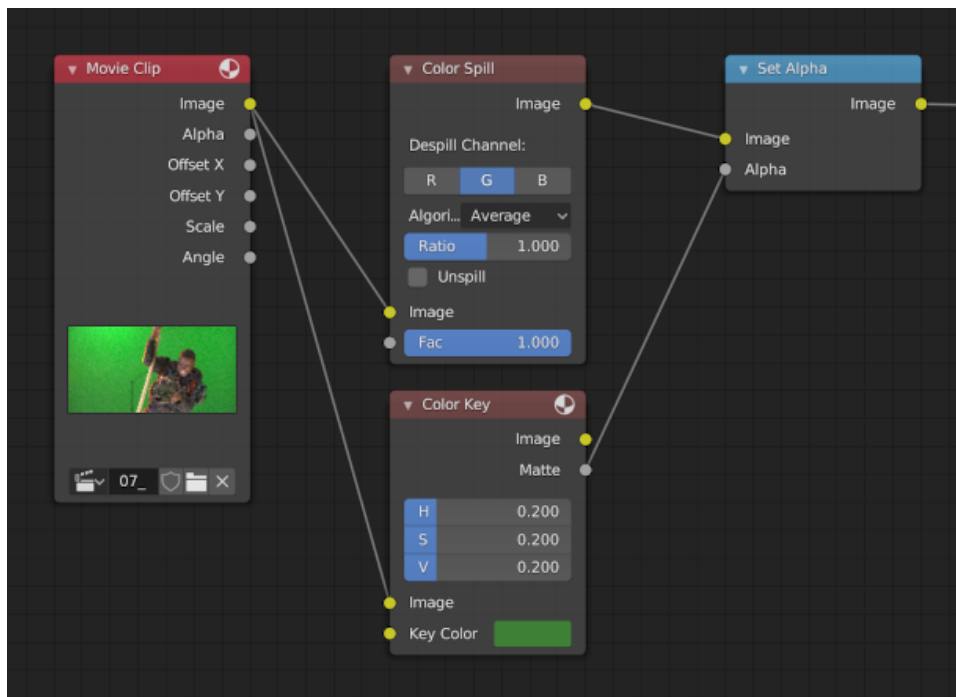
would be updating automatically along the movie. Tracks might have an offset for easier tracking of feature-less screens.

## 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.



Fig. 312: Example from the *Mango Open Movie*, *Tears of Steel*.

That can result in a bad matte.

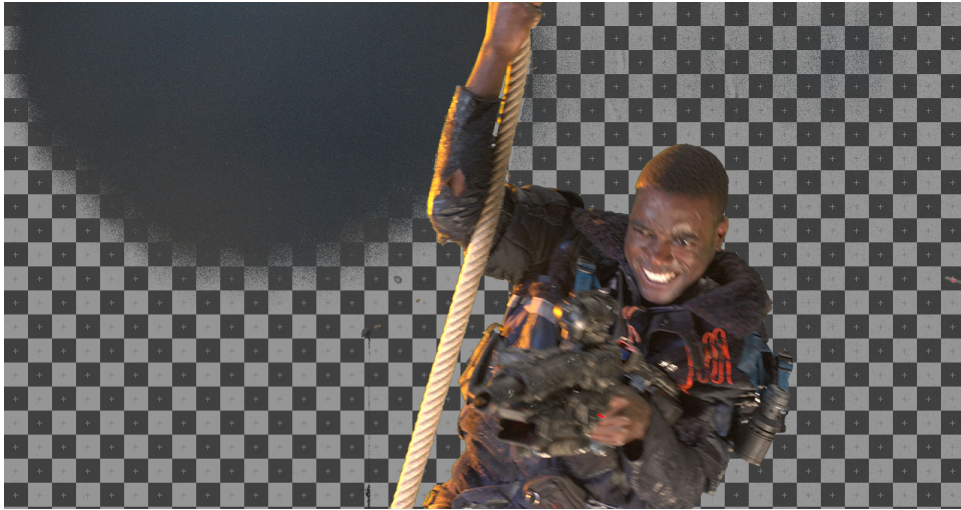
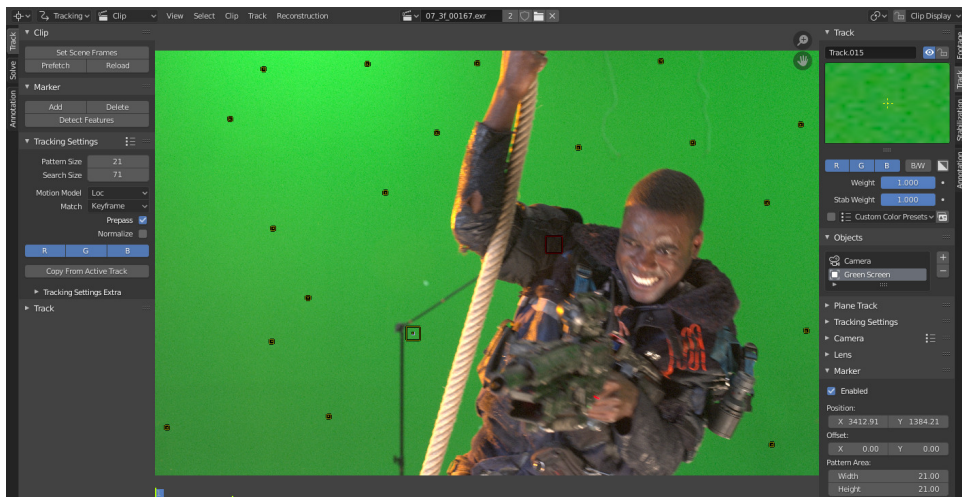


Fig. 313: Example of a poor mask: Some of the backdrop is opaque, and some parts of the gun in the foreground are transparent.

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.

The resulting image now has a better matte.



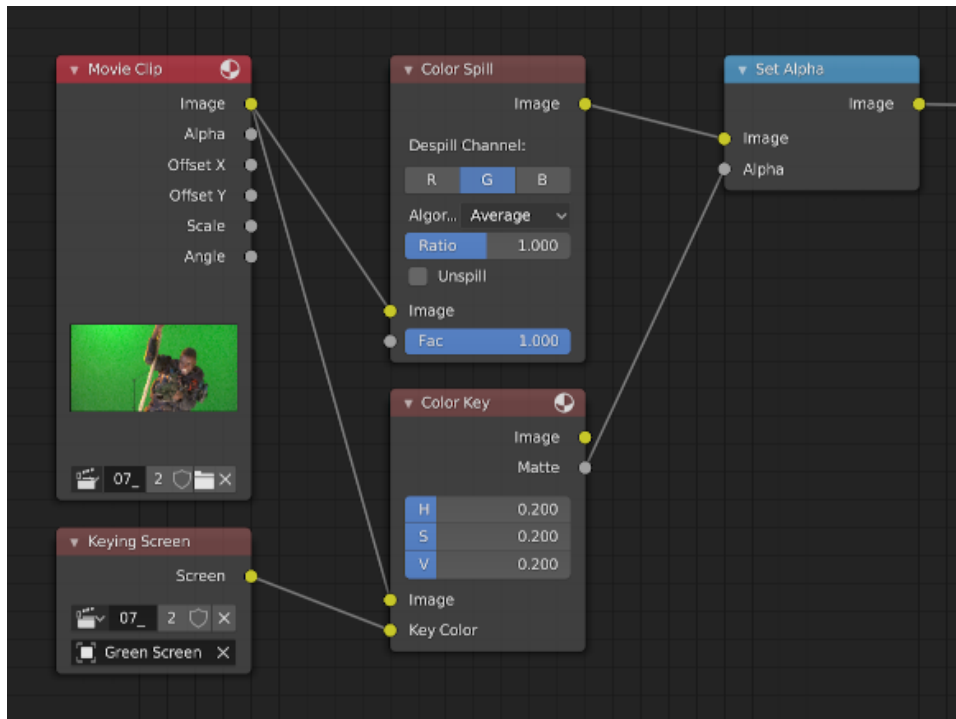


Fig. 314: Node configuration with *Keying Screen*'s generated gradient plate connected to the Color input of the Keying node.



Fig. 315: Gradient plate generated by *Keying Screen*.





## Luminance Key Node

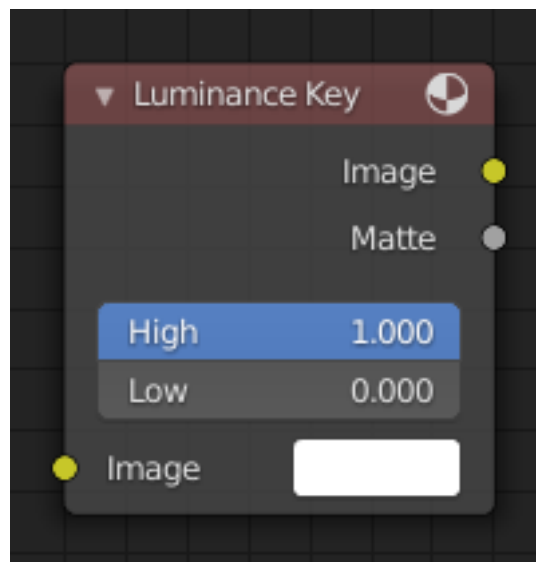


Fig. 316: Luminance Key Node.

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 image 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.

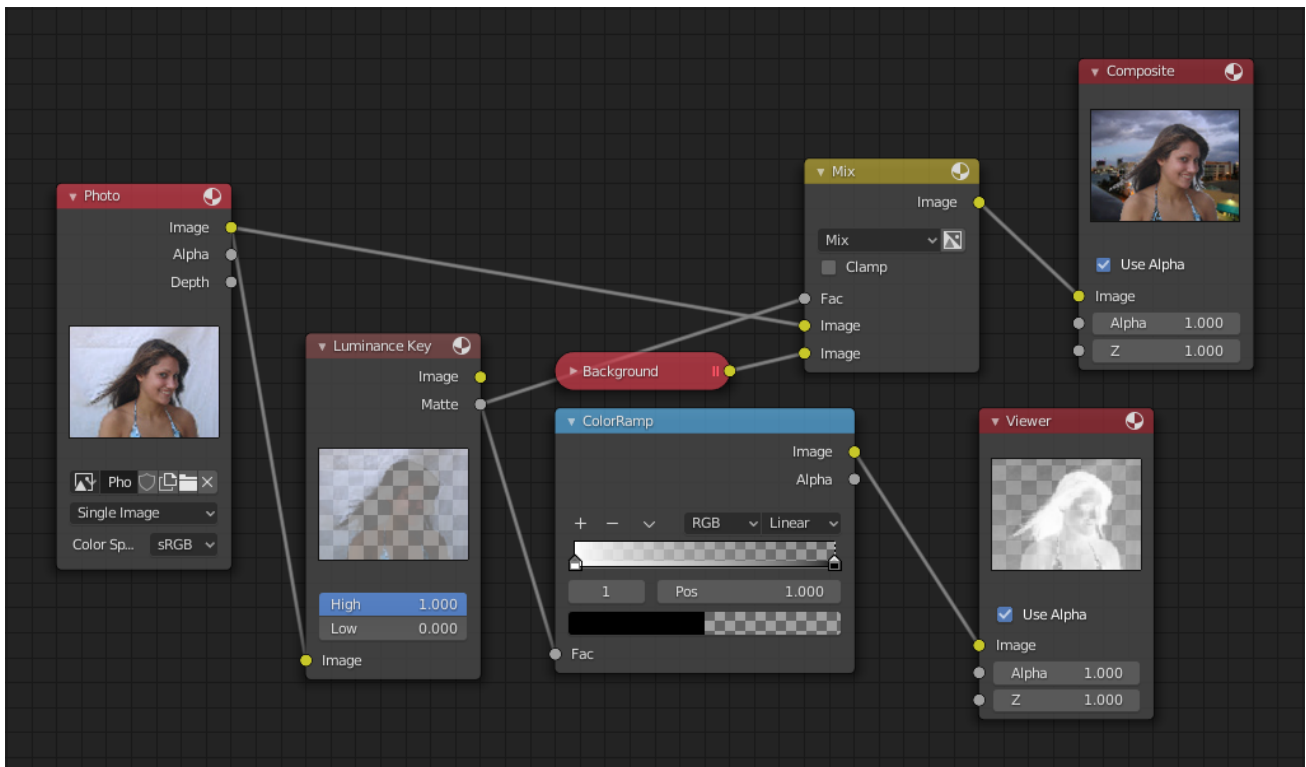


Fig. 317: Using Luma Key with a twist.

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.

### **Distort 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.

## Corner Pin Node

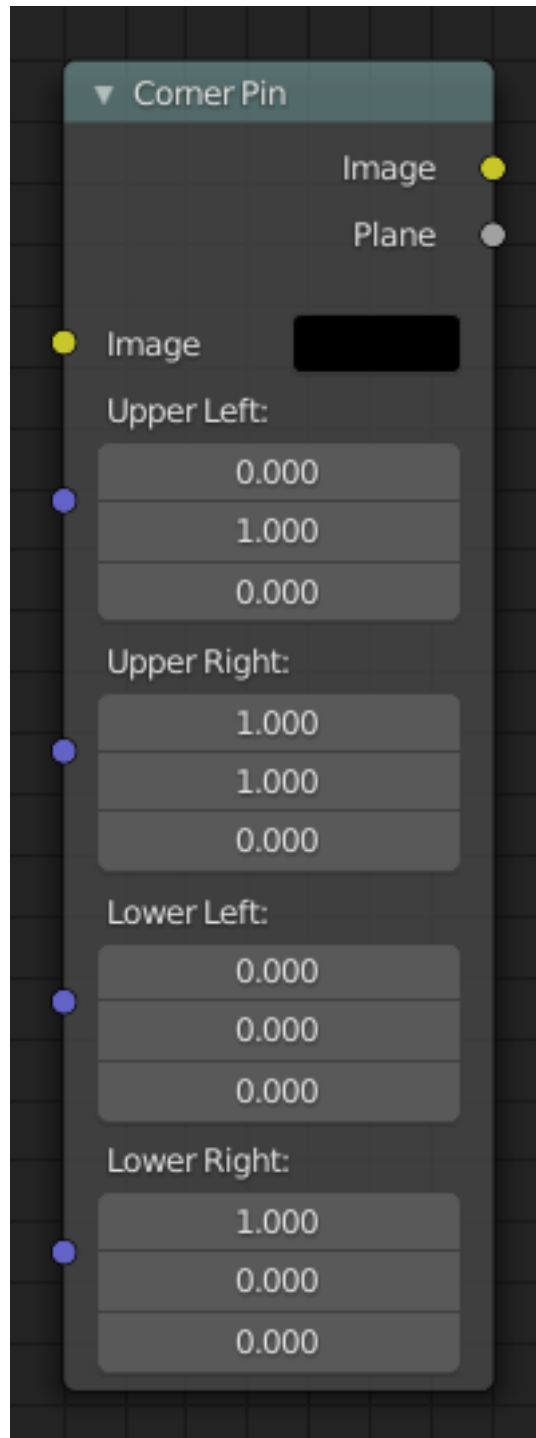


Fig. 318: Corner Pin Node.

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 image 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 image output. (The image after distorting.)

**Plane** A black-and-white alpha mask of the plane.

## Example

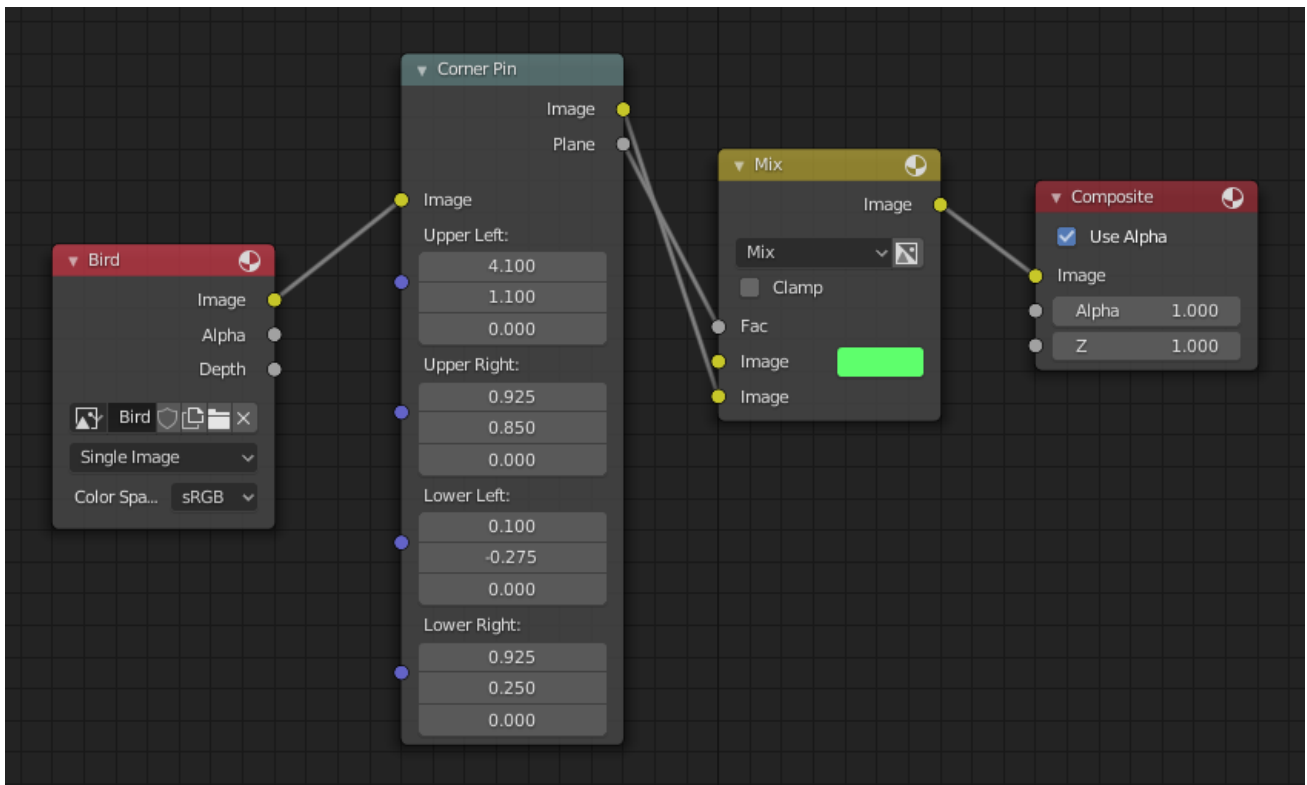


Fig. 319: An example of the Corner Pin node.

In the example above, the image of the bird is distorted by the vectors specified by the Corner Pin node.



Fig. 320: An example of the distorted image.

## Crop Node

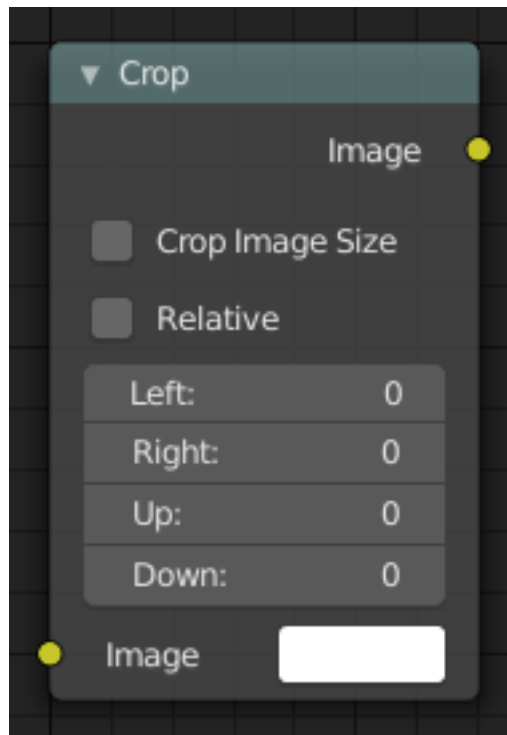


Fig. 321: Crop Node.

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 image 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 image 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

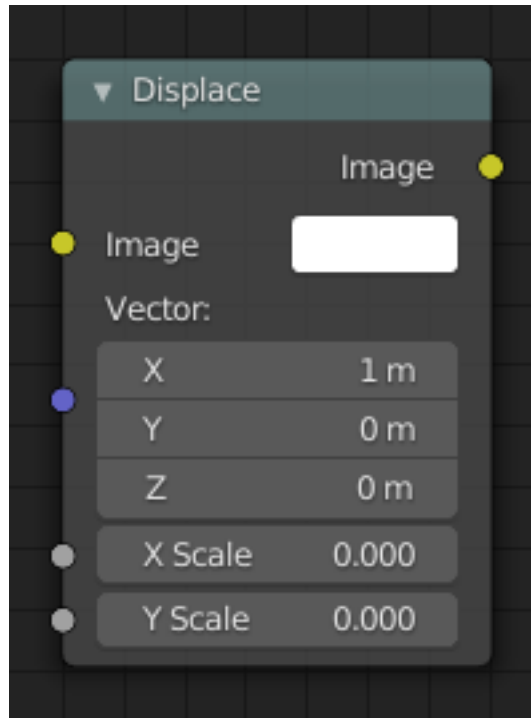


Fig. 322: Displace Node.

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 image 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 image output.

## Flip Node

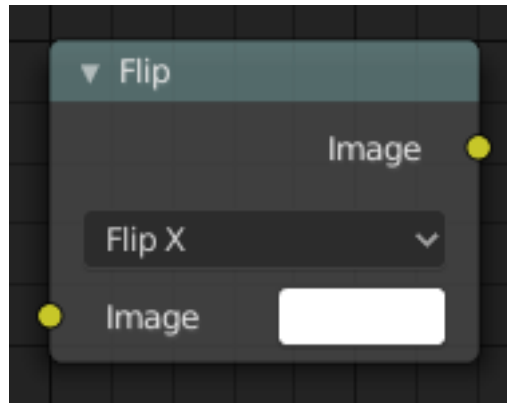


Fig. 323: Flip Node.

This node flips an image at 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 image 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 image output.

## Lens Distortion Node

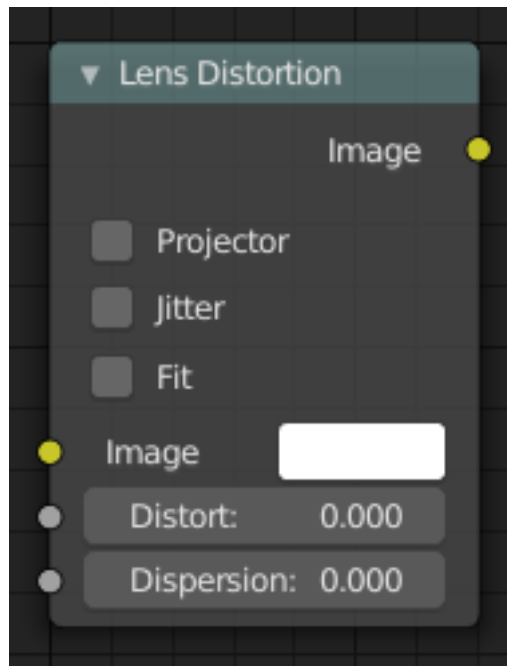


Fig. 324: Lens Distortion Node.

Use this node to simulate distortions that real camera lenses produce.

### Inputs

**Image** Standard image input.

**Distort** 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 image output.

## Map UV Node

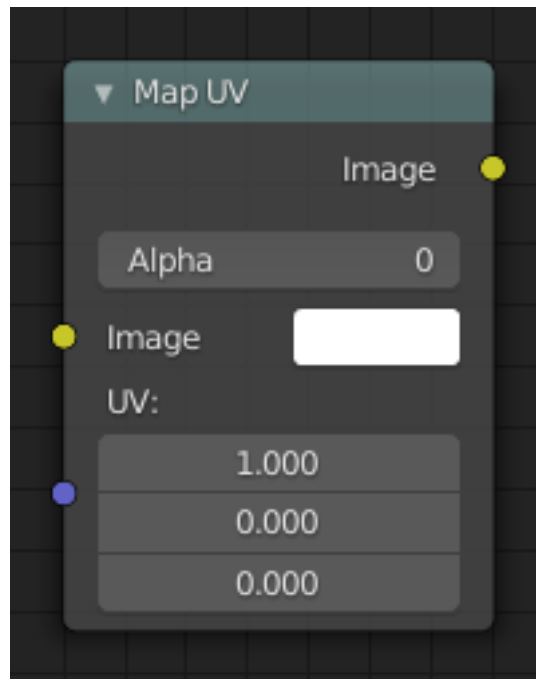


Fig. 325: Map UV node.

With this node objects can be “re-textured” after they have been rendered. To apply a texture to individual enumerated objects the *ID Mask Node* could be used.

### 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

**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.

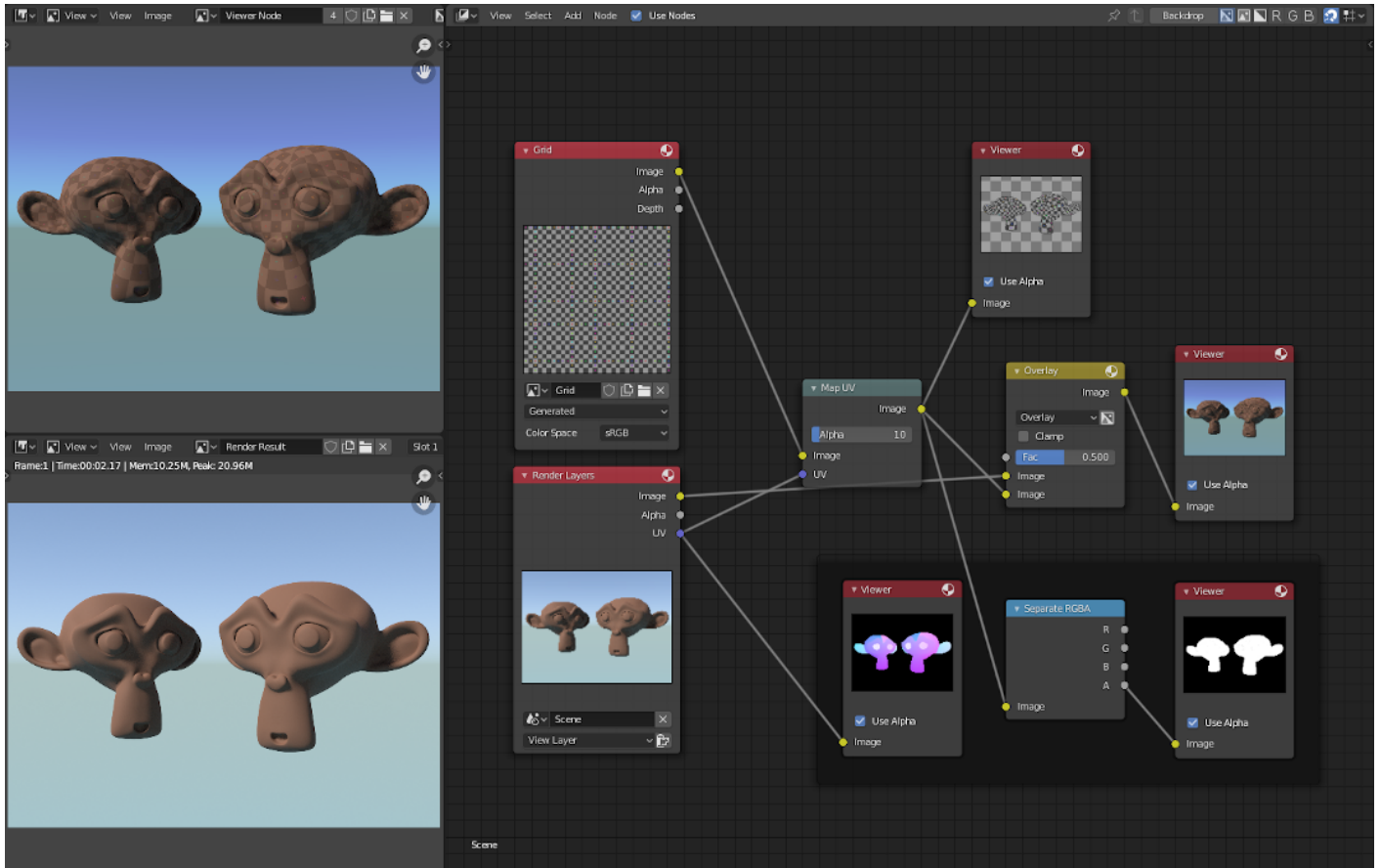


Fig. 326: Adding a grid UV textures for motion tracking.

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”.

---

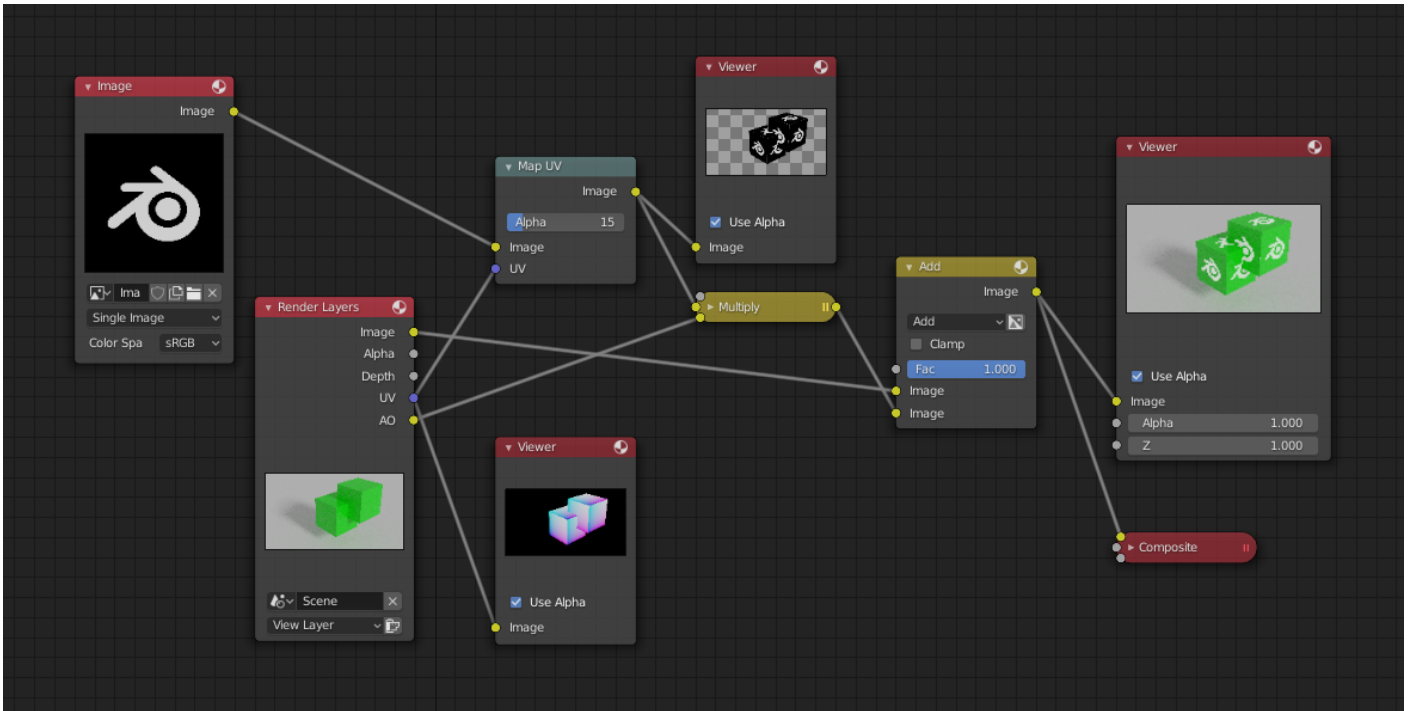


Fig. 327: Adding UV textures in post-production.

### Movie Distortion Node

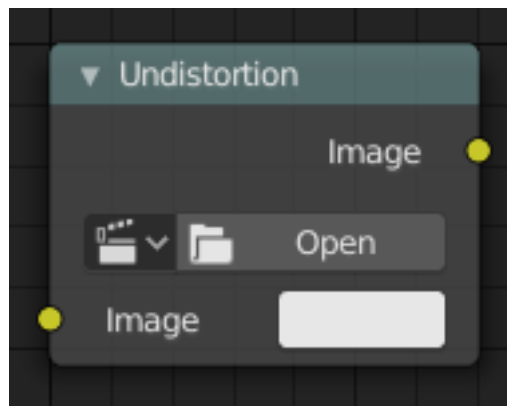


Fig. 328: Movie Distortion Node.

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 image 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.

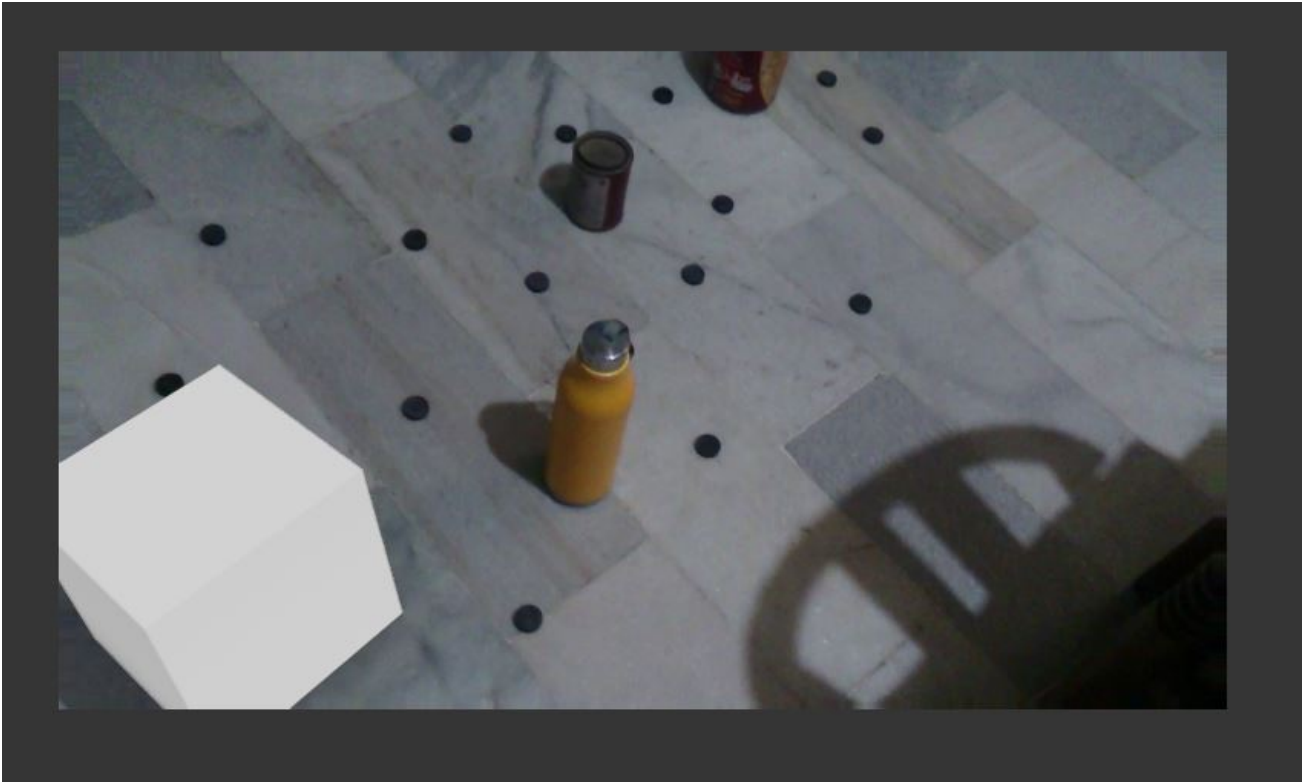


Fig. 329: Problems (notice the edges?)

### Plane Track Deform Node

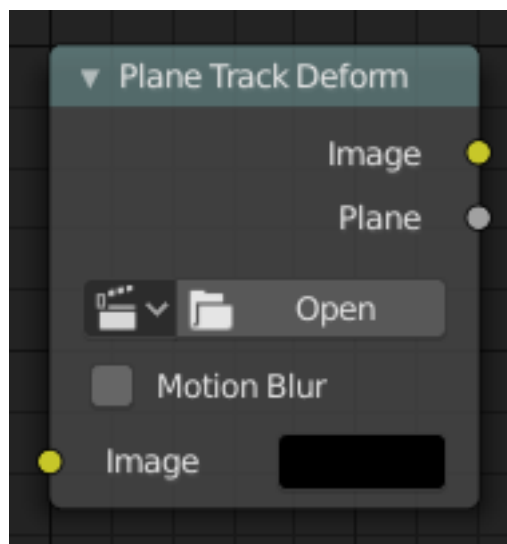


Fig. 330: Plane Track Deform Node.

The Plane Track Deform Node is used to incorporate the special “plane track” in your composite by checking areas which are planes, and replacing their footage with some other image.

### 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.

### Using Plane Output

This can be achieved by mixing the movie clip and the image by using the plane output as the factor.

### 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:

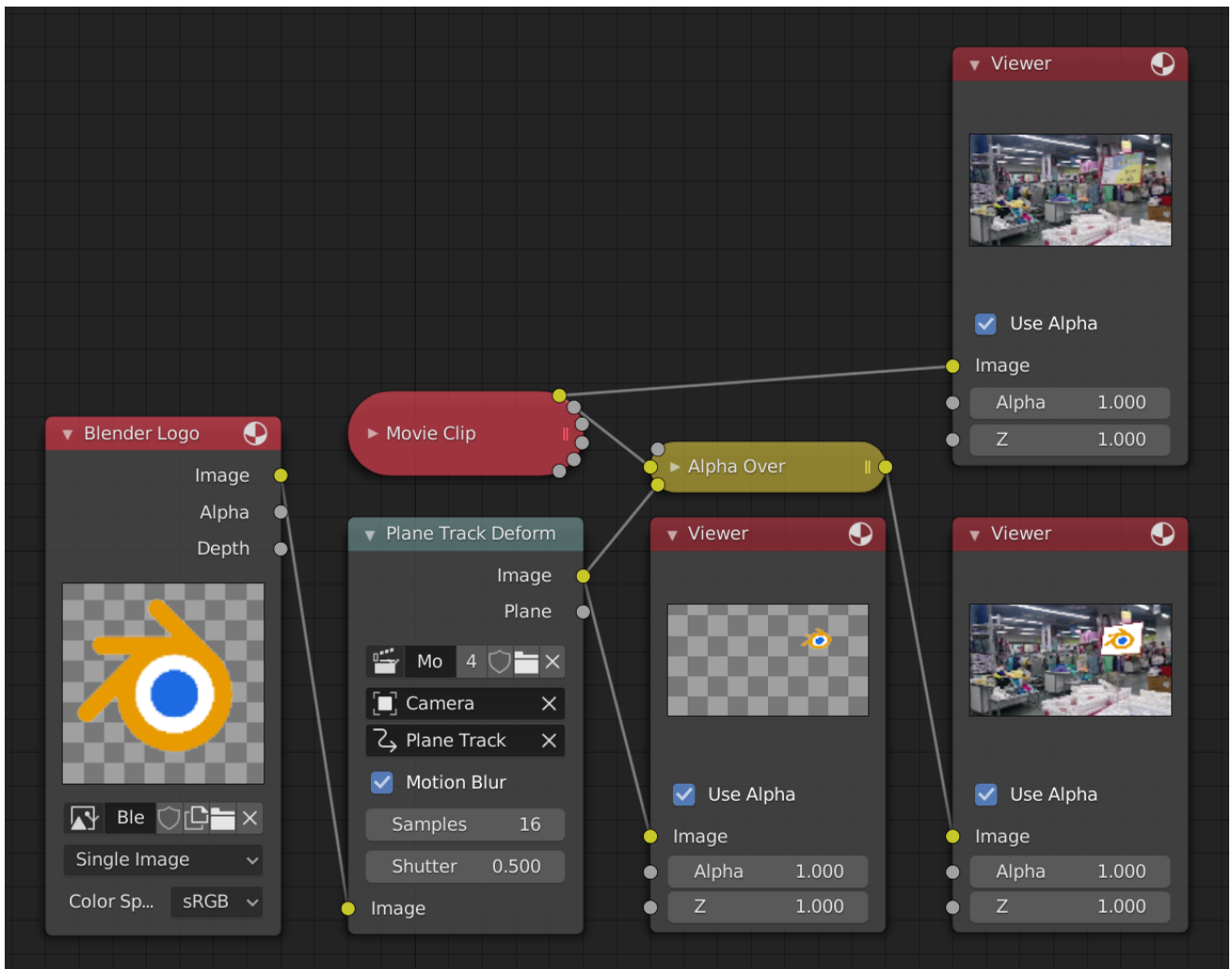


Fig. 331: Image output.

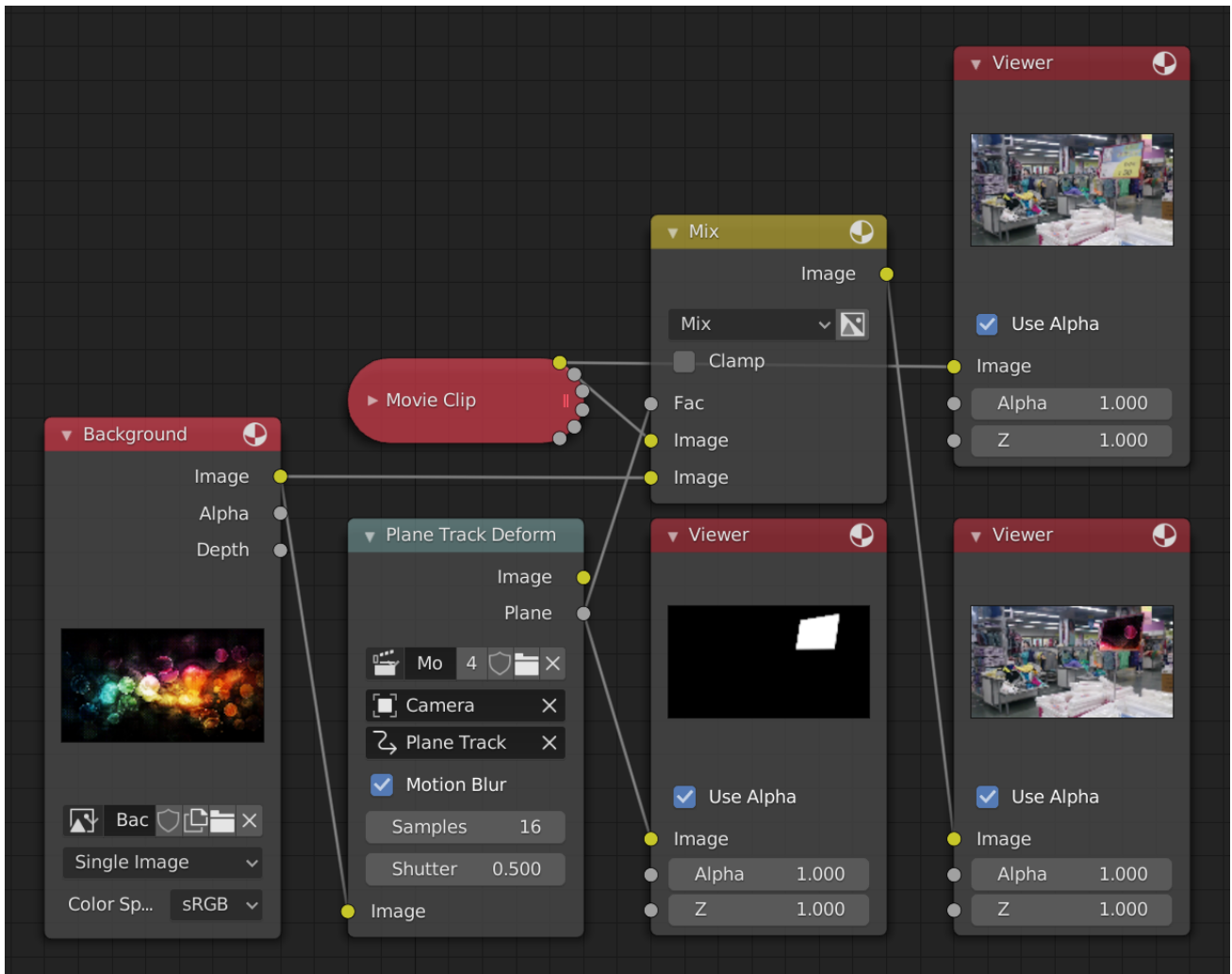


Fig. 332: Plane output.

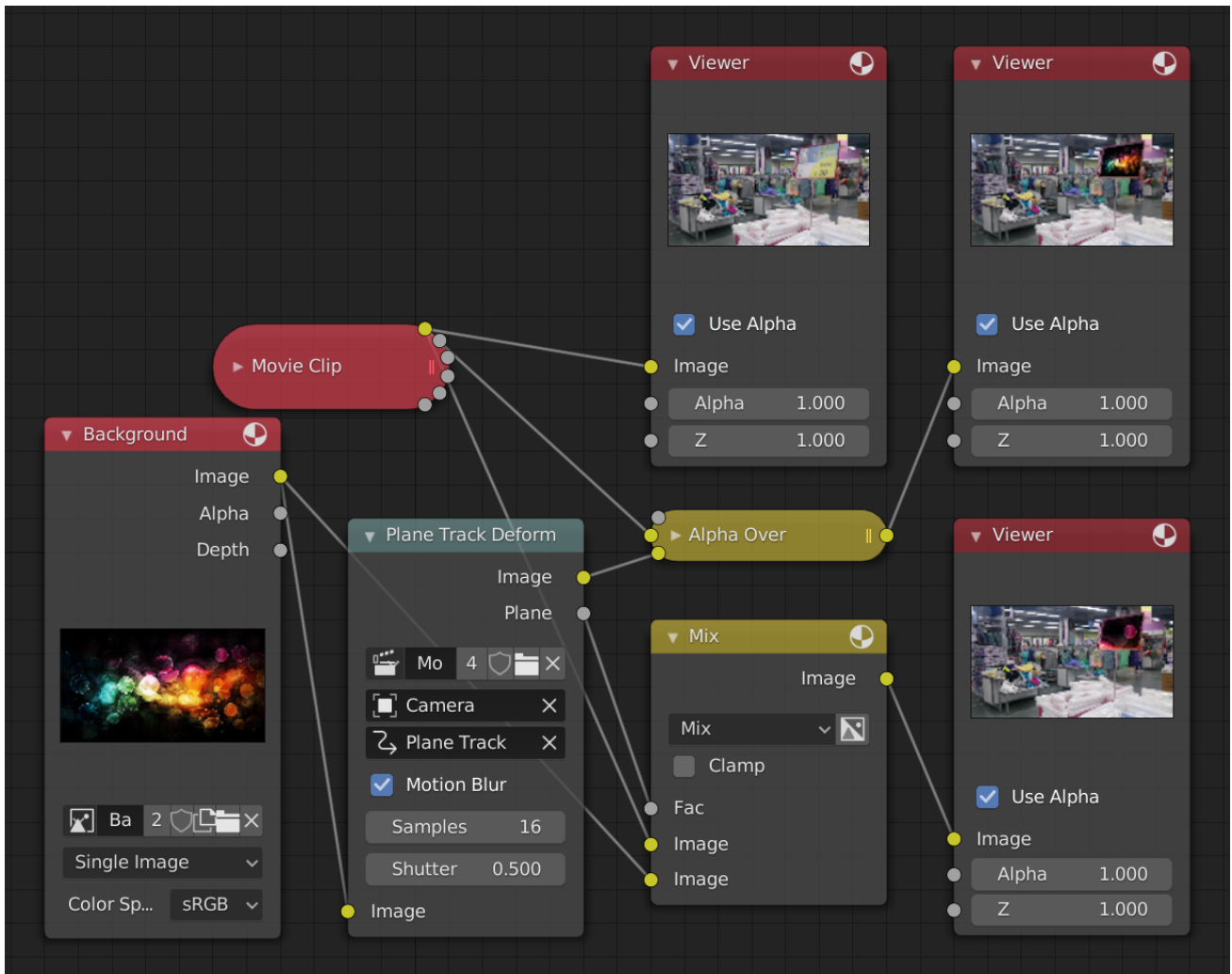


Fig. 333: Comparison between image output and original image (see Viewer nodes carefully).

## Rotate Node

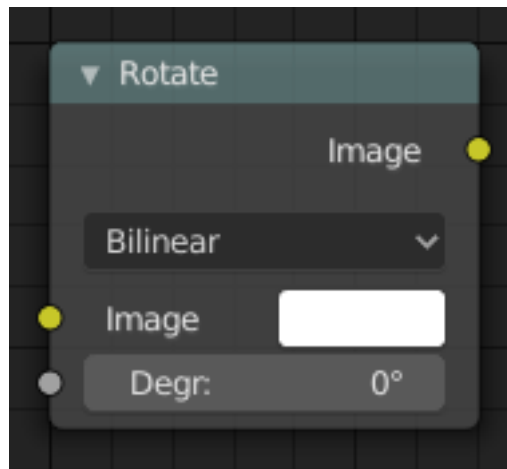


Fig. 334: Rotate Node.

This node rotates an image.

### Inputs

**Image** Standard image 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 image output.

## Scale Node

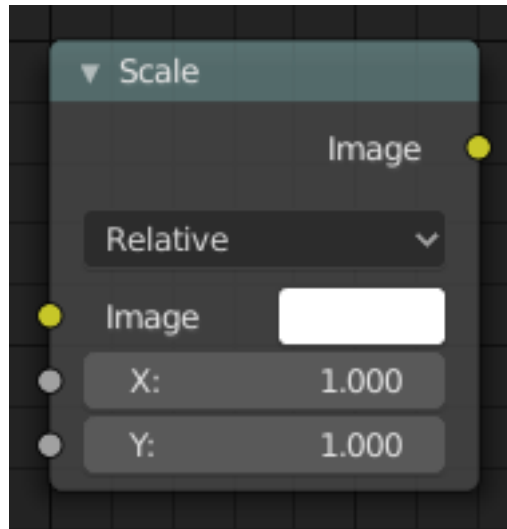


Fig. 335: Scale Node.

This node scales the size of an image.

### Inputs

**Image** Standard image 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 image 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.

## Stabilize 2D Node

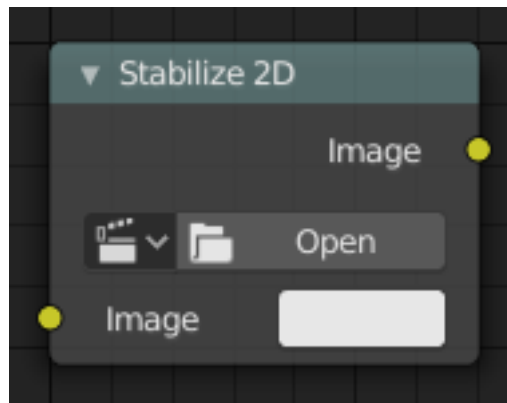


Fig. 336: Stabilize 2D Node.

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 image 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 image input.

## Transform Node

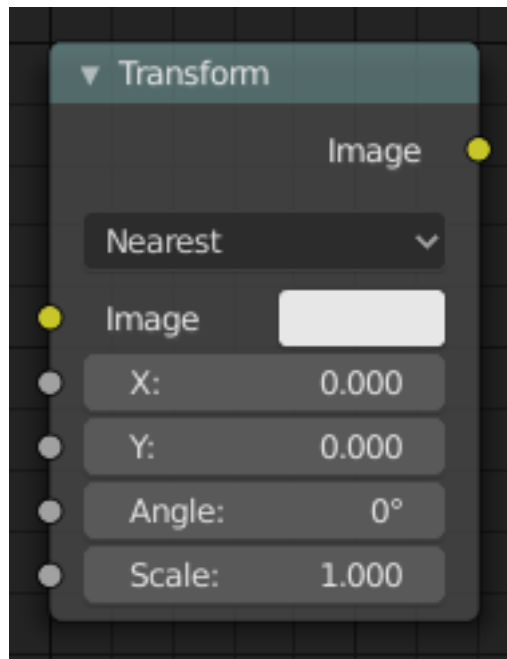


Fig. 337: Transform Node.

This node combines the functionality of three other nodes: *Scale*, *translate*, and *rotate* nodes.

### Inputs

**Image** Standard image 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 image output.



## Translate Node

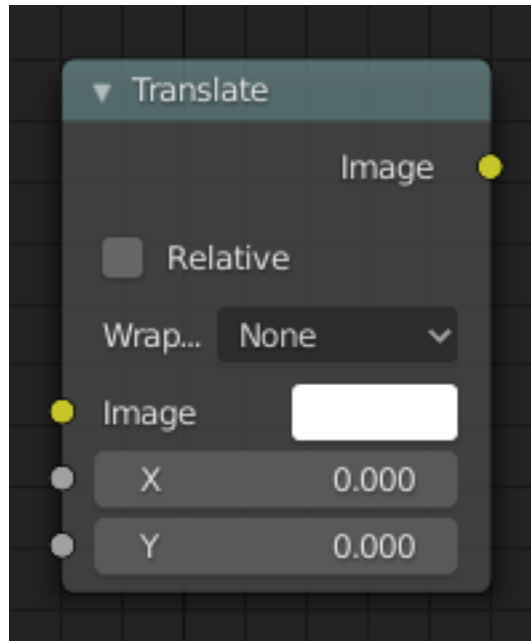


Fig. 338: Translate Node.

The Translate node moves an image.  
 Could also be used to add a 2D camera shake.

### Inputs

**Image** Standard image 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

### Outputs

**Image** Standard image output.

## Node Groups

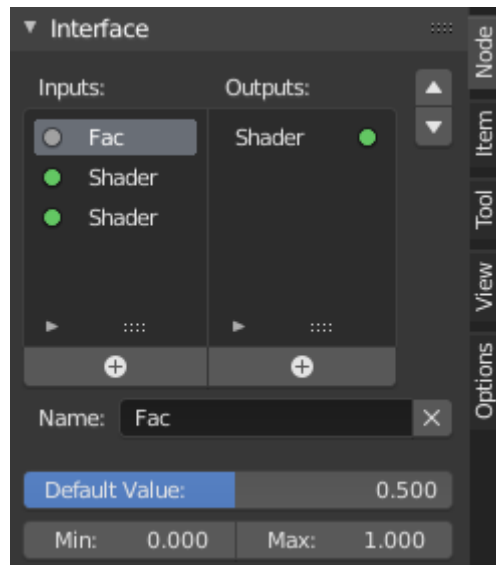


Fig. 339: Example of a node group.

Grouping nodes can simplify a node tree by allowing instancing and hiding parts of the tree. Both material and composite nodes can be grouped.

Conceptually, grouping nodes allows you to specify a *set* of nodes that you can treat as though it were “just one node”. Node groups are similar to functions in programming, they can be reused in many places in a node tree and can be customized by changing the “parameters” of the node group.

As an example: If you have created a material that you would like to use with different inputs e.g. diffuse color: red plastic, green plastic. You could create different materials with *Make Single User* for each different color with a copy of the tree part describing the plastic material. If you like to edit the material you would need to redo the edit on all materials. A better method of reuse is to create node groups, exposing only the variable inputs (e.g. diffuse color).

Also nested node groups are supported. I.e. a node group can be inserted or created inside another 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).

---

## Make Group

### Reference

**Mode** All Modes

**Menu** *Node* → *Make Group*

**Hotkey** Ctrl-G

---

To create a node group, select the nodes you want to include, then press Ctrl-G, *Group* → *Make Group*. A node group will have a green title bar. All of the 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 easily distinguish between the two types.

**Tip:** What **not** to include in node groups:

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not** include:

**Input nodes** If you include a source node in your group, you will end up having the source node appearing *twice*: once inside the group, and once outside the group in the new material node tree.

**Output node** If you include an output node in the group, there will not be an output socket available *from* the group!

## Edit Group

### Reference

**Mode** All Modes

**Menu** *Node* → *Edit Group*

**Header** *Go to Parent Node Tree*

**Hotkey** Tab, Ctrl-Tab

With a node group selected, Tab expands the node to a frame, and the individual nodes within it are shown. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of the editor view. You will not be able, though, to thread them to a node outside the group; you have to use the external sockets on the side of the node group. While Tab can be used to both enter and exit a group, Ctrl-Tab only exits.

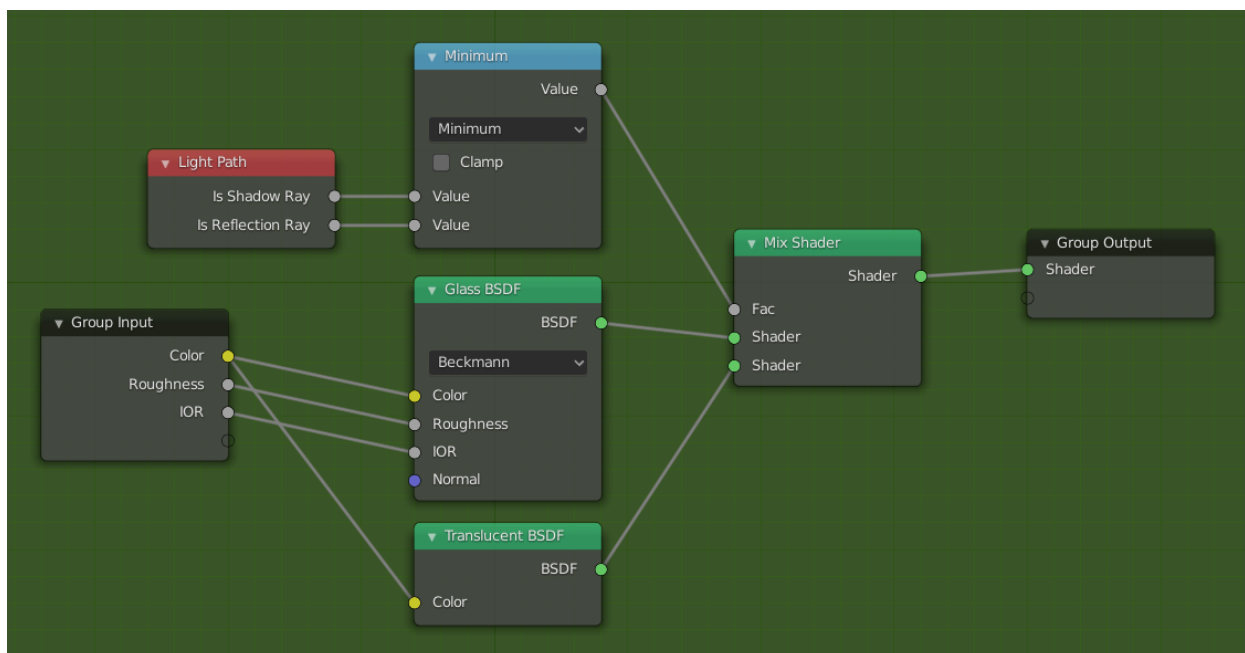


Fig. 340: Example of an expanded node group.

## Interface

### Interactively

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. When created, 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 during node group development an additional parameter needs to be passed into the group, an additional socket must be added to the *Group Input* node. This is easily done by adding a connection from the hollow socket on the right side of the *Group Input* node to the desired input socket on the node requiring input. The process is similar for the *Group Output* regarding data you want to be made available outside the group.

### Panel

#### Reference

**Mode** All Modes

**Panel** *Sidebar region* → *Node* → *Interface*

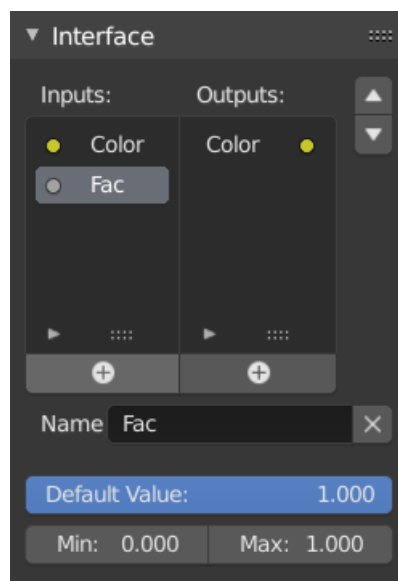


Fig. 341: The interface panel for editing groups.

Sockets can be added, re-ordered, or removed, descriptive names can be added and the details of the input data value defined here.

If you have multiple inputs or outputs, they can be re-ordered by selecting the socket in the list and then moving it up or down with the arrow buttons on the right side of the panel. The larger plus sign buttons below the list will add an unconnected socket of the same type as the selected socket or a value socket if there is no selection. The triangle at the bottom of the list has filtering functions to facilitate finding nodes if the group has a large number of sockets.

### Ungroup

---

**Reference****Mode** All Modes**Menu** *Group* → *Ungroup***Hotkey** Ctrl-Alt-G

---

The Ctrl-Alt-G tool removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread 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.**Group Insert**

---

**Reference****Mode** All Modes**Menu** *Node* → *Group Insert*

---

Selecting a set of nodes, ending with the destination group node, and pressing *Node* → *Group Insert* 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.

**Appending Node Groups**

---

**Reference****Editor** Topbar**Mode** All Modes**Menu** *File* → *Link/Append*

---

Once you have appended a Node Tree to your blend-file, you can make use of it in a node editor by pressing Shift-A, *Add* → *Group*, then selecting the appended group. The “control panel” of the Group include the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

**Layout Nodes**

These are nodes which help you control the layout and connectivity of nodes within the Compositor.

## Switch Node

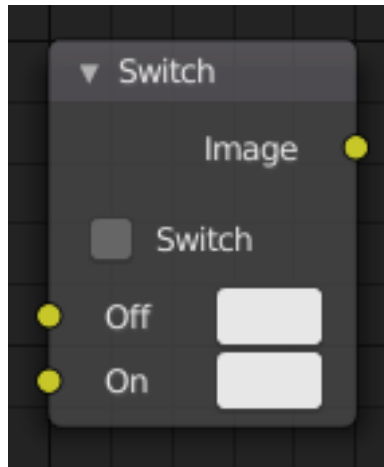


Fig. 342: Switch Node.

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 image 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.

---

## 2.3.6 Texture Nodes

### Introduction

---

**Note:** The texture node system is legacy and will be replaced soon by a new system. Due to this, the manual is not up to date with the latest version of Blender.

---

Blender includes a node-based texture generation system, which enables textures creation by combining colors, patterns and other textures in the same way as shader writing with *material nodes*.

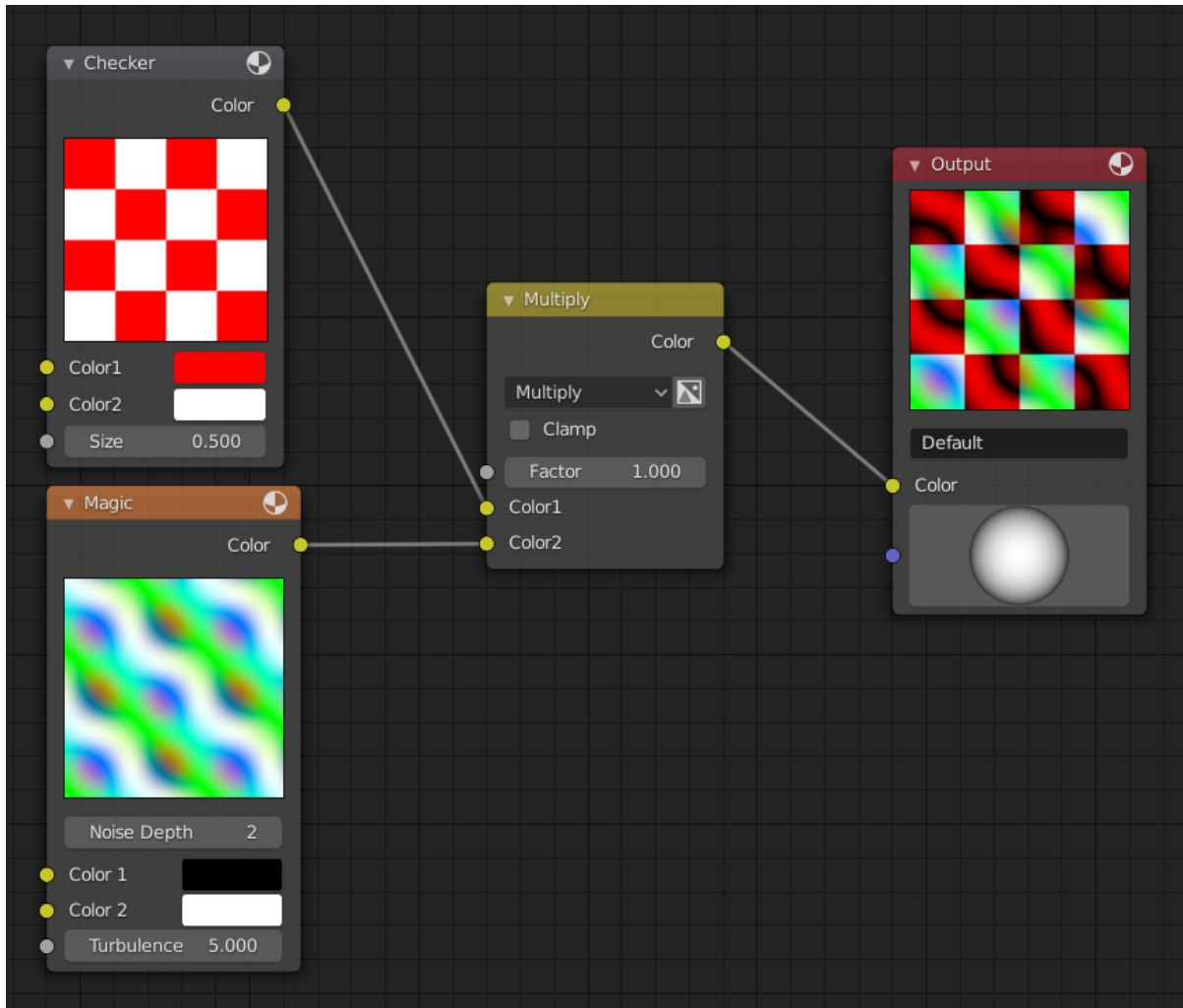


Fig. 343: Combined textures based on nodes.

These textures can be used for brushes, compositing and inside particle systems.

### Using Texture Nodes

To use texture nodes with the current texture, open the Texture Node editor.

A new texture can be created by either clicking the *New* button in the Texture Node editor, or the *New* button in the Texture panel. Once a texture is selected, it can be toggled to a function as a regular texture or a node texture by clicking the *Use Nodes* option in the Texture Node editor.

The default node setup will appear: a red-and-white checkerboard node connected to an *Output* named “Default”. For texture nodes, multiple Outputs can exist in the node setup. Compare to other types of node contexts, which are limited to one active Output node. See the next section for details.

For instructions on how to add, remove and manipulate the nodes in the tree, see the *nodes* reference.

## Using Multiple Outputs

Each texture defined with texture nodes can have several outputs, which can then be used for different things. For example, a texture that defines both a diffuse (color) map and a normal map. This can be done by:

1. Create two texture slots in the texture list, and set them to the same texture data-block.
2. Add two *Output* nodes to the node tree, and type new names into their *Name* text fields: e.g. “Diffuse” for one and “Normal” for the other.
3. Underneath the texture list view in the Texture panel, a selector with the names of the outputs is shown. For each entry in the Texture list, select the desired output by changing the menu entry (e.g. set one to *Diffuse* and the other to *Normal*).

These named outputs could be used, when the material is defined with material nodes. In this case, Texture Channels are probably not used. Instead, insert the texture nodes into the material node tree by using *Add* → *Input* → *Texture*. Inside the just added texture node the output to use can then be selected (e.g. *Diffuse* or *Normal*).

## Header

**Pin (pin icon)** The pin button will keep the current texture selection fixed. When a texture is pinned, it will remain visible in the Texture Editor even when another object or simulation is selected elsewhere.

## Node Types

### Color Nodes

#### Invert Node

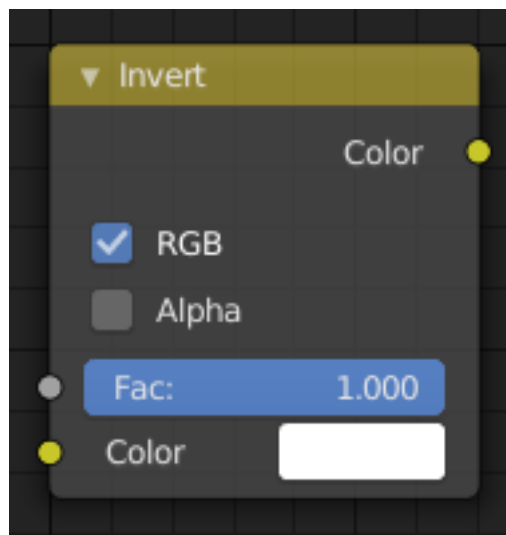


Fig. 344: Invert Node.

The *Invert Node* inverts the colors in the input image, producing a negative.



## Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Color** Standard image input.

## Properties

In the compositing context this node has the following properties.

**RGB** De/activation of the color channel inversion.

**Alpha** De/activation of the alpha channel inversion.

## Outputs

**Color** Standard image output.

## Example

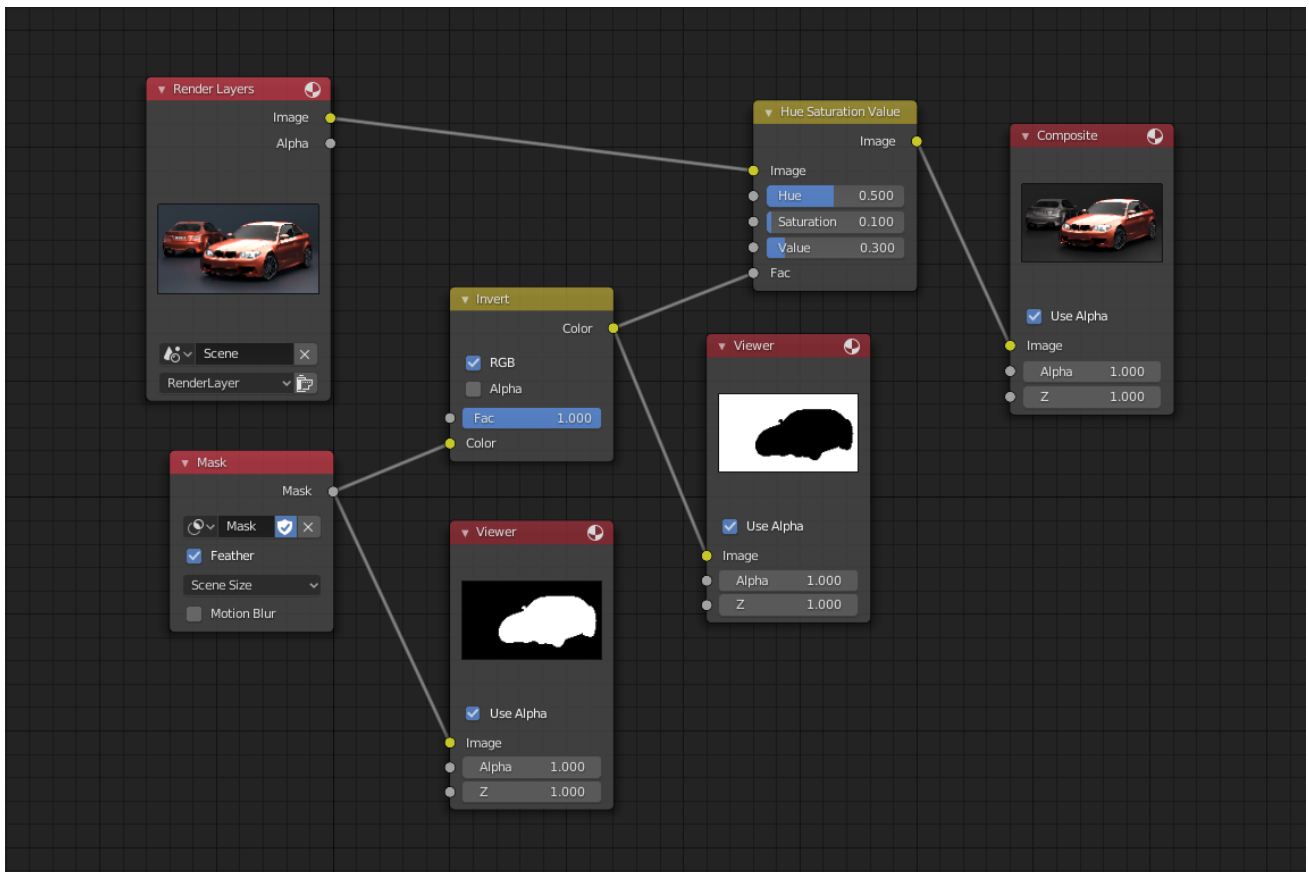


Fig. 345: The Invert node is used to invert the mask.

## Mix Node

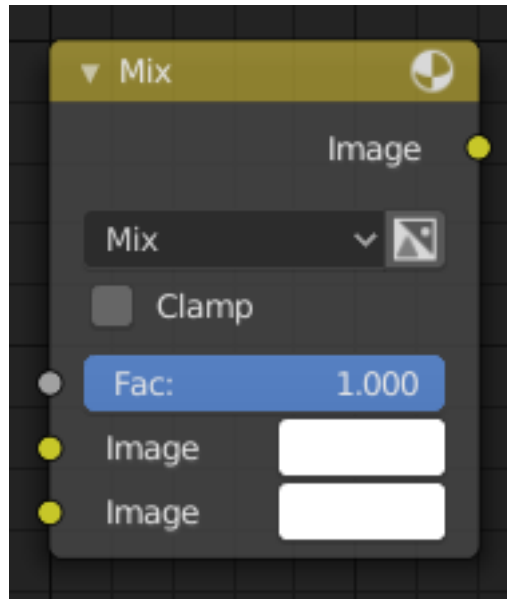


Fig. 346: Mix Node.

The *Mix Node* mixes images by working on the individual and corresponding pixels of the two input images. Called “MixRGB” in the shader and texture context.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** The background image. The image size and resolution sets the dimensions of the output image.

**Image** The foreground image.

### Properties

**Mix** 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

**Use Alpha** If activated, by clicking on the *Color and Alpha* icon, the Alpha channel of the second image is used for mixing. When deactivated, the default, the icon background is a light gray. The alpha channel of the base image is always used.

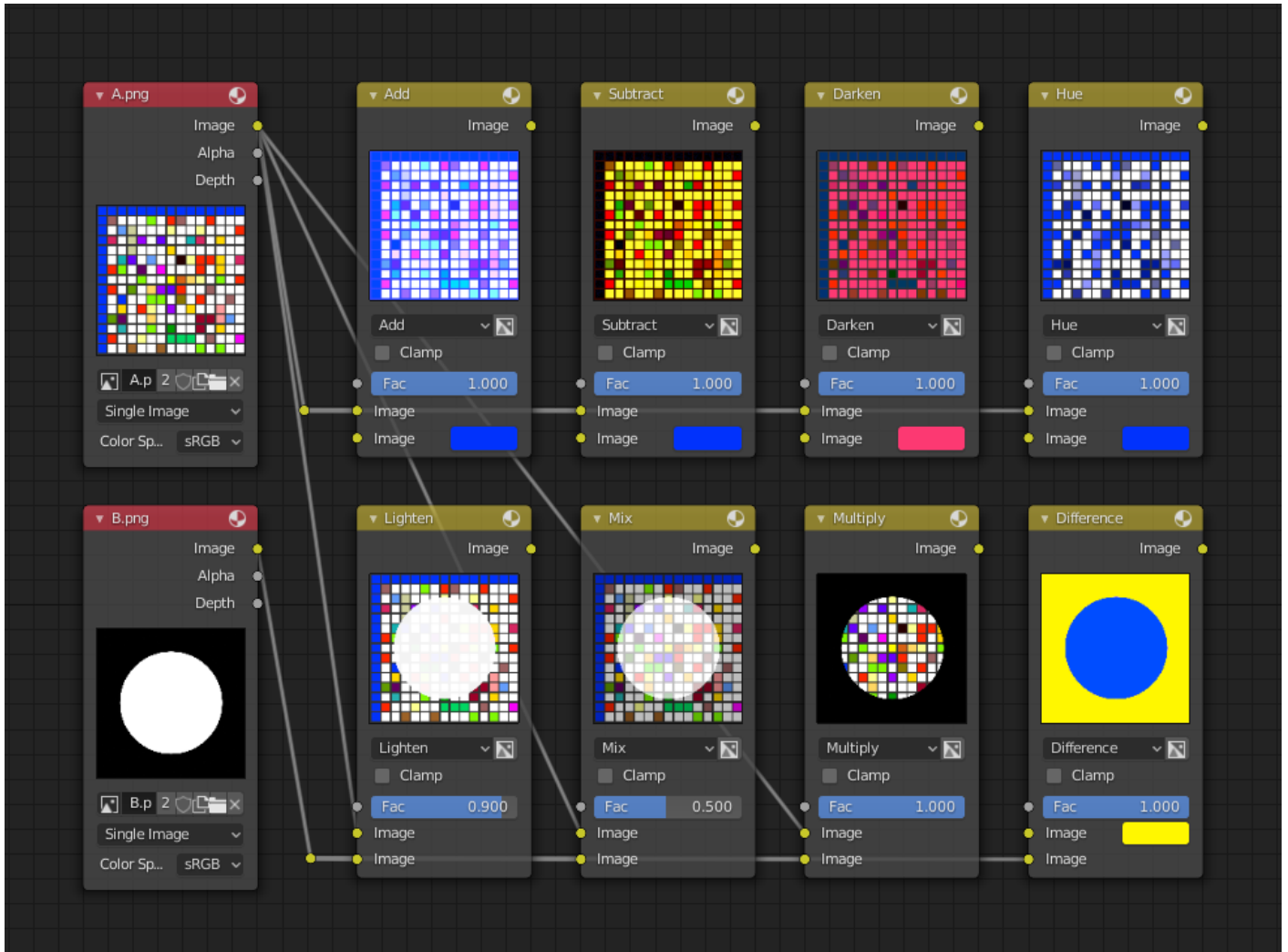
**Clamp** Limit the output value between 0.0 and 1.0.

### Outputs

**Image** Standard image output.

## Examples

Below are samples of common mix modes and uses, mixing a color or checker with a mask.



Some explanation of the mixing methods above might help you use the Mix node effectively:

**Add** Adding blue to blue keeps it blue, but adding blue to red makes purple. White already has a full amount of blue, so it stays white. Use this to shift a color of an image. Adding a blue tinge makes the image feel colder.

**Subtract** Taking Blue away from white leaves Red and Green, which combined make Yellow. Taking Blue away from Purple leaves Red. Use this to desaturate an image. Taking away yellow makes an image bluer and more depressing.

**Multiply** Black (0.0) times anything leaves black. Anything times White (1.0) is itself. Use this to mask out garbage, or to colorize a black-and-white image.

**Hue** Shows you how much of a color is in an image, ignoring all colors except what is selected: makes a monochrome picture (style 'Black & Hue').

**Mix** Combines the two images, averaging the two.

**Lighten** Like bleach makes your whites whiter. Used with a mask to lighten up a little.

**Difference** It takes out a color. The color needed to turn Yellow into White is Blue. Use this to compare two very similar images to see what had been done to one to make it the other; sort of like a change log for images. You can use this to see a watermark (see *Watermark images*) you have placed in an image for theft detection.

**Darken** With the colors set here, it's like looking at the world through rose-colored glasses.

**Note:** Only add, subtract, multiply and divide are suitable for *Scene Referenced* images.

## Contrast Enhancement

Here is a small node tree showing the effects of two other common uses for the RGB Curve: *Darken* and *Contrast Enhancement*. You can see the effect each curve has independently, and the combined effect when they are *mixed* equally.

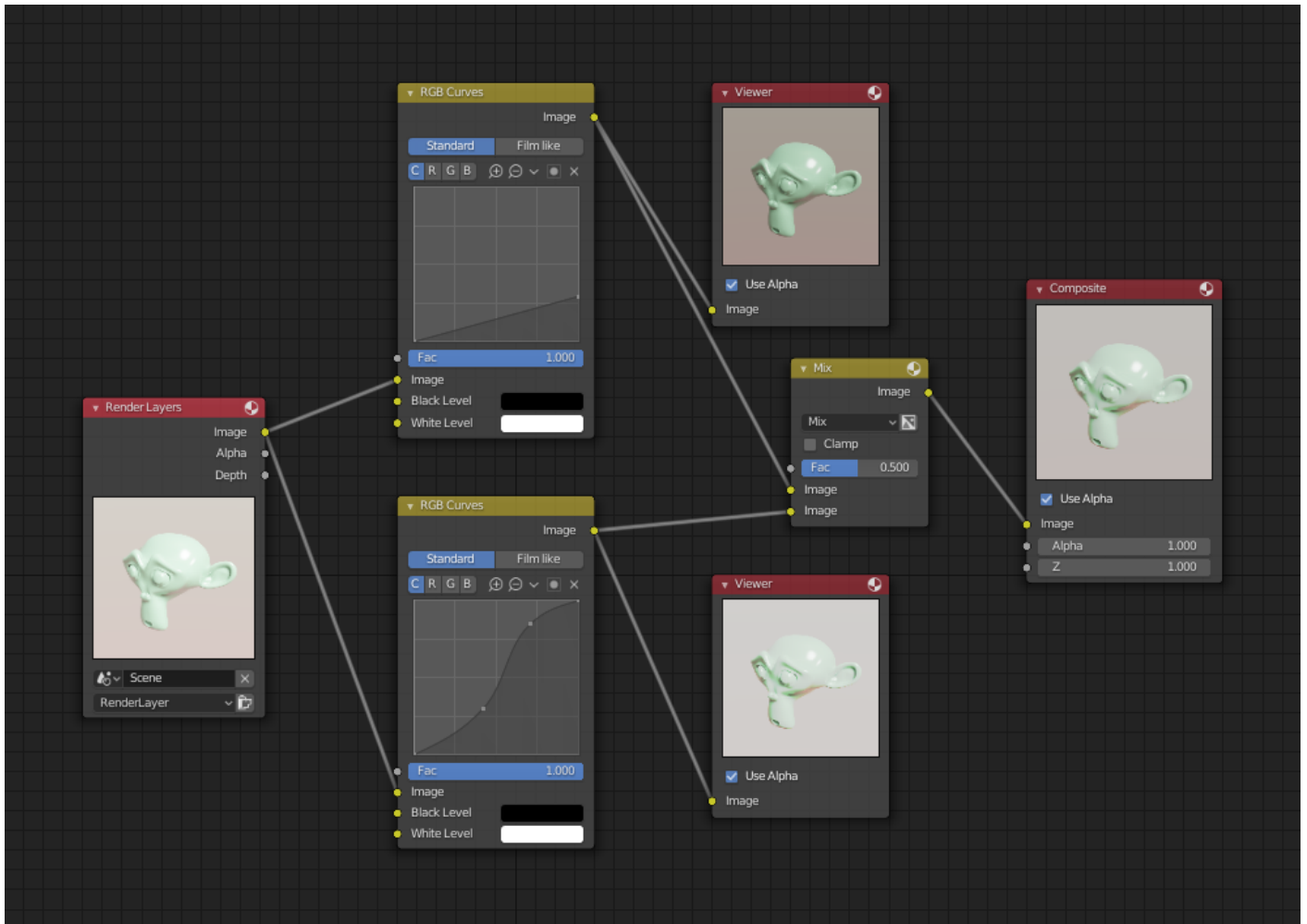


Fig. 347: Example node setup showing “Darken”, “Enhance Contrast” and “Mix” nodes for composition.

As you can hopefully see, our original magic monkey was overexposed by too much light. To cure an overexposure, you must both darken the image and enhance the contrast.

In the top RGB curve, *Darken*, only the right side of the curve was lowered; thus, any X input along the bottom results in a geometrically less Y output. The *Enhance Contrast* RGB (S-shaped) curve scales the output such that middle values of X change dramatically; namely, the middle brightness scale is expanded, and thus, whiter whites and blacker blacks are output. To make this curve, simply click on the curve and a new control point is added. Drag the point around to bend the curve as you wish. The Mix node combines these two effects equally, and Suzanne feels much better.

## 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 image to numbers that the Map Value node can use to make the image subliminal. In this case, it reduces the mark to one-tenth of its original intensity. The Add node adds the corresponding pixels, making the ones containing the mark ever-so-slightly brighter.

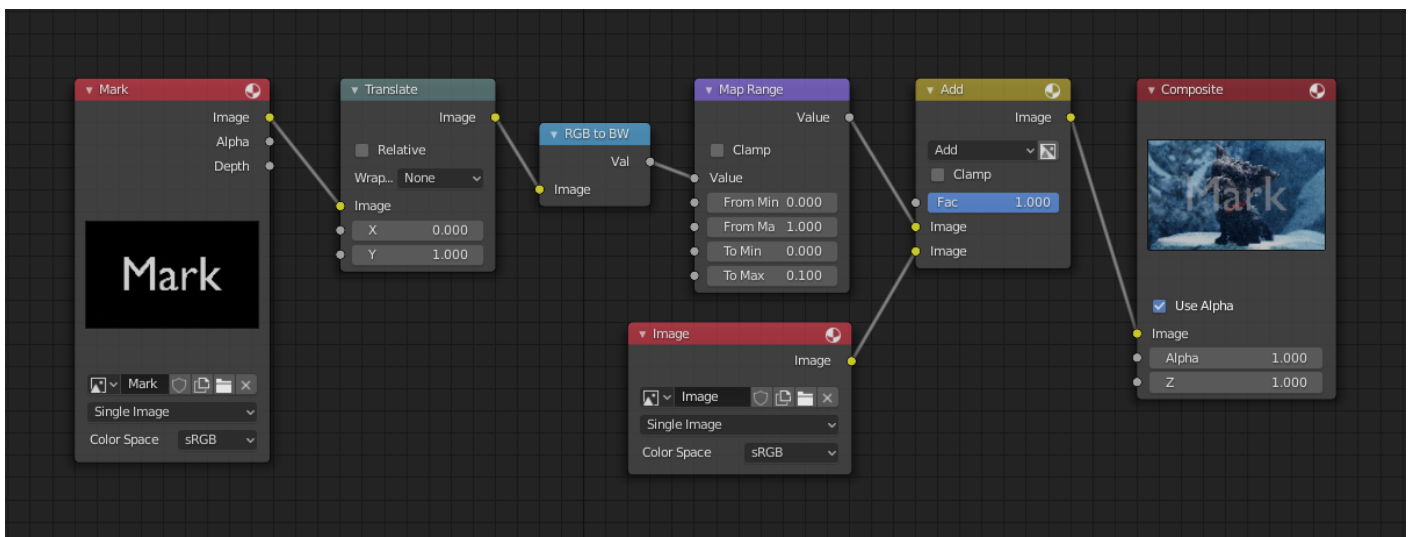


Fig. 348: Embedding your mark in an image using a mark and specific position.

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!

---

#### Hint: Additional uses

You can also use this technique, using settings that result in visible effects, in title sequences to make the words appear to be cast on the water's surface, or as a special effect to make words appear on the possessed girl's forearm.

---

### 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. The result is routed to a viewer, and you can see how the original mark clearly stands out.

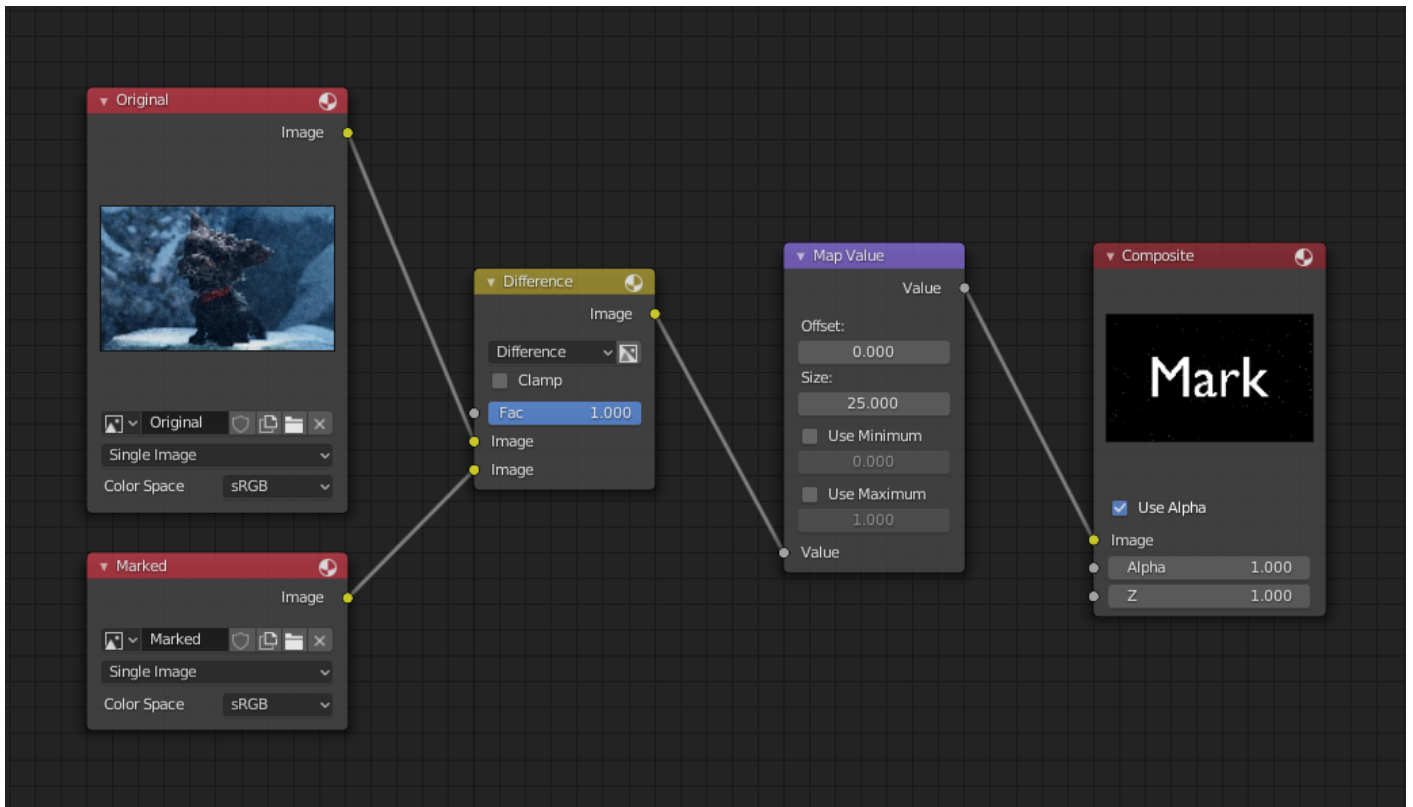


Fig. 349: Checking an image for your watermark.

Various image compression algorithms lose some of the original; the difference shows as noise. Experiment with different compression settings and marks to see which works best for you by having the encoding node group in one scene, and the decoding group in another. Use them while changing Blender's image format settings, reloading the watermarked image after saving, to get an acceptable result. In the example above, the mark was clearly visible all the way up to JPEG compression of 50%.

## RGB Curves Node

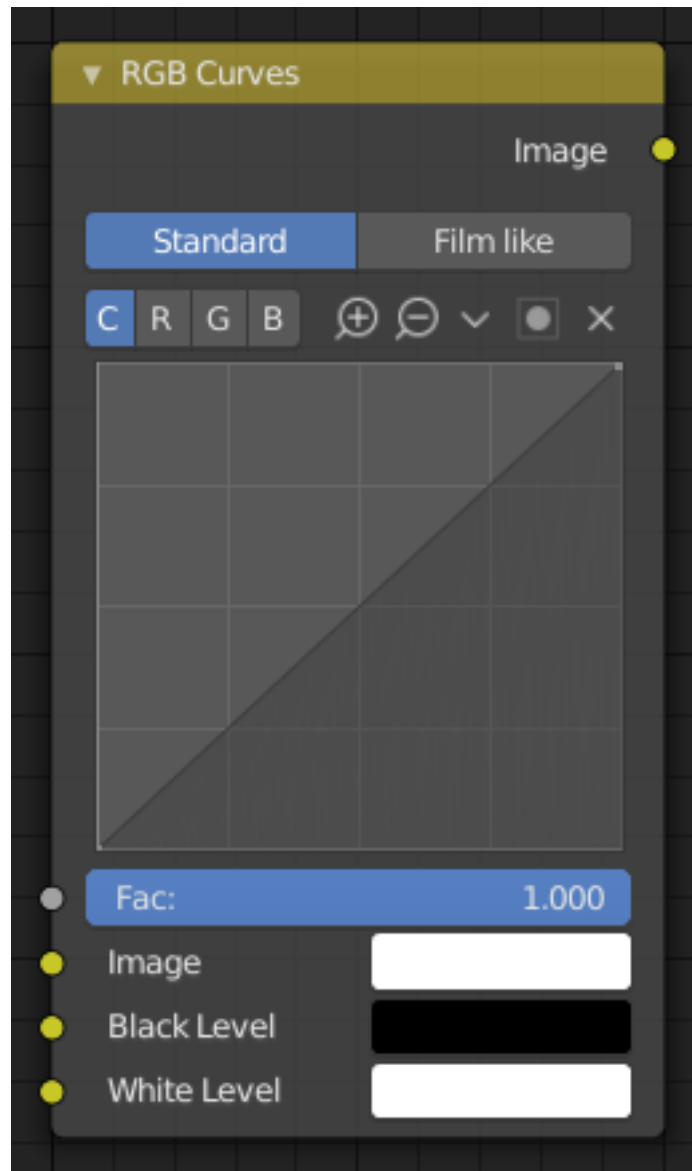


Fig. 350: RGB Curves Node.

The *RGB Curves Node* allows color corrections for each color channel and levels adjustments in the compositing context.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** Standard image input.

**Black Level *Compositor Only*** Defines the input color that is (linear) mapped to black.

**White Level *Compositor Only*** Defines the input color that is (linear) mapped to white.

---

**Tip:** To define the levels, use the *eyedropper* to select a color sample of a displayed image.

---

## Properties

### Tone

**Standard** TODO 2.8

**Film like** TODO 2.8

**Channel** Clicking on one of the channels displays the curve for each.

C (Combined RGB), R (Red), G (Green), B (Blue)

**Curve** A Bézier curve that varies the input levels (X axis) to produce an output level (Y axis). For the curve controls see: *Curve widget*.

### Outputs

**Image** Standard image output.

### Examples

Below are some common curves you can use to achieve desired effects.

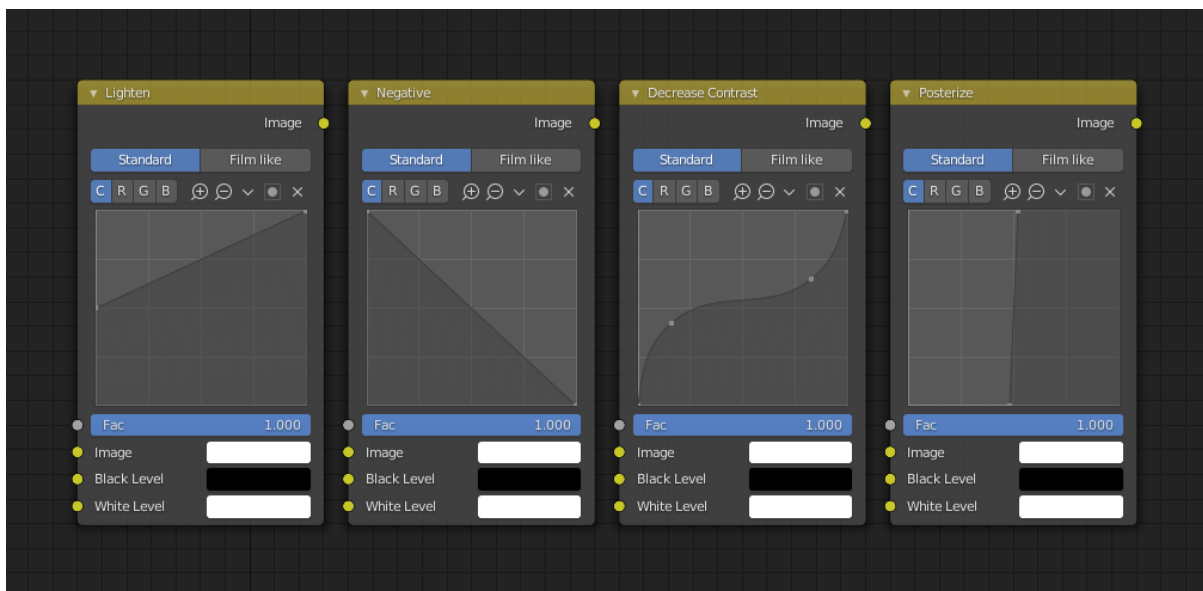


Fig. 351: From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

### Color Correction using 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.

Also, read on for examples of the Darken and Contrast Enhancement curves, [here](#).

### Color Correction using 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.



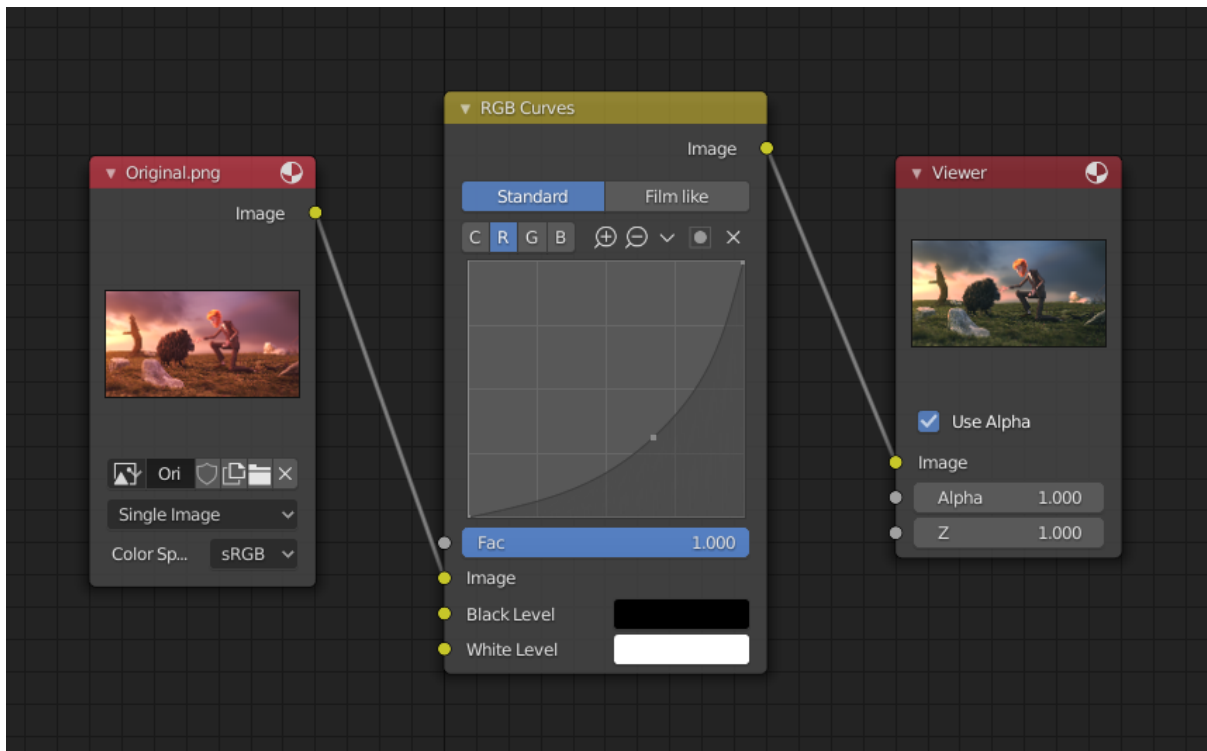


Fig. 352: Color correction with curves.

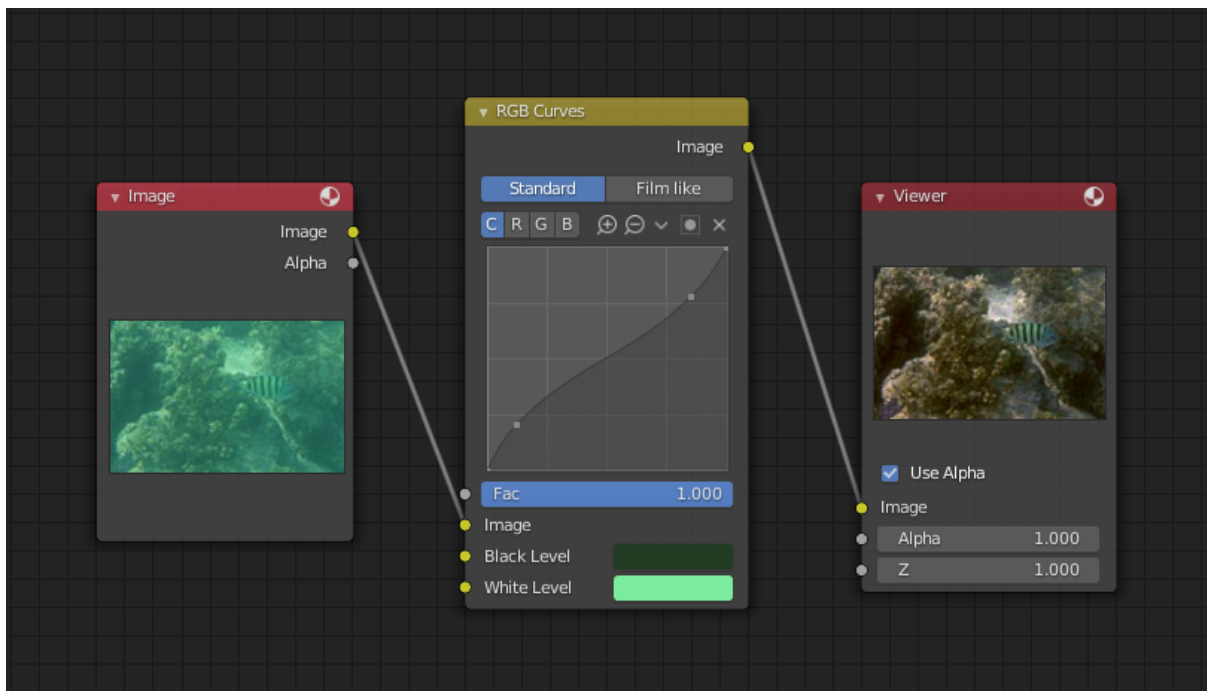


Fig. 353: Color correction with Black/White Levels.

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

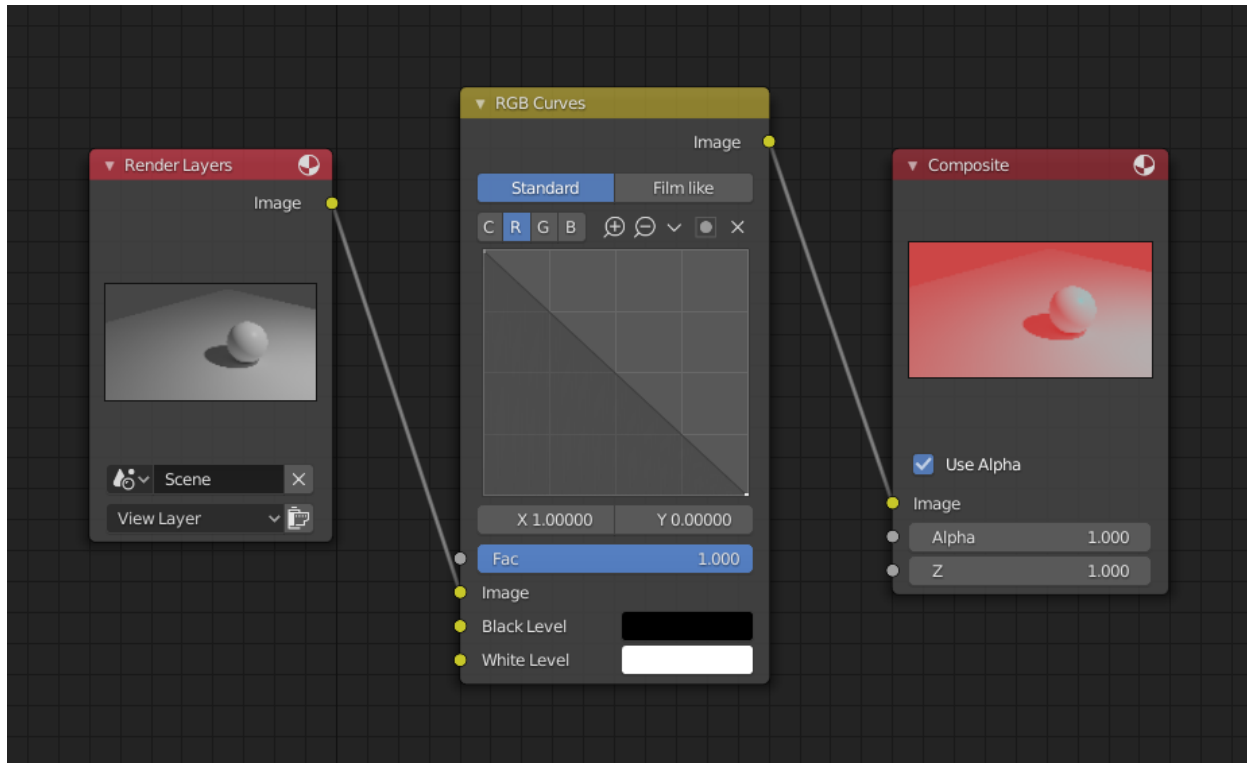


Fig. 354: Changing colors by inverting the red channel.

## Hue Saturation Value Node

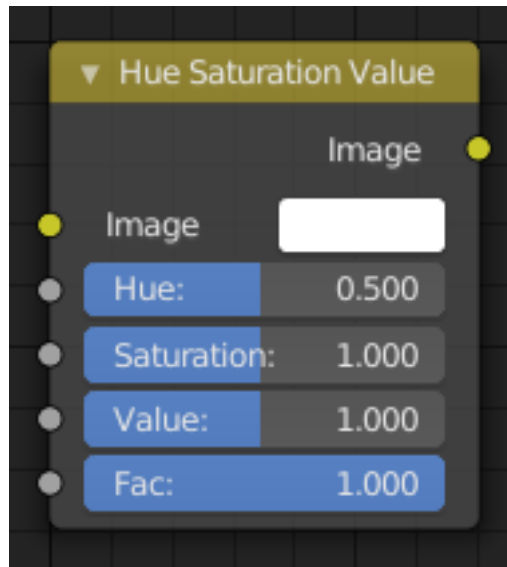


Fig. 355: Hue Saturation Node.

The *Hue Saturation Value Node* applies a color transformation in the HSV color space.

### Inputs

**Factor** Controls the amount of influence the node exerts on the output image.

**Image** Standard image input.

### Properties

The transformations are relative shifts. In the shader and texture context the following properties are available as input sockets.

**Hue** Specifies the hue rotation of the image.  $360^\circ$  are mapped to (0 to 1). The hue shifts of 0 ( $-180^\circ$ ) and 1 ( $+180^\circ$ ) 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.

### Outputs

**Image** Standard image output.

### Hue/Saturation Tips

Some things to keep in mind that might help you use this node better:

**Hues are vice versa** A blue image, with a Hue setting at either end of the spectrum (0 or 1), is output as yellow (recall that white, minus blue, equals yellow). A yellow image, with a Hue setting at 0 or 1, is blue.

**Hue and Saturation work together.** So, a Hue of 0.5 keeps the blues the same shade of blue, but *Saturation* can deepen or lighten the intensity of that color.

**Gray & White are neutral hues** A gray image, where the RGB values are equal, has no hue. Therefore, this node can only affect it with *Value*. This applies to all shades of gray, from black to white; wherever the values are equal.

**Changing the effect over time** The Hue and Saturation values can be animated with a *Time Node* or by animating the property.

**Note:** Tinge

This HSV node simply shifts hues that are already there. To colorize a gray image, or to add a tint to an image, use a *Mix* node to add in a static color from an RGB input node with your image.

## HSV Example



Fig. 356: A basic example.

## Combine/Separate Nodes

All of these nodes do essentially the same thing:

- Separate: Split out an image into its composite color channels.
- Combine: Re/combine an image from its composite color channels.

These nodes can be used to manipulate each color channel independently. Each type is differentiated in the applied *Color Space*.

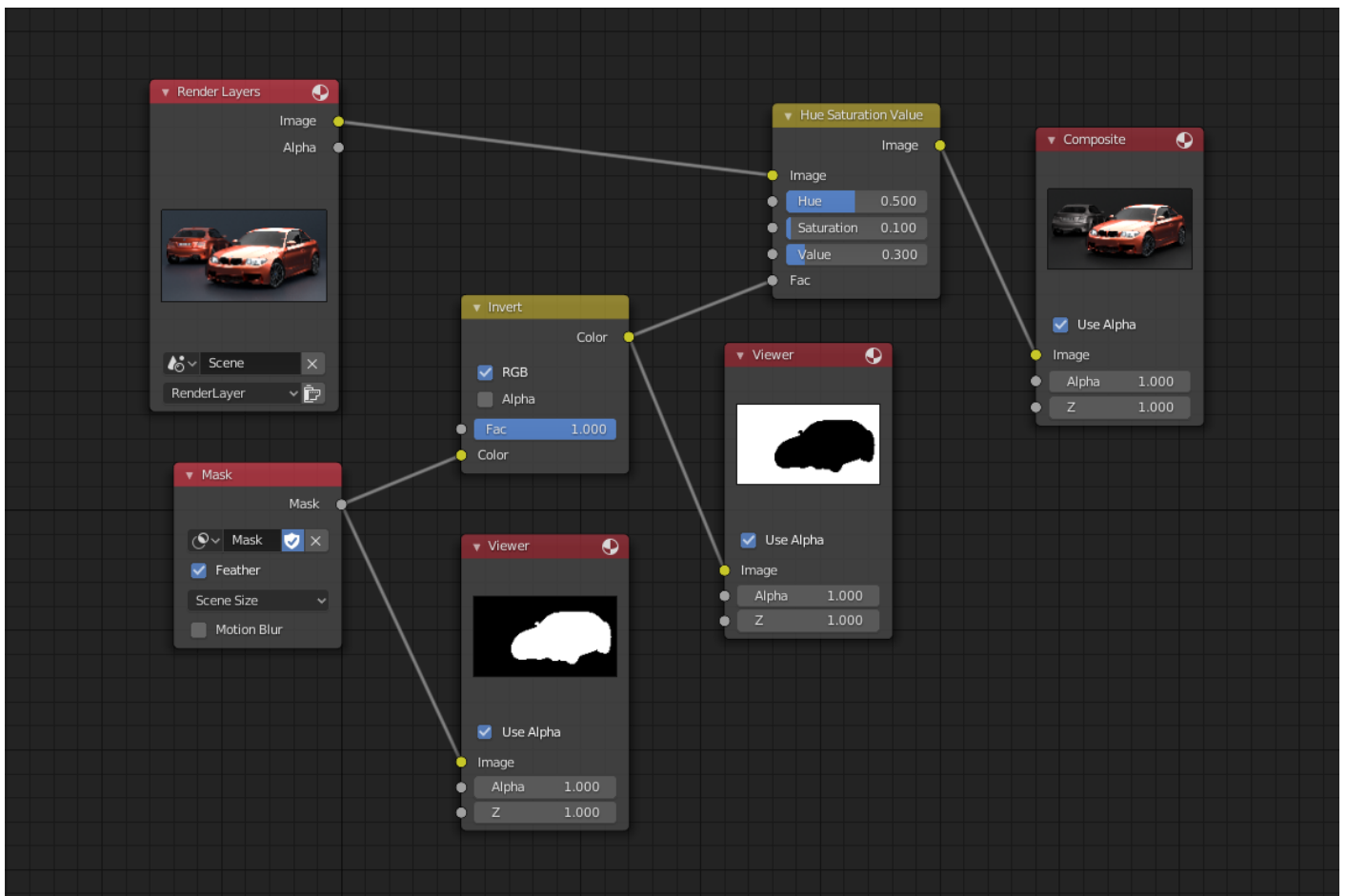


Fig. 357: An example of using the Factor input for masking.

In compositing and texture context each node supports the Alpha channel. In the texture context only RGB color space is available. In the shading context of Cycles combine and separate nodes are added for HSV and vectors (XYZ).

The Combine nodes can also be used to input single color values. For RGBA and HSVA color spaces it is recommended to use the *RGB Node*. Some common operations could easier be executed with the *Color Nodes*.

### Separate/Combine RGBA Nodes

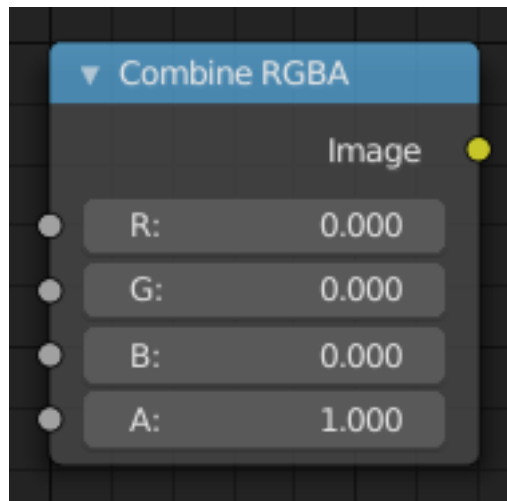


Fig. 358: Combine RGBA Node.

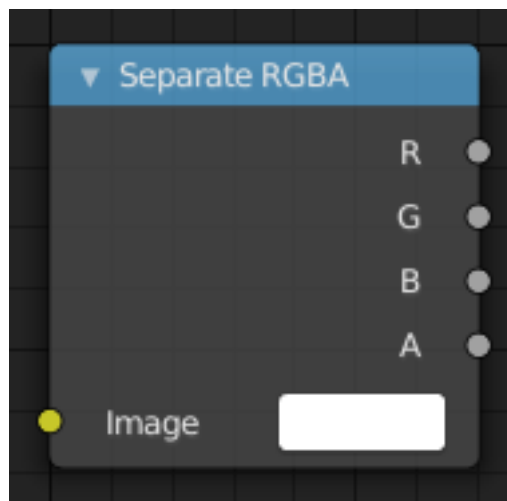


Fig. 359: Separate RGBA Node.

### Input/Output

**Image** Standard image in/output.

- R (Red)
- G (Green)
- B (Blue)

- A (Alpha)

## Properties

This node has no properties.

## Examples

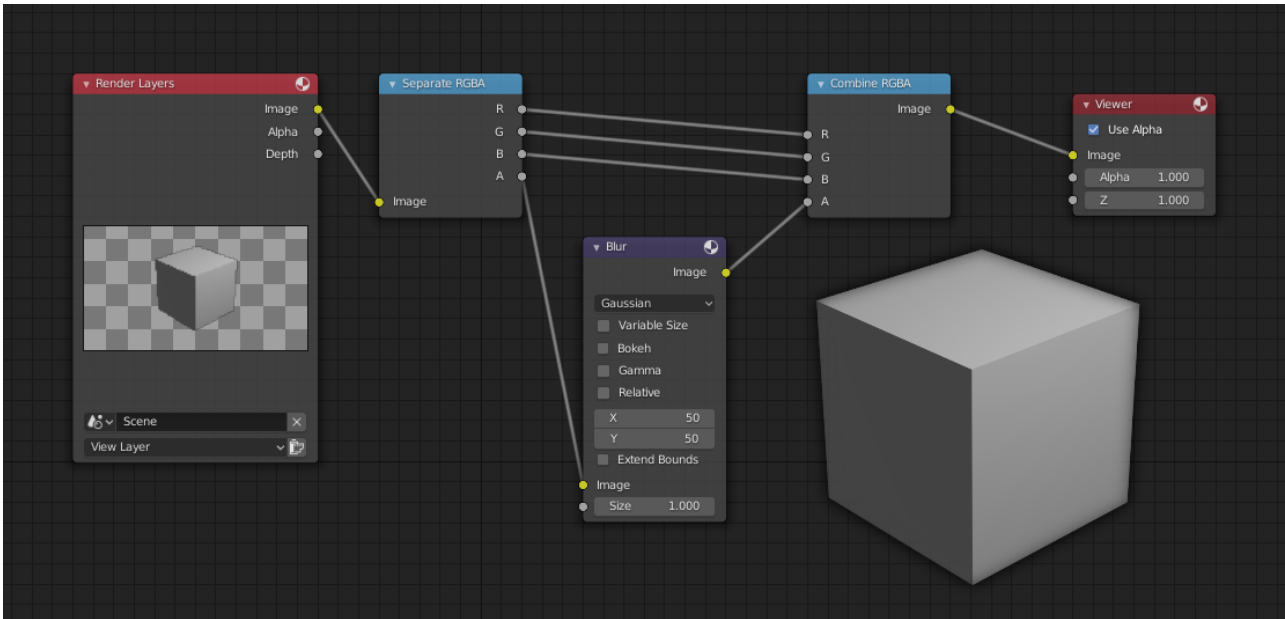


Fig. 360: An example of blurring the alpha channel.

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.

## Separate/Combine HSVA Nodes

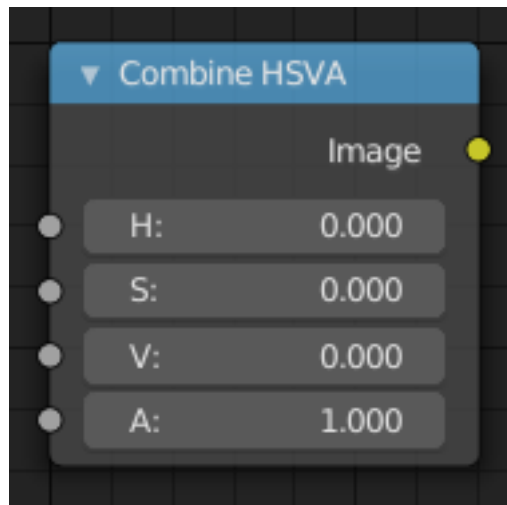


Fig. 361: Combine HSVA Node.

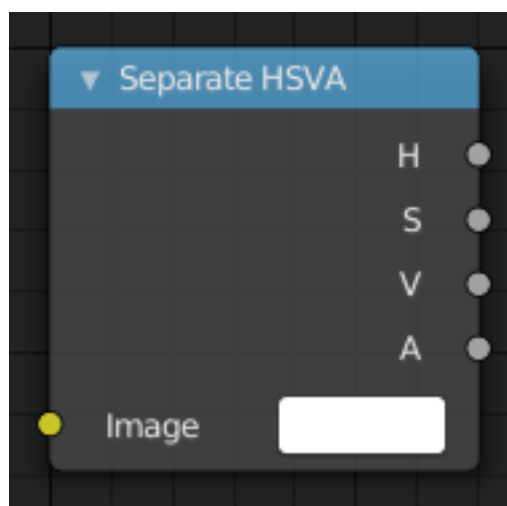


Fig. 362: Separate HSVA Node.

### Input/Output

**Image** Standard image in/output.

- H (Hue)
- S (Saturation)
- V (Value)
- A (Alpha)

### Properties

This node has no properties.



## Separate/Combine YUVA Nodes

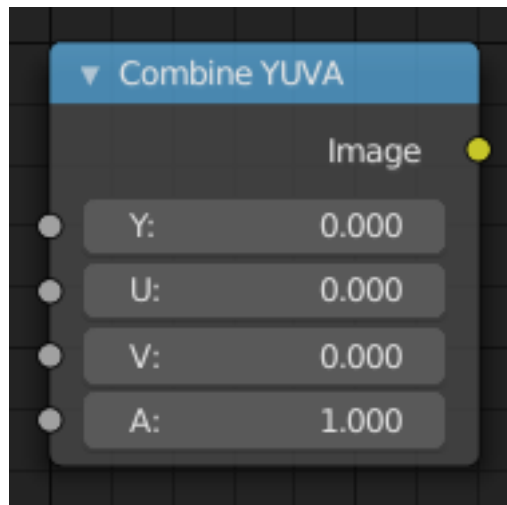


Fig. 363: Combine YUVA Node.



Fig. 364: Separate YUVA Node.

### Input/Output

**Image** Standard image in/output.

- Y (Luminance)
- U (U chrominance)
- V (V chrominance)
- A (Alpha)

### Properties

This node has no properties.

## Separate/Combine YCbCrA Node

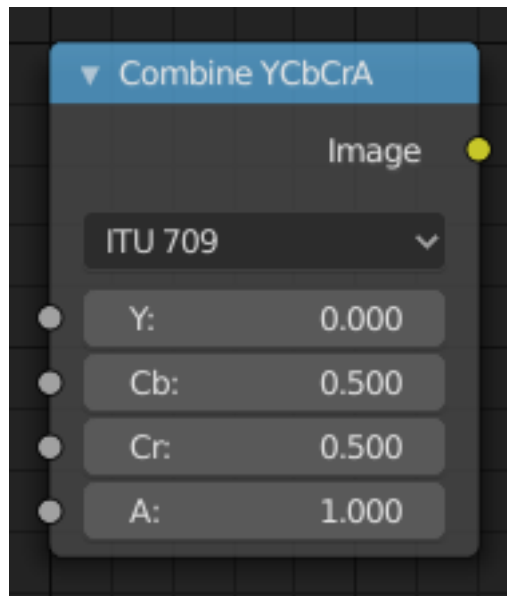


Fig. 365: Combine YCbCrA Node.

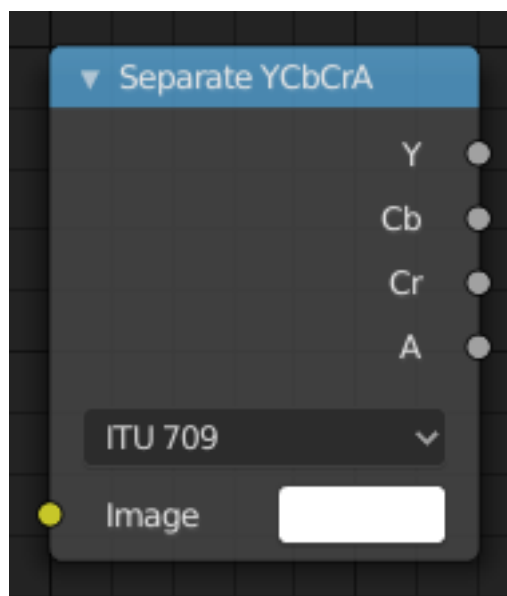


Fig. 366: Separate YCbCrA Node.

### Input/Output

**Image** Standard image in/output.

- Y (Luminance)
- Cb (Chrominance Blue)
- Cr (Chrominance Red)
- A (Alpha)

## Properties

**Mode** ITU 601, ITU 709, Jpeg

## Examples

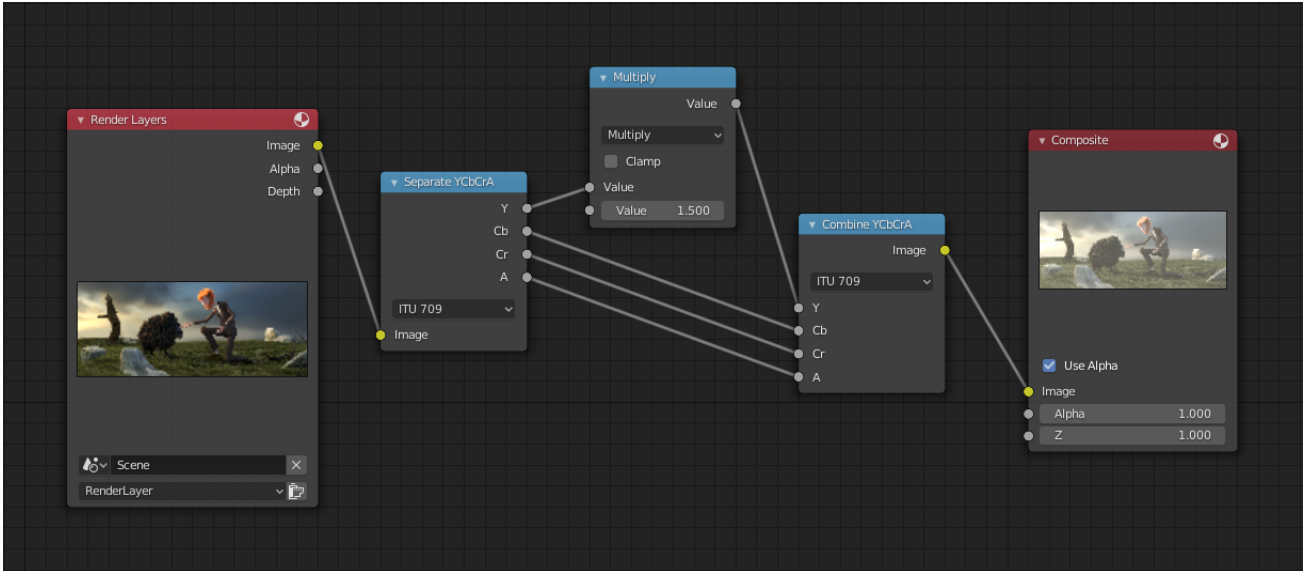


Fig. 367: An example of the scaling the Luminance channel.

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.

---

## Converter Nodes

### Color Ramp Node

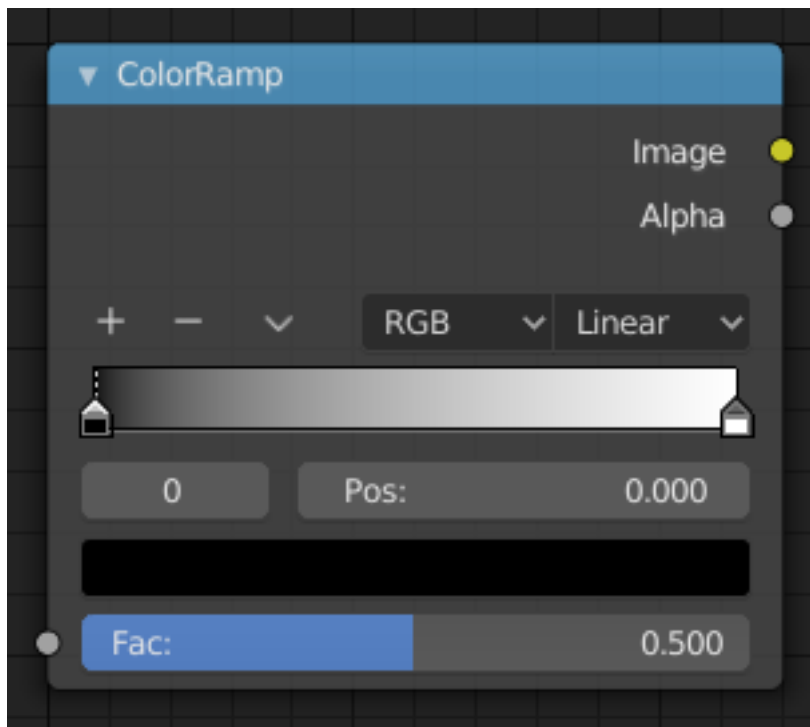


Fig. 368: Color Ramp Node.

The Color Ramp Node is used for mapping values to colors with the use of a gradient.

#### Inputs

**Factor** The Factor input is used as an index for the color ramp.

#### Properties

**Color Ramp** For controls see *Color Ramp Widget*.

#### Outputs

**Image** Standard image output.

**Alpha** Standard alpha output.

#### Examples

##### Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to create an alpha mask, or a mask that is overlaid on top of another image. Such a mask allows you to select parts of the background to be show through.

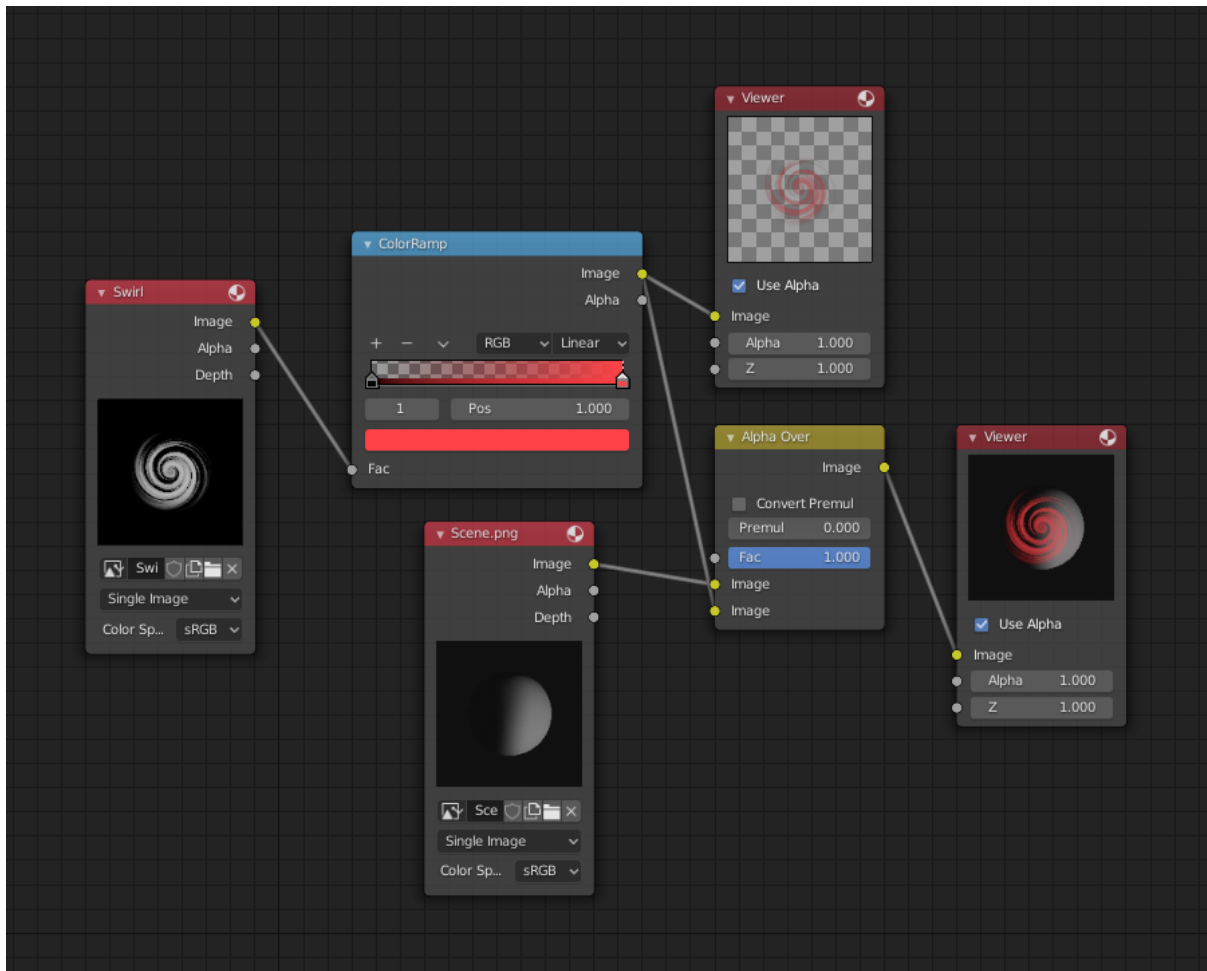


Fig. 369: Using the Color Ramp node to create an alpha mask.

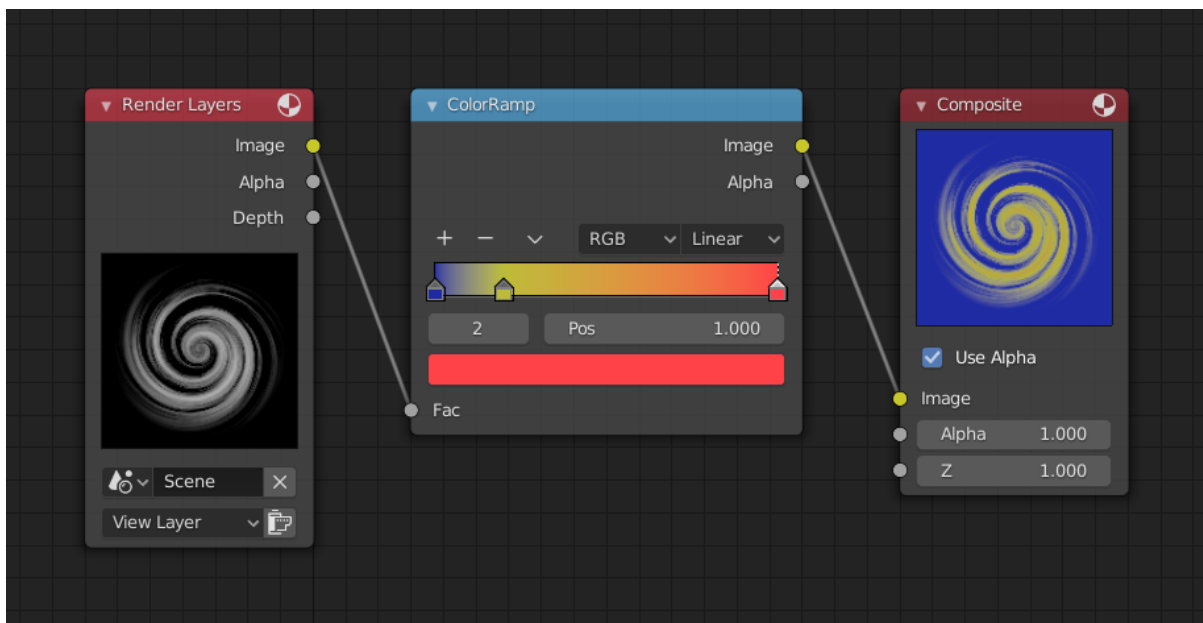
In the map above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

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 puts out a mask that is fully transparent where the image is black. Black is zero, so Color Ramp outputs the color at the left end of the gradient, which is set to transparent. The Color Ramp image is fully red and opaque where the image is white (which is 1).

You can verify that the output image mask is indeed transparent by overlaying it on top of another image.

### 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 (alpha of 1). Where the image is black, Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, 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

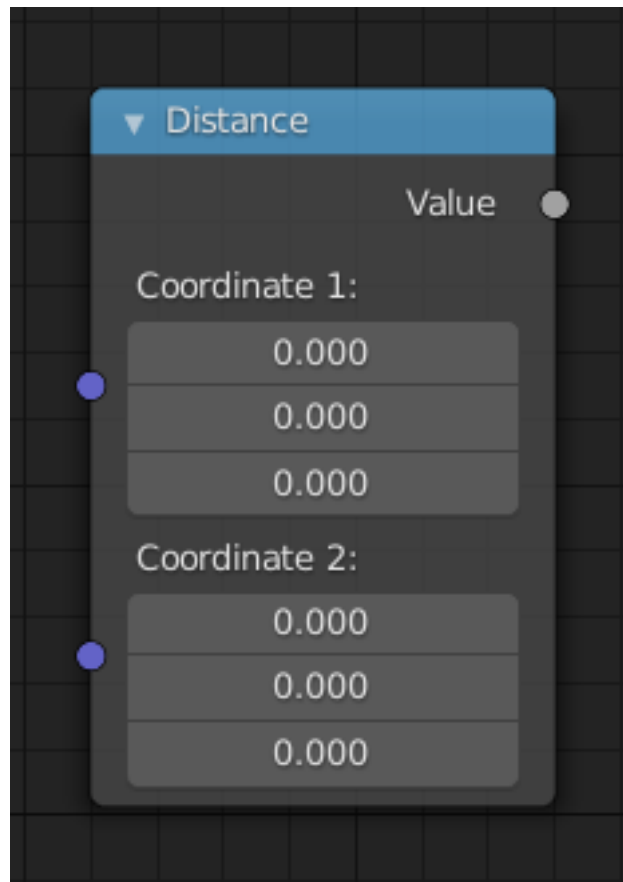


Fig. 370: Distance node.

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

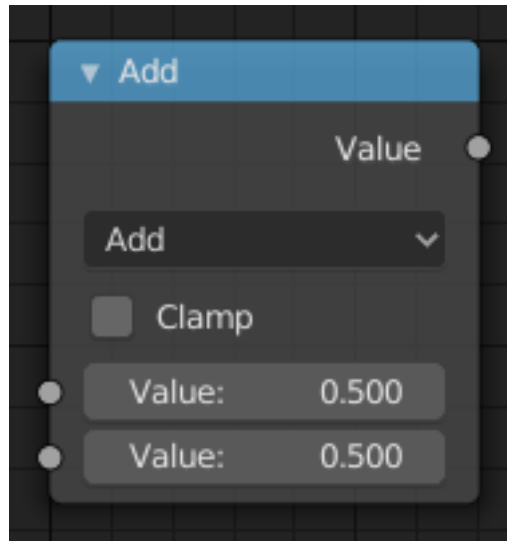


Fig. 371: Math node.

The *Math Node* performs math operations.

### Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Addend* input is only available in 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 with 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** Round 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** [Fraction](#).

**Modulo** Outputs the remainder once the first value is divided by the second value.

**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** Round the input value to down to the nearest integer multiple of *Increment*.

**Ping-pong** The output value is moved between 0.0 and the *Scale* based on the input value.

### 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

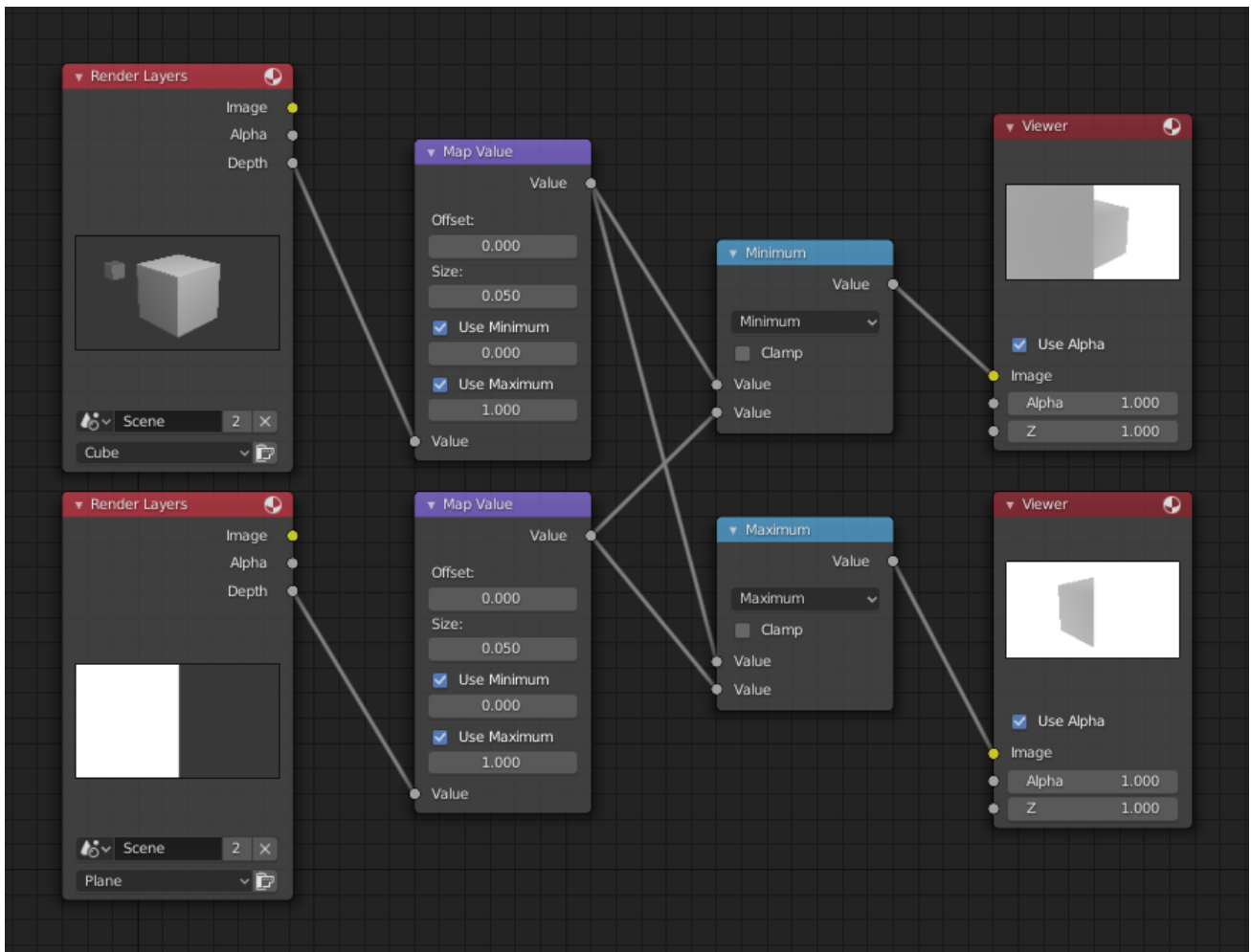


Fig. 372: Minimum and maximum function example.

This example has one scene input by the top *Render Layers* node, which has a cube that is about 10 units from the camera. The bottom *Render Layers* node inputs a scene with 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 divide the Z-buffer by 20 (multiply by 0.05, as shown in the Size field) and clamped to be a min/max of 0.0/1.0 respectively.

For the minimum function, the node selects those Z values where the corresponding pixel is closer to the camera; so it chooses the Z values for the plane and part of the cube. The background has an infinite Z value, so it is clamped to 1.0 (shown as white). In the maximum example, the Z values of the cube are greater than the plane, so they are chosen for the left side, but the plane *Render Layers Z* are infinite (mapped to 1.0) for the right side, so they are chosen.

## Using Sine Function to Pulsate

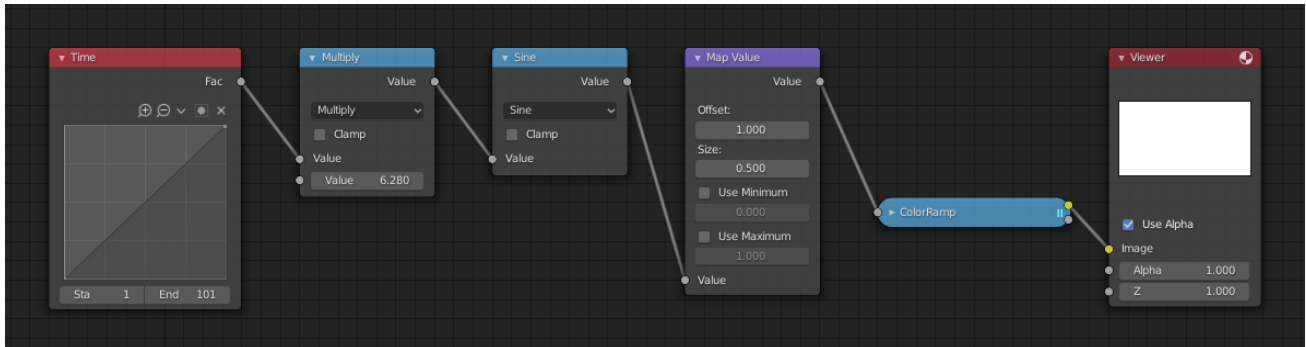


Fig. 373: Using sine function example.

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  $2 \times \pi$  (6.28) and converted to 1.0 by the Sine function, since  $\sin(2 \times \pi / 4) = \sin(\pi / 2) = +1.0$ .

Since the sine function can put out 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 one-half (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

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 *Bright/Contrast node* that might give simpler control over brightness.

## Restrict Color Selection (Posterization)

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:  $\text{round}(x \times n - 0.5) / (n - 1)$ , 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

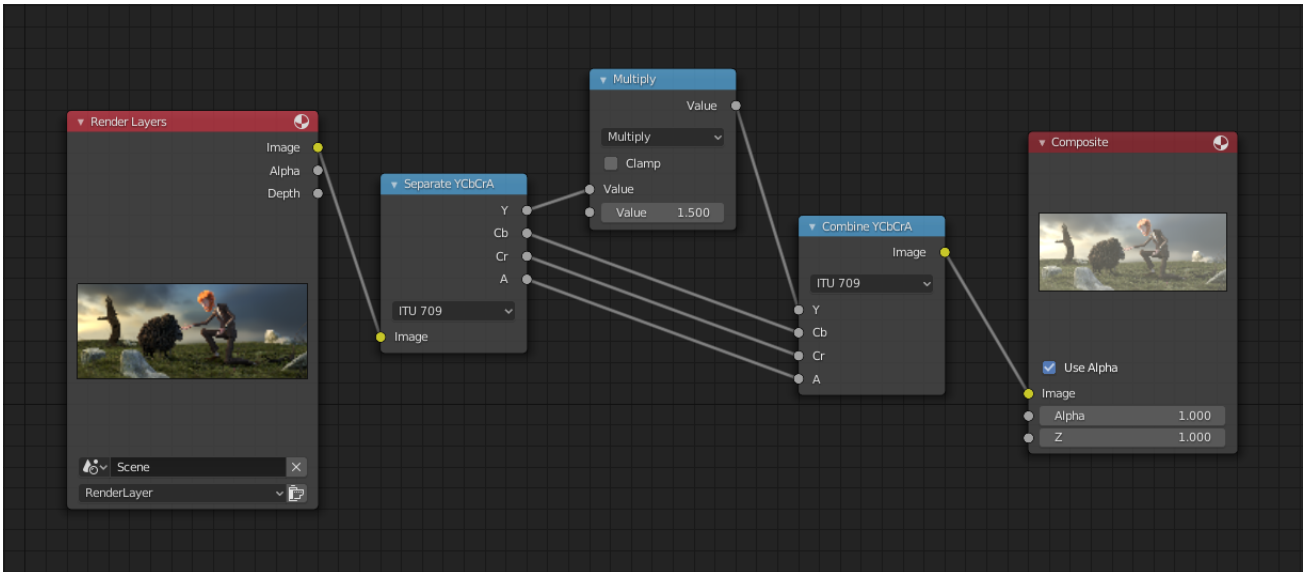


Fig. 374: Scaling a channel example.

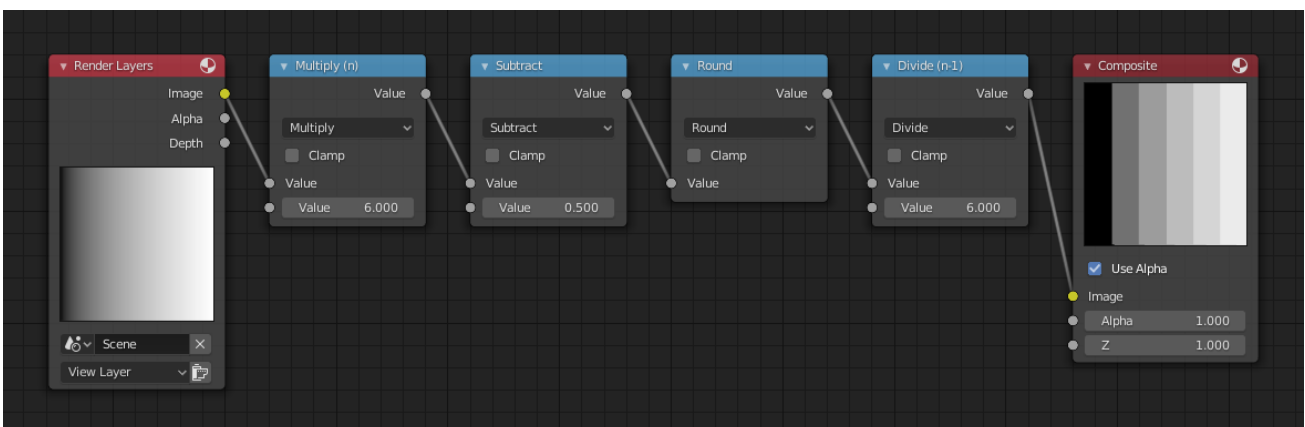


Fig. 375: Posterization example.

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 split it into separate RGB channels using *Separate/Combine RGBA* nodes and perform this operation on each channel independently.

### RGB to BW Node

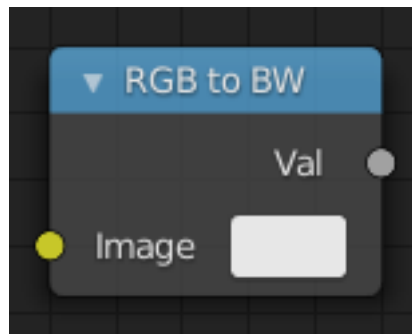


Fig. 376: RGB to BW Node.

The *RGB to BW Node* maps an RGB color image to a gray-scale by the luminance.

#### Inputs

**Image** Color image input.

#### Properties

This node has no properties.

#### Outputs

**Value** Gray-scale value output.

## Value to Normal Node

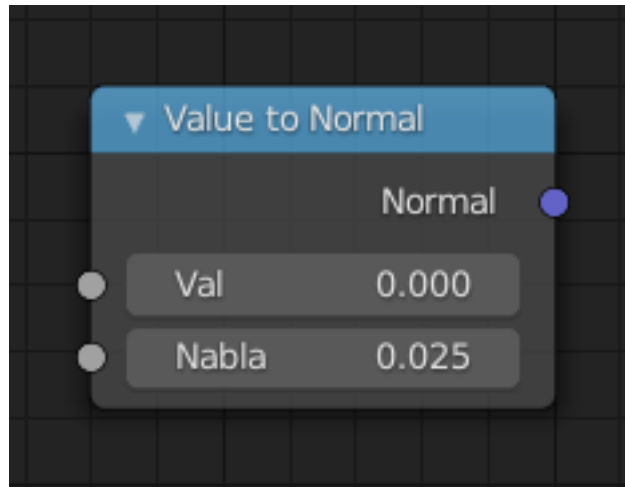


Fig. 377: Value to Normal node.

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

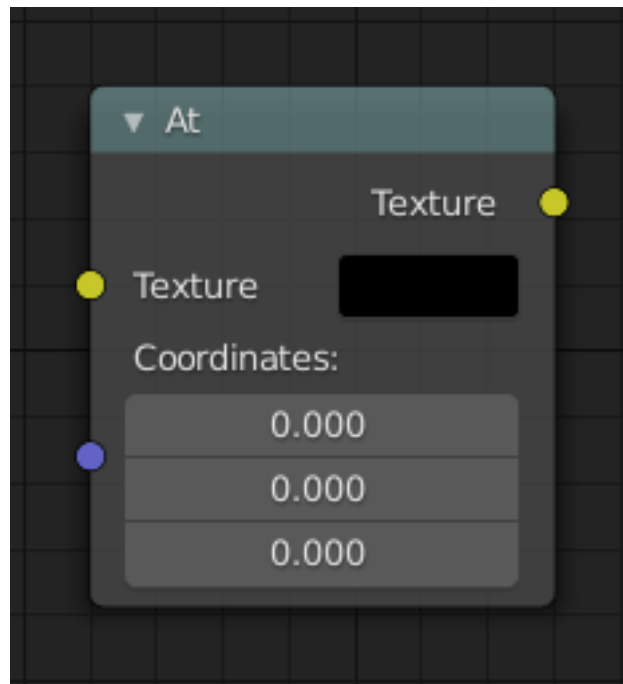


Fig. 378: At node.

Returns the color of a texture at the specified coordinates.

### Inputs

**Texture** Standard image 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 image output.

## Rotate Node

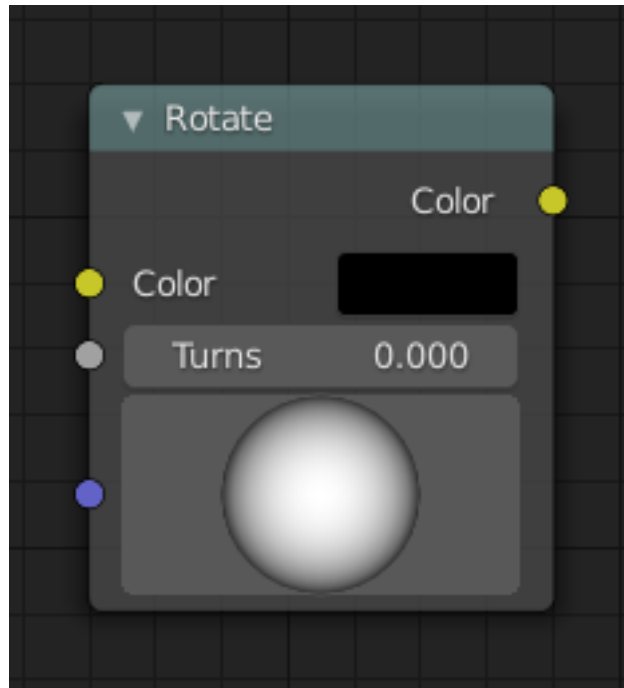


Fig. 379: Rotate node.

Rotate the texture coordinates of an image or texture.

### Inputs

**Color** Standard image 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 image output.



## Scale Node

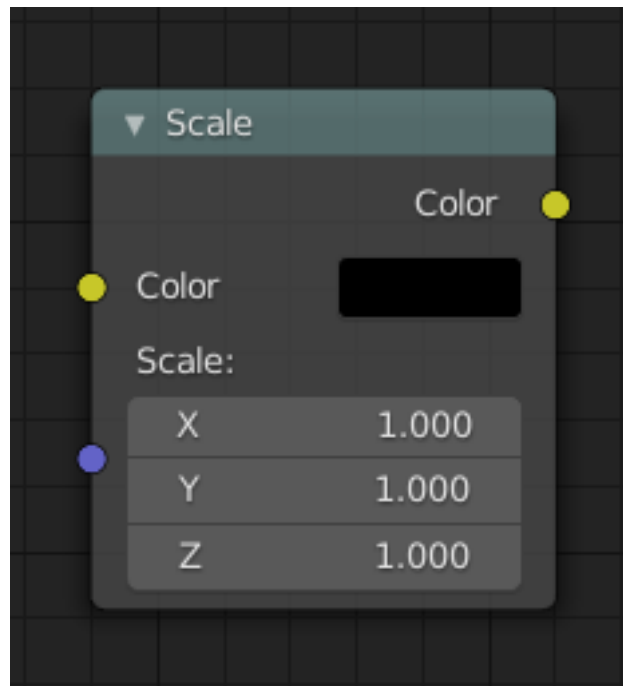


Fig. 380: Scale node.

Scale the texture coordinates of an image or texture.

### Inputs

**Color** Standard image input.

**Scale** The amount to scale the coordinates in each of the three axes.

### Properties

This node has no properties.

### Outputs

**Color** Standard image output.

## Translate Node

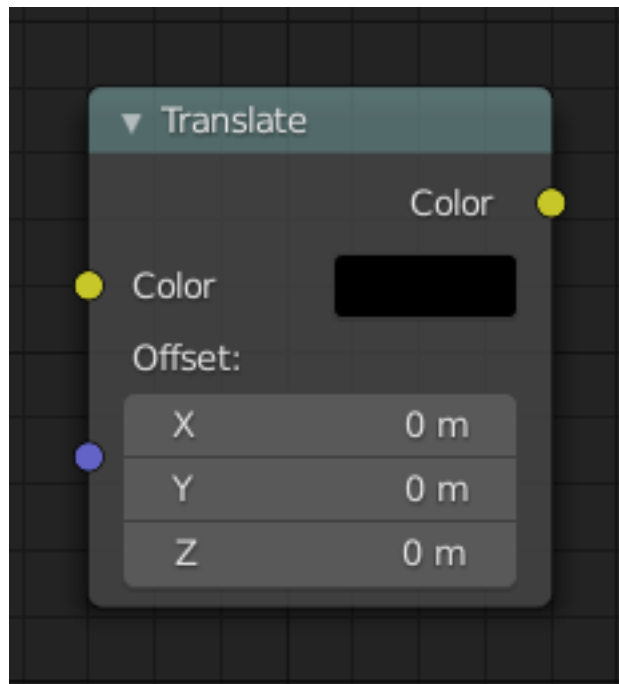


Fig. 381: Translate node.

Translate the texture coordinates of an image or texture.

### Inputs

**Color** Standard image input.

**Offset** The amount to offset the coordinates in each of the three axes.

### Properties

This node has no properties.

### Outputs

**Color** Standard image output.

### Input Nodes

Input nodes provide input data for other nodes.

## Coordinates Node

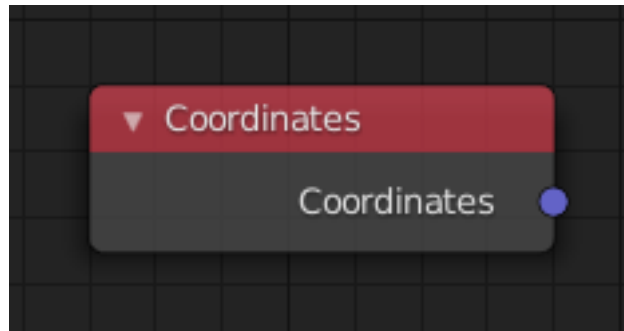


Fig. 382: Coordinates node.

### Inputs

This node has no inputs.

### Properties

This node has no properties.

### Outputs

**Coordinates** The Coordinates node outputs the geometry local coordinates, relative to its bounding box as RGB colors:

- Red channel corresponds to X value.
- Green channel corresponds to Y value.
- Blue channel corresponds to Z value.

## Image Node

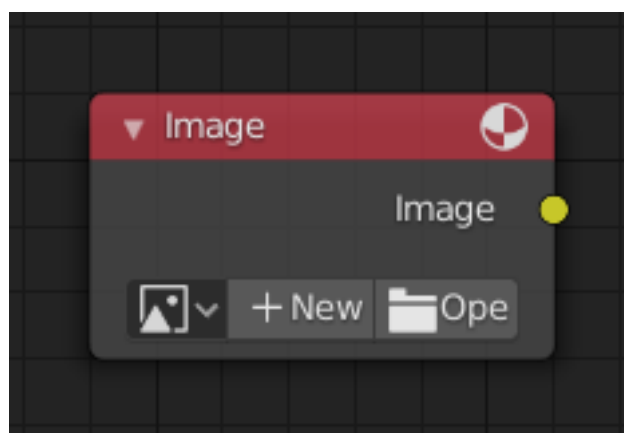


Fig. 383: Image node.

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

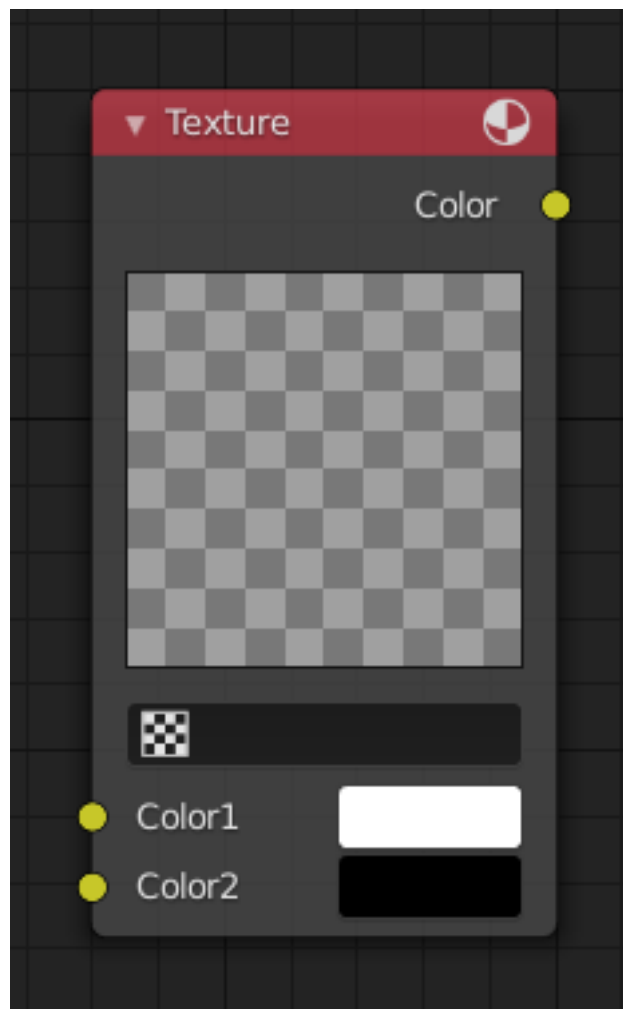


Fig. 384: Texture node.

The Texture node can be used to load 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 could be selected from a list of textures available in the current blend-file or link in textures. The textures themselves could not be edited in this node, but in the Texture panel.

## Outputs

**Color** Standard color output.

## Time Node

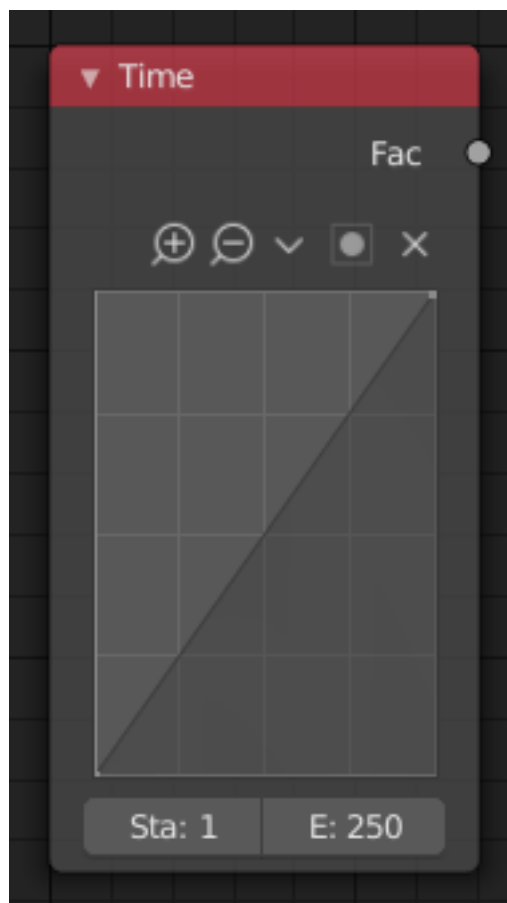


Fig. 385: Time Node.

The *Time node* generates a factor value (from 0.0 to 1.0) that changes according to the curve as time progresses through the *Timeline*.

## 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** A speed of time factor (from 0.0 to 1.0) relative to the frame rate defined in the *Render Dimensions Panel*. The factor changes according to the defined curve.

---

**Hint:** Output values

The *Map Value* node can be used to map the output to a more appropriate value. With sometimes curves, it is possible that the Time 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

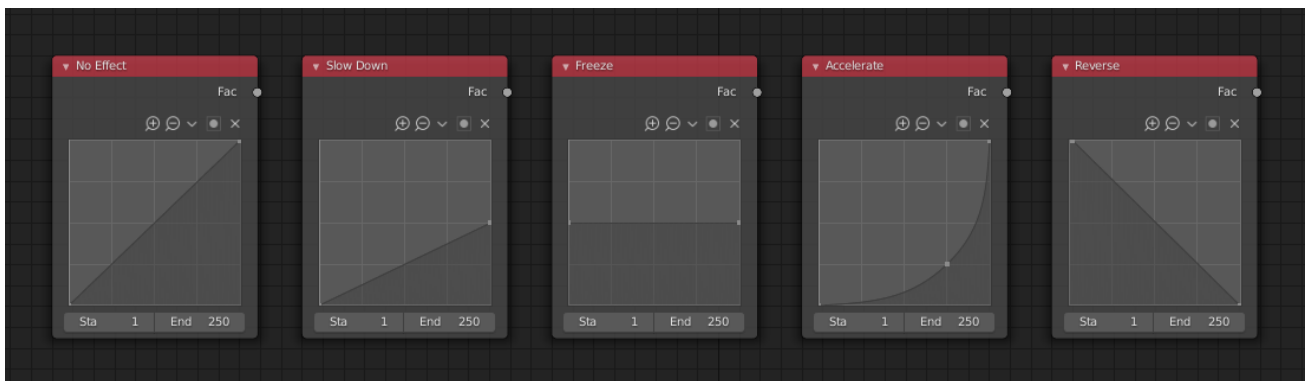


Fig. 386: Time controls from left to right: no effect, slow down, freeze, accelerate, reverse.

## Output Nodes

These nodes serve as outputs for node textures.

## Output Node

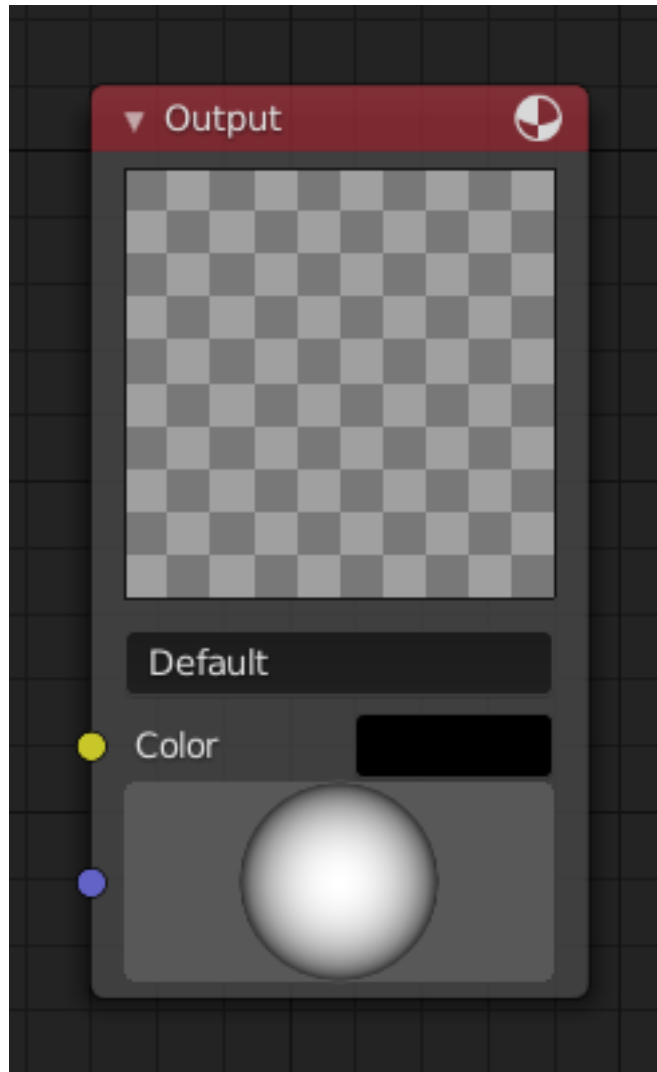


Fig. 387: Output node.

This node contains the result of the node texture.

Multiple output nodes can exist in a node texture, however, only one of them is active. The active one is set in the Texture Panel in the *Output* selector.

### Inputs

**Color** The color data that the texture renders.

**Normal** The normal map that the texture will output.

### Properties

**File Path** Output ID.

## Outputs

This node has no outputs.

## Viewer Node

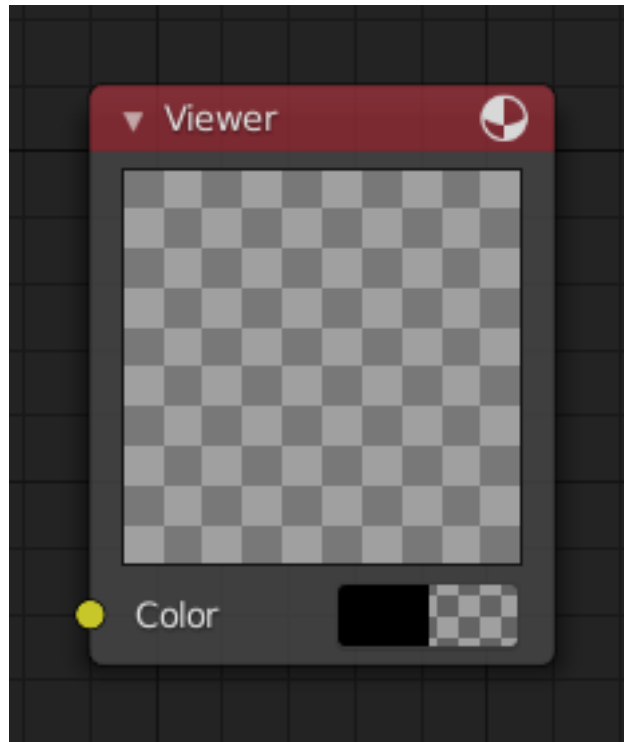


Fig. 388: Viewer node.

The Viewer node can be used to preview the results of a node.

## Inputs

**Color** Standard image input.

## Properties

This node has no properties.

## Outputs

This node has no outputs.



## Pattern Nodes

### Checker Node

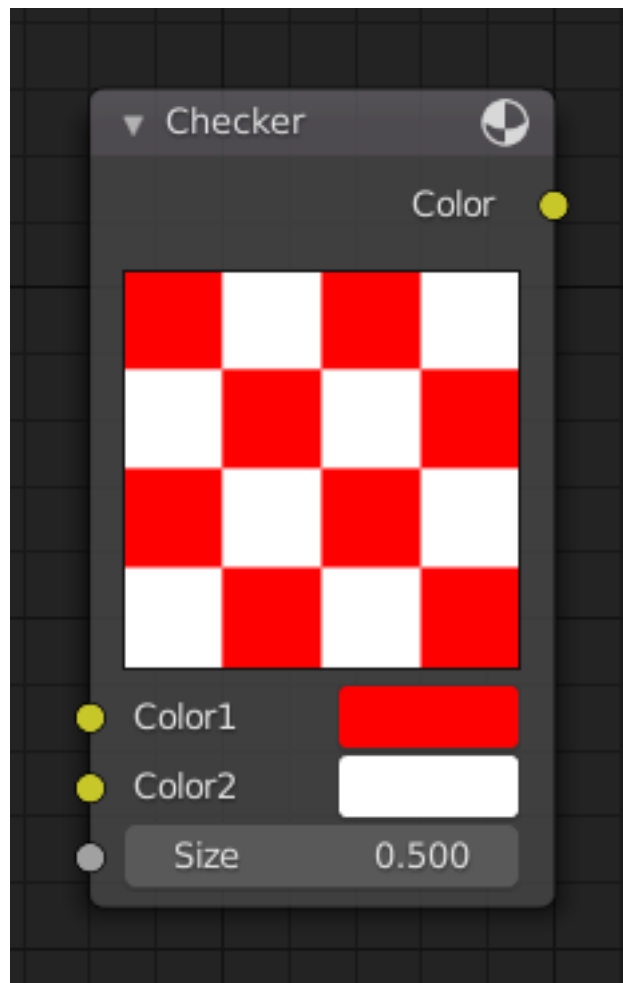


Fig. 389: Checker node.

The Checker node creates a checkerboard pattern.

#### Inputs

**Color 1, Color 2** Image inputs setting the color of the squares.

**Size** The scale of the checker pattern.

#### Properties

This node has no properties.

#### Outputs

**Color** Standard image output.

## Bricks Node

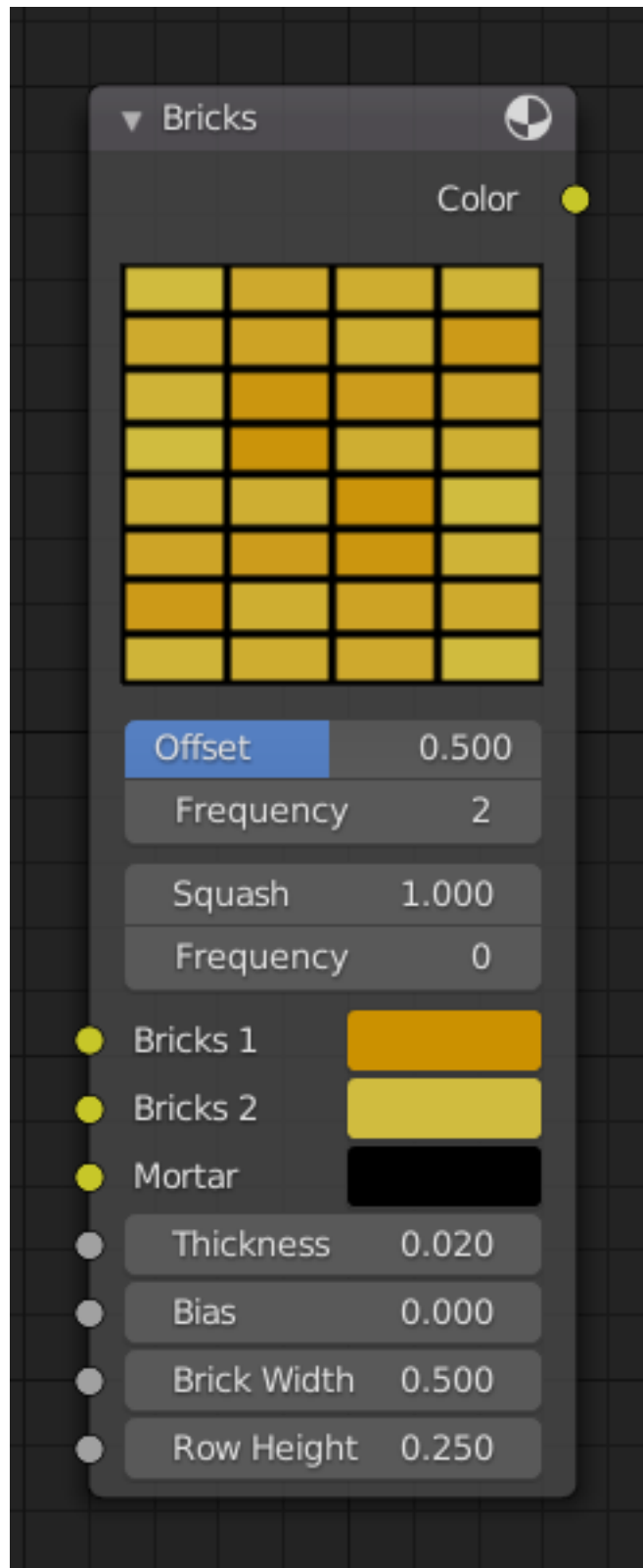


Fig. 390: Bricks node.

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

See *Here*.

### Clouds Node

See *Here*.

### Distorted Noise Node

See *Here*.

### Magic Node

See *Here*.

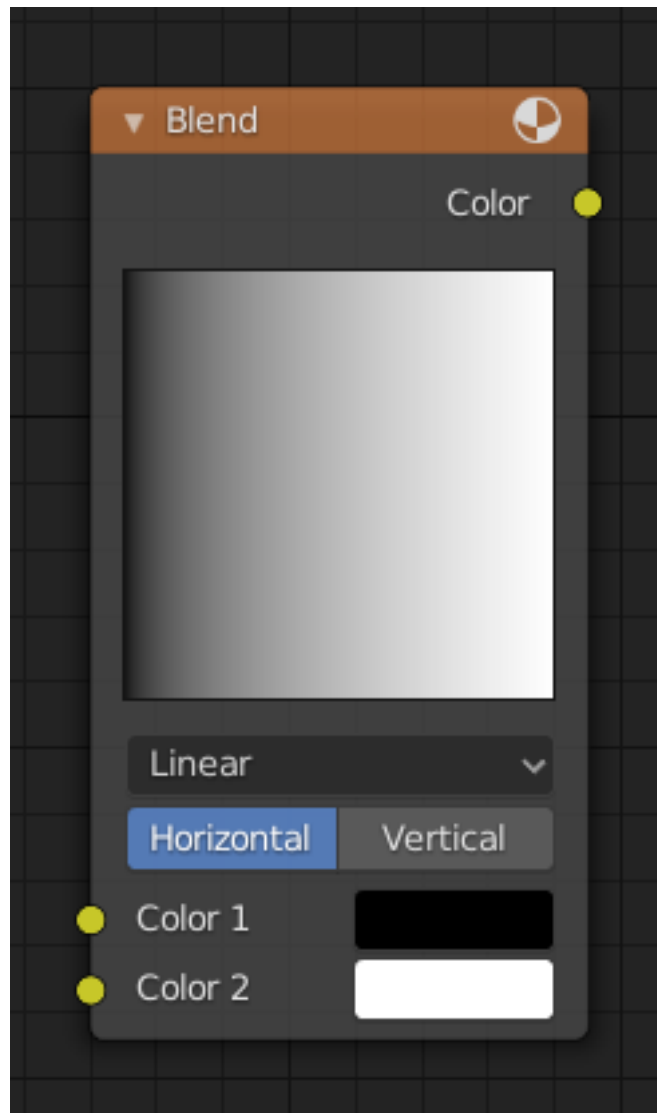


Fig. 391: Blend node.

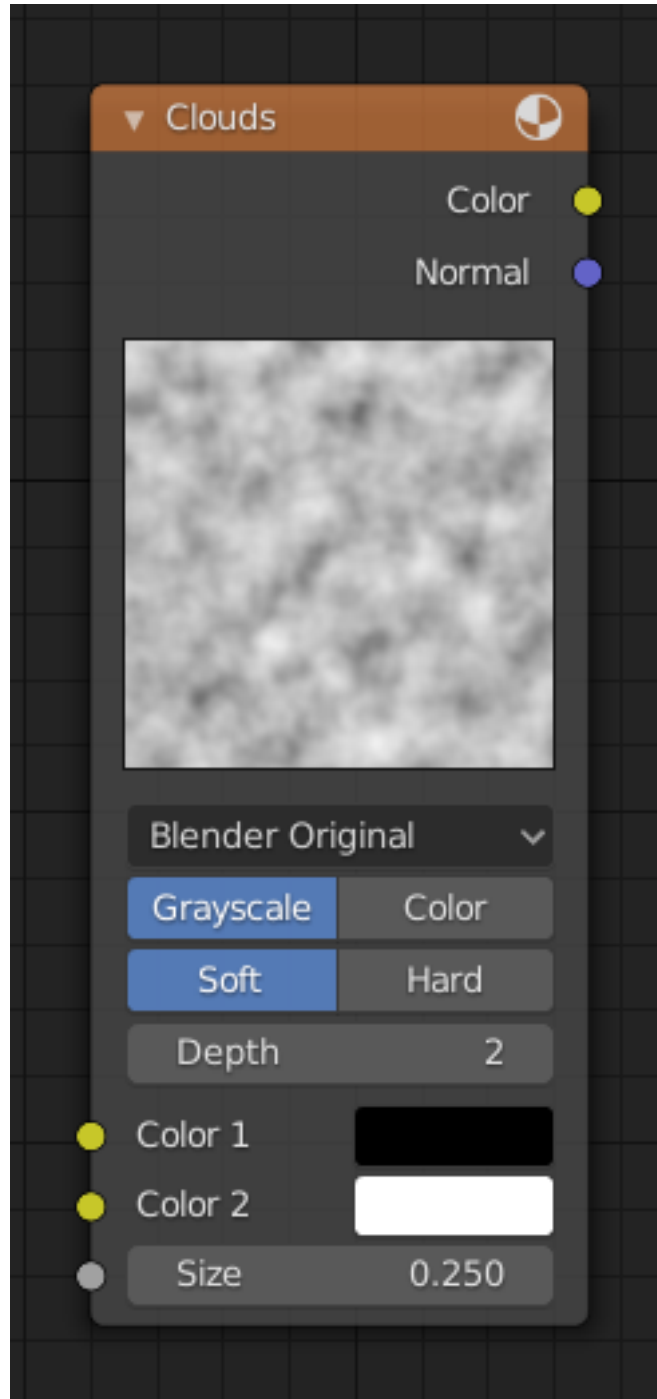


Fig. 392: Clouds node.

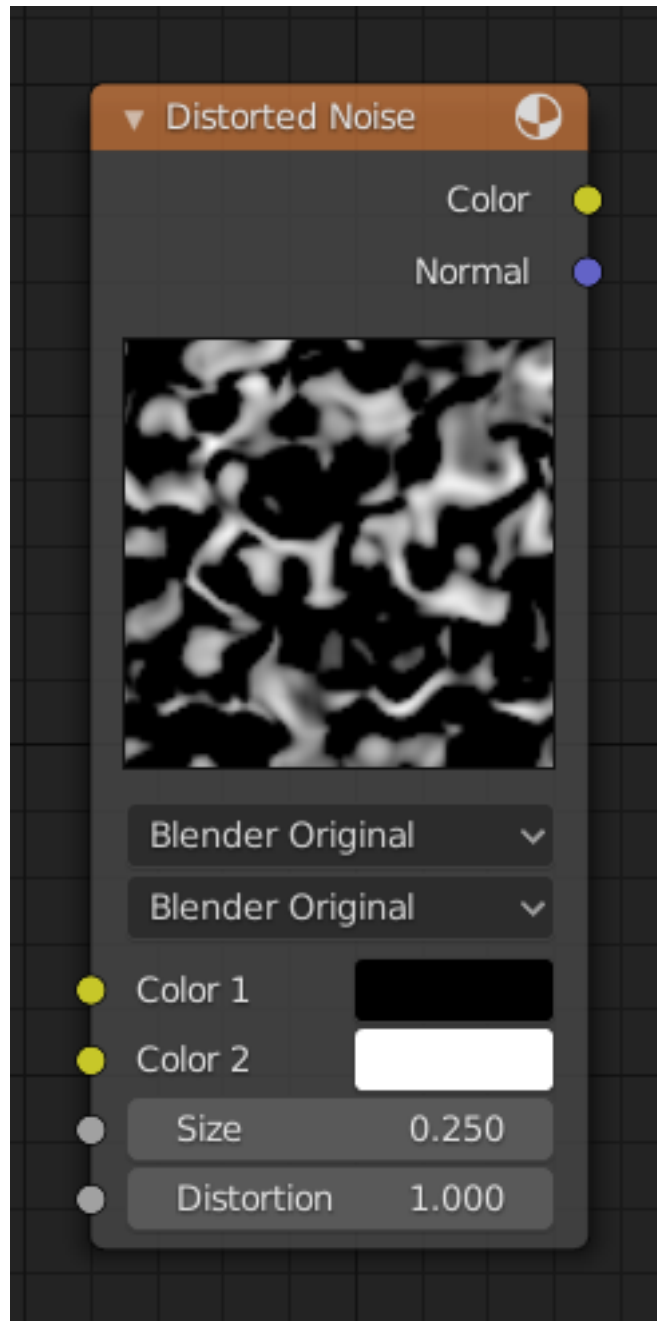


Fig. 393: Distorted Noise node.

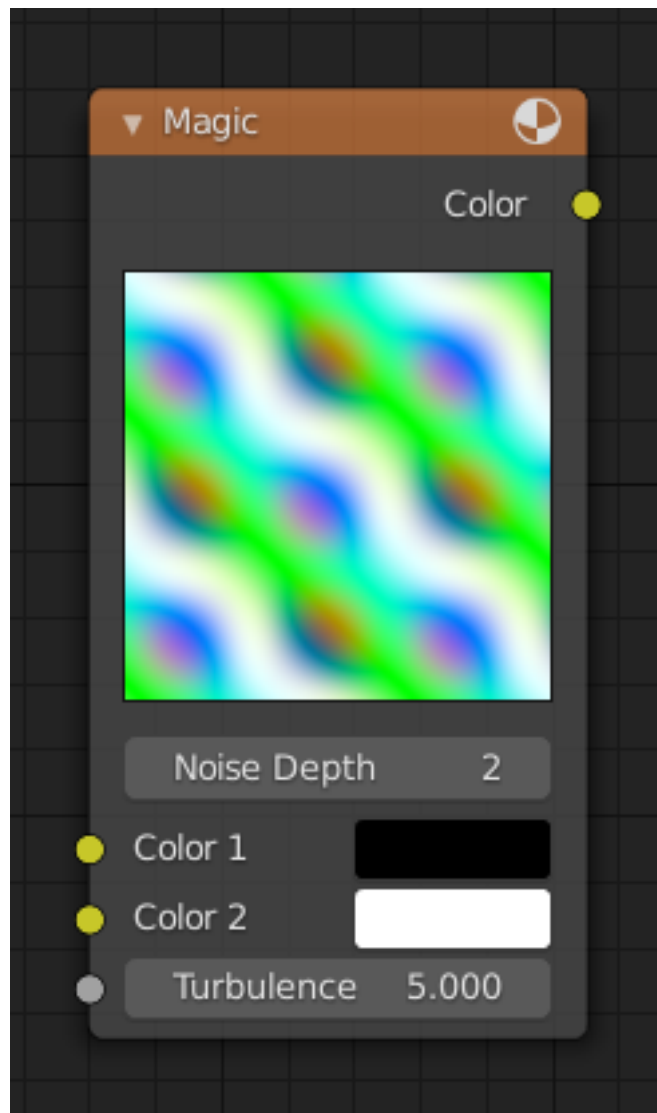


Fig. 394: Magic node.

### Marble Node

See *Here*.

### Musgrave Node

See *Here*.

### Noise Node

See *Here*.

### Stucci Node

See *Here*.

### Voronoi Node

See *Here*.

### Wood Node

See *Here*.

### Node Groups

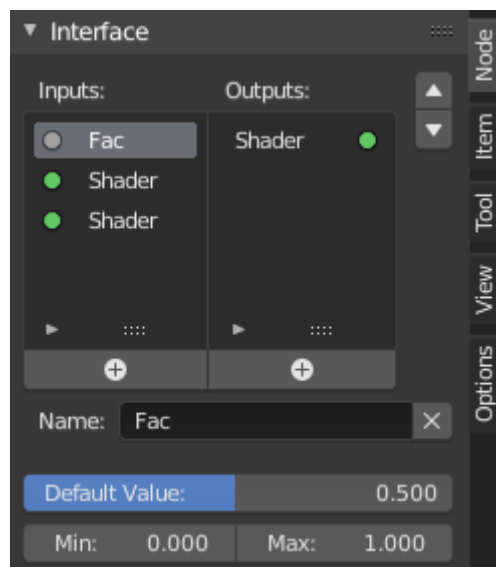


Fig. 401: Example of a node group.

Grouping nodes can simplify a node tree by allowing instancing and hiding parts of the tree. Both material and composite nodes can be grouped.



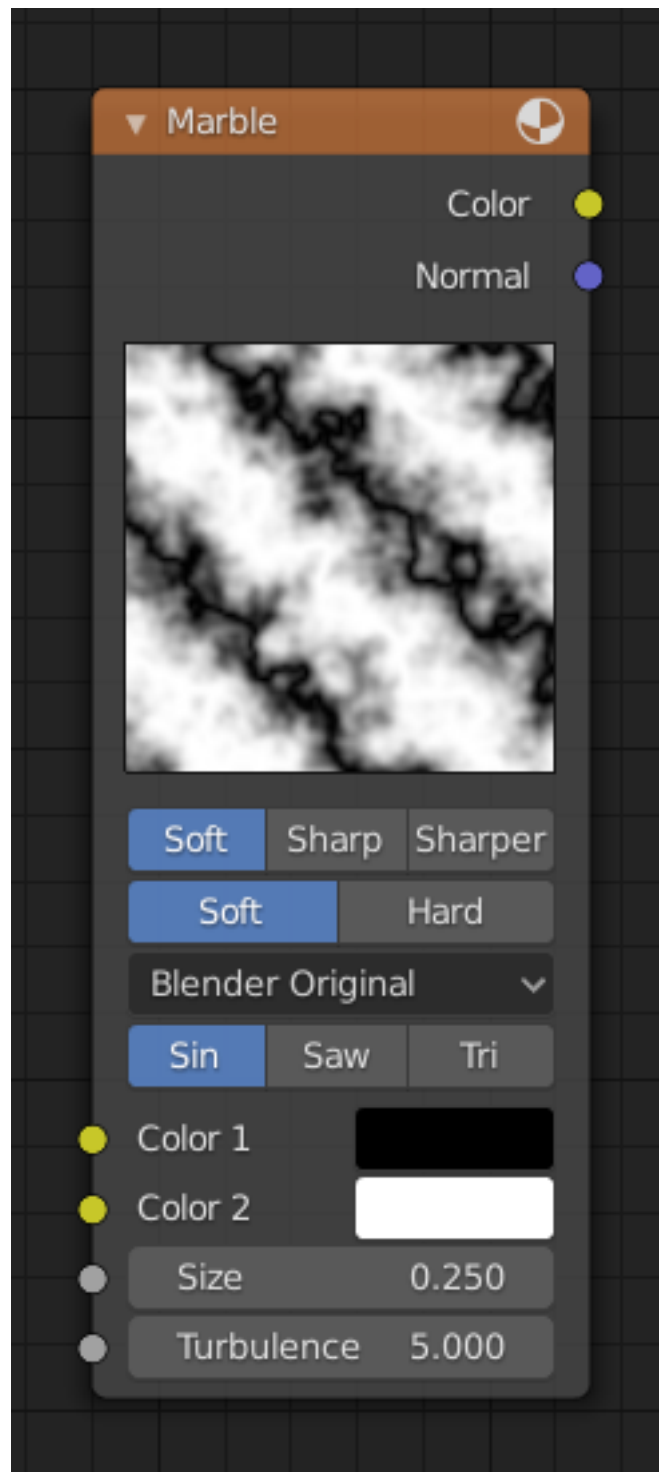


Fig. 395: Marble node.

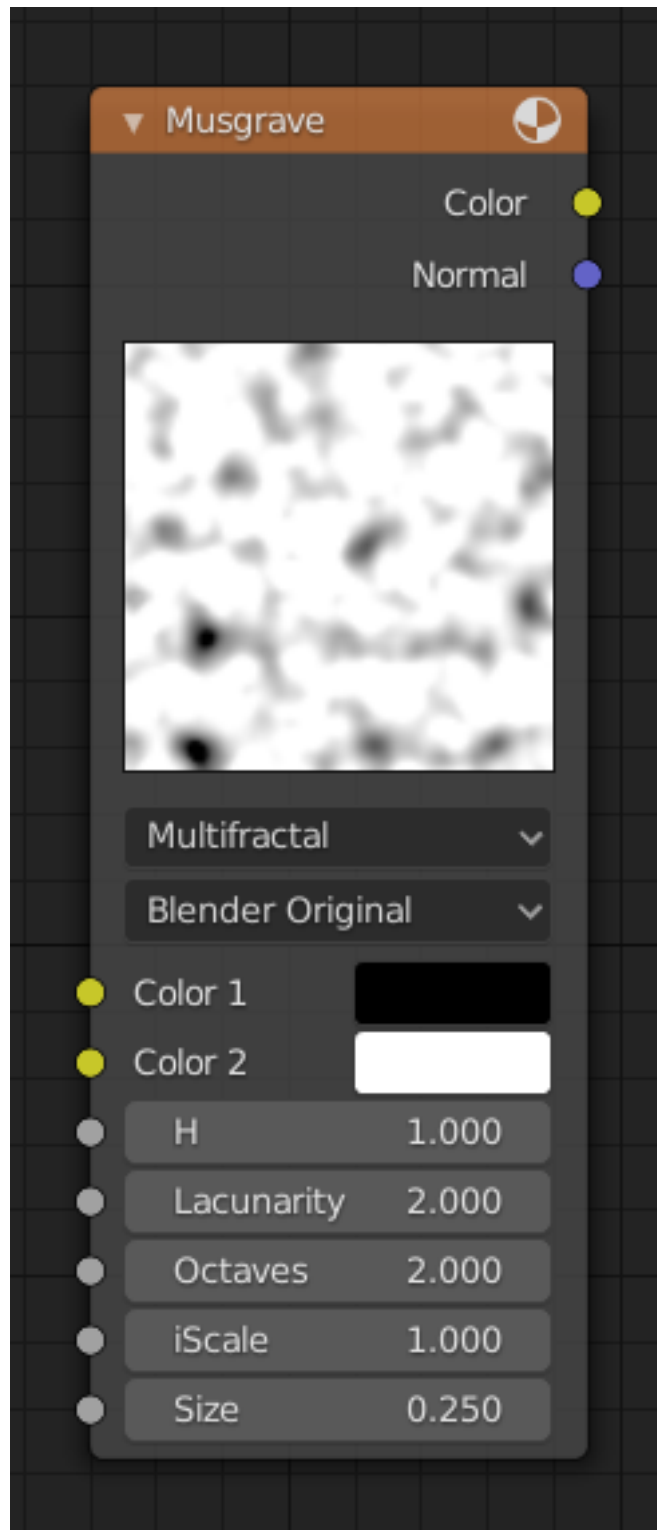


Fig. 396: Musgrave Node.

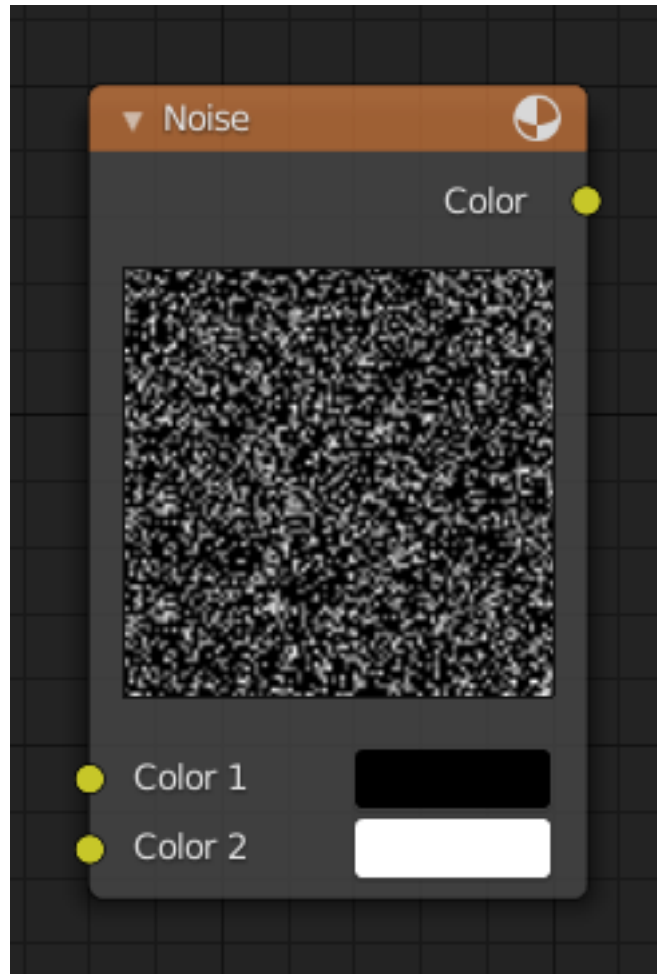


Fig. 397: Noise Node.

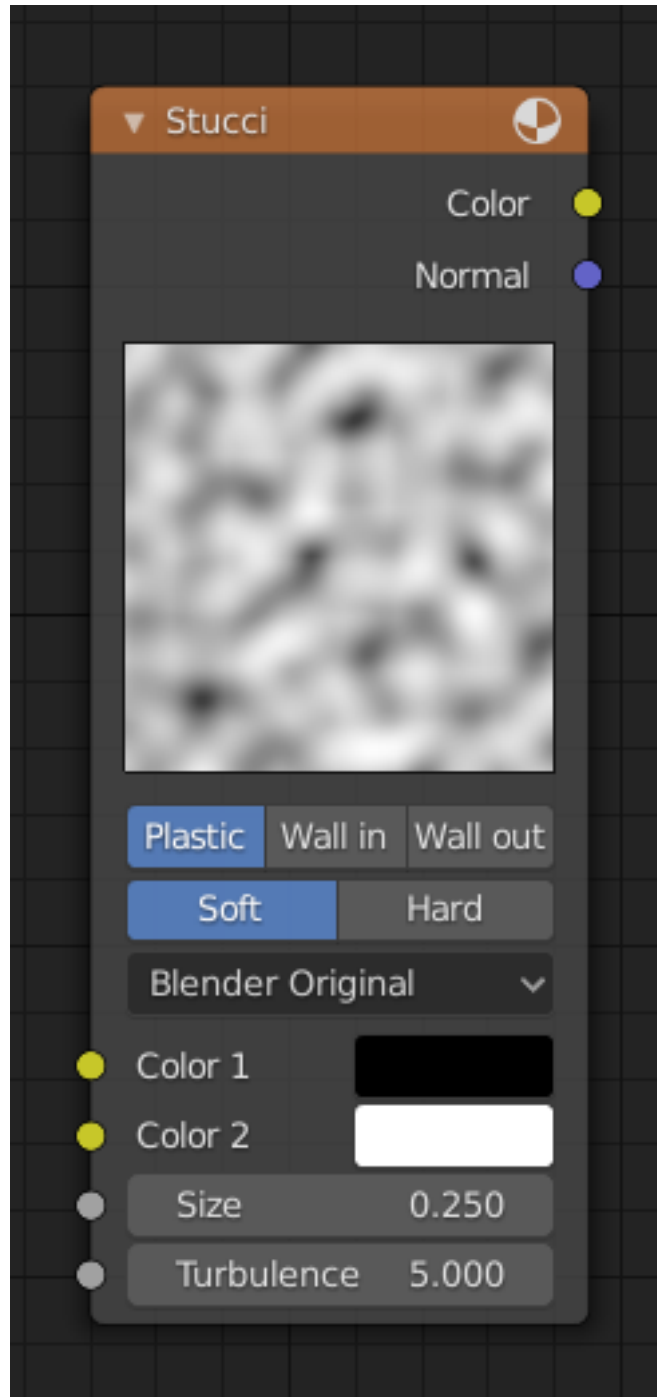


Fig. 398: Stucci Node.

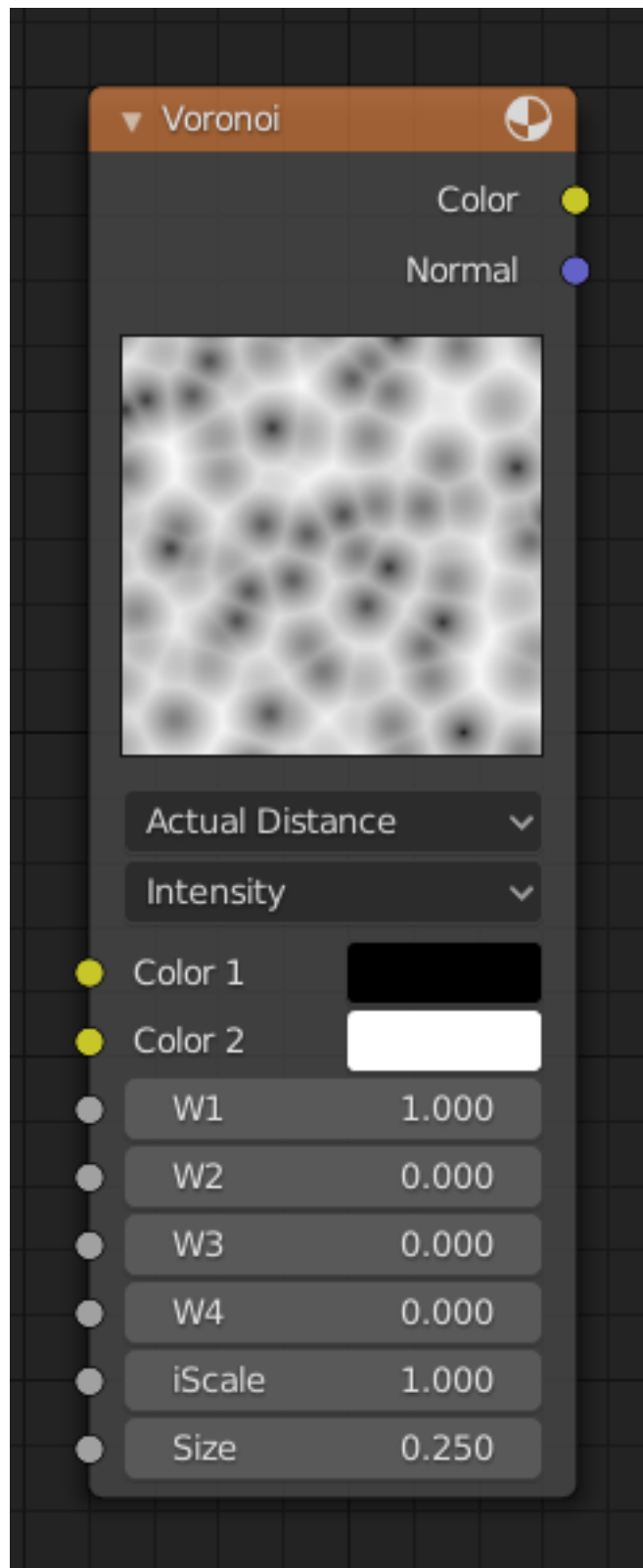


Fig. 399: Voronoi node.

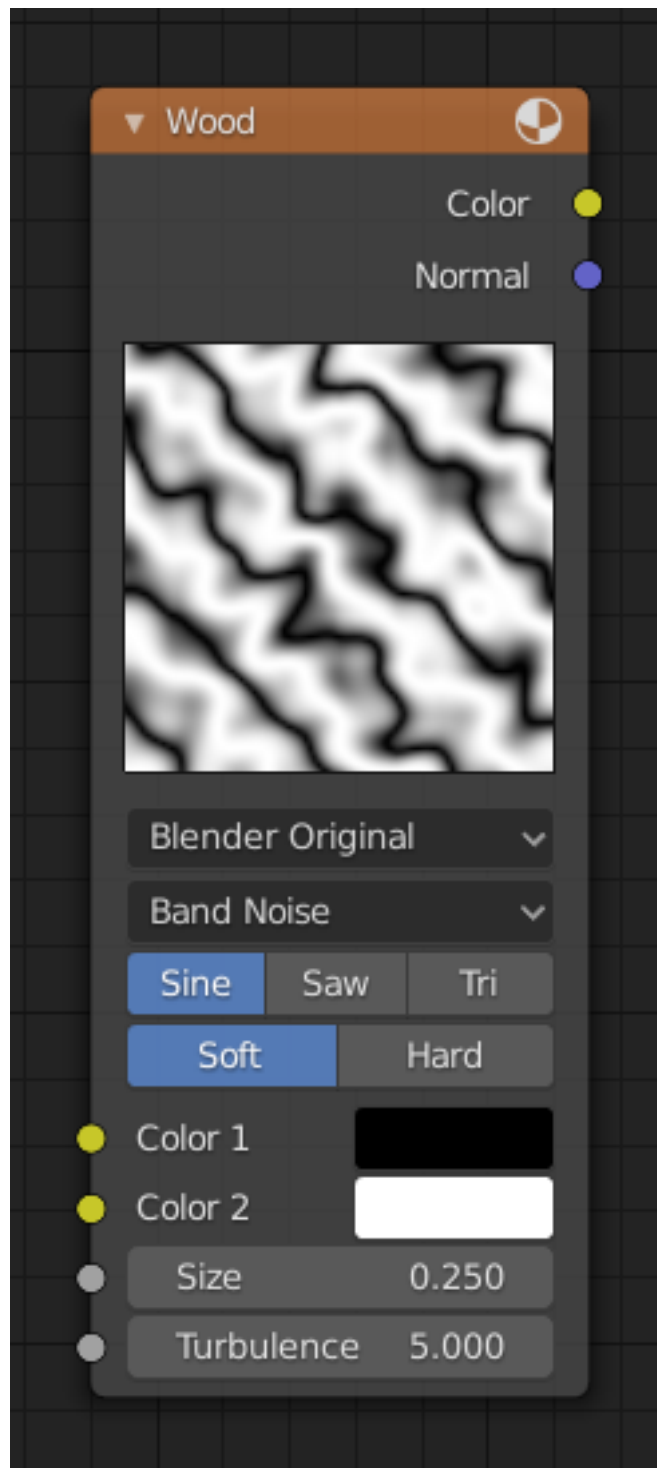


Fig. 400: Wood node.

Conceptually, grouping nodes allows you to specify a *set* of nodes that you can treat as though it were “just one node”. Node groups are similar to functions in programming, they can be reused in many places in a node tree and can be customized by changing the “parameters” of the node group.

As an example: If you have created a material that you would like to use with different inputs e.g. diffuse color: red plastic, green plastic. You could create different materials with *Make Single User* for each different color with a copy of the tree part describing the plastic material. If you like to edit the material you would need to redo the edit on all materials. A better method of reuse is to create node groups, exposing only the variable inputs (e.g. diffuse color).

Also nested node groups are supported. I.e. a node group can be inserted or created inside another 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).

---

## Make Group

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Make Group*

**Hotkey** Ctrl-G

---

To create a node group, select the nodes you want to include, then press Ctrl-G, *Group* → *Make Group*. A node group will have a green title bar. All of the 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 easily distinguish between the two types.

---

**Tip:** What **not** to include in node groups:

Remember that the essential idea is that a group should be an easily-reusable, self-contained software component. Material node groups should **not** include:

**Input nodes** If you include a source node in your group, you will end up having the source node appearing *twice*: once inside the group, and once outside the group in the new material node tree.

**Output node** If you include an output node in the group, there will not be an output socket available *from* the group!

---

## Edit Group

---

### Reference

**Mode** All Modes

**Menu** *Node* → *Edit Group*

**Header** *Go to Parent Node Tree*

---

## Hotkey Tab, Ctrl-Tab

With a node group selected, Tab expands the node to a frame, and the individual nodes within it are shown. You can move them around, play with their individual controls, re-thread them internally, etc. just like you can if they were a normal part of the editor view. You will not be able, though, to thread them to a node outside the group; you have to use the external sockets on the side of the node group. While Tab can be used to both enter and exit a group, Ctrl-Tab only exits.

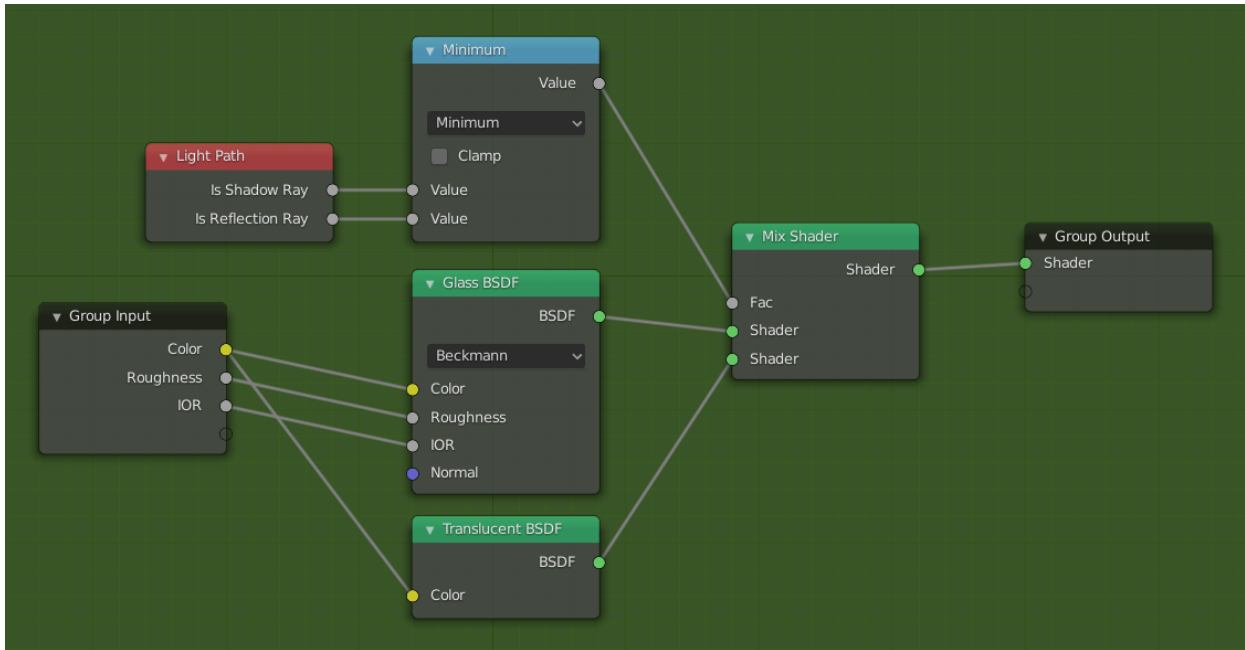


Fig. 402: Example of an expanded node group.

## Interface

### Interactively

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. When created, 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 during node group development an additional parameter needs to be passed into the group, an additional socket must be added to the *Group Input* node. This is easily done by adding a connection from the hollow socket on the right side of the *Group Input* node to the desired input socket on the node requiring input. The process is similar for the *Group Output* regarding data you want to be made available outside the group.

## Panel

### Reference

**Mode** All Modes

**Panel** *Sidebar region* → *Node* → *Interface*



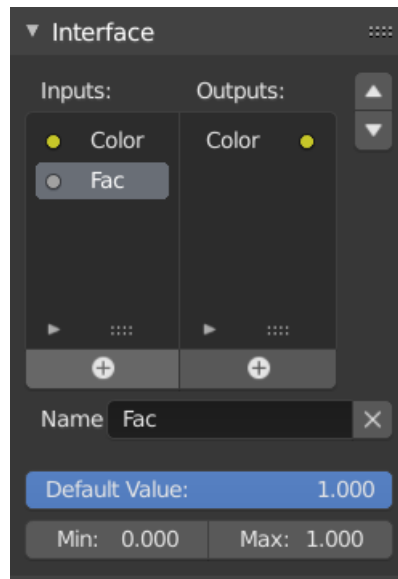


Fig. 403: The interface panel for editing groups.

Sockets can be added, re-ordered, or removed, descriptive names can be added and the details of the input data value defined here.

If you have multiple inputs or outputs, they can be re-ordered by selecting the socket in the list and then moving it up or down with the arrow buttons on the right side of the panel. The larger plus sign buttons below the list will add an unconnected socket of the same type as the selected socket or a value socket if there is no selection. The triangle at the bottom of the list has filtering functions to facilitate finding nodes if the group has a large number of sockets.

## Ungroup

### Reference

**Mode** All Modes

**Menu** *Group* → *Ungroup*

**Hotkey** Ctrl-Alt-G

The Ctrl-Alt-G tool removes the group and places the individual nodes into your editor workspace. No internal connections are lost, and now you can thread 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.

## Group Insert

### Reference

**Mode** All Modes

---

**Menu** *Node → Group Insert*

---

Selecting a set of nodes, ending with the destination group node, and pressing *Node → Group Insert* 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.

---

**Appending Node Groups**

---

**Reference****Editor** Topbar**Mode** All Modes**Menu** *File → Link/Append*

---

Once you have appended a Node Tree to your blend-file, you can make use of it in a node editor by pressing *Shift-A, Add → Group*, then selecting the appended group. The “control panel” of the Group include the individual controls for the grouped nodes. You can change them by working with the Group node like any other node.

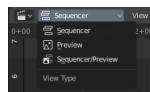
## 2.3.7 Video Editing

### Introduction

In addition to modeling and animation, Blender can be used to edit video. There are two possible methods for this, one being the *Compositor*. However, this chapter describes the other, the Video Sequence Editor (VSE), sometimes shortened to just “Sequencer”. The 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 VSE, 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.

### View Types



The Video Sequence Editor has three view types for the main view:

**Sequencer** View timeline and strip properties.**Preview** View preview window and preview properties.**Sequencer/Preview** Combined view of preview and timeline and properties of both.

It is possible to create multiple instances of any view type in single workspace.

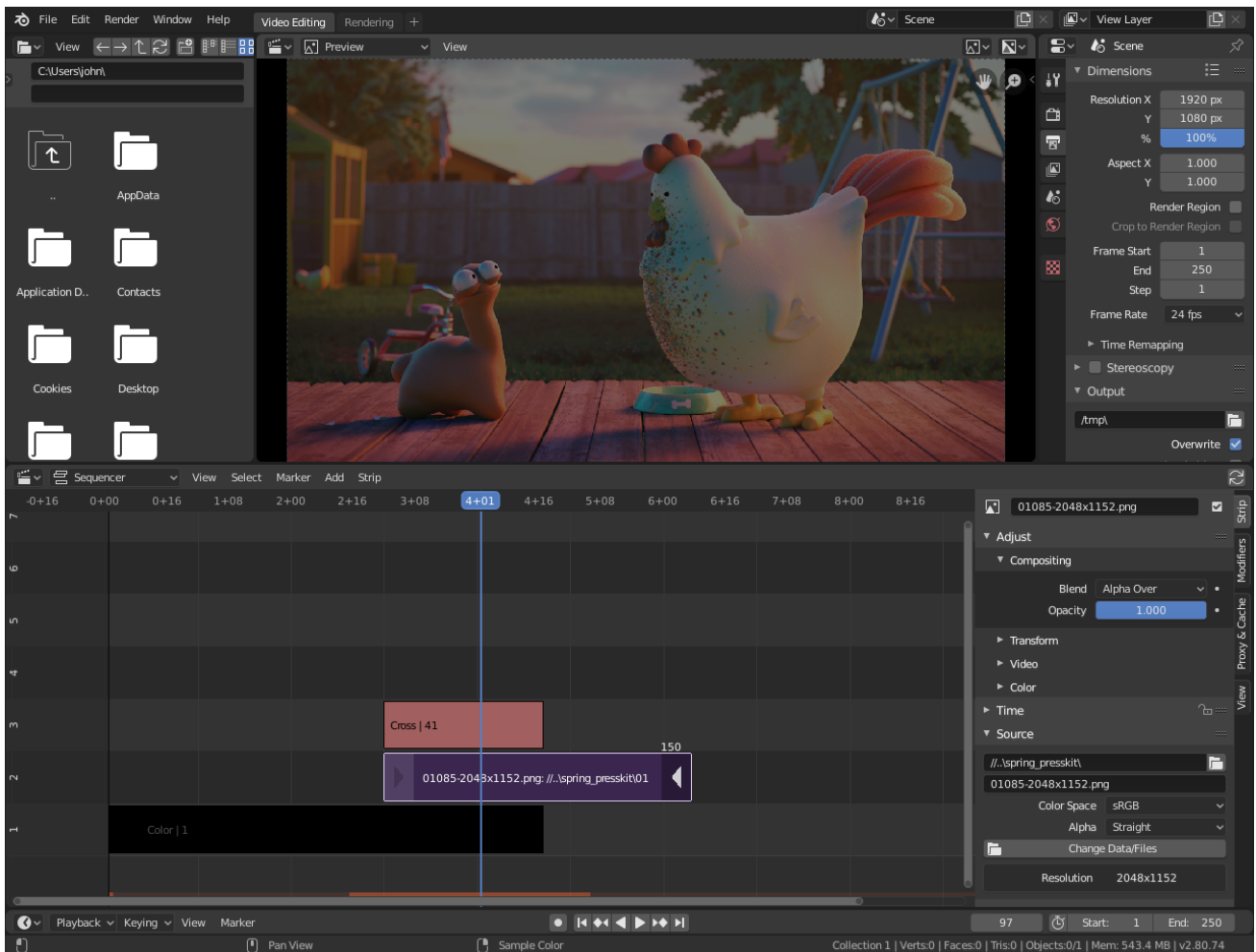


Fig. 404: Default Video Editing screen layout.

**Note:** By default the Sequencer is enabled, however, it can be disabled in the *Post Processing Panel*.

---

## Performance

Playback performance can be improved through several ways. The biggest impact on performance is to allow the Video Sequencer to cache the playback. There are two levels of cache, the first is a RAM cache, this is enabled by default but can be increased based on the amount of RAM available. The next level of cache is a disk cache which stores cached strips on disk. A disk cache can generally cache more than a RAM cache, but it can be slower. Both of these cache options can be configured in the *Preferences*.

Another way to improve performance is by using *Strip Proxies* These are used to cache images or movies in a file that is easier to playback by reducing the image quality by either decreasing the resolution and/or compressing the image.

## Sequencer

### Introduction

The Sequencer region is horizontally divided into channels, each channel can contain what is called a strip. A strip can be an image, animation, or any number of effects. Each channel is numbered consecutively on the Y axis, starting from zero and allows up to 32 total channels. The X axis represents time. Each channel can contain as many strips as it needs as long as they do not overlap. If a strip needs to overlap another, it needs to be placed on a channel above or below the other strip. When strips are stacked, they stack from bottom to top where the lowest channel forms the background and the highest the foreground.

---

**Note:** The first channel 0 is unusable as a place to put strips. This is because it is used by the *Sequencer Display* to show a composite of all strips above channel 0.

---

This region is where strips can be *selected*, *modified* by moving, cutting, or extending strips. There are also several built-in *effects* that can be combined with other strips to change their appearance.

## Toolbar

### Introduction

**Select** Select or move.

**Select Box** Select strip by dragging a box.

**Blade** Create a cut along the strip.

## Blade

---

### Reference

**Mode** Sequencer Mode

**Tool** *Toolbar* → *Blade*

## Hotkey Shift-K

**Soft** This cuts the strip in two at the location of the click. This will result in two strips which use the same source, fitting the original strip's timing and length.

**Hard** Like *Soft Blade*, it cuts a strip in two distinct strips; but you will not be able to drag the endpoints to show the frames past the cut of each resulting strip.

## Navigating

### Header

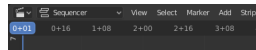


Fig. 405: Video Sequencer Header.

### View Menu

As usual, the View Menu controls the editor's view settings.

**Sidebar N** Show or hide the *Sidebar*.

**Toolbar T** Show or hide the *Toolbar*.

**Adjust Last Operation** Displays a pop-up panel to alter properties of the last completed operation. See *Adjust Last Operation*.

**Preview as Backdrop** Displays the current frame in the background of the main view like in the *Compositor*.

**Frame Selected NumpadPeriod** Zooms in the display to fit only the selected strips.

**Frame All Home** Zooms the display to show all strips.

**Zoom Shift-B** Click and drag to draw a rectangle and zoom to this rectangle.

### Navigation

**Play Animation Spacebar** Start or stop playback of animation. 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** Current frame will jump to beginning of strip.

**Jump to Next Strip PageUp** Current frame will jump to end of strip.

**Jump to Previous Strip (Center) Alt-PageDown** Jump to previous center of the strip.

**Jump to Next Strip (Center) Alt-PageUp** Jump to next center of the strip.

### Range

**Set Preview Range P** Interactively define frame range used for playback. Allows you to define a temporary preview range to use for animation playback (this is the same thing as the *Playback Range* option of the *Timeline editor header*).

**Set Preview Range to Strips** Sets the frame range of preview to the range of the selected strips.

**Clear Preview Range Alt-P** Clears preview range.

**Set Start Frame Ctrl-Home** Set Start of animation range to current playhead position.

**Set End Frame Ctrl-End** Set End of animation range to current playhead position.

**Set Frame Range to Strips** Sets the frame range of preview and render animation to the frame range of the selected strips.

**Refresh All** To force Blender to re-read in files, and to force a re-render of the 3D Viewport, click the *Refresh Sequencer* button. Blender will update and synchronize all cached images and compute the current frame.

Certain operations, like moving an object in the 3D Viewport, may not force the *Sequencer* to call for a refresh of the rendered image (since the movement may not affect the rendered image). If an image or video, used as a strip, is changed by some application outside of Blender, Blender has no real way of being notified from your operating system.

**Sync Visible Range** Synchronize the visible range with other time based editors.

**Show Seconds Ctrl-T** Shows seconds instead of frames on the time axis.

**Show Offsets** Shows overflow bars of “extra” content from either cutting or sliding strips.

**Show F-Curves** Show animation curves for opacity and volume values as darkened sections of the strip.

**Show Markers** Shows the markers region. When disabled, the *Markers Menu* is also hidden and markers operators are not available in this editor.

**Show Cache** Show *Cache* Show all enabled types; Final Images, Raw Images, Preprocessed Images, Composite Images

**Show Waveforms** Global option to either display the waveform, or the strip info, or use the individual *strip option*.

**Sequence Render Image** Render an image of the current frame.

**Sequence Render Animation** Render timeline from Preview Start to Preview End Frame to a Video file or series of images.

**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 display type between Sequencer and Preview.

## Markers 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.

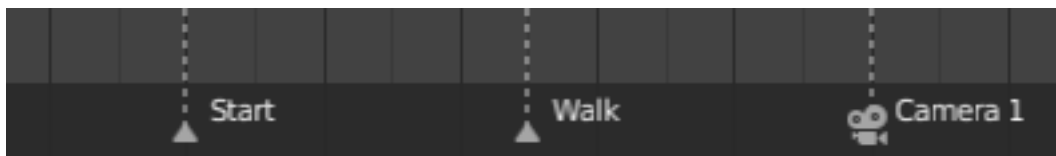


Fig. 406: Markers in animation editor.

For descriptions of the different marker tools see *Editing Markers*.

## Main View

### Adjusting the View

Use these shortcuts to adjust the sequence area of the VSE:

- Pan: MMB
- Zoom: Wheel

- Vertical Scroll: use Shift-Wheel, or drag on the left scrollbar.
- Horizontal Scroll: use Ctrl-Wheel, or drag on the lower scrollbar.
- Scale View: Ctrl-MMB and drag up/down (vertical scale) or left/right (horizontal scale).
- Scale View Vertically: drag on the circles on the vertical scrollbar.
- Scale View Horizontally: drag on the circles on the horizontal scrollbar.

## Playhead

The playhead is the blue vertical line with the current frame number at the top. It can be set or moved to a new position by pressing or holding LMB in scrubbing area at the top of the timeline. You can move the playhead in increments by pressing Left or Right, or by using Alt-Wheel. You can also jump to the beginning or end frame by pressing Shift-Left or Shift-Right. As you do, the image for that frame is displayed in the Preview region.

When you drag the frame indicator with Shift-RMB directly on a sequence strip, this will show the strip *solo*, (temporarily disregarding effects and other strips, showing only this strip's output) and the strip will be highlighted.

When holding Ctrl while dragging it will snap to the start and endpoints of strips.

Real-time preview is possible on reasonable computers when viewing an image sequence or movie (avi/mov) file. Scene strips can use viewport previews or proxies for real-time playback, otherwise displaying rendered frame is supported, but typically too slow for real-time playback.

---

**Hint:** Every other synced editor can be used for scrubbing e.g. the Timeline.

---

## Strips

### 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.

Fig. 407: Strip schematic.

### Adding Strips

---

#### Reference

**Menu** Add

**Hotkey** Shift-A

---

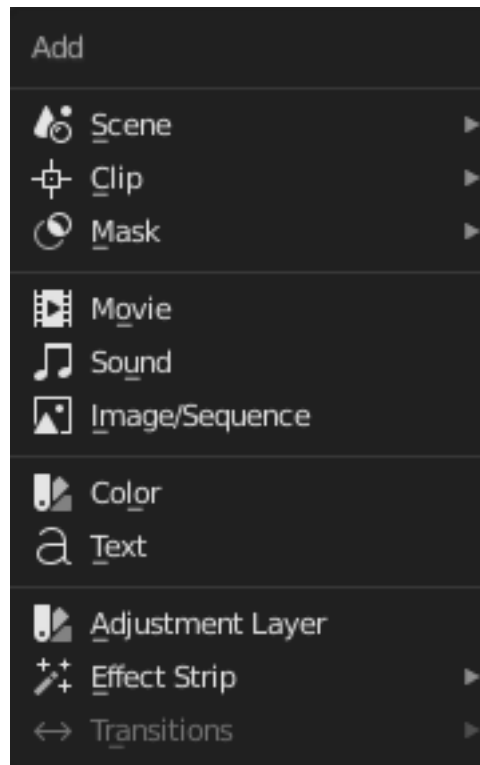


Fig. 408: The Add Menu.

The Add menu is the main menu you will be using to add content to the VSE. 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 VSE editor 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.

### 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 VSE:

- Scene strip: Light green.
- Clip strip: Dark blue.
- Mask strip: Red.
- Movie strip: Aquamarine.
- Image strip: Purple.
- Sound strip: Turquoise.

Each of the effect strips has its own color.

## Types

### Scene Strips

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.

---

## Options

**Use Sequence** Expand the scenes sequence strips, 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.

**Volume** Volume of the audio taken from the chosen scene.

**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 Grease Pencil** Shows *Grease Pencil* in non render preview i.e. *Solid* mode.

**Transparent** Creates a transparent background. This is useful for doing overlays like rendering out Grease Pencil films via the Sequencer.

### Clip Strips

Clip can be modified within the *Movie Clip Editor*.

## Options

This strip has no options.

## Mask Strips

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 Strips

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.

---

## Example

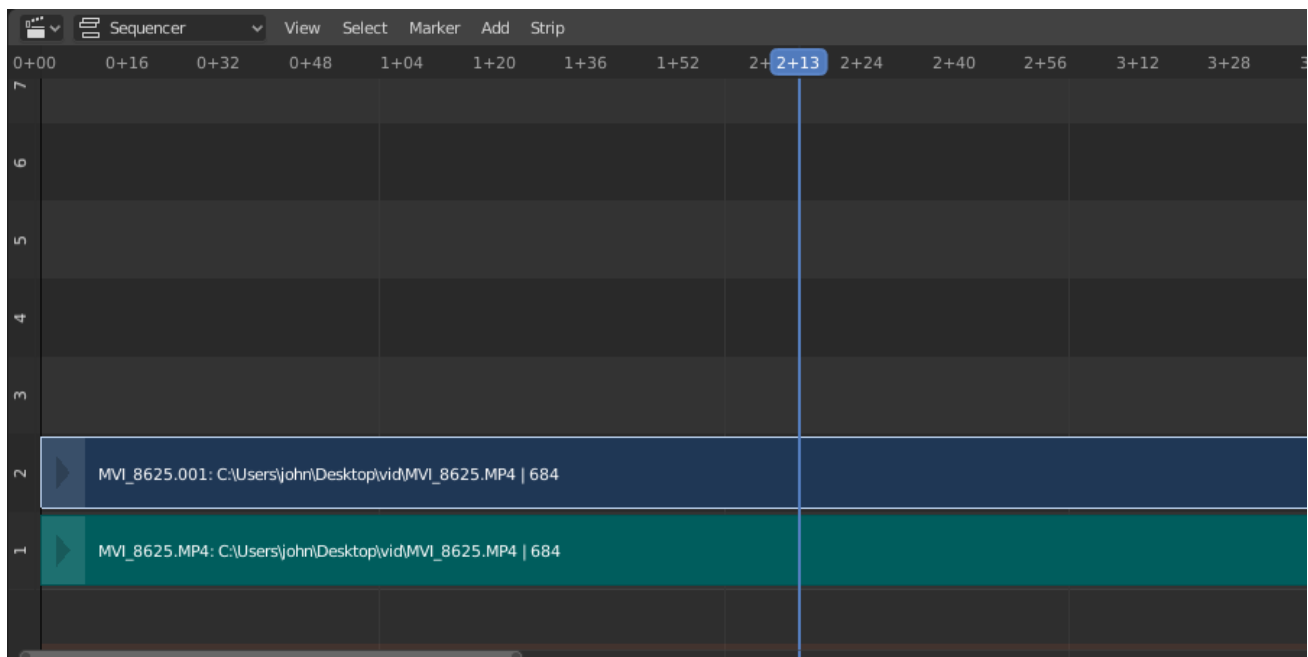


Fig. 409: Imported movie strip with audio track underneath.

In the strip itself, you can see strip name, path to source file, and strip length.

## Image/Sequence Strips

### 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 VSE 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

### Placeholder Images

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.

### Sound Strips

As well as images and movies the VSE 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.

### Working with Audio Tracks

A sound strip is just like any other strip in the VSE. 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



## Options

**Color** Click on the color field in the Effect panel in the Sidebar region, to pick a different color.

## Text Strips

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.

**Size** Size of the text.

**Color** The text color.

**Shadow** Creates a shadow of the specified color under the text.

## 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

### Adjustment Layer Strips

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.

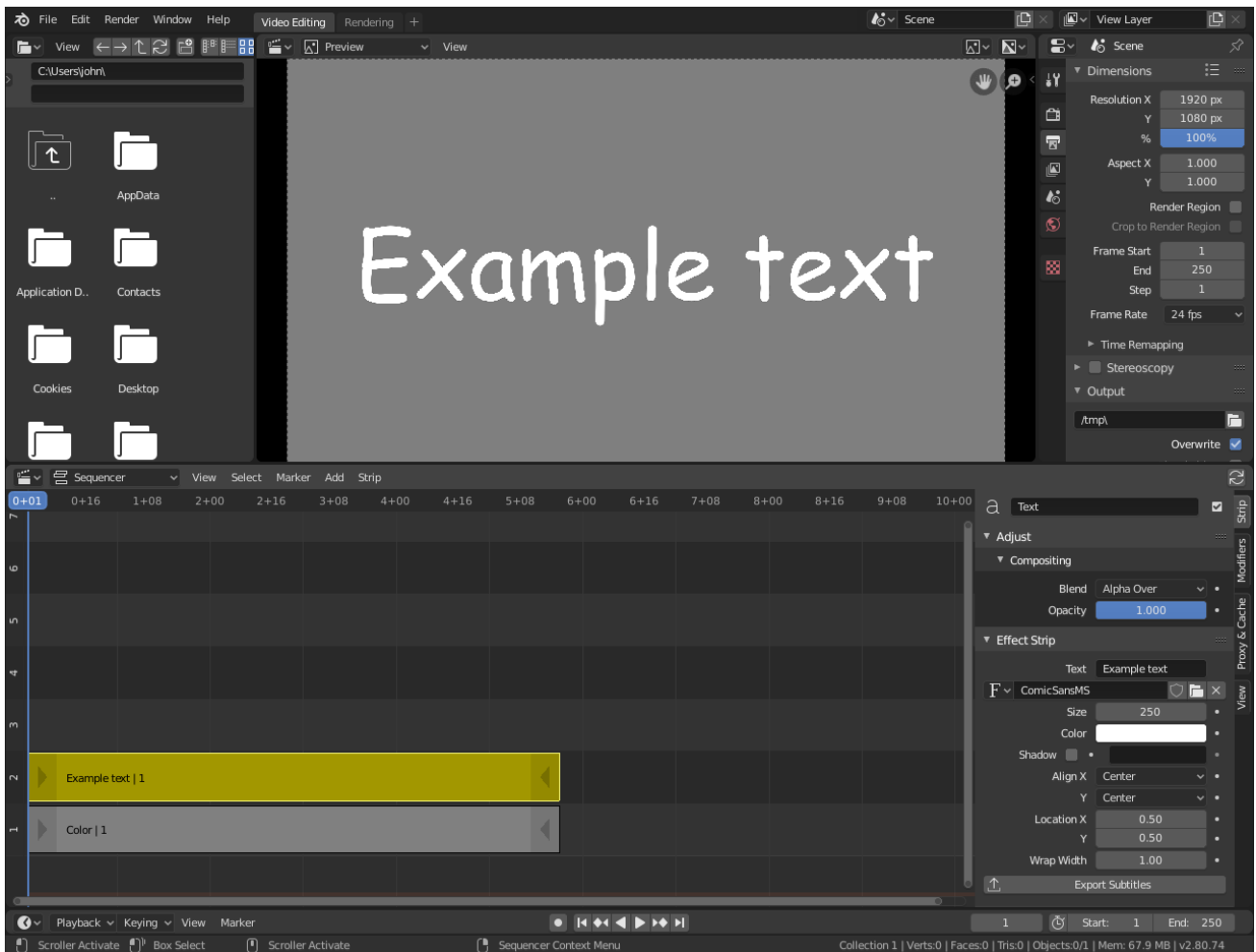


Fig. 411: Text Effect.

## Effect Strips

### Add Effect

The Add Effect 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

#### Subtract Effect

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

#### Multiply

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).

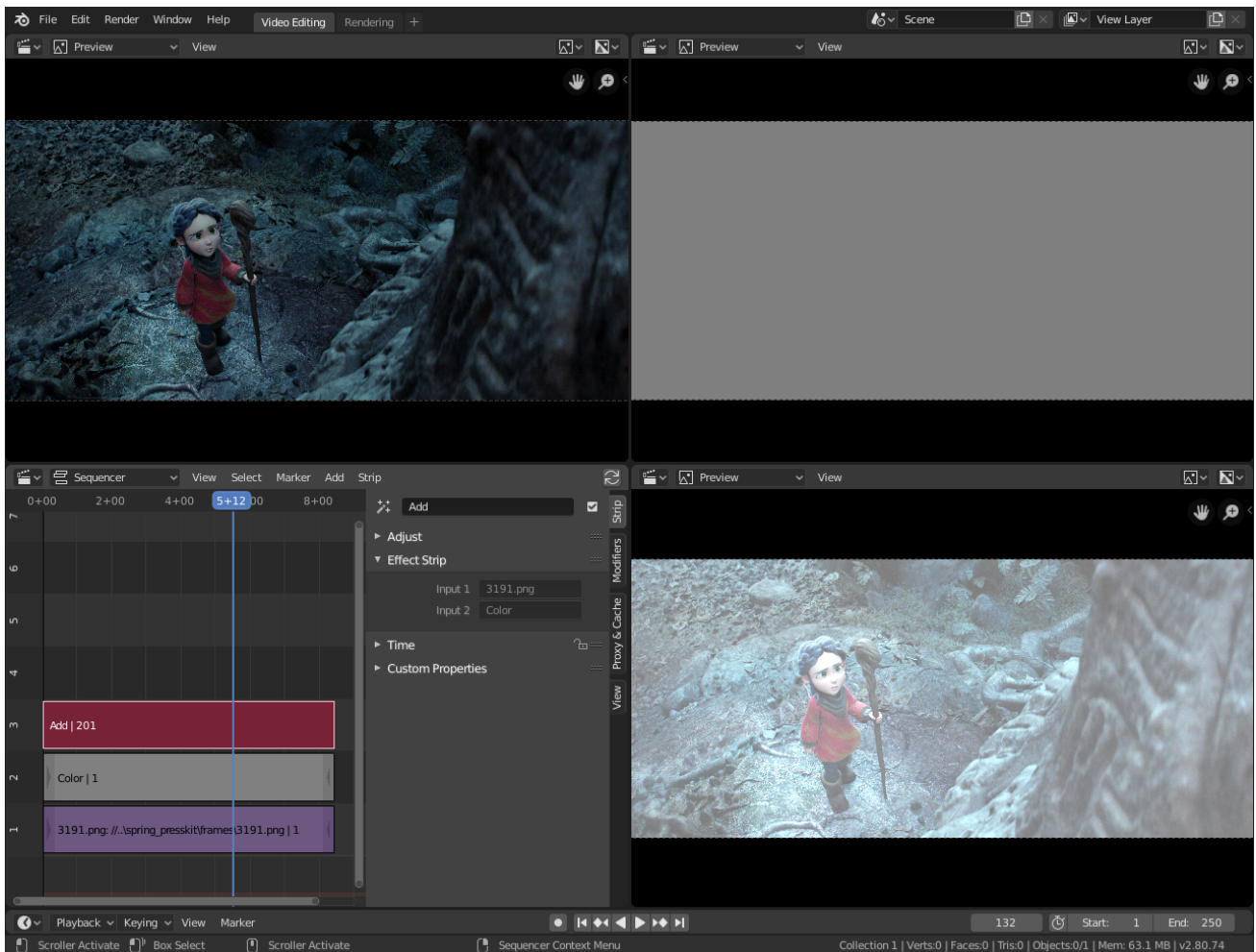


Fig. 412: Add Effect.



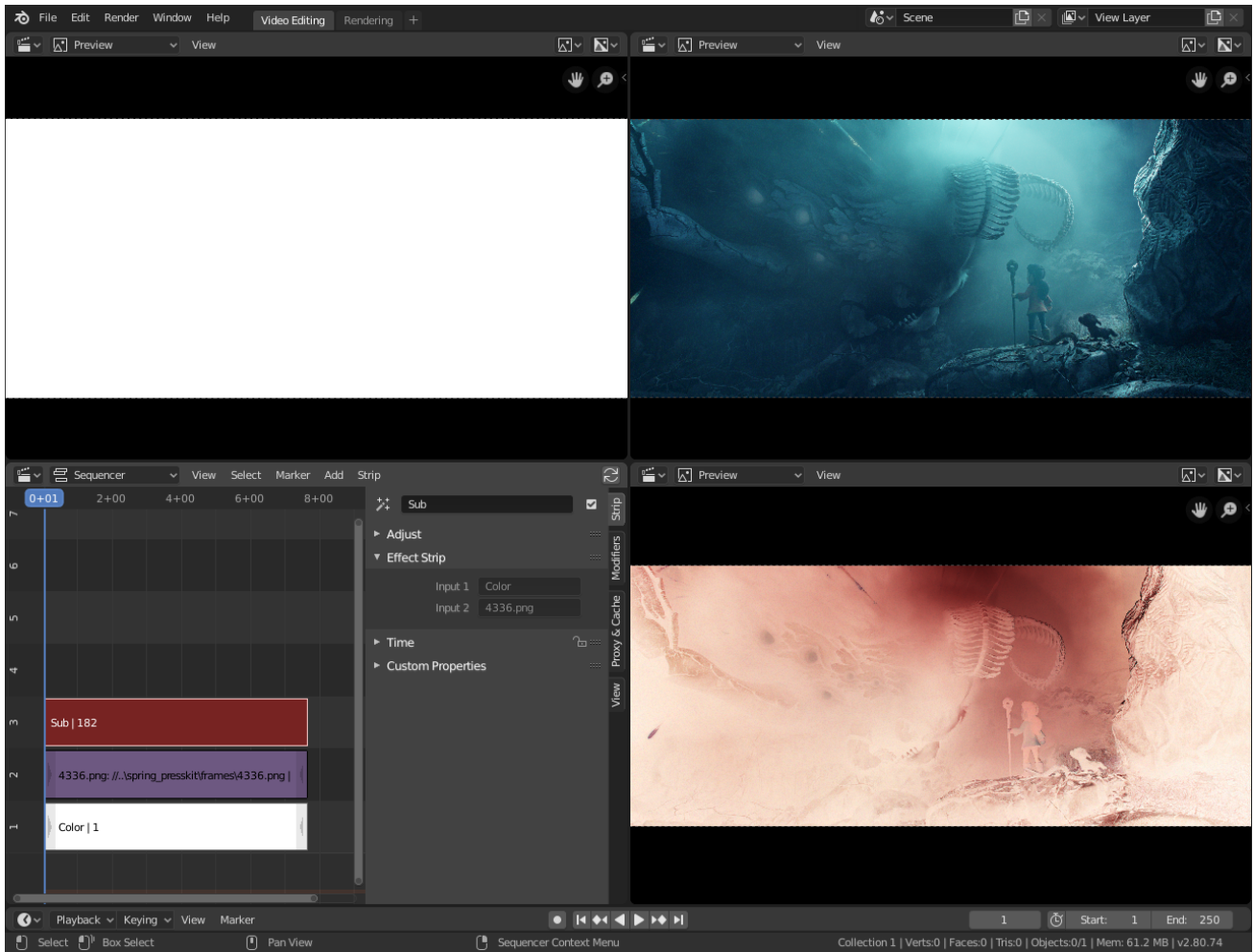


Fig. 413: Subtract Effect.

The opening title sequence to James Bond movies, where the camera is looking 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, 0.5). 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

#### Alpha Over, Under & Over Drop

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*.

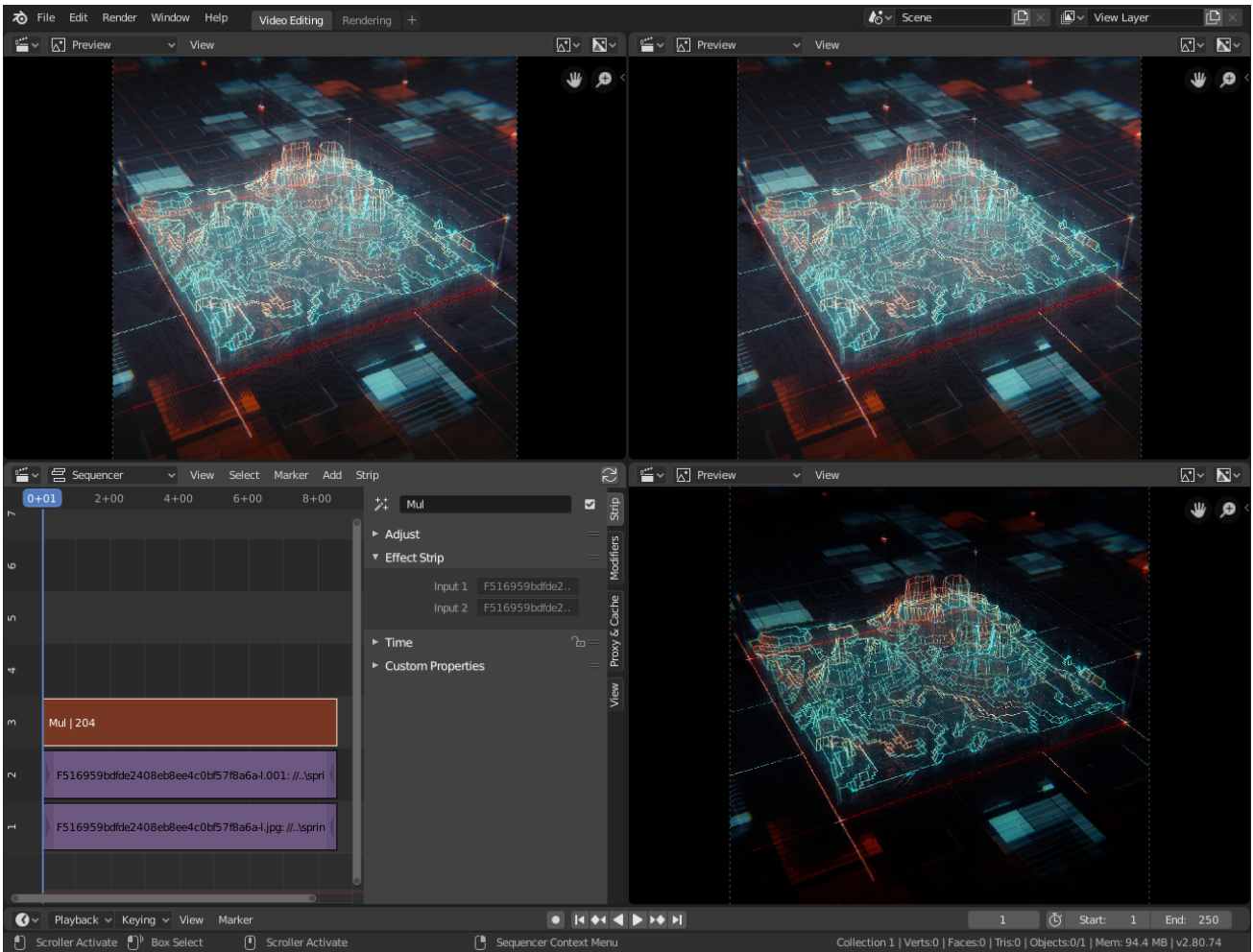


Fig. 414: Multiply Effect.

## Example

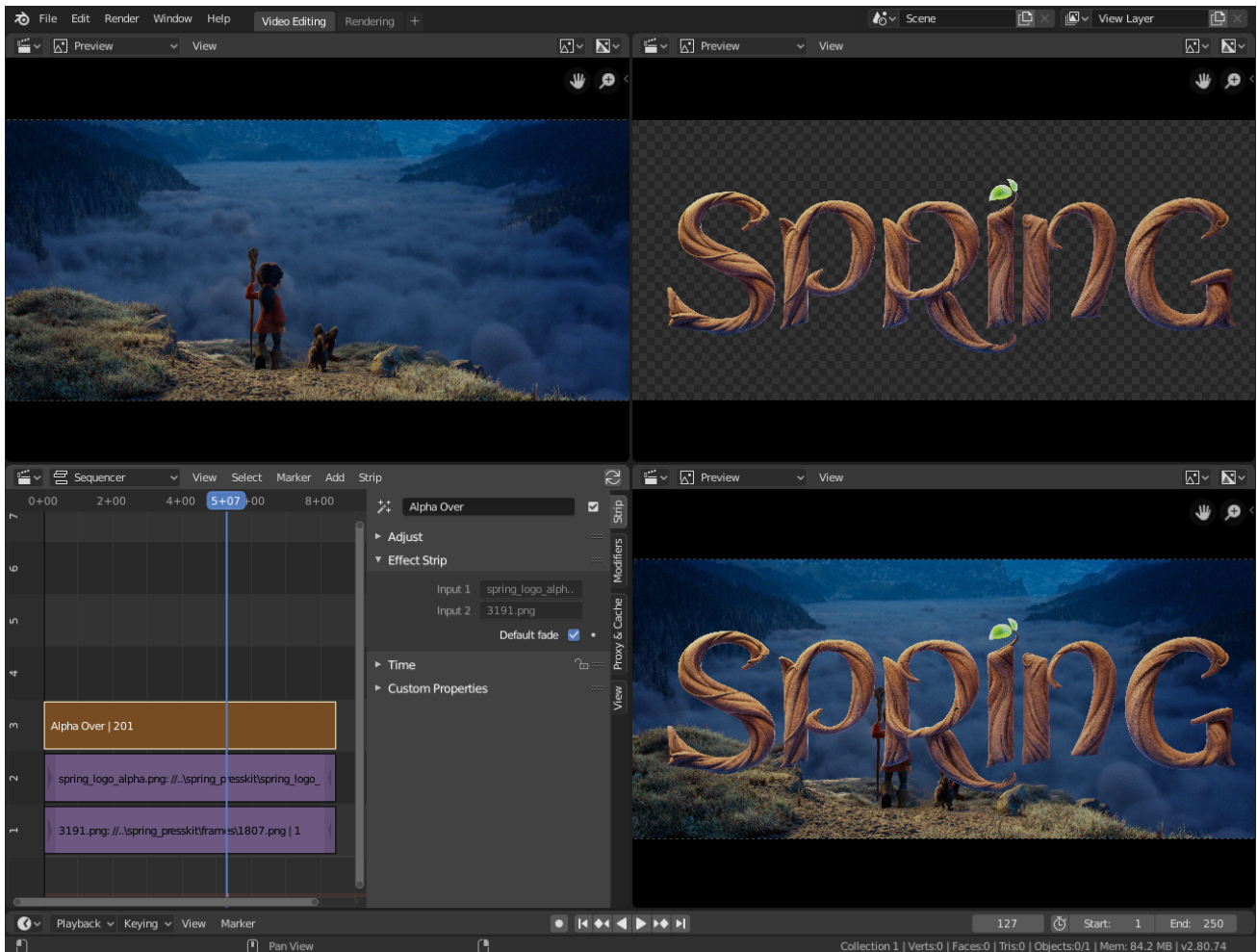


Fig. 415: Alpha Over Effect.

## 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

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

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. This process can be rather easy in the VSE (Video Sequence Editor) if you properly setup every to improve your workflow.

## 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. Add a viewer region for every input channel and to improve the performance use proxies.
4. Add a Multicam Selector strip *above* all the channel tracks.  
After completing these steps you should get something similar to the image below:
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.

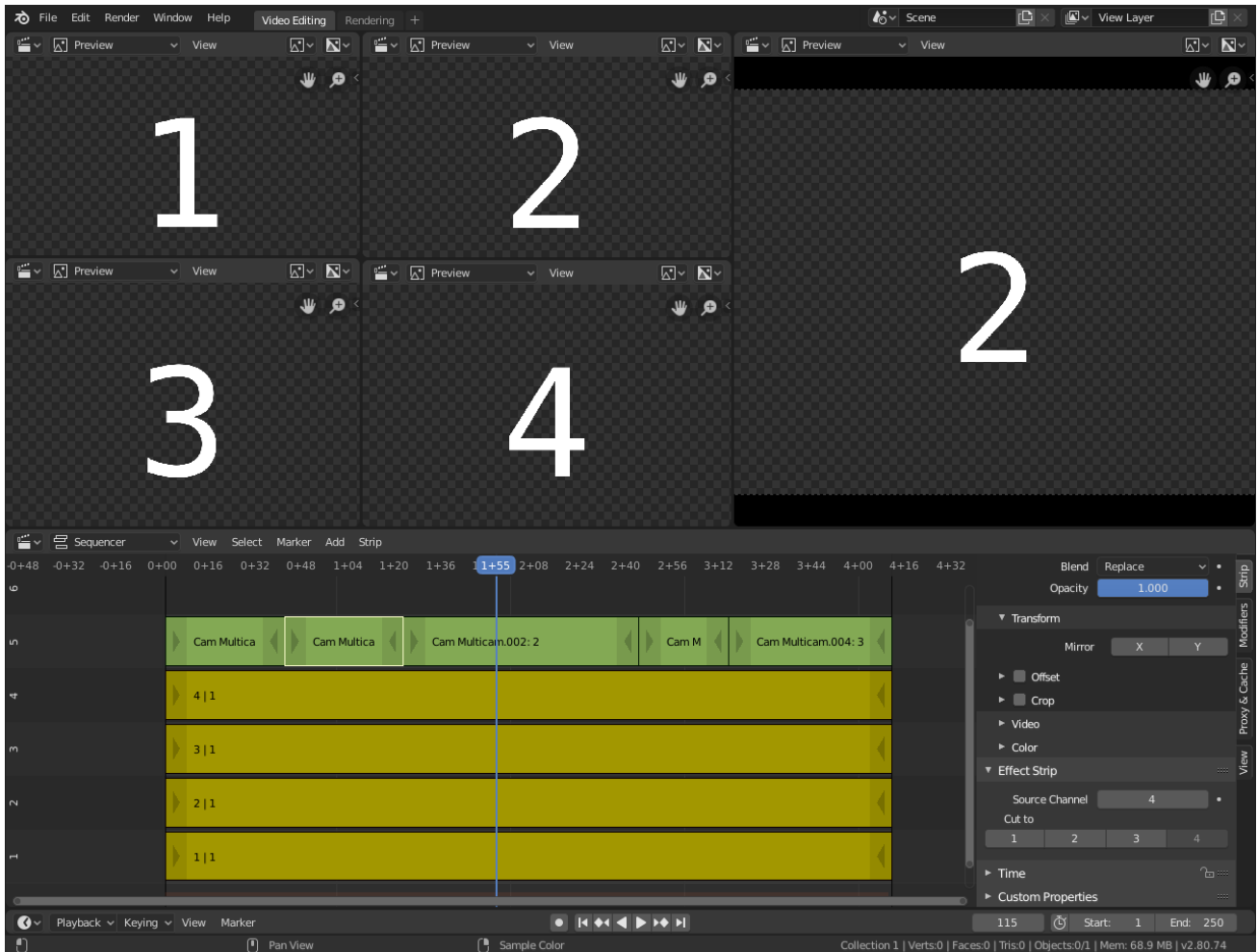


Fig. 416: Multi-camera editing setup.



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, do some fine-tuning by selecting the outer handles of two neighboring Multicam for A/B rolling.

## Transform

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

### Speed Control

Speed Control time-warps the strip, making it play faster or slower than it normally would. A *Global Speed* less than 1.0 makes the strip play slower; greater than 1.0 makes it play faster. 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 will appear to freeze. To avoid this, position the next strip under the original at a point where you want the motion to continue.

## Options

**Stretch to Input Strip Length** Automatically calculates the *Speed Factor* based on the length of the input strip. So if you make a strip 1/2 the original size the sequence will play back at 2 times the speed.

**Use as Speed** Calculates the scale value based on a *Speed Factor*.

---

**Note:** You will have to manually re-adjust the length of the strip accordingly.

---

**Speed Factor** Multiplies the current speed of the sequence by this value. So, a value of 0.5 will make the sequence half as fast while 2 would make the sequence twice as fast.

**Frame Number** Specifies a frame to remap the current frame to, for example, setting this value to 50 displays the 50th frame. This can then be manually *keyframed* to recreate the animation.

**Scale to Length** Maps the frame range on a 0-1 scale. For example, using this and a *Frame Number* of 0.5 will select the frame halfway through the sequence.

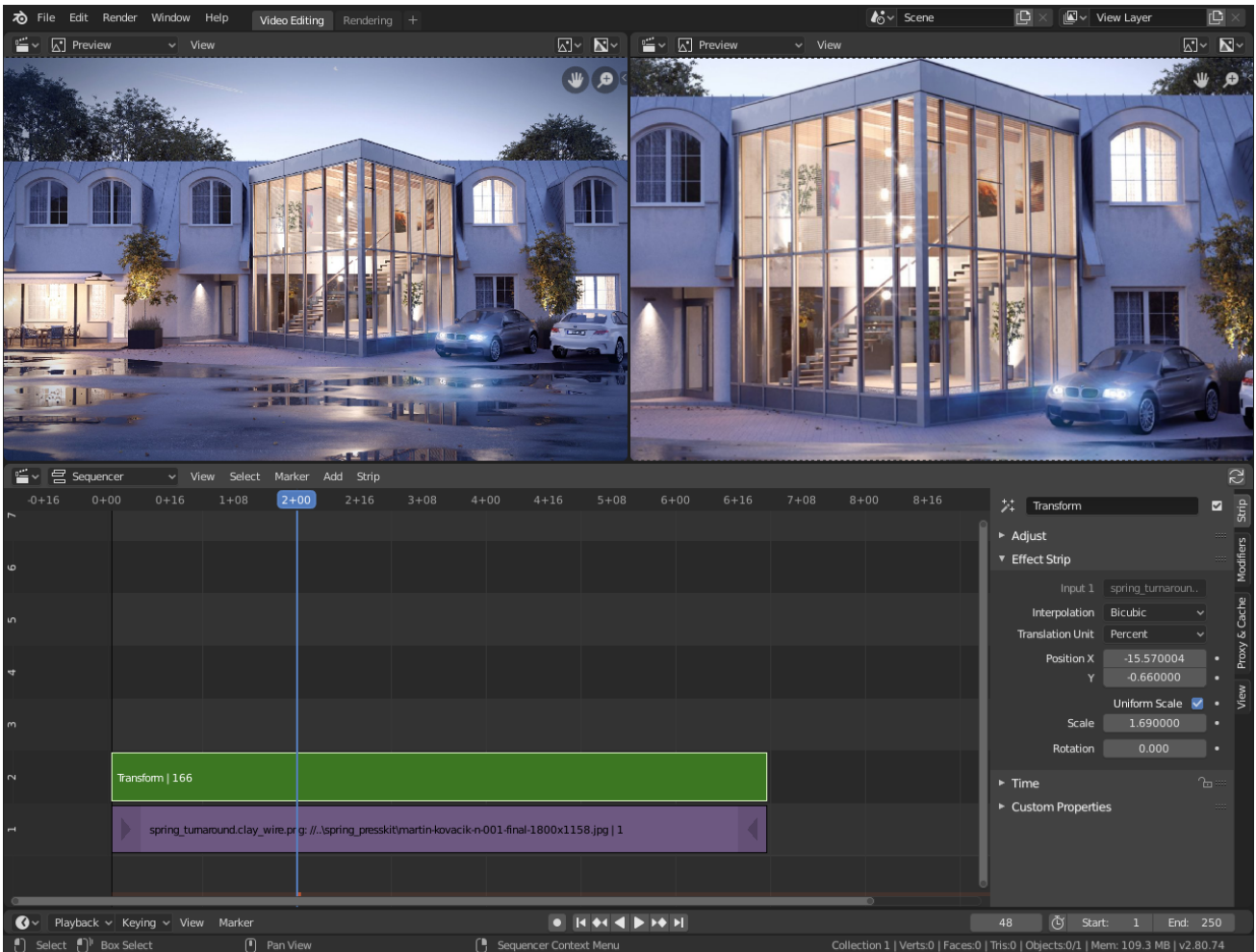


Fig. 417: Transform Effect.



**Multiply Speed** A multiplier applied after all other calculations.

**Frame 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. Click to drop it and press **N** to get the Properties. Uncheck the *Stretch to input strip length* option in the Effect Strip section. Set the Speed 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:

$$\text{new\_length} = \text{real\_length} / \text{speed\_factor}$$

That is it, set your render to animate (in this example) all 550 frames.

### Keyframing the Speed Control

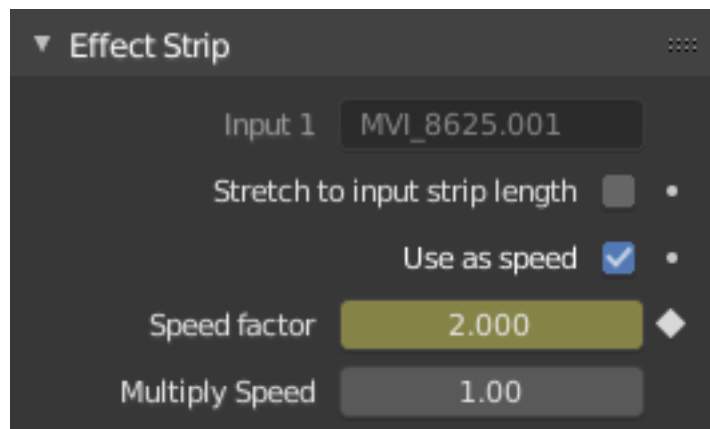


Fig. 418: Keyframing the Frame number.

To get even finer control over your clip timing, you can use curves! While it is possible to keyframe the Speed factor, usually you want to keyframe the Frame number directly.

Uncheck *Stretch to input strip length* and uncheck *Use as speed*. 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.

**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 Global Speed 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 Global Speed of 30/24, or 1.25 (and Enable Frame Blending to give that film blur feel). Instead of producing  $5 \times 60 \times 30 = 9000$  frames, Blender would produce  $9000 / 1.25 = 7200 = 5 \times 60 \times 24$  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

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

### Gaussian Blur

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.

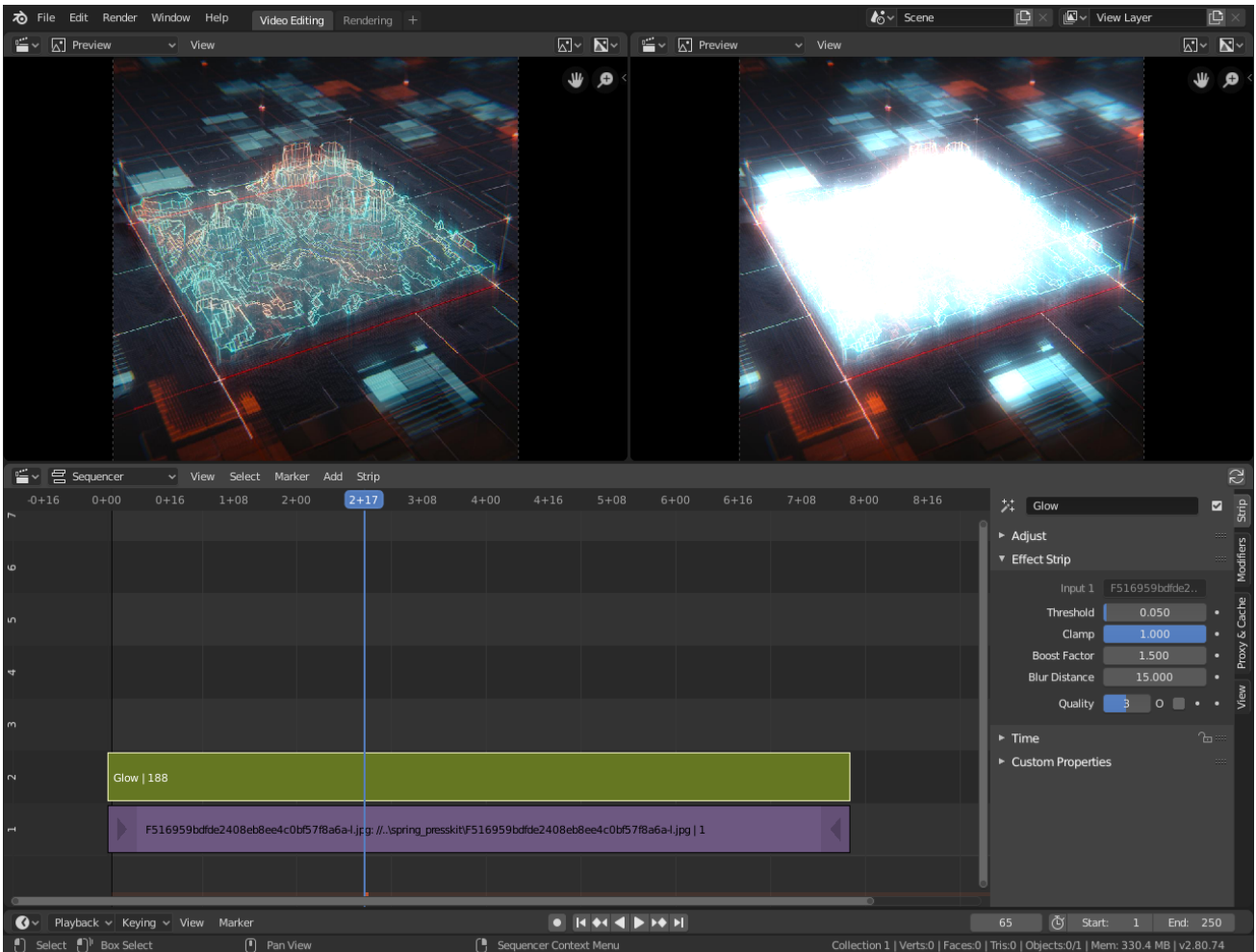


Fig. 419: Glow effect.

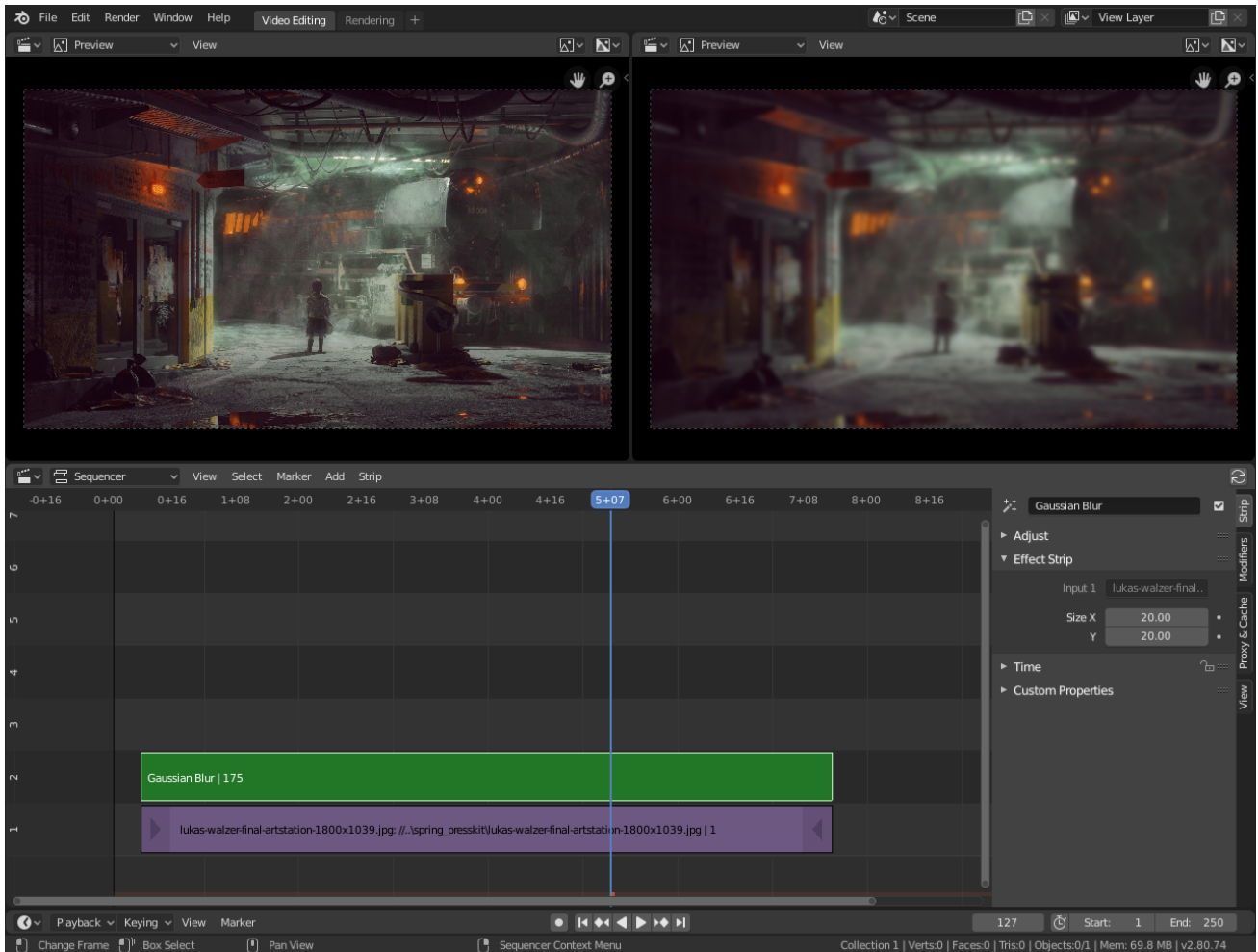


Fig. 420: Gaussian Blur Effect.

## Example

### Transitions

#### 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

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

#### Gamma Cross

The *Gamma Cross* transition is similar to the *Cross* transition, however, the *Gamma Cross* 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

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.

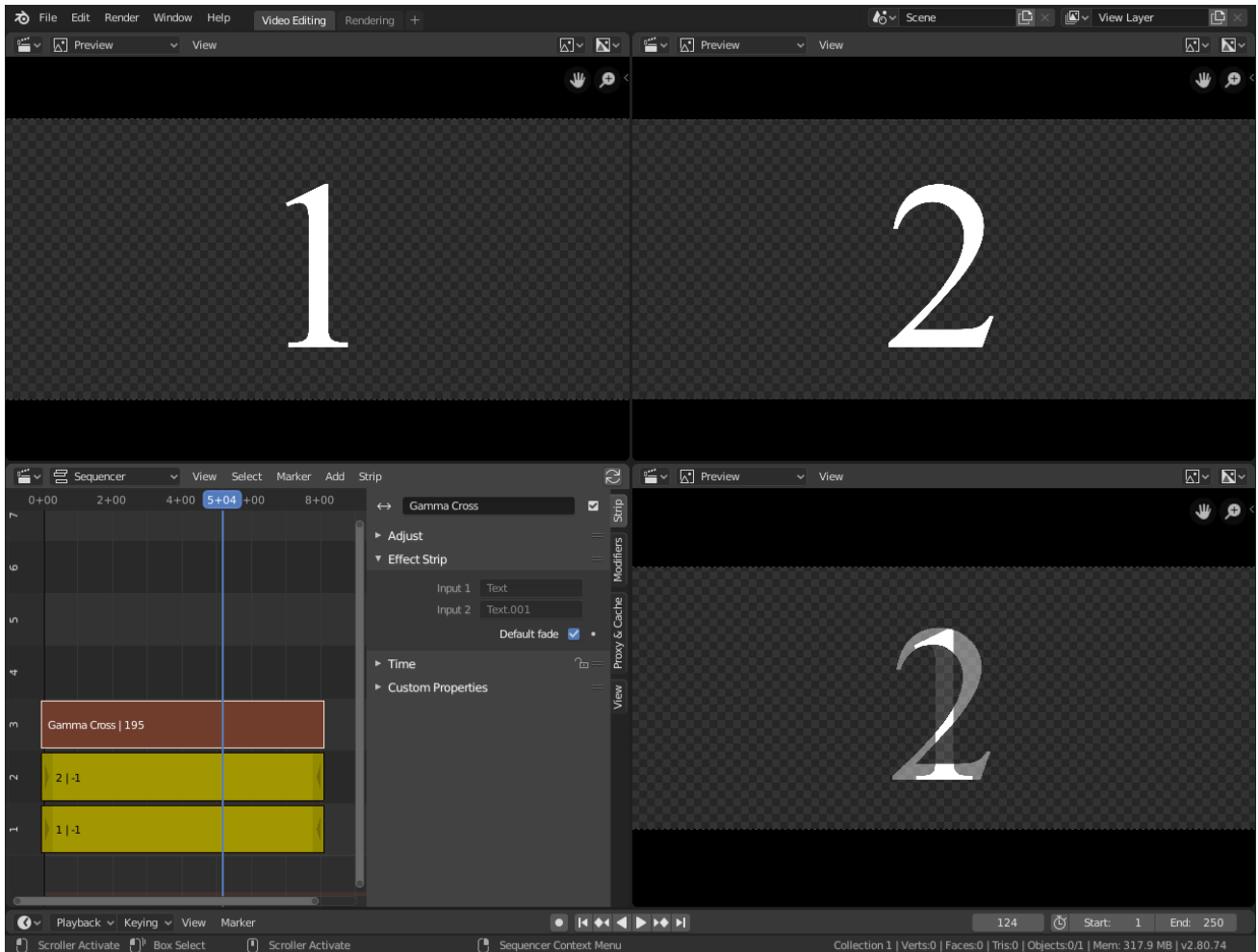


Fig. 421: Cross Effect.

**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



Fig. 422: 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 helps 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** Click and drag a rectangular lasso around a region of strips in your Sequence workspace. Selects strips all intersecting this rectangle.

**Box Select (Include Handles) Ctrl-B** Works the same as *Box Select* but it selects only the strip's handles, if just one handle is selected moving the strip after selecting will change the strip's length. If both handles are selected the strip will move and behave the exact same as *Box Select*.

### Side of Frame

**Left/Right [/]** Select strips laying left or right to the current frame.

**Handle** Select left, right or both handles of selected strips.

---

**Note:** Select with this method when you want to change the timing of a cut.

---

**Channel** Select strips in the same channel laying left or right to active strip.

### Linked

**All Ctrl-L / Less Ctrl-NumpadMinus / More Ctrl-NumpadPlus** Selects strips, that are placed next to each other without any gaps.

**Grouped Shift-G** Selects strips according to their relation with other strips.

**Type** Selects any strips of the same type within a category for example, if you have a cross strip selected this will select all other effect strips.

**Global Type** Selects any strips of the same type, e.g. Effect, Image, Movie, etc.

**Effect Type** Selects all effect strips.

**Data** Selects strips that share the same data, for example, two image strips sharing the same image file.

**Effect** Selects the strip that shares an effect strip.

**Effect/Linked** Selects the effect strips, if any, linked to the currently selected strip.

**Overlap** Selects any strips that occur on the same frame as the current.

## Editing

### Transform

### Move

---

## Reference



**Menu** *Strip* → *Transform* → *Move*

**Hotkey** 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 snaps to the start and endpoints of other strips. The position of the mouse relative to the selection influences where the strips are snapped. If it is closer to the start of the selection, then the start frame of the selection gets snapped, else the end frame will get snapped.

To “ripple edit” (make room for strips you drag) hold Alt when placing a strip.

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 fourth one.

---

### Move/Extend from Current Frame

---

#### Reference

**Menu** *Strip* → *Transform* → *Move/Extend from Current Frame*

## Hotkey 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 the current frame.

## Slip Strip Contents

---

### Reference

**Menu** *Strip* → *Transform* → *Slip Strip Contents*

**Hotkey** 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*

**Hotkey** Shift-S

---

Moves the strip or control point to the current frame.

## Clear Strips Offset

---

### Reference

**Menu** *Strip* → *Transform* → *Clear Strips Offset*

**Hotkey** Alt-0

---

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*

**Hotkey** Backspace

---

Remove blank frames between the current frame and the first strip to the right, independent of selection or locked state of strips.

## Insert Gaps

---

### Reference

**Menu** *Strip* → *Transform* → *Insert Gaps*

**Hotkey** Equals

---

Insert blank frames between the current frame and the first strips to the right, independent of selection or locked state of strips.

## Split

---

### Reference

**Menu** *Strip* → *Split*

**Hotkey** 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*

**Hotkey** 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*

**Hotkey** 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*

**Hotkey** Delete, X

---

Delete the selected strip(s).

## Separate Images

---

### Reference

**Menu** *Strip* → *Separate Images*

**Hotkey** 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*

**Hotkey** 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*

**Hotkey** Alt-S

---

Swaps the first two inputs for the effect strip.

### Lock/Mute

**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 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.

## 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.

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.

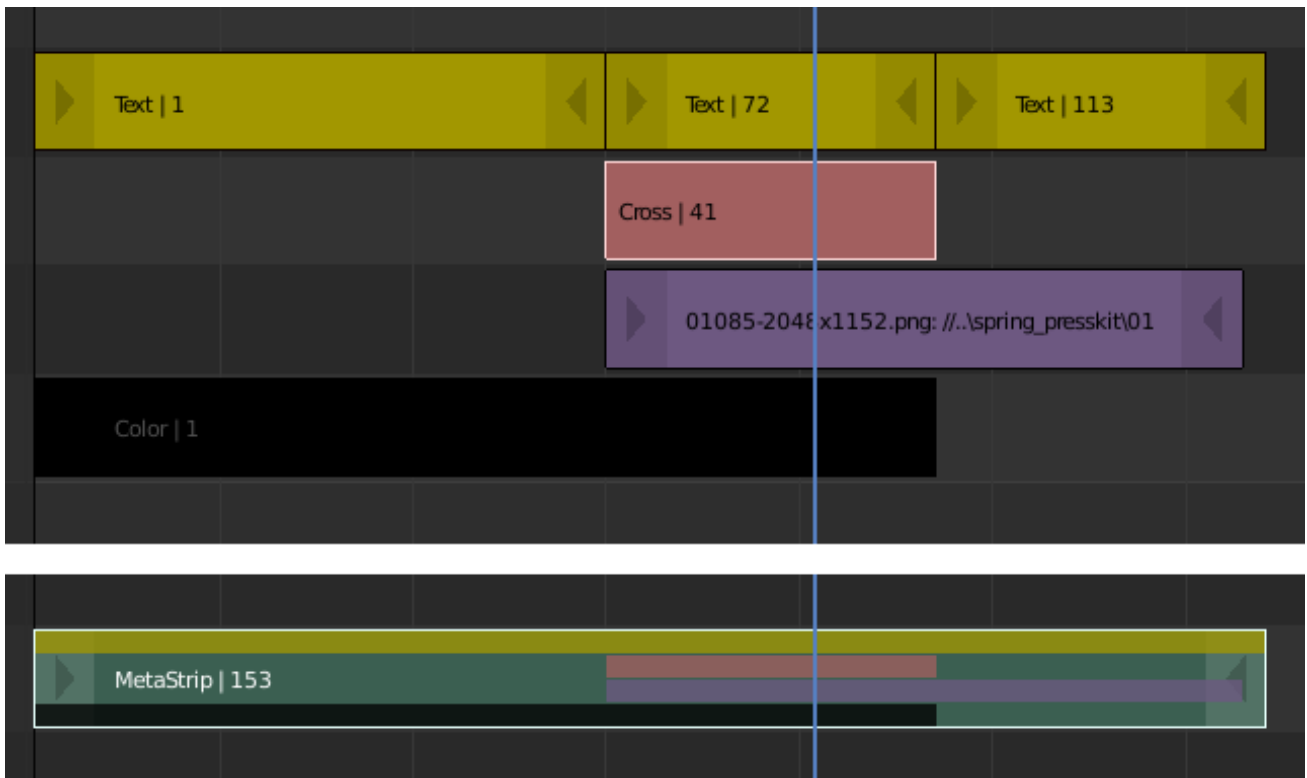


Fig. 423: Example of Meta strips.

---

**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.

## Properties

### Strip

#### Header

**Type** Strip type represented by an icon.

**Name** You can name or rename your strips here.

**Mute** If checked the strip will not produce any output.

### Adjust Panel

---

## Reference

**Panel** *Sidebar region* → *Strip* → *Adjust*

---

The *Adjust* panel is used to control visual properties of strips.

### Compositing

**Blend** Mode of blending strip with lower channels.

**Opacity** Set the opacity (alpha) of the strip. This value, when animated and with the *View* → *Show F-Curves* option turned on, is drawn on the strip as a dark section that follows the animation curve.

### Transform

**Mirror X** Mirrors the image along the X axis (left to right) or the Y axis (top to bottom).

### Offset

Used to move the frames along the X and Y axis. Additionally it disables the auto-scaling of the image.

### Crop

Used to crop the source image, use *Top*, *Left*, *Bottom*, and *Right* to control which part of the image is cropped.

### Video

**Strobe** To only display each nth frame. For example, if you set this to 10, the strip will only display frames 1, 11, 21, 31, 41... of the source. *Strobe* is a float value - this way you can get a strobe effect synced exactly to a beat, for example, by using non-integer values.

**Reverse Frames** Strip is played backwards starting from the last frame in the sequence.

### Color

**Saturation** Increase or decrease the saturation of an image.

**Multiply** Multiplies the colors by this value. This will increase the brightness.

**Convert to Float** Converts input to float data.

### Sound

**Volume** The volume of the sound. This value, when animated and with the *View* → *Show F-Curves* option turned on, is drawn on the strip as a dark section that follows the animation curve. The value is also reflected in the waveform.

**Pitch** Coefficient of playback speed. This value will affect length of the strip, that will not be represented in the timeline.



**Pan** Used to pan the audio from left and right channels. Only works for mono sources. Values can be between -2 and 2, where 0 means front/center, -1 means to the left and 1 to the right. In case of multichannel audio (rear speakers) you can pan to those with the higher values: -2, 2 is back. So this value basically represents the angle at which it's played.

**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*.

**Mono** Mixdown all audio channels into a single one.

## Time Panel

---

### Reference

**Panel** *Sidebar region* → *Strip* → *Time*

---

The Time panel is used to control source and timeline position of the strip.

**Lock (padlock icon)** Prevents the strip from being moved (found in the panel header).

**Channel** Changes the channel number, or row, of the strip.

**Start** Changes the starting frame number 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** Specifies the ending time and ending frame number for the strip.

**Strip Offset Start/End** Can be used to either extend the strip beyond the end frame by repeating the last frame. Or it can be used to shorten the strip, as if you were cropping the end frame. This is the same as adjusting the strip handles.

**Hold Offset Start/End** Offset of the uncut strip content.

**Current Frame** Position of the Playhead relative to the start of the active strip.

## Source Panel

---

### Reference

**Panel** *Sidebar region* → *Strip* → *Source*

---

The Source panel is used to control sources of the strip such as filename and file path and various methods of interpreting these files.

**Path** The directory that contains the source file. When the file is moved this can be updated instead of re-create the strip.

**File** The file name of the source file. For image strips showing an image sequence, this will be different for each frame.

**Change Data/Files** Same as the *Path* and *File* fields, but this time combined to open the File Browser in order to find the file(s) you search. Same as *Strip* → *Inputs* → *Change Paths/Files*.

**MPEG Preseek** Movie strip only - Use Preseek field to tell Blender to look backward and compose an image based on the specified amount of previous frames (e.g. 15 for MPEG-2 DVD).

**Color Space** To specify the color space of the source file.

**Alpha Mode** If the source file has an Alpha (transparency) channel, you can choose:

*Straight Alpha* or *Premultiplied Alpha*

**Stream Index** Movie strip only - For files with several movie streams, use the stream with the given index.

**Deinterlace** Removes fields in a video file. For example, if it is an analog video and it has even or odd interlacing fields.

**Resolution** Resolution of the active strip image output.

### Options for Sound Strips

**Sound** *Data-block menu* to select a sound.

**Path** Path to the sound file used by this *data-block* menu.

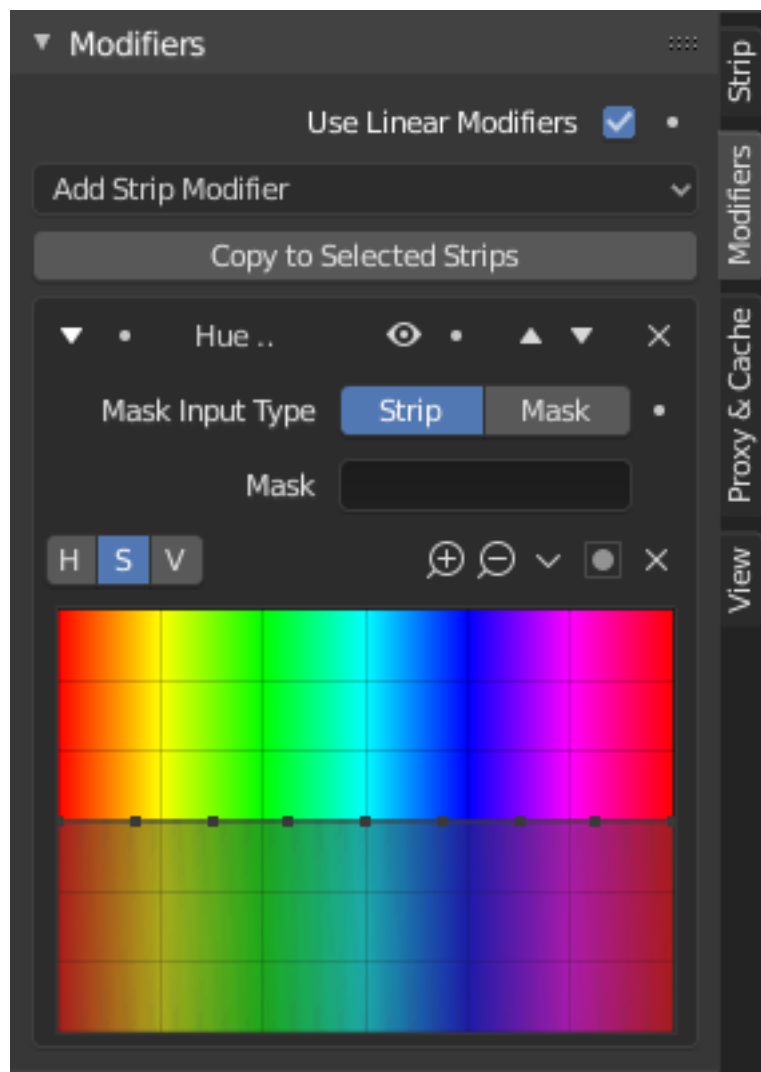
**Pack** Pack sound into the blend-file.

**Caching** Sound file is decoded and loaded into the RAM.

### Modifiers Panel

#### Reference

**Panel** *Sidebar region* → *Modifiers* → *Modifiers*



Modifiers are used to make adjustments on the image, like contrast, brightness, saturation, color balance and applying masks.

You can add these modifiers directly to the selected strip, or you can use it within an “Adjustment Layer” effect strip, which allows you to apply these modifiers onto several strips the same time.

**Use Linear Modifiers** Calculate modifiers in linear space instead of sequencer space.

**Copy to Selected Strips** Allows you to copy the modifiers to selected strips. This works two ways, you can either replace the old modifiers or append/add to the previous modifiers.

### Common Options

Each modifier has several buttons at its top:

**Mute (eye icon)** Disables the modifier. Very useful to compare the image, with / without modifications.

**Move (up/down arrow icon)** The next two buttons are used to change the modifier’s position in the stack.

**Remove X** The cross is to delete the modifier from the stack.

### Input Mask Type

**Strip** Use this to apply the modification on the whole image, or to use another strip’s image (with alpha channel) for masking the modifier (and only this modifier), by choosing it in the “Mask” select menu.

**Mask** This allows you to choose a Mask created in the Mask editor which will limit the modification to the masked image’s zones.

### Types

Currently, the following modifiers are supported:

#### Color Balance Modifier

Color balance adjustments, through Lift, Gamma, and Gain.

This modifier works the same as the *Color Balance Node*.

#### Curves Modifier

Color and RGB curves.

This modifier works the same as the *Curves Node*.

#### Hue Correct Modifier

HSV multi points curves.

This modifier works the same as the *Curves Node*.

#### Bright/Contrast Modifier

Adjusts the brightness and contrast of the modifier input.

## Mask Modifier

Use it for masking the other modifiers in the stack which are below.

For example, to correct the brightness only on a certain zone of the image, you can filter the Bright/Contrast modifier by placing a Mask modifier, just before it in the stack. You can choose to use a Mask created in the Mask editor, or to use another strip as a mask (the image of this strip must have an alpha channel). This mask will be applied on all the others modifiers below it in the stack.

## White Balance Modifier

Use it to adjust the white balance by choosing the color that should be white.

## 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*.

## Proxy & Cache Settings

### Cache Settings

---

## Reference

**Panel** *Sidebar region* → *Proxy & Timecode* → *Cache Settings*

---

The Cache is used to save frames in memory for preview, so they can be later displayed much faster than rendered from scratch. Cache capacity can be set in *System tab* of the Preferences.

In this panel you can set up types of images that will be cached for all strips.

### Cache

**Raw** Cache raw images read from drive, for faster tweaking of strip parameters at the cost of memory usage.

**Pre-processed** 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.

**Final** Cache final image for each frame.

**Recycle Up to Cost** Only frames with cost lower than this value will be recycled.

Each stored image has a cost assigned. Cost is calculated as ratio of time spent on rendering to maximum possible time to keep up with chosen frame rate. The higher the cost, the harder it is to render image.

Maximum image cost is limited to arbitrary value of 10.

---

## Strip Cache

---

### Reference

**Panel** *Sidebar region* → *Proxy & Timecode* → *Cache Settings*

---

Similar to *Cache Settings* panel, this panel sets the types of images that will be cached for the active strip.

**Enable Strip Cache** Enable overriding the cache defaults. When disabled, *Cache Settings* will be used.

**Cache Raw** Cache raw images read from drive, for faster tweaking of strip parameters at the cost of memory usage.

**Cache Preprocessed** Cache preprocessed images, for faster tweaking of effects at the cost of memory usage.

**Cache Composite** Cache intermediate composited images, for faster tweaking of stacked strips at the cost of memory usage.

## Proxy Settings

---

### Reference

**Panel** *Sidebar region* → *Proxy & Timecode* → *Proxy Settings*

---

**Storage** Defines whether the proxies are for individual strips or the entire sequence.

**Per Strip** Proxies are stored in the directory of the input.

**Project** All proxies are stored in one directory.

**Proxy Directory** The location to store the proxies for the project.

**Set Selected Strip Proxies** Set proxy size and overwrite flag for all selected strips.

**Rebuild Proxy and Timecode Indices** Generates Proxies and Timecodes for all selected strips, same as doing *Strip* → *Rebuild Proxy and Timecode indices*.

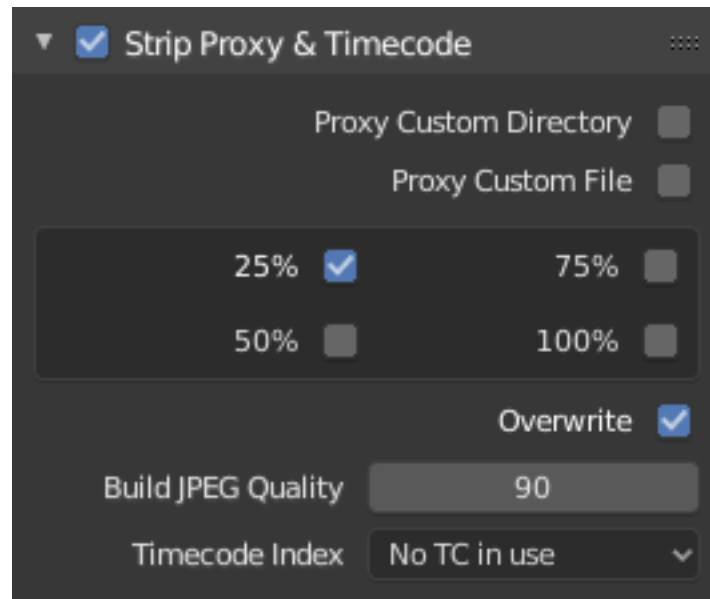
## Strip Proxy & Timecode Panel

---

### Reference

**Panel** *Sidebar region* → *Proxy & Timecode* → *Strip Proxy & Timecode*

---



## Proxy

Once you have chosen the *Proxy/Timecode* parameters, you need to select all strips for which you want proxies to be built. Then use *Strip* → *Rebuild Proxy and Timecode indices*, or button in *Proxy Settings* panel. Once all proxies are built, they will be ready to use.

In order to use proxies, you have to select matching *Proxy Render Size* in Sequencer preview Sidebar panel.

### Custom Proxy

**Directory** By default, all generated proxy images are storing to the <path of original footage>/BL\_proxy/<clip name> folder, but this location can be set by hand using this option.

**File** Allows you to use pre-existing proxies.

**Resolutions** Buttons to control how big the proxies are. The available options are 25%, 50%, 75%, 100 percent of original strip size.

**Overwrite** Saves over any existing proxies in the proxy storage directory.

**Build JPEG Quality** Defines the quality of the JPEG images used for proxies.

**Timecode Index** When you are working with footage directly copied from a camera without pre-processing it, there might be bunch of artifacts, mostly due to seeking a given frame in sequence. This happens because such footage usually does not have correct frame rate values in their headers. 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 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 Proxy/Timecode option in Blender.

The following timecodes are supported:

- No TC in use - do not use any timecode
- Record Run
- Free Run
- Free Run (rec date)
- Record Run No Gaps

---

**Note:** Record Run is the timecode which usually is best to use, but if the clip's file is totally damaged, *Record Run No Gaps* will be the only chance of getting acceptable result.

---

## Preview

### Introduction

Sequencer preview is used to display result of rendering Sequencer timeline. This can be further configured to display output from certain channel, overlay or image analyzer (scope). You can adjust the view by zooming in with NumpadPlus and zoom out with NumpadMinus.

### Header



Fig. 424: Sequencer Display header.

### View Menu

**Sidebar N** Show or hide the *Sidebar*.

**Toolbar T** Show or hide the *Toolbar*.

**Fit Preview in Window Home** Resize the preview so that it fits in the area.

**Zoom Shift-B** Click and drag to draw a rectangle and zoom to this rectangle.

**Fractional Zoom** Resize the preview in steps from 1:8 to 8:1.

**Show Frame Overlay** Displays the *Frame Overlay*, to compare the current frame to a reference frame.

**Show Safe Areas** Display an overlay on the preview, marking where title safe region is.

**Show Metadata** Display Image metadata in the preview area.

**Show Annotations** Displays *Annotations* in the preview region.

**Sequence Render Image** Render the an image at the current frame.

**Sequence Render Animation** Render timeline from Preview Start to Preview End Frame to a Video file or series of images.

**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 display type between Sequencer and Preview.

### Display Mode

Mode to show different aspects of the composite result, for the current frame:

**Image Preview** Render image preview.

**Luma Waveform** Brightness/contrast analyzer.

**Chroma Vectorscope** Color hue and saturation analyzer.

**Histogram** RGB distribution histogram.

## Display Channels

**Color and Alpha** Display preview image with transparency over checkerboard pattern.

**Color** Ignore transparency of preview image (fully transparent areas will be black).

## Gizmos

You can use gizmos to pan and zoom image in the Sequencer preview region.

## Display Mode

There are different display modes available, each having a specific purpose.

## Image Preview

The Image Preview mode shows you what the resulting video will look like when saved. This is the main working mode for adding strips and moving them around, cutting, grouping (making meta) and splicing them through special effects.

## Luma Waveform

For the selected channel, brightness, or luminosity, is mapped with this display.

A luma waveform allows you to judge the quality of the luminance distribution across the video signal, you can view a luma waveform instead of the usual output display on every control monitor.

The display plots for every scanline the luminance value. The lines are all drawn on top of each other. The points get brighter if the lines cross (which is very likely with several hundred scanlines). You will understand the picture most easily if you plug an oscilloscope to the Luma-video-output of your television set. It will basically look the same.

In this mode, the vertical axis represents the luminosity: 0 at the bottom, 1 at the top; the horizontal axis is a mapping from the horizontal axis of the frame. There are as many curves as scanlines in the frame: each one of these curves represents the luminosity of the pixels of one line. Moreover, the color of a pixel in this mode represents the number of pixels from the matching column of the frame sharing the same luminosity, i.e. the number of curves that cross at this point (black/transparent, for no pixel, white/opaque for at least three pixels).

**Separate Colors** Separates RGB channels into separate graphs.

This mode is good for:

- If the waveform does not fill the whole picture you might want to play with the Bright/Contrast modifier until it fills the whole picture (contrast autostretch).
- With the more advanced Curves or Color Balance modifiers, you can be more precise.
- You can judge if you want to dump the whole thing since it is completely distorted and clips at the top or the bottom.



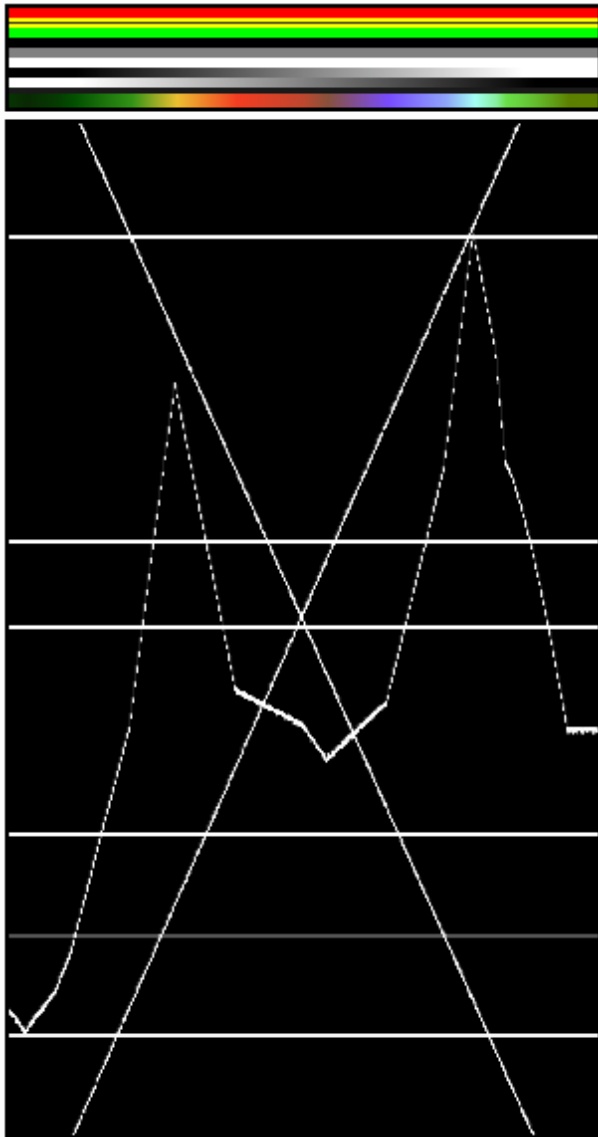


Fig. 425: The various horizontal lines in the Luma waveform match the uniform-colored lines of the picture. Note that the 'gray 20%' one-pixel width line (inside the yellow strip) is represented in the Luma waveform by a gray line. The two lines drawing an "X" are from the two linear tone shades (white -> black and black -> white). Finally, the broken line matches the complex tone shade at the bottom of the picture.

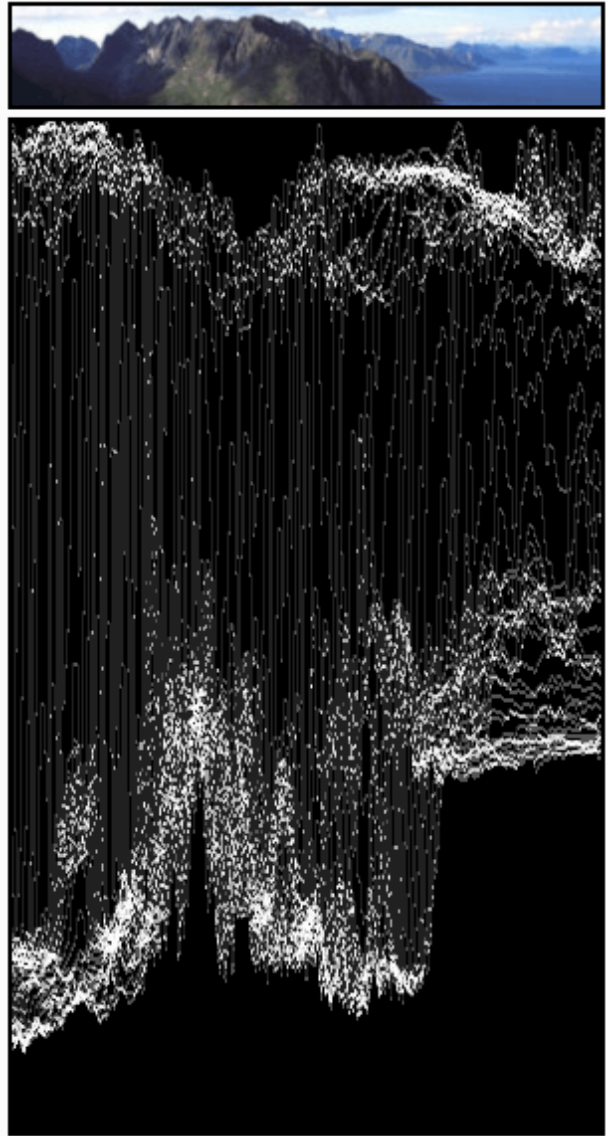


Fig. 426: The curves are quite visible. We found a luma of 80-100% for the sky, a luma around 40% for the sea, and a luma of 10-20% for the mountains, growing around 40% for the sunny part.

**Note:** The pictures (first green frame, at the top) are only 50px high, to limit the number of curves displayed in the *Luma waveform*.

Use this display to check for appropriate contrast and luminosity across all frames in the channel. When spots in the film that should have even illumination do not, it looks like a flashbulb went off or an extra light was suddenly turned on. This can happen if two strips were rendered or shot under different lighting conditions but are supposed to be contiguous.

## Chroma Vectorscope

Use this mode to judge the quality of the color-distribution and saturation, you can also view a U/V scatter-plot.

The picture is converted to YUV-format. The U and V values represent the angle of the color. For pixel of the picture, one point is plotted in the display at the U and V value position. If several pixels happen to have the same UV value the pixel in the plot gets brighter.

To help you understand what color is meant, a hexagram marking the extreme positions (red, magenta, blue, cyan, green, yellow) is shown and a red cross to mark the origin.

In other words, for the selected channel, this display shows the color space of the image inside a hexagon. Each point of the hexagon is a primary color: red, magenta, blue, cyan, green, and yellow. Black is at the center, and overall saturation is scaled as dots closer to the outside. The example to the right shows that the image has a lot of red (50% saturation) and small amount of blue, with no green.

Always: remember to activate an additional control monitor of the end result. Color calibration is a matter of taste and depends on what you want.

Use this display to check for too much color saturation. While over-saturated images look great for op-art and computer displays, they might not when shown on the big screen TV.

This mode is good for:

- If your picture looks very moody or desaturated you might want to take a look at the U/V plot. You will most likely see all pixels building a crowd at the origin. If you add saturation using the *Saturation* slider in the Filter panel or any modifiers that change color, you can see in the U/V plot if you distort the color.
- If you do color-matching on a by hand basis you can match the angle you see of different channels monitors.



Fig. 427: Example image.

## Histogram

This mode displays a graph showing the distribution of color information in the pixels of the currently displayed image. The X axis represents values of pixel, from 0 to 1 (or 0 to 255), while the Y axis represents the number of pixels in that tonal range. A predominantly dark image would have most of its information toward the left side of the graph.

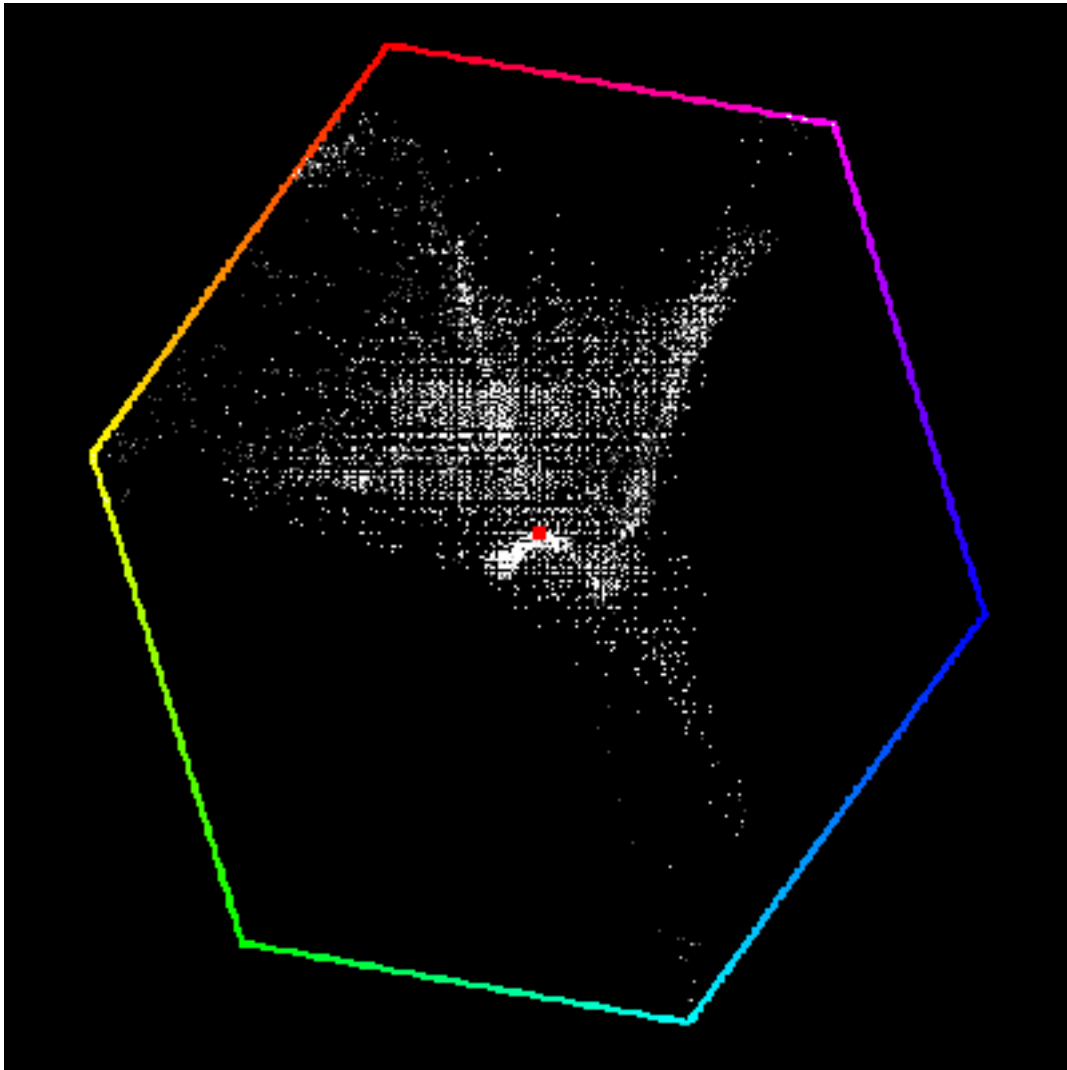


Fig. 428: Example of a Chroma Vectorscope Preview.

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.



Fig. 429: Example image.

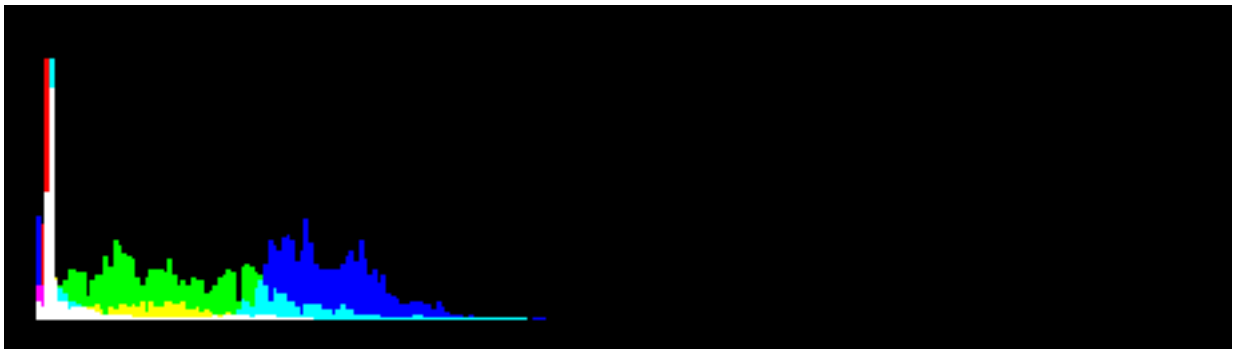


Fig. 430: Example of Histogram Preview.

## Toolbar

**Sample** Used to sample a pixel's color from anywhere within Blender.

**Annotate** Draw free-hand annotation.

**Annotate Line** Draw straight line annotation.

**Annotate Polygon** Draw a polygon annotation.

**Annotate Eraser** Erase previous drawn annotations.

## Properties

### Metadata

A list of metadata of the displayed image.

---

**Note:** The metadata will only be displayed for the image, that has not been processed by any effect. By default images are processed by alpha over blending.

---

## Scene Strip Display

It allows you to control how the images of *Scene Strips* are displayed in the preview.

**Shading** Method for rendering the viewport. See the 3D Viewport's *Viewport Shading* options.

**Override Scene Settings** Use the Workbench render settings from the sequencer scene, instead of each individual scene used in the strip.

## View Settings

**Channel** Selects the channel to show in the preview.

Channel 0 is the compositing result of all strips. Channel 1 is the current frame's image from the strip in channel 1 only (channel 1 is at the bottom of the stack). The display of these modes is either the composite (channel 0) or the frame from the strip (channels 1 through n).

**Show Overexposed** Shows overexposed (bright white) areas using a zebra pattern. The threshold can be adjust with the slider.

**Proxy Render Size** Size to display proxies at in the preview region. Using a smaller preview size will increase speed.

**Prefetch Frames** Automatically fill the cache with frames after the current frame in the background. Use this feature to achieve more consistent playback speed. This feature currently doesn't support rendering scene strips.

## Frame Overlay

Option to enable the overlay. It can be used for comparing the current frame to a reference frame.

**Set Overlay Region** Selects the rectangular bounds for the overlay region. This area can be defined by pressing 0 key over the preview.

**Frame Offset** The slider controls the offset of the reference frame relative to current frame.

**Overlay Type** It describes the way the reference frame should be displayed.

**Rectangle** Which means the rectangle area of reference frame will be displayed on top of current frame.

**Reference** Only the reference frame is displayed in the preview region.

**Current** Only the current frame is displayed in the preview region.

---

**Tip:** It is possible to have several Sequence Editors opened and they can use different overlay types. So it is possible to have current and reference frames displayed in different editor spaces.

---

**Overlay Lock** It's still possible to lock the reference frame to its current position.

## Safe Areas

Shows guides used to position elements to ensure that the most important parts of the video can be seen across all screens.

### See also:

See *Safe Areas* in the camera docs.

## Annotations

Allows you to use *Annotations* in the Sequencer.

## 2.3.8 Movie Clip Editor

### Introduction

The Movie Clip Editor has two main purposes, it can be used for *tracking or masking* movies. The empty editor looks like the image below.

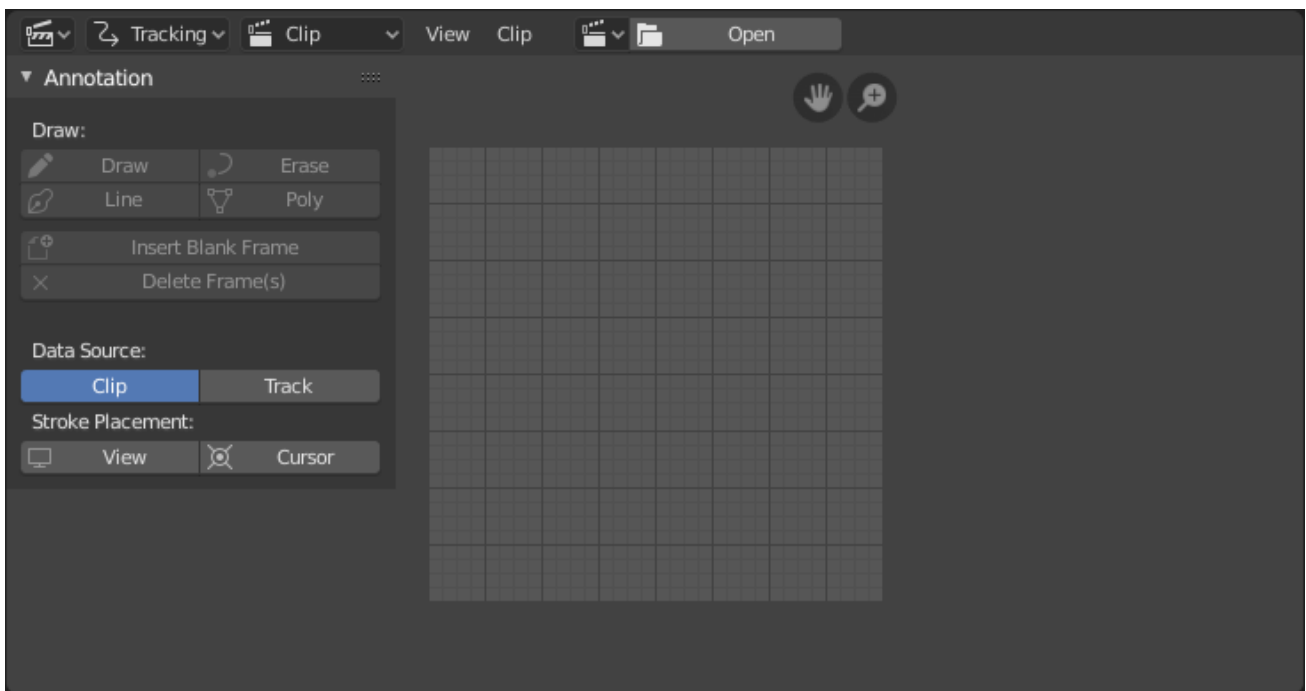


Fig. 431: Movie Clip Editor interface.

### Header

### Menus

**View** Tools for controlling how the content is displayed in the editor.

**Center View to Cursor** Centers the view so that the cursor is in the middle of the view.

**Select** TODO.

### Clip

**Open Clip Alt-0** Open a new clip from an image file sequence.

**Prefetch Frames** TODO.

**Reload Clip** TODO.

**Proxy** TODO.

## Controls

**Clip** A *data-block menu* used for add a movie file. Both movie files and image sequences can be used in the Clip editor. When a movie clip is loaded into the Clip editor, extra panels are displayed in the interface.

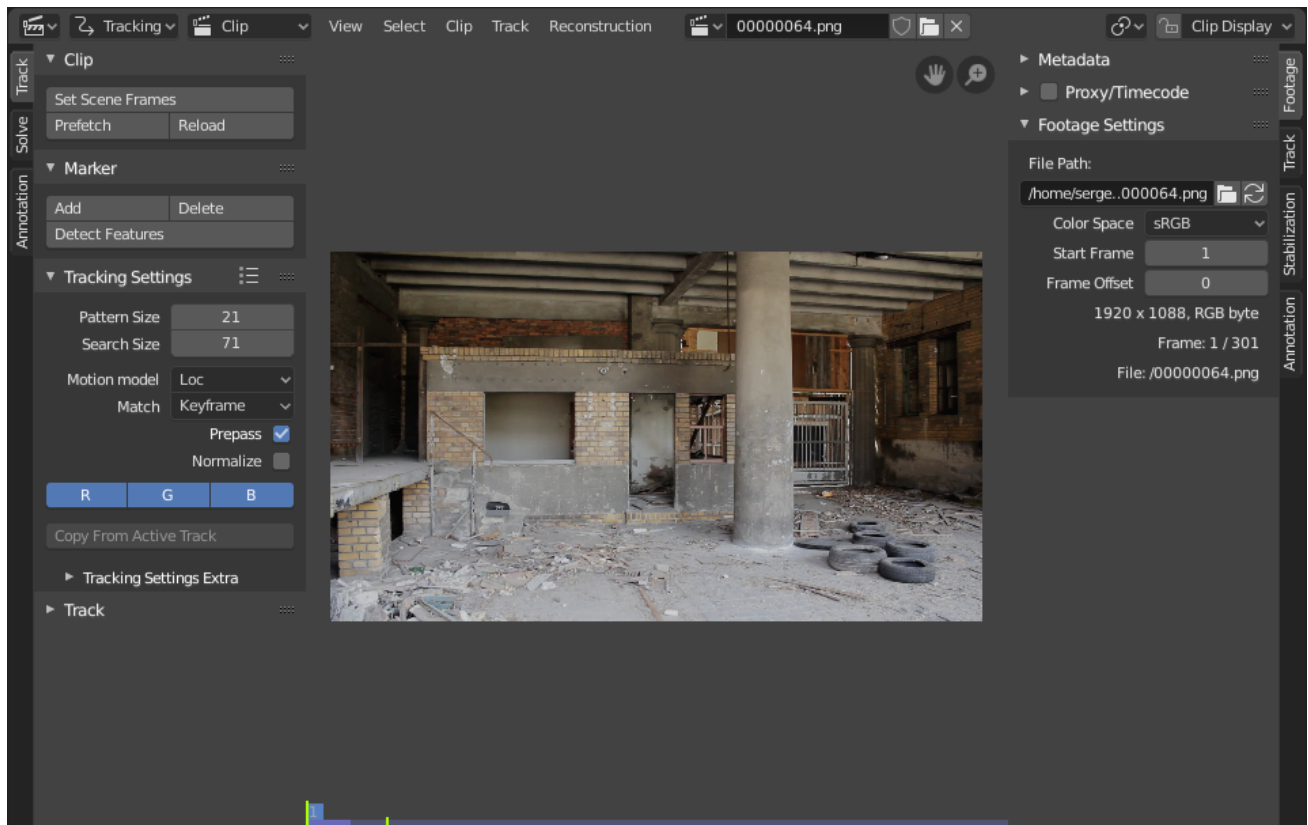


Fig. 432: Movie Clip Editor with an opened clip.

## Modes

- *Motion Tracking*
- *Masking*

**Pivot Point** See *Pivot Points*.

**Clip Display** This pop-over contains display settings related to editor itself.

**Channels** The R, G, B toggles control the color channels used for frame preview. It is needed because the tracking algorithm works with grayscale images and it is not always obvious to see which channels disabled will increase contrast of feature points and reduce noise.

**Grayscale Preview (B/W)** Shows the whole frame gray-scale.

**Mute (eye icon) M** Shows black frames in the preview instead of the movie clip. It helps to find tracks which are tracked inaccurately or which were not tracked at all.

**Render Undistorted** Applies the *Lens Distortion* settings to the viewport image in order to display the footage undistorted. It is only a preview option, which does not actually change the footage itself.



**Lock to Selection L** Makes the editor display selected tracks at the same screen position along the whole footage during playback or tracking. This option helps to control the tracking process and stop it when the track is starting to slide off or when it jumped.

**Show Stable** This option makes the displayed frame be affected by the 2D stabilization settings (available in reconstruction mode only). It is only a preview option, which does not actually change the footage itself.

**Grid** Displays a grid which is originally orthographic, but is affected by the distortion model (available in distortion mode only). This grid can be used for manual calibration - distorted lines of grids are equal to straight lines in the footage.

**Calibration** Applies the distortion model for annotation strokes (available in distortion mode only). This option also helps to perform manual calibration. A more detailed description of this process will be added later.

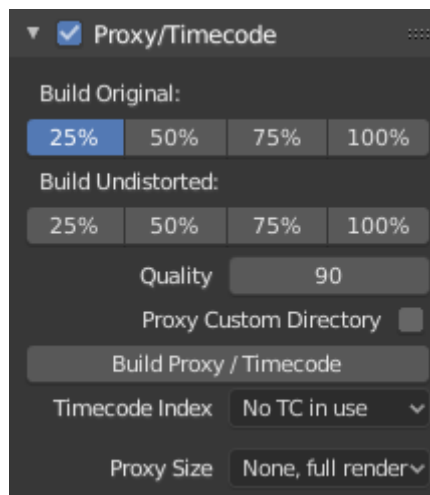
**Display Aspect Ratio** Changes the aspect ratio for displaying only. It does not affect the tracking or solving process.

## Sidebar Region

### Proxy/Timecode Panel

Once you have chosen the Proxy/Timecode parameters, you need to use *Clip* → *Proxy* → *Rebuild Proxy and Timecode indices* to generate the proxy clip and it will be available after Blender makes it.

### Proxy



A proxy is a smaller image (faster to load) that stands in for the main image. When you rebuild proxies Blender computes small images (like thumbnails) for the big images and may take some time. After computing them, though, editing functions like scrubbing and scrolling is much faster but gives a low-res result. Make sure to disable proxies before final rendering.

**Build Original** Used to define which resolutions of proxy images should be built.

**Build Undistorted** Builds images from undistorted original images for the sizes set above. This helps provide faster playback of undistorted footage.

**Quality** Defines the quality of the JPEG images used for proxies.



**Proxy Custom Directory** By default, all generated proxy images are storing to the <path of original footage>/BL\_proxy/<clip name> folder, but this location can be set by hand using this option.

**Rebuild Proxy** Regenerates proxy images for all sizes set above and regenerate all timecodes which can be used later.

**Timecode** See *Timecode*.

**Proxy Render Size** Defines which proxy image resolution is used for display. If *Render Undistorted* is set, then images created from undistorted frames are used. If there are no generated proxies, render size is set to “No proxy, full render”, and if render undistorted is enabled, undistortion will happen automatically on frame refresh.

## Timecode

When you are working with footage directly copied from a camera without pre-processing it, there might be bunch of artifacts, mostly due to seeking a given frame in sequence. This happens because such footage usually does not have correct frame rate values in their headers. 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 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 Proxy/Timecode option in Blender.

The following timecodes are supported:

- No TC in use - do not use any timecode
- Record Run
- Free Run
- Free Run (rec date)
- Record Run No Gaps

---

**Note:** Record Run is the timecode which usually is best to use, but if the clip’s file is totally damaged, *Record Run No Gaps* will be the only chance of getting acceptable result.

---

## Footage Settings

See *Image Settings*.

## Main View

### Mini Timeline

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

## Animation

### 2.3.9 Dope Sheet

#### Introduction

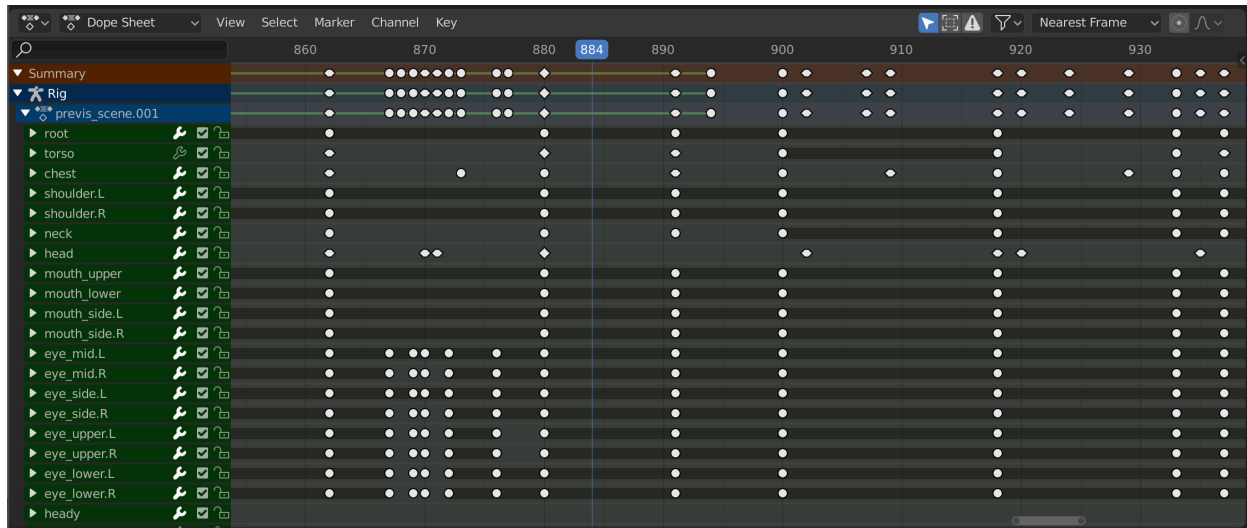


Fig. 433: The Dope Sheet.

The Dope Sheet gives the animator a birds-eye-view of the keyframes inside the scene.

The Dope Sheet is inspired by classical hand-drawn animation process, in which animators will make use of a chart, showing exactly when each drawing, sound and camera move will occur, and for how long. This is called an exposure sheet or 'dope sheet'. While CG foundations dramatically differ from classical hand-drawn animation, Blender's *Dope Sheet* inherits a similar directive.

#### Dope Sheet Modes

While the Dope Sheet Mode allows you to edit multiple actions at once, the other ones are dedicated to view and edit specific data-blocks used in different context of animation.

- Dope Sheet
- *Action Editor*
- *Shape Key Editor*
- *Grease Pencil*
- *Mask*
- Cache File: Alembic Todo 2.78.

#### Main Region

#### Navigation

As with most editors, you can:

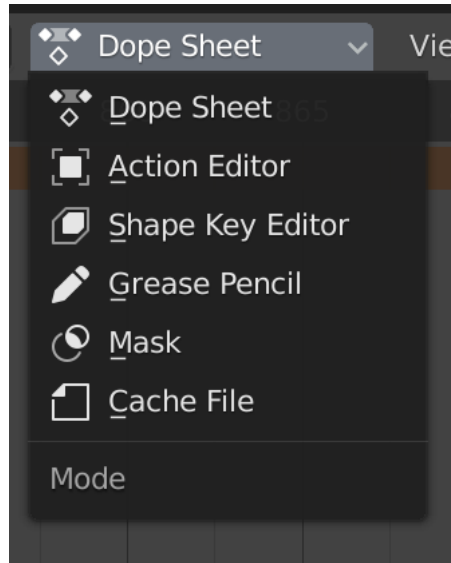


Fig. 434: Dope Sheet modes.

**Pan** Pan the view vertically (values) or horizontally (time) with click and drag (MMB).

**Zoom** Zoom in and out with the mouse wheel (Wheel).

**Scale View** Scale the view vertically or horizontally (Ctrl-MMB).

In addition, you can also use the scrollbars to pan and zoom the view.

## Keyframes

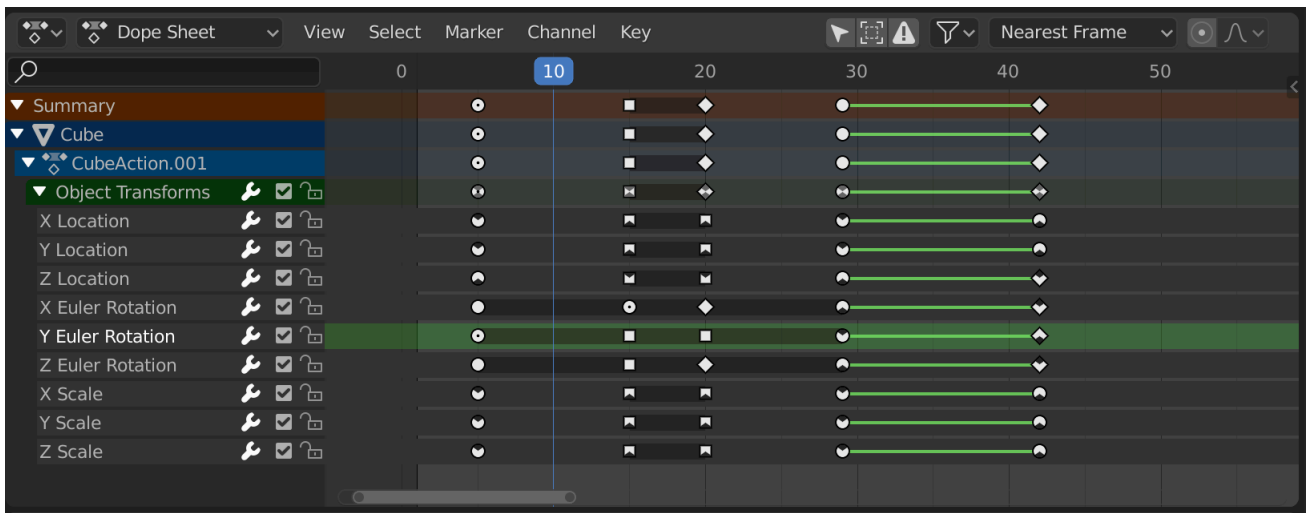


Fig. 435: The Dope Sheet Editor with object channels.

This area contains keyframes for all visible action channels. As with the other time-based editors, the X axis represents time. The Y axis represents a stack of action channels.

On these channels lay the keyframes, which can show different information:

Gray	Unselected
Yellow	Selected
Diamond	Free Keyframe Handle
Round	Auto-Clamped Keyframe Handle
Circle	Automatic Keyframe Handle
Square	Vector Keyframe Handle
Rhombus	Aligned Keyframe Handle
Various colors	These represent custom keyframe tags set by the user ( <i>Key</i> → <i>Keyframe Type</i> )
Gray bar between keys	Held key (the two keyframes are identical)
Green line between keys	Fixed keyframe interpolation (set in <i>Key</i> → <i>Interpolation Mode</i> )
Upwards arrow	Maximum Extreme keyframe (visible if <i>View</i> → <i>Show Curve Extremes</i> are enabled)
Downwards arrow	Minimum Extreme keyframe (visible if <i>View</i> → <i>Show Curve Extremes</i> are enabled)

## Selecting Keyframes

Selection tools are available in the Select menu in the header, and the main shortcuts are listed below:

**Selecting** Click on a key to select it. Hold Shift to extend the current selection.

**Box Selecting** Click and drag to box select multiple keyframes at once. You can hold Shift to extend or Ctrl to subtract from the current selection.

### Select/Deselect All

- To select all keys, press A.
- To deselect all keys, press Alt-A.
- To inverse the selection, press Ctrl-I.

**Before/After Current Frame** [, ] Select all to the right or left. Or hold Shift-Ctrl and click on either side of the Playhead.

See the Select menu for a full list of selection tools.

## Manipulating Keyframes

Keyframe tools are available in the Key menu in the header, and the main shortcuts listed below:

**Moving Keyframes** To move a single keyframe, click and drag on a key. To move multiple keyframes, make sure several keys are selected and press G.

**Scaling Keyframes** To scale (stretch) selected keys, press S.

**Extending Keyframes** To extend the time between two keys, select all with A, place the Playhead between two keyframes and press E.

See the Key menu for a full list of selection tools.

## Channels Region

See *Channels*.

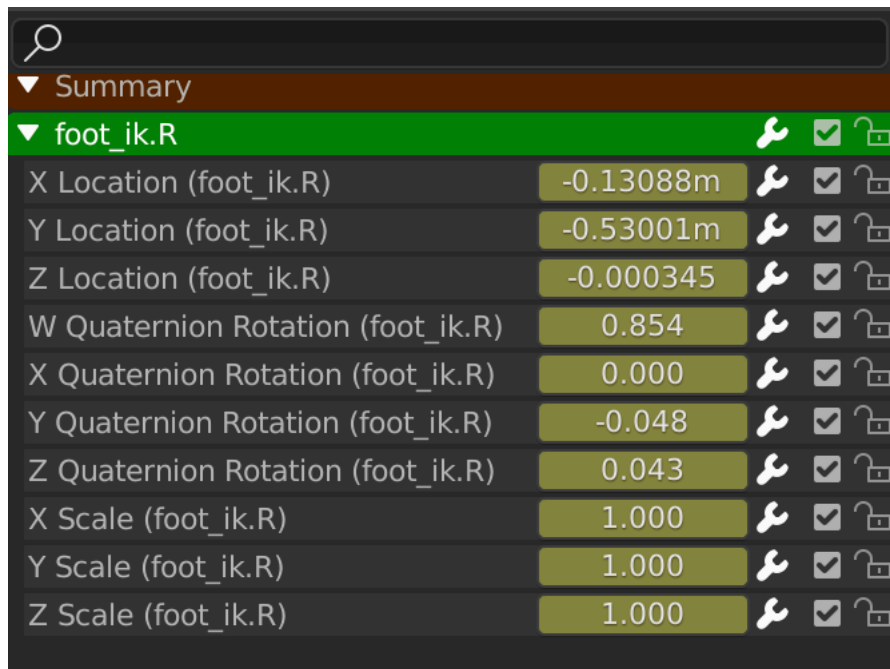


Fig. 436: The Action editor's channels region.

## Header

Here you find the menus, a first group of controls related to the editor "mode", a second one concerning the action data-blocks, and a few other tools (like the copy/paste buttons, and snapping type).

## View Menu



Fig. 437: Handle types.

**Show Handles and Interpolation** Instead of displaying all keyframes as diamonds, different icons are used to show the Bézier handle type. When curves use a different interpolation type, a line is shown between keys to highlight that.

See *Handles & Interpolation Display*.



Fig. 438: Extreme markers.

**Show Extremes** Detect keys where the curve changes direction based on comparing with the adjacent key values, and display that by changing the keyframe icons to resemble an arrow.

A muted version of the icon is used if the curve overshoots the extreme, or for groups with different results for contained curves.

See Graph editor's *View Menu*.

## Markers 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.

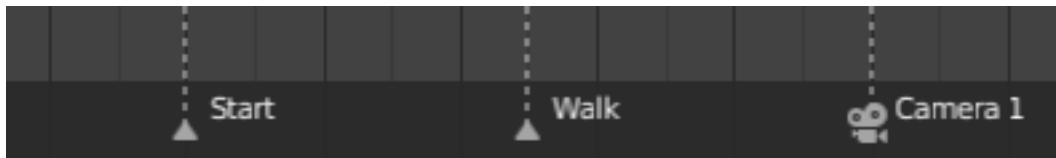


Fig. 439: Markers in animation editor.

There are some options that are exclusive to the Dope Sheet editor:

**Sync Markers** Sync Markers with keyframe edits.

**Show Pose Markers** Available in *Action* and *Shape Key* modes. Shows pose markers owned by the active action instead of the scene ones.

**Make Markers Local** Available in *Action* and *Shape Key* modes. Converts selected scene markers in pose markers, assigning them to the active action.

For more information and the description of the other marker tools, see *Editing Markers*.

## Key Menu

**Keyframe Type R** Sets the *Keyframe Types* of the selected keyframes.

See *F-Curve*.

## Filters

**Only Show Selected** Only include keyframes related to the selected objects and data.

**Show Hidden** Include keyframes from objects or bones that are not visible.

**Only Show Errors** Only include curves and drivers that are disabled or have errors. Useful for debugging.

**F-Curve Name Filter** Fuzzy/Multi-Word name filtering matches word snippets/partial words, instead of having to match everything. It breaks down the search text based on white-space placement. e.g. "lo ro" will filter all location and rotation, while "lc rt" will *not* work.

**Filter by Type** Filter curves by property type.

**Filtering Collection** Select a collection to only show keyframes from objects contained in that collection.

**Sort Data-Blocks** Objects data-blocks appear in alphabetical order, so that it is easier to find where they occur (as well as helping to keep the animation of related objects together in the NLA editor for instance).

If you find that your playback speed suffers from this being enabled (it should only really be an issue when working with lots of objects in the scene), you can turn this off.

**Summary** Toggles the "Dope Sheet Summary" channel at the top of the *Channels Region*. This is used to give an overview of all the channels by combining all the actions into one channel.

## Editing

### Control

### Auto Snapping

Todo.

### Proportional Editing

Todo.

## Modes

### Action Editor

The *Action Editor* is where you can define and control *actions*. It enables you to view and edit the F-curve data-blocks you defined as Actions in the *F-Curve editor*. So it takes place somewhere in between the low-level *F-Curves*, and the high-level *NLA editor*.

It gives you a slightly simplified view of the F-curve data-blocks (somewhat similar to F-curve shown without handles). The editor can list all Action data-blocks of an object at once.

Each Action data-block forms a top-level channel (see below). Note that an object can have several *Constraint* (one per animated constraint) and *Pose* (for armatures, one per animated bone) F-curve data-blocks, and hence an action can have several of these channels.

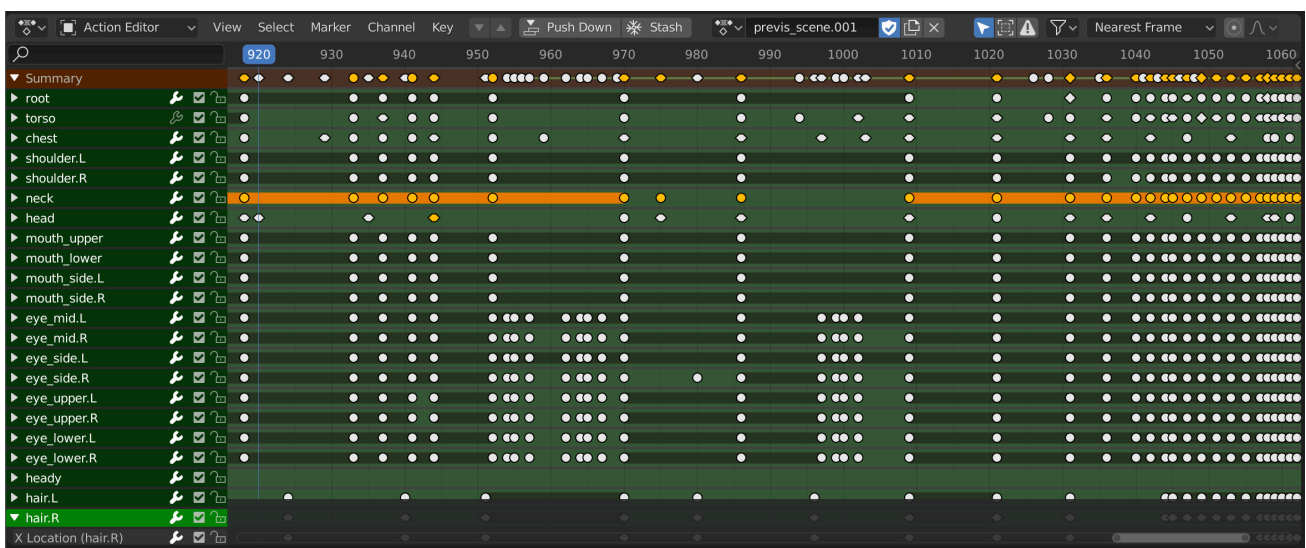


Fig. 440: The Action Editor.

## Header

**Layer Previous/Next (down/up arrow peak icon)** Switch between different actions stacked/stashed on top of each other in the NLA Stack, without having to go to the NLA Editor and leaving tweak mode and reentering it on the other strip.

Clicking on the up/down arrow buttons to go to the action in the NLA Track above/below the NLA Strip being whose action is being tweaked in the Action Editor.

If there are multiple actions/strips on the same layer/track, then only the one closest to the current frame will be used.

The operators will take into account the settings to view/edit the action in isolation (i.e. Solo and NLA Muting). This was done to make it easier to preview different stashed actions.

- If moving from a solo'd NLA Track to the active action, the NLA stack will be muted so that the action can be edited in isolation.
- Likewise, if the NLA stack is muted when editing the action, the NLA Track below it will be edited with solo enabled.
- If switching between NLA Tracks, the solo status for the previous track will be transferred to the new track.

---

**Note:** These still work when you're not editing the action used by an NLA Strip. If you're just animating a new action normally, it is possible to use the "down arrow" to temporarily jump down to the previous action without losing the new action you're working on, and then use the "up arrow" to get back to it once you're done checking the other action(s).

---

**Action** *A data-block menu.*

**Add +** When an action is created it is stored in an NLA Action Stash.

**Unlink X** When Shift-LMB clicking it clears the Fake User and removes the stashed action from the NLA stack too.

**Push Down (double down arrow peak icon)** Adds the active action on to the NLA stack as a contributing strip. This is basically the same as pressing the Push Down button in the NLA Editor.

**Stash (snowflake icon)** Stashes the active action on to the NLA stack. i.e. it is added as a non-contributing stack in the same way that it would if you were creating a new action instead.

---

**Note:** In both of these cases (Push Down and Stash), once the action has been added to the NLA stack, it is cleared/unassigned from the active action slot (i.e. it cannot be edited anymore from the Action/Graph Editors, unless you enter "Tweak Mode" on the corresponding strips later).

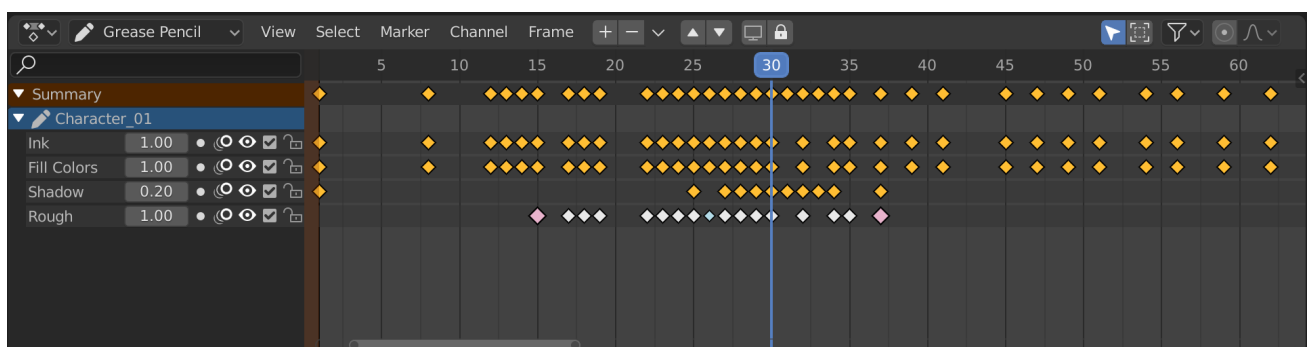
---

## Grease Pencil

This Mode allows you adjust the timing of the *Grease Pencil object's* animation frames. It is especially useful for animators blocking out shots, where the ability to re-time blocking is one of the main purposes of the whole exercise.

This mode can be accessed by changing the Dope Sheet editor's *Mode* selector (found in the header to the far left) to Grease Pencil.

To use this editor mode, make sure you have a *Grease Pencil object* selected.





## Channels Region

**Grease Pencil (light blue)** The channels' region shows the Grease Pencil data-blocks containing the layers. Multiple blocks are used for each area (e.g. one for the 3D Viewport and the Image editor).

**Layers (gray)** These channels contain the keyframes to which the *layers* are bound.

**Opacity** Controls the layers *Opacity*.

**Mask (mask icon)** Toggle the *Masks* visibility.

**Onion Skinning (onion skin icon)** Toggle the use the layer for *Onion Skinning*.

**Viewport/Render Visibility (eye icon)** Toggle layer visibility in the viewport and in render.

**Lock Animation (checkbox icon)** Toggles playback of animation or keep the channel static.

**Lock (padlock icon)** Toggle layer from being editable.

## Header

**Layer Move** Moves the selected layer/channel up or down in the evaluation stack.

**Layer Add/Remove** Adds/removes Grease Pencil layers/channels.

## Insert Keyframe

Insert Keyframe I can be used for creating blank Grease Pencil frames at a particular frame. It will create blank frames if *Additive Drawing* is disabled, otherwise it will make a copy of the active frame on that layer, and use that.

## 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 View

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*.

## Shape Key

### Shape Key Editor

The *Shape Key Editor* is used to adjust the animation timing of *shape keys*. These are stored inside an Action data-block. It lets you edit the *value* of shape keys over time.

This mode of the *Dope Sheet* uses a similar interface as the *Action Editor* Mode with the distinction of the absence of header filter controls and tools for channels.

## Mask

In the Dope Sheet's *Mask Mode* mask shape keyframes can be selected and edited. All Mask data-blocks in the blend-file are shown.

See *Masking*.

## 2.3.10 Timeline

The *Timeline* editor, identified by a clock icon, is used for manipulating keyframes and scrubbing the playhead.

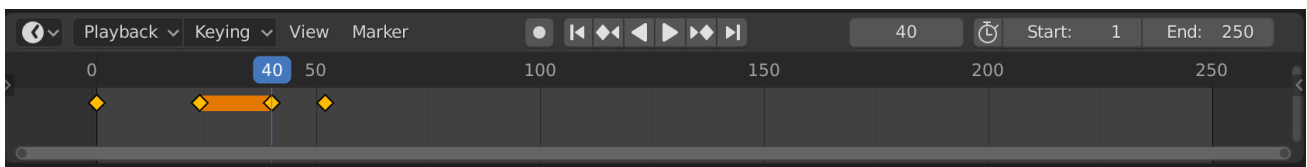


Fig. 441: The Timeline.

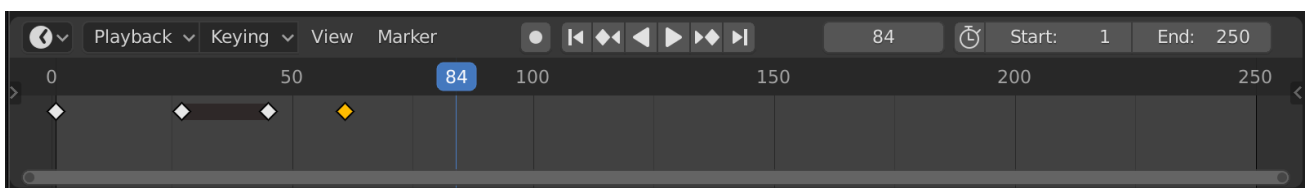
The *Timeline* gives the user a broad overview of a scene's animation, by showing the current frame, the keyframes of the active object, the start and end frames of your animation sequence, as well as markers set by the user.

The *Timeline* includes *Transport Controls*, to play, pause, and skip through an animation sequence.

It also includes tools for manipulating *Keyframes*, *Keying Sets*, and *Markers*.

## Main View

The main *Timeline* region displays the animation frames over time.



Here you can see the *Keyframes* (diamond shapes), *Playhead* (blue handle), *Scrollbar* (along the bottom).

## Adjusting the View

The *Timeline* can be panned by holding MMB, then dragging the area left or right.

You can zoom the *Timeline* by using Ctrl-MMB, the mouse Wheel, or pressing NumpadMinus and NumpadPlus.

You can also use the scrollbars, located at the bottom or the right of the editor, to pan and zoom the view.

## Playhead

The *Playhead* is the blue vertical line with the current frame number at the top.

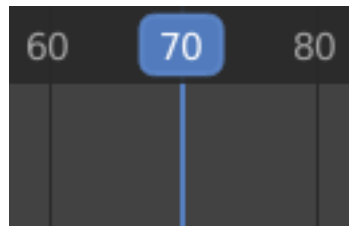


Fig. 442: Playhead.

The *Playhead* can be set or moved to a new position by pressing or holding LMB in scrubbing area at the top of the timeline.

The *Playhead* can be moved in single-frame increments by pressing Left or Right, or you can jump to the beginning or end frame by pressing Shift-Left or Shift-Right.

## Frame Range

By default, the *Frame Range* is set to start at frame 1 and end at frame 250. You can change the frame range in the top right of the Timeline header, or in the Output Properties.

## Keyframes

For the active and selected objects, keyframes are displayed as diamond shapes.

You can click to select one at a time, or select several by holding Shift, or by dragging a box around the keyframes. You can then move single keys by dragging them, and you can move multiple keys by pressing G and scale them with S.

*Only Selected Channels* can be enabled. *Timeline* → *View* → *Only Selected Channels*. For *Armatures*, this will display the object keyframes, and the keyframes for the active and selected pose bones.

## Markers

See the *Markers page* for more information.

## Header

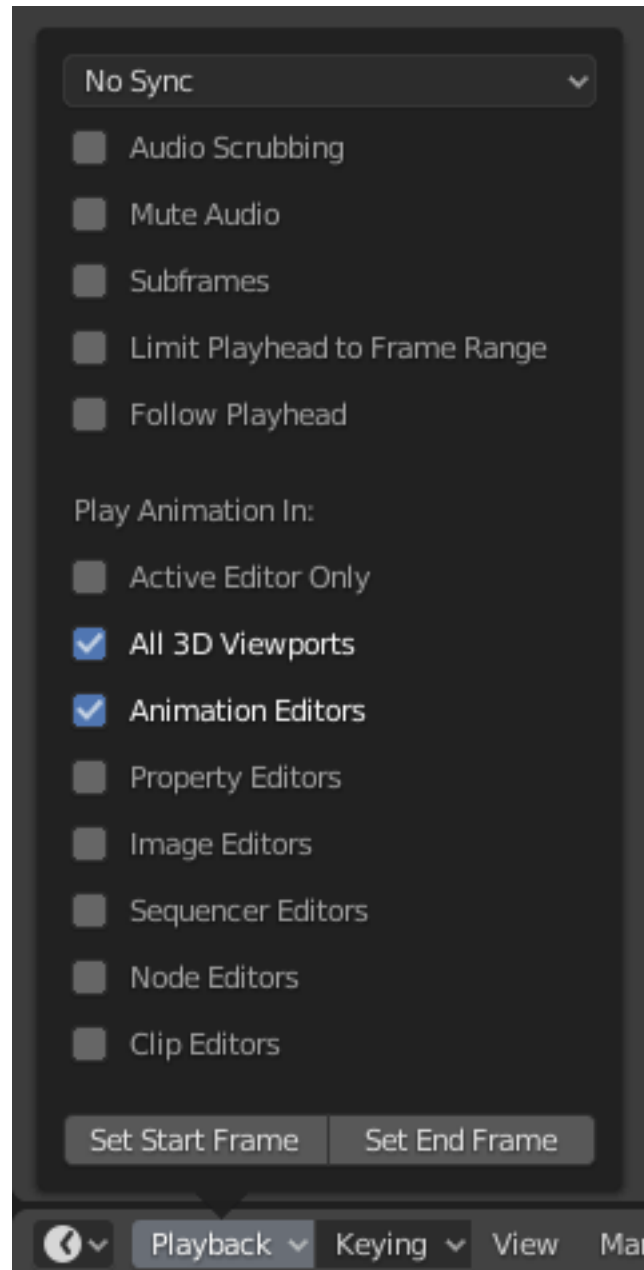
## Popovers

### Playback Popover

The *Playback* popover contains options controlling the animation playback.

### Audio

#### Sync Mode



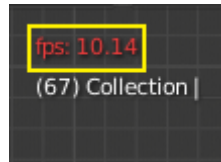


Fig. 443: 3D Viewport red FPS.

When you play an animation, the frame rate is displayed at the top left of the 3D Viewport. If the scene is detailed and playback is slower than the set *Frame Rate* (see *Dimensions Panel*), these options are used to synchronize the playback.

**No Sync** Do not sync, play every frame.

**Frame Dropping** Drop frames if playback is too slow.

**AV-sync** (Audio/Video Synchronization). Sync to audio clock, dropping frames if playback is slow.

**Scrubbing** If your animation has sound, this option plays bits of the sound wave while you move the playhead with LMB or keyboard arrows (like a moving playhead).

**Mute** Mute the sound from any audio source.

### Playback

**Limit Playback to Frame Range** Don't allow selecting frames outside of the playback range using the mouse.

**Follow Current Frame** Animation editors can be setup to always follow the time indicator as animation is being played back. Following will be done when animating and changing frame: When the cursor reaches the end of the screen, the next range of frames of the same width will be displayed.

### Play In

**Active Editor** While playing, updates the Timeline, if *Animation Editors* and *All 3D Viewports* disabled.

**3D Viewport** While playing, updates the 3D Viewport and the Timeline.

**Animation Editors** While playing, updates the Timeline, Dope Sheet, Graph Editor, Video Sequencer.

**Image Editor** The Image editor in Mask mode.

**Properties Editor** When the animation is playing, this will update the property values in the UI.

**Movie Clip Editor** While playing, updates the Movie Clip Editor.

**Node Editors** While playing, updates the Node properties for the node editors.

**Video Sequencer** While playing, updates the Video Sequencer.

### Show

**Subframes** Display and allow changing the current scene subframe.

### Keying Popover

The *Keying* popover contains options that affect keyframe insertion.

### Active Keying Set

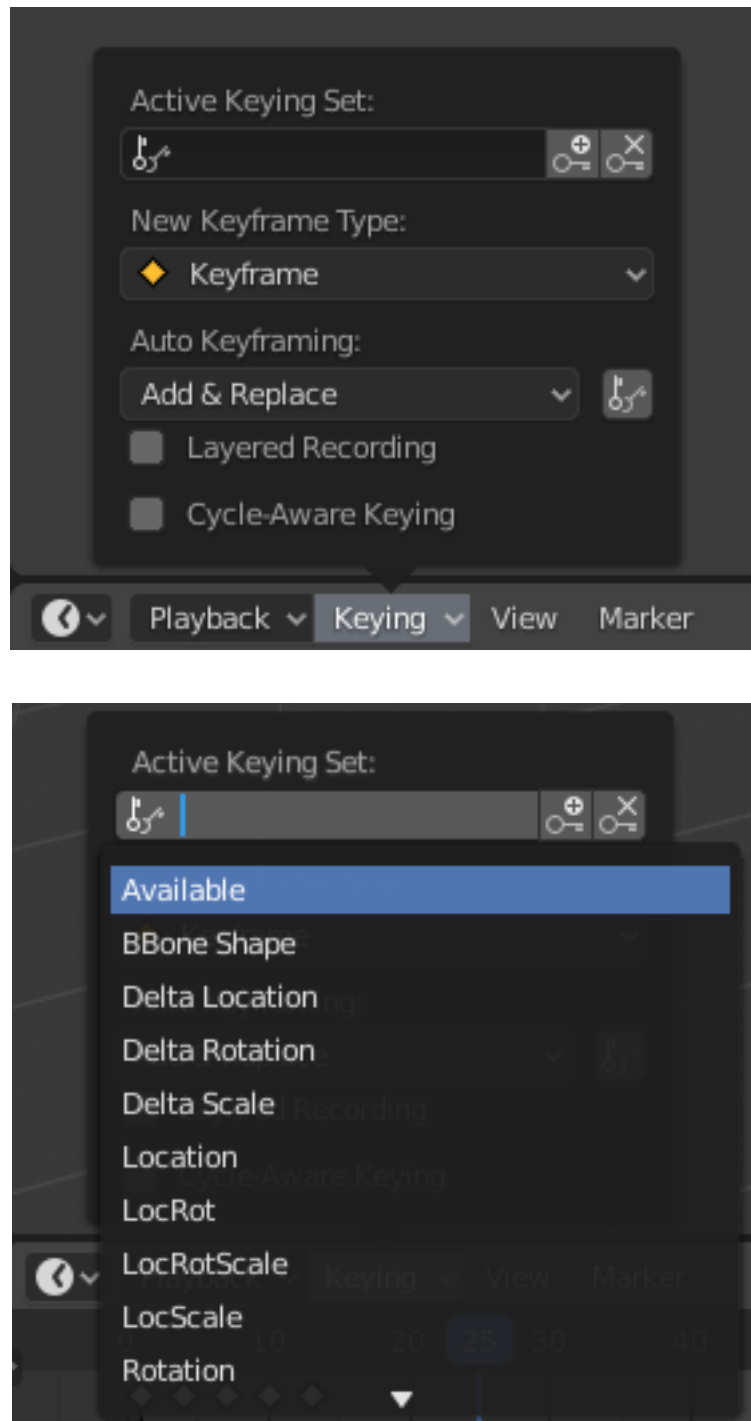


Fig. 444: Timeline Keying Sets.

*Keying Sets* are a set of keyframe channels in one. They are made so the user can record multiple properties at the same time. With a keying set selected, when you insert a keyframe, Blender will add keyframes for the properties in the active *Keying Set*. There are some built-in keying sets, *LocRotScale*, and also custom keying sets. Custom keying sets can be defined in the panels *Properties* → *Scene* → *Keying Sets* + *Active Keying Set*.

**Insert Keyframes (plus icon)** Insert keyframes on the current frame for the properties in the active *Keying Set*.

**Delete Keyframes (minus icon)** Delete keyframes on the current frame for the properties in the active *Keying Set*.

**New Keyframe Type** *Keyframe Types* on insertion.

## Auto Keying Popover

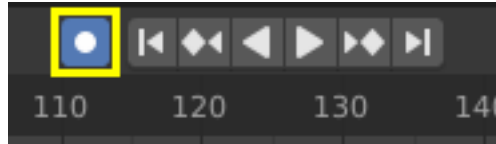



Fig. 445: Timeline Auto Keyframe.

The record button () enables *Auto Keyframe*: It will add and/or replace existing keyframes for the active object when you transform it in the 3D Viewport. For example, when enabled, first set the Playhead to the desired frame, then move an object in the 3D Viewport, or set a new value for a property in the UI.

When you set a new value for the properties, Blender will add keyframes on the current frame for the transform properties. Other use cases are *Fly/Walk Navigation* to record the walk/flight path and *Lock Camera to View* to record the navigation in camera view.

---

**Note:** Note that *Auto Keyframe* only works for transform properties (objects and bones), in the 3D Viewport (i.e. you can't use it e.g. to animate the colors of a material in the Properties...).

---

**Add & Replace / Replace** This controls how the auto keyframe mode works. Only one mode can be used at a time.

**Add & Replace** Add or replace existing keyframes.

**Replace** Only replace existing keyframes.

**Only Active Keying Set** When enabled, new keyframes for properties will be inserted into the active *Keying Set*.

**Layered Recording** Adds a new NLA Track and strip for every loop/pass made over the animation to allow non-destructive tweaking.

**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 appropriately.

## Menus

### View Menu

The *View Menu* controls what you see, and what it looks like.

**Show Seconds Ctrl-T** Whether to show the time in the X axis and the *Playhead* as frames (based on the FPS) or as seconds.

**Sync Visible Range** It synchronizes the horizontal panning and scale of the current editor with the other editors (Graph, Dope Sheet, NLA and Sequencer) when this option is set. That way you always have these editors showing the same section of frames.

**Show Markers** Shows the markers region. When disabled, the *Markers Menu* is also hidden and markers operators are not available in this editor.

**Only Keyframes from Selected Channels** For *Armatures*, this will display the object keyframes, and the keyframes for the active and selected pose bones.

## Cache

**Show Cache** Show all enabled types.

Soft Body, Particles, Cloth, Smoke, Dynamic Paint, Rigid Body.

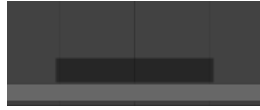


Fig. 446: Timeline Cache.

**Frame All Home** Maximize the area based on the Animation Range.

**Go to Current Frame Numpad0** Centers the Timeline to the Playhead.

## Markers 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.

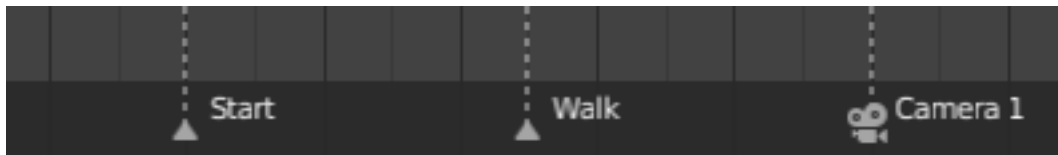


Fig. 447: Markers in animation editor.

For descriptions of the different marker tools see *Editing Markers*.

## Header Controls

The Timeline header controls.



Fig. 448: Timeline header controls.

1. Popovers for Playback and Keying, 2. Transport Controls, 3. Frame Controls

## Transport Controls

These buttons are used to set, play, rewind, the *Playhead*.

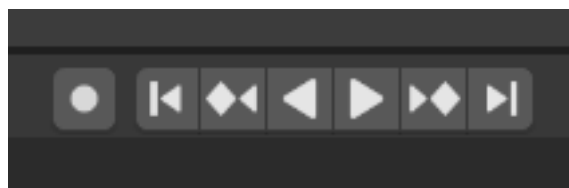


Fig. 449: Transport controls.



**Jump to Start (⏪) Shift-Left** This sets the cursor to the start of frame range.

**Jump to Previous Keyframe (⏮) Down** This sets the cursor to the previous keyframe.

**Rewind (↶) Shift-Ctrl-Spacebar** This plays the animation sequence in reverse. When playing the play buttons switch to a pause button.

**Play (▶) Spacebar** This plays the animation sequence. When playing the play buttons switch to a pause button.

**Jump to Next Keyframe (⏭) Up** This sets the cursor to the next keyframe.

**Jump to End (⏩) Shift-Right** This sets the cursor to the end of frame range.

**Pause (⏸) Spacebar** This stops the animation.

## Frame Controls

**Current Frame Alt-Wheel** The current frame of the animation/playback range. Also the position of the *Playhead*.

**Preview Range (clock icon)** This is a temporary frame range used for previewing a smaller part of the full range. The preview range only affects the viewport, not the rendered output. See *Preview Range*.

**Start Frame** The start frame of the animation/playback range.

**End Frame** The end frame of the animation/playback range.

## 2.3.11 Graph Editor

### Introduction

The Graph Editor allows users to adjust animation curves over time for any animatable property. *F-Curves*.

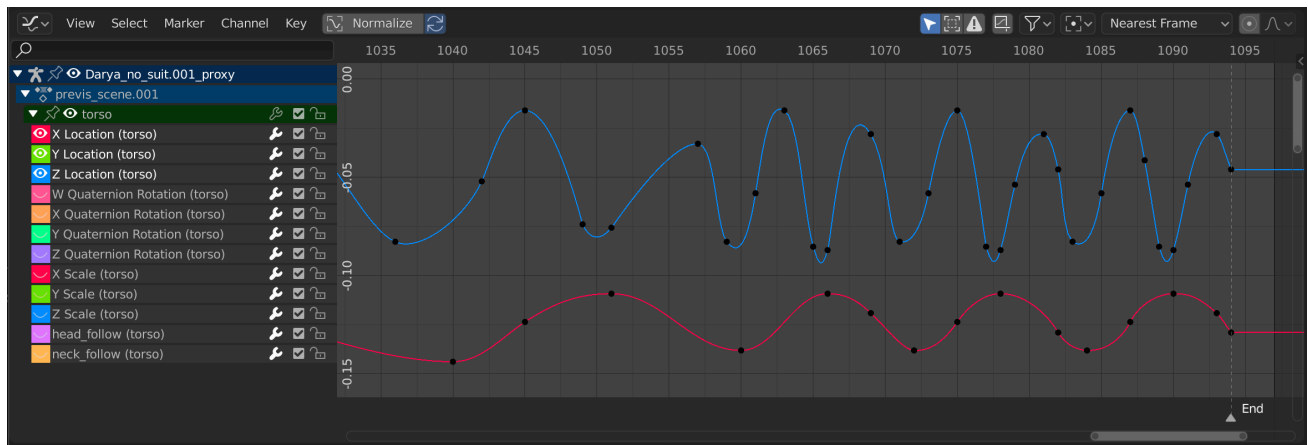


Fig. 450: The Graph Editor.

### Main Region

The curve view allows you to view and edit F-curves. An F-curve has several key parts:

**Curve** The curve defines the value (Y axis) of the property over time (X axis).

See *F-Curves*.

**Keyframes** Keyframes are user-defined values on certain frames and are represented by little black squares which become orange if selected.

See *Keyframes* for more information.

**Handles** Each keyframe has a handle that helps determine the values of the curve between keyframes. These handles are represented by extruding lines with circular ends and can be selected and modified to change the shape of the curve.

See *Handle Type* for more information.

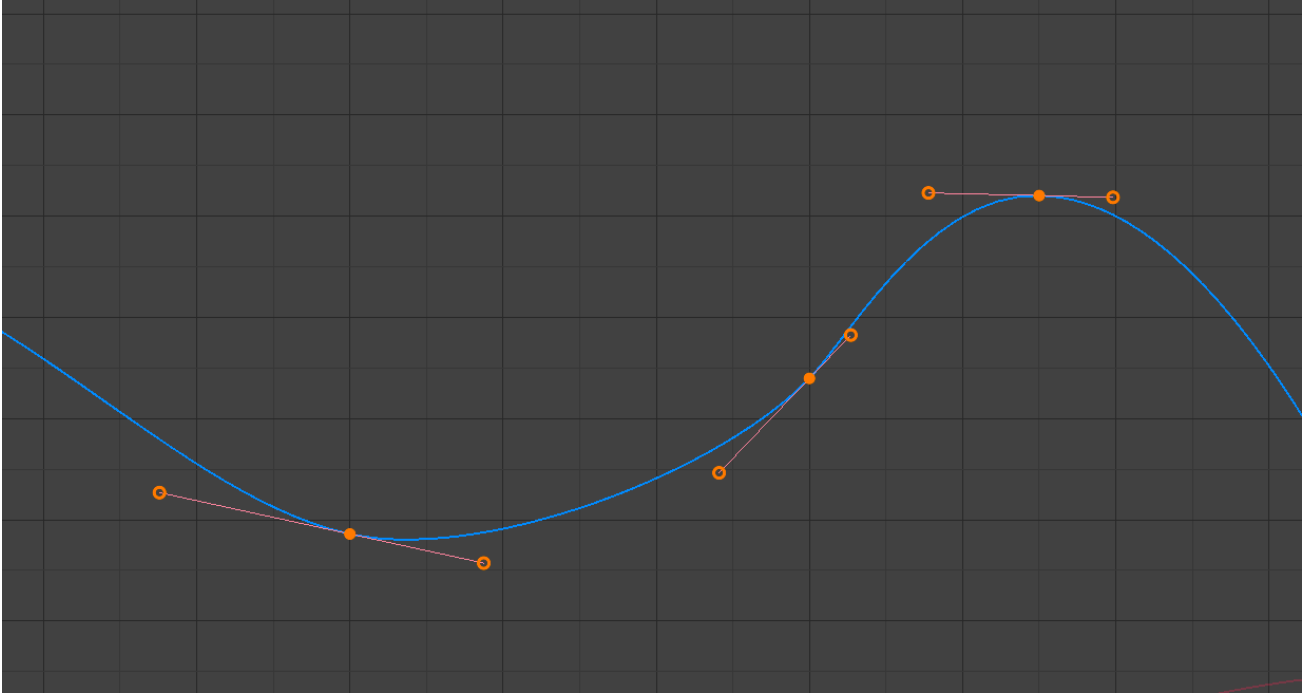


Fig. 451: A simple curve.

**See also:**

See *F-Curves* for more info.

### Navigation

As with most editors, you can:

**Pan** Pan the view vertically (values) or horizontally (time) with click and drag MMB.

**Zoom** Zoom in and out with the mouse wheel Wheel.

**Scale View** Scale the view vertically or horizontally Ctrl-MMB.

In addition, you can also use the scrollbars to pan and zoom the view.

## Playhead & 2D Cursor

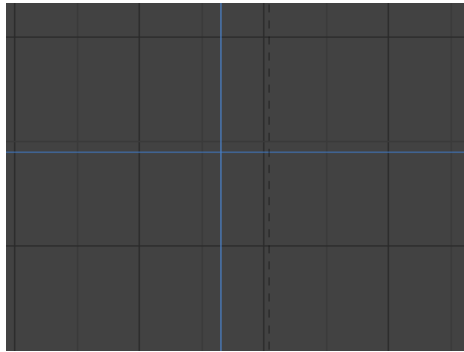


Fig. 452: Graph Editor 2D Cursor.

The current frame is represented by a blue vertical line called the *Playhead*.

As in the *Timeline*, you can change the current frame by LMB-dragging in the scrubbing area at the top of the editor.

The blue horizontal line is called the *2D Cursor*. This can be enabled or disabled via the *View Menu* or the *View Properties* panel.

These two lines can be used as a reference for moving and scaling keyframe handles.

### See also:

See Graph Editor's *View Tab*.

## View Axes

For *Actions* the X axis represents time, the Y axis represents the value to set the property.

Depending on the selected curves, the values have different meaning: for example rotation properties are shown in degrees.

## Header

### View Menu

**Realtime Updates** When transforming keyframes, changes to the animation data are propagated to other views.

**Show Cursor** Toggles the visibility of the *Playhead & 2D Cursor*.

**Show Sliders** A toggle option that shows the value sliders for the channels. See the Fig. *The Action editor's channels region*..

**Show Group Colors** Display groups and channels with colors matching their corresponding groups.

**AutoMerge Keyframes** Automatically merge nearby keyframes.

**Show Markers** Shows the markers region. When disabled, the *Markers Menu* is also hidden and markers operators are not available in this editor.

**Use High Quality Display** Display F-curves using *Anti-Aliasing* and other fancy effects (disable for a better performance).

**Show Handles Ctrl-H** Toggles the display of a curve's handles in the curve view.

**Only Selected Curve Keyframes** Only shows the keyframes markers on the selected curves.

**Only Selected Keyframes Handles** Only shows the handles for the currently selected curves.

**Frame All Home** Reset viewable area to show all keyframes.

**Frame Selected NumpadPeriod** Reset viewable area to show selected keyframes.

**Go to Current Frame Numpad0** Centers the area to the Playhead.

**See also:**

- See Graph Editor's *View Tab*.
- See Timeline's *View Menu*.

## Preview Range

**Set Preview Range P** Interactively define frame range used for playback. Allows you to define a temporary preview range to use for animation playback (this is the same thing as the *Playback Range* option of the *Timeline editor header*).

**Clear Preview Range Alt-P** Clears the preview range.

**Auto-Set Preview Range Ctrl-Alt-P** Automatically sets the preview range to playback the whole action or the selected NLA strips.

## Markers 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.

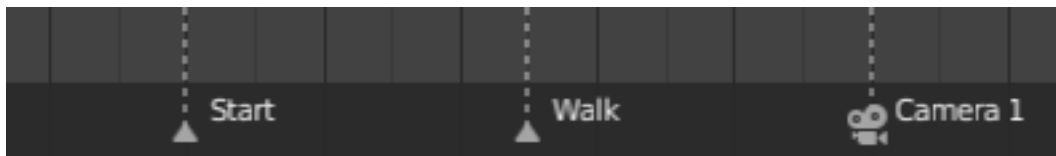


Fig. 453: Markers in animation editor.

For descriptions of the different marker tools see *Editing Markers*.

## View Controls



Fig. 454: View controls.

**Show Only Selected** Only include curves related to the selected objects and data.

**Show Hidden** Include curves from objects/bones that are not visible.

**Show Only Errors** Only include curves and drivers that are disabled or have errors. Useful for debugging.

**Filter (funnel icon)** Only include curves with keywords contained in the search field.

**Multi-Word** Fuzzy/Multi-Word name filtering matches word snippets/partial words, instead of having to match everything. It breaks down the search text based on white-space placement. e.g. “lo ro” will filter all location and rotation, while “lc rt” will *not* work.

**Type Filter** Filter curves by property type.

**Filtering Collection** Select a collection to only show keyframes from objects contained in that collection.

**Sort Data-Blocks** Objects data-blocks appear in alphabetical order, so that it is easier to find where they occur (as well as helping to keep the animation of related objects together in the NLA for instance).

If you find that your playback speed suffers from this being enabled (it should only really be an issue when working with lots of objects in the scene), you can turn this off.

**Normalize** Normalize curves so the maximum or minimum point equals 1.0 or -1.0.

**Auto** Automatically recalculate curve normalization on every curve edit. This is useful to prevent curves from jumping after tweaking it.

## F-Curve Controls

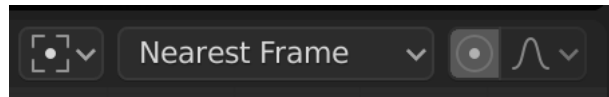


Fig. 455: F-Curve controls.

**Proportional Editing 0** See *Proportional Editing*.

**Auto Snap** Auto snap the keyframes for transformations.

- No Auto-Snap
- Frame Step
- Second Step
- Nearest Frame
- Nearest Second
- Nearest Marker

**Pivot Point** Pivot point for rotation.

**Bounding Box Center** Center of the selected keyframes.

**2D Cursor** Center of the *2D Cursor*. *Playhead + Cursor*.

**Individual Centers** Rotate the selected keyframe *Bézier* handles.

**Create Ghost Curves (framed F-curve icon)** Creates a picture with the current shape of the curves.

## Sidebar Region

The panels in the *Sidebar region*.

## View Tab

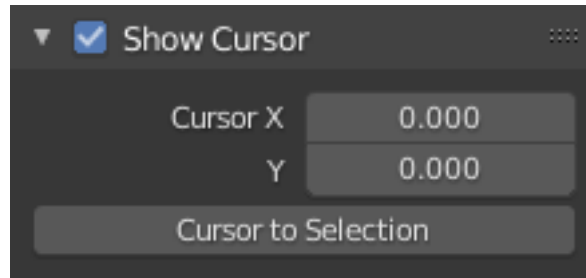


Fig. 456: View Tab.

**Show Cursor** Toggles the visibility of the *2D Cursor*.

**Cursor X, Y** Moves the cursor to the specified frame (X value) and value (Y value).

**Cursor to Selection** Places the *2D Cursor* at the midpoint of the selected keyframes.

### See also:

Graph Editor's *View Menu*.

## Further Tabs

**F-Curve Tab** See *F-Curve*.

**Modifiers Tab** See *F-Curve Modifiers*.

## Channels

### Channels Region

The channels region is used to select and manage the curves for the Graph editor. This part shows the objects and their animation data hierarchy each as headers. Each level can be expanded/collapsed by the small arrow to the left of its header.

- Scenes, Objects (dark blue)
- *Actions, Shape keys, etc.* (light blue)
- Groups (green)
- Channels (gray)

**Name Filter** Only displays channels that match the search text. Pressing the invert button displays all channels except the channels that match the search text.

### Controls

On the headers, there are toggles to control channel's setting:

**Pin (pin icon)** Make the channel always visible regardless of the current selection (Graph editor only).

**Hide (eye icon)** Hides the channel(s)/curve (Graph editor only).

**Modifiers (wrench icon)** Deactivates the F-curve modifiers of the selected curve or all curves in the channel.

**Mute (speaker icon)** Deactivates the channel/curve.



Fig. 457: The Channels region.

**Lock (padlock icon) Tab** Toggle channel/curve from being editable. Selected channels can be locked by pressing Tab.

---

**Note:** In the Dope Sheet this is also working inside the NLA, but that it does not prevent edition of the underlying F-curve.

---

## Selecting

- Select channel (text in white/black): LMB
- Multi Select/Deselect: Shift-LMB
- Select All: A
- Deselect All: Alt-A
- Box Select: (LMB drag) or B (LMB drag)
- Box Deselect: (Ctrl-LMB drag) or B (Shift-LMB drag)
- Select all keyframes in the channel: double LMB on a channel header.

## Editing

- Rename: Ctrl-LMB
- Delete selected: X or Delete
- Lock selected: Tab

- Enable Channel Setting: Shift-Ctrl-W
- Disable Channel Setting: Alt-W
- Toggle Channel Setting: Shift-W

## Sliders

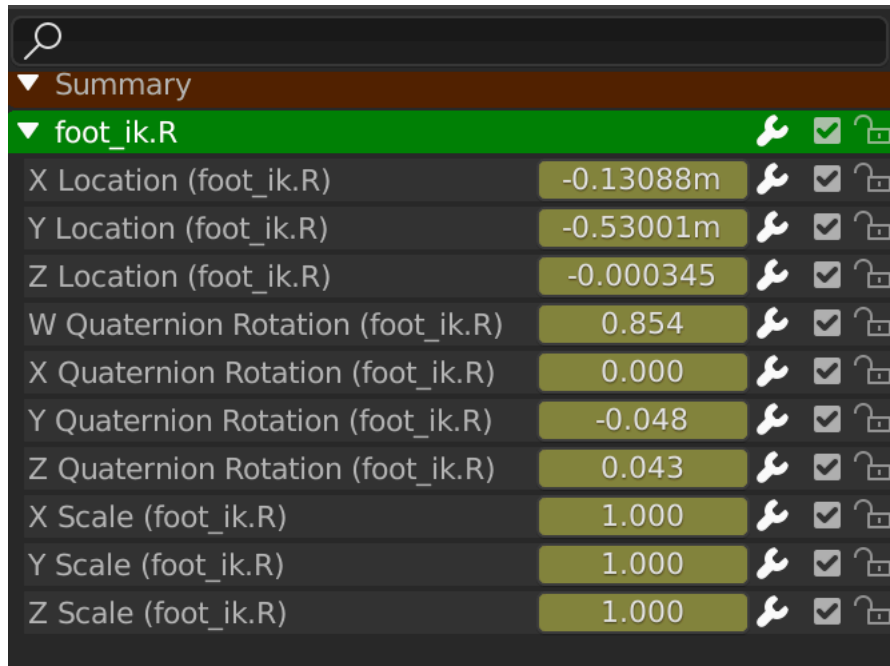


Fig. 458: The Action editor showing sliders.

On channels headers you can have another column with number fields or sliders, allowing you to change the value on the current keyframes, or to add new keyframes. See *View Menu* for how to show these sliders.

## Menu

**Delete Channels X** Deletes the whole channel from the current action (i.e. unlink the underlying F-curve data-block from this action data-block).

**Warning:** The X shortcut is area-dependent: if you use it in the left list part, it will delete the selected channels, whereas if you use it in the main area, it will delete the selected keyframes.

**Un/Group Channels Ctrl-Alt-G, Ctrl-G** Un/Groups the selected channels into a collection that can be renamed by double clicking on the group name. For example, this helps to group channels that relate a part of an armature to keep the editor more organized.

**Settings Toggle/Enable/Disable, Shift-W, Shift-Ctrl-W, Alt-W** Enable/disable a channel's setting (selected in the menu that pops up).

Protect, Mute

**Toggle Channel Editability Tab** Locks or unlocks a channel for editing.

**Extrapolation Mode Shift-E** Change the *extrapolation* between selected keyframes.



**Expand Channels, Collapse Channels** `NumpadPlus`, `NumpadMinus` Expands or collapses selected channels.

**Hide Selected Curves** `H` Hides the selected curves.

**Hide Unselected** `Shift-H` Show only the selected curve (and hide everything else).

**Reveal Curves** `Alt-H` Show all previous hidden curves.

**Move...** This allows you to move selected channels up/down `PageUp`, `PageDown`, or directly to the top/bottom `Shift-PageUp`, `Shift-PageDown`.

**Revive Disabled F-Curves** Clears “disabled” tag from all F-curves to get broken F-curves working again.

## F-Curves

### Introduction

After animating some property in Blender using keyframes you can edit their corresponding curves. When something is “animated”, it changes over time. This curve is shown as something called an F-curve. Basically what an F-curve does is an interpolation between two animated properties. In Blender, animating an object means changing one of its properties, such as the object’s location, or its scale.

As mentioned, Blender’s fundamental unit of time is the “frame”, which usually lasts just a fraction of a second, depending on the *frame rate* of the scene. As animation is composed of incremental changes spanning multiple frames, usually these properties are **not** manually modified *frame-by-frame*, because:

- It would take ages!
- It would be very difficult to get smooth variations of the property (unless you compute mathematical functions and type a precise value for each frame, which would be crazy).

This is why nearly all direct animation is done using *interpolation*.

The idea is simple: you define a few Keyframes, which are multiple frames apart. Between these keyframes, the properties’ values are computed (interpolated) by Blender and filled in. Thus, the animators’ workload is significantly reduced.

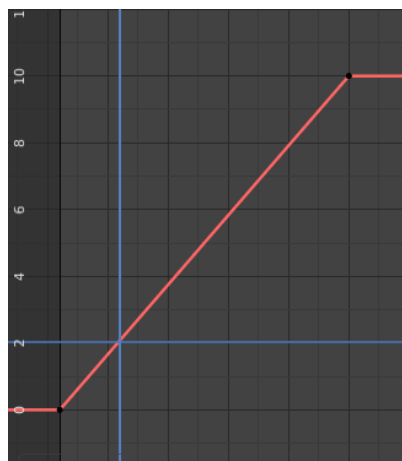


Fig. 459: Example of an interpolation.

For example, if you have:

- A control point of value 0 at frame 0,
- another one of value 10 at frame 25,

- and you use linear interpolation, then, at frame 5 we get a value of 2.

The same goes for all intermediate frames: with just two points, you get a smooth increase from (0 to 10) along the 25 frames. Obviously, if you would like the frame 15 to have a value of 9, you would have to add another control point (or keyframe)...

## Direction of Time

Although F-curves are very similar to *Bézier*, there are some important differences.

For obvious reasons, a property represented by a curve cannot have more than **one** value at a given time, hence:

- When you move a control point ahead of a control point that was previously ahead of the point that you are moving, the two control points switch their order in the edited curve, to avoid the curve going back in time.
- For the above reason, it is impossible to have a closed F-curve.

Table 15: Two control points switching: the curve cannot go back in time!

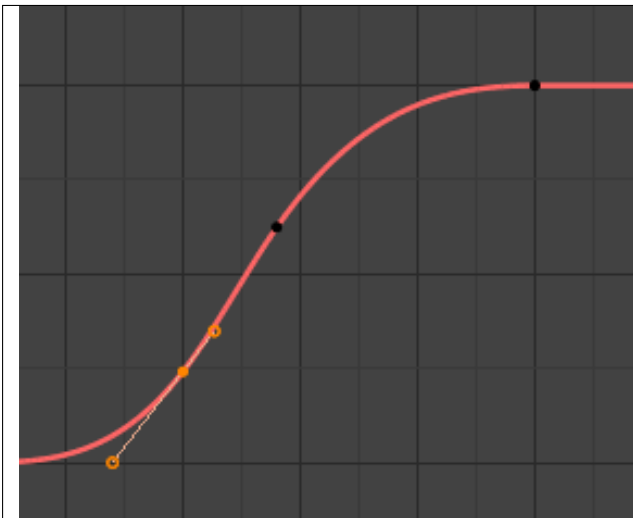


Fig. 460: Before moving the second keyframe.

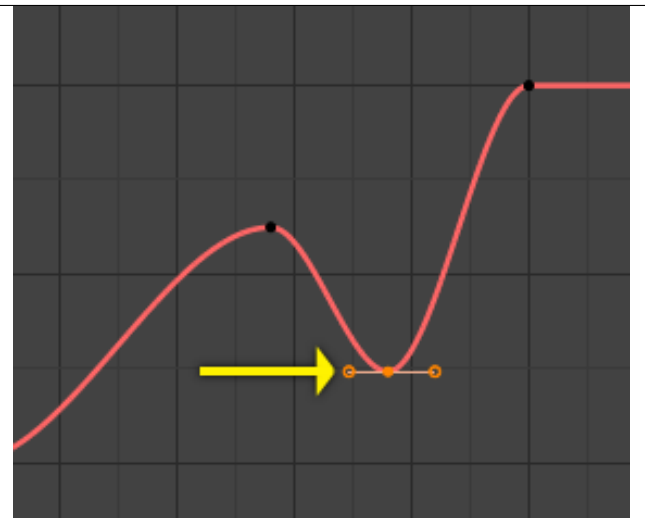


Fig. 461: After moving the second keyframe.

## Settings

F-Curves have three additional properties, which control the interpolation between points, extension behavior, and the type of handles.

## Interpolation Mode

---

### Reference

**Menu** *Key* → *Interpolation Mode*

**Hotkey** T

---

Mode for the *Interpolation* between the current and next keyframe.

## Interpolation

**Constant** There is no interpolation at all. The curve holds the value of its last keyframe, giving a discrete (stairway) “curve”. Usually only used during the initial “blocking” stage in pose-to-pose animation workflows.



Fig. 462: Constant.

**Linear** This simple interpolation creates a straight segment, giving a non-continuous line. It can be useful when using only two keyframes and the *Extrapolation* extend mode, to easily get an infinite straight line (i.e. a linear curve).

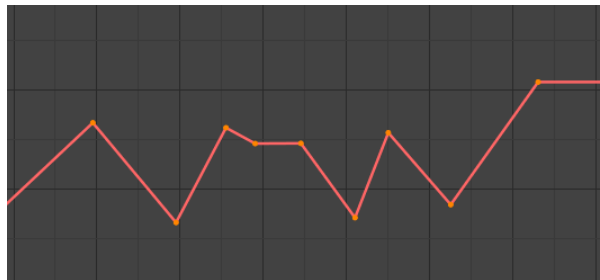


Fig. 463: Linear.

**Bézier** The more powerful and useful interpolation, and the default one. It gives nicely smoothed curves, i.e. smooth animations!

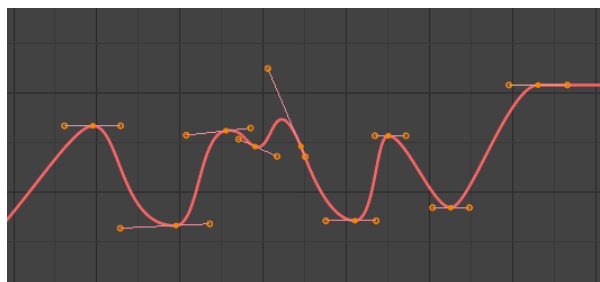


Fig. 464: Bézier.

---

**Note:** Remember that some F-curves can only take discrete values, in which case they are always shown as if constant interpolated, whatever option you chose.

---

## Easing (by strength)

Different methods of easing interpolations for F-curve segment. The “Robert Penner easing equations” (basically, equations which define some preset ways that one keyframe transitions to an

other) which reduce the amount of manual work (inserting and tweaking keyframes) to achieve certain common effects. For example, snappy movements.

- Linear
- Sinusoidal
- Quadratic
- Cubic
- Quartic
- Quintic
- Exponential
- Circular

**See also:**

For more info and a few live demos, see <https://easings.net> and <http://www.robertpenner.com/easing/>

## Dynamic Effects

These additional easing types imitate (fake) physics-based effects like bouncing/springing effects. The corresponding settings can be found in the *Sidebar region* → *Active Keyframe panel*.

**Elastic** Exponentially decaying sine wave, like an elastic band. This is like bending a stiff pole stuck to some surface, and watching it rebound and settle back to its original state.

**Amplitude** The amplitude property controls how strongly the oscillation diverges from the basic curve. At 0.0, there is no oscillation (i.e. it just snaps to the B-value like an extreme exponential transition), and at 1.0 a profile similar to the one shown in the icon occurs.

**Period** The period property controls the frequency with which oscillations occur. Higher values result in denser oscillations.

**Bounce** Exponentially decaying parabolic bounce, like when objects collide. e.g. for Bouncing balls, etc.

**Back** Cubic easing with overshoot and settle. Use this one when you want a bit of an overshoot coming into the next keyframe, or perhaps for some wind-up anticipation.

**Back** The back property controls the size and direction (i.e. above/below the curve) of the overshoot.

## Easing Type

---

### Reference

**Menu** *Key* → *Easing Type*

**Hotkey** Ctrl-E

---

The Easing Type controls which end of the segment between the two keyframes that the easing effects apply to.

**Automatic Easing** The most commonly expected of the below behaviors is used. For the transitional effects, this is basically *ease in*, while for the physics effects it is *ease out*.

**Ease In** Effect builds up to the second keyframe.

**Ease Out** Effect fades out from the first keyframe.

**Ease In Out** Effect occurs on both ends of the segment.

## Extrapolation Mode

### Reference

**Menu** *Channel* → *Extrapolation Mode*

**Hotkey** Shift-E

Extrapolation defines the behavior of a curve before the first and after the last keyframes.

There are two basic extrapolation modes:

**Constant** The default one, curves before their first keyframe and after their last one have a constant value (the one of these first and last keyframes).

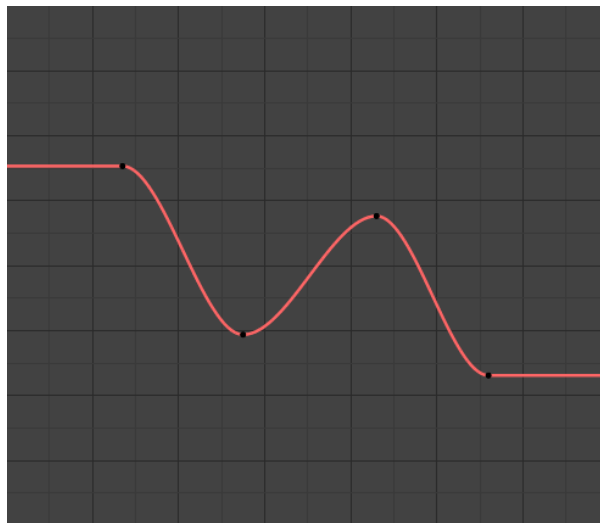


Fig. 465: Constant extrapolation.

**Linear** Curves ends are straight lines (linear), as defined by the slope of their first and last keyframes.

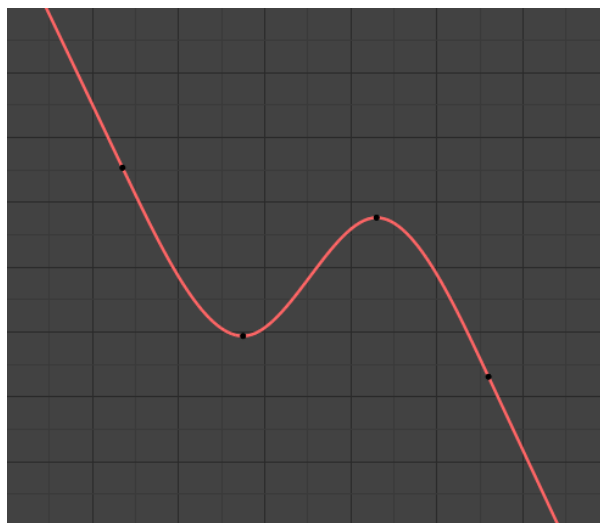


Fig. 466: Linear extrapolation.

Additional extrapolation methods (e.g. the *Cycles* modifier) are located in the *F-Curve Modifiers*.

## Handle Type

### Reference

**Menu** *Key* → *Handle Type*

**Hotkey** V

When using Bézier-interpolated curves it is possible to control the slope of the curve at the control points. This is done via the curve point *handles*; you can set the type of handle to use for the curve points by pressing V or choosing *Key, Handle Type* in the Graph editor menu. Each curve point can have a different handle type, even within the same curve.

There are three automatic modes, *Automatic*, *Auto Clamped*, and *Vector*, where Blender automatically determines the curve's slope at each control point. The neighboring control points have the most influence of the slope, and points further away have a smaller influence. This can be controlled per F-curve with the *Auto Handle Smoothing* properties.

By using the other, non-automatic modes, you have full manual control over the slope.

**Automatic** Handle positions are automatically chosen to produce smooth curves.

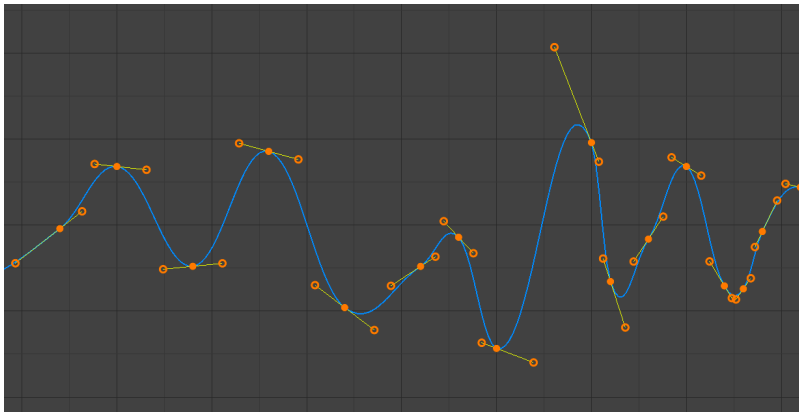


Fig. 467: Auto handles.

**Auto Clamped** Automatic handles clamped to prevent overshoots and changes in the curve direction between keyframes (S-shapes).

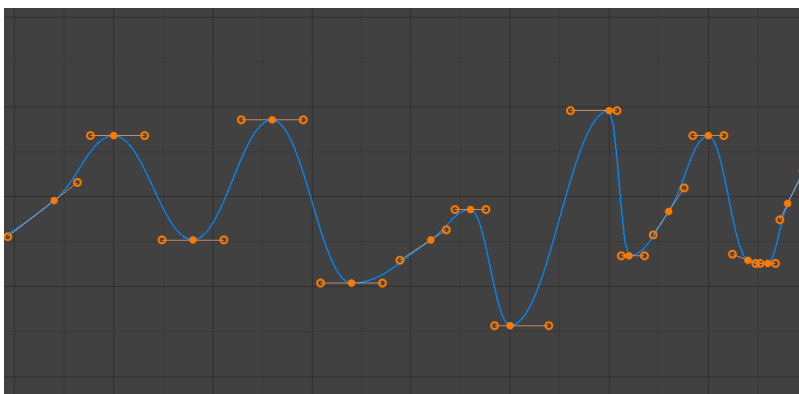


Fig. 468: Auto clamped handles.

**Vector** Creates automatic linear interpolation between keyframes. The segments remain linear when keyframe centers are moved. However, when the handles are moved, the handle type switches to *Free*.

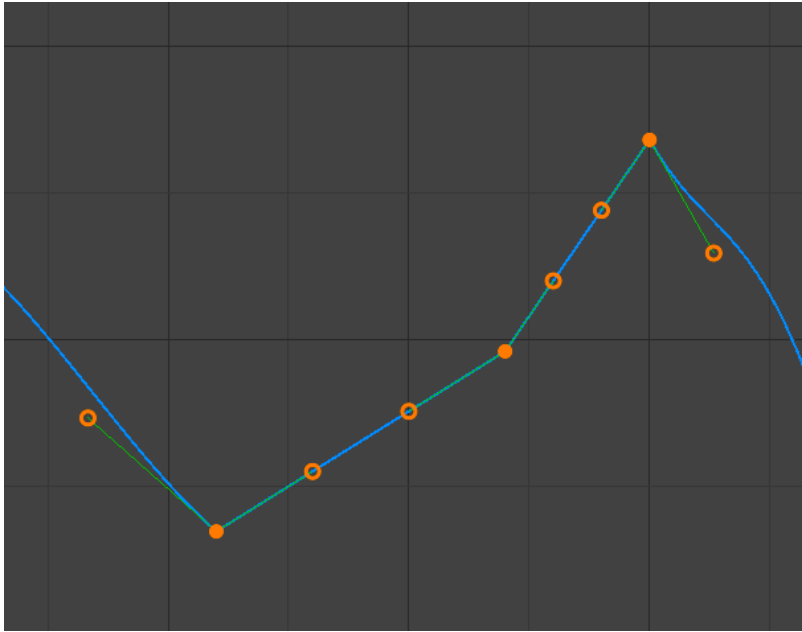


Fig. 469: Vector handles.

**Aligned** 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.

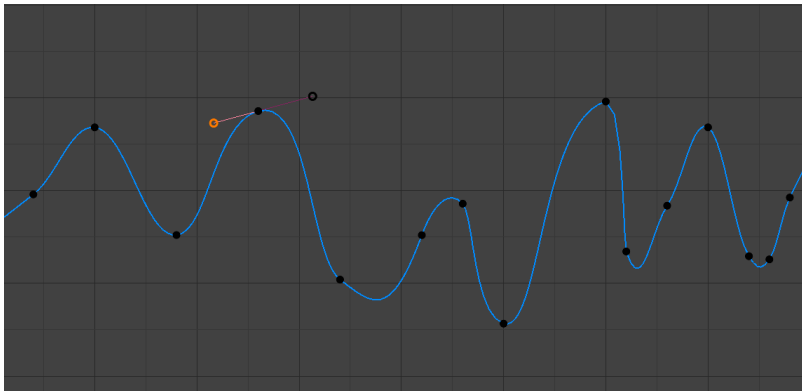


Fig. 470: Aligned handles.

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

## Editing

### Transform

---

## Reference

**Mode** Edit Mode

**Menu** Key → Transform

---

An F-curve can be edited by transforming the locations of the keyframes.

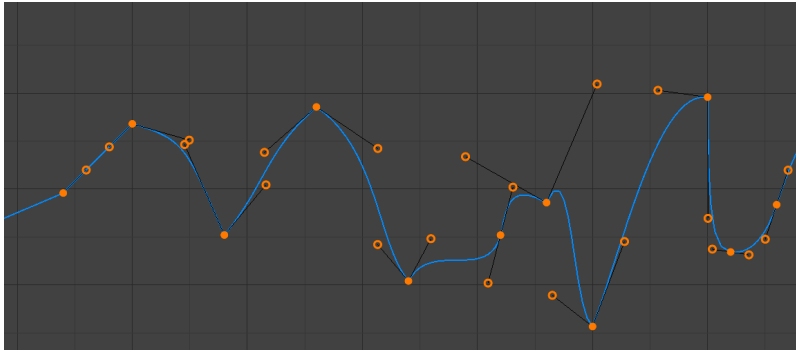


Fig. 471: Free handles.

**Move, Rotate, Scale** Like other elements in Blender, keyframes can be moved, rotated, or scaled as described in *Basic Transformations*.

**Extend** Moves keyframes relative to the Playhead. If the mouse is to the left of the Playhead, this operator only affects the selected keyframes that are to the left of the Playhead. On the contrary, if the mouse is to the right of the Playhead, this operator only affects the selected keyframes that are to the right of the Playhead.

---

**Tip:** For precise control of the keyframe position and value, you can set values in the *Active Keyframe* of the Sidebar region.

---

## Snap

---

### Reference

**Menu** *Key* → *Snap*

**Hotkey** Shift-S

---

Keyframes can be snapped to different properties by using the *Snap Keys* tool.

**Selection to Current Frame** Snap the selected keyframes to the *Playhead*.

**Selection to Cursor Value** Snap the selected keyframes to the *2D Cursor*.

**Selection to Nearest Frame** Snap the selected keyframes to their nearest frame individually.

**Selection to Nearest Second** Snap the selected keyframes to their nearest second individually, based on the *FPS* of the scene.

**Selection to Nearest Marker** Snap the selected keyframes to their nearest marker individually.

**Flatten Handles** Flatten the *Bézier* handles for the selected keyframes.



Table 16: Flatten Handles snapping example.

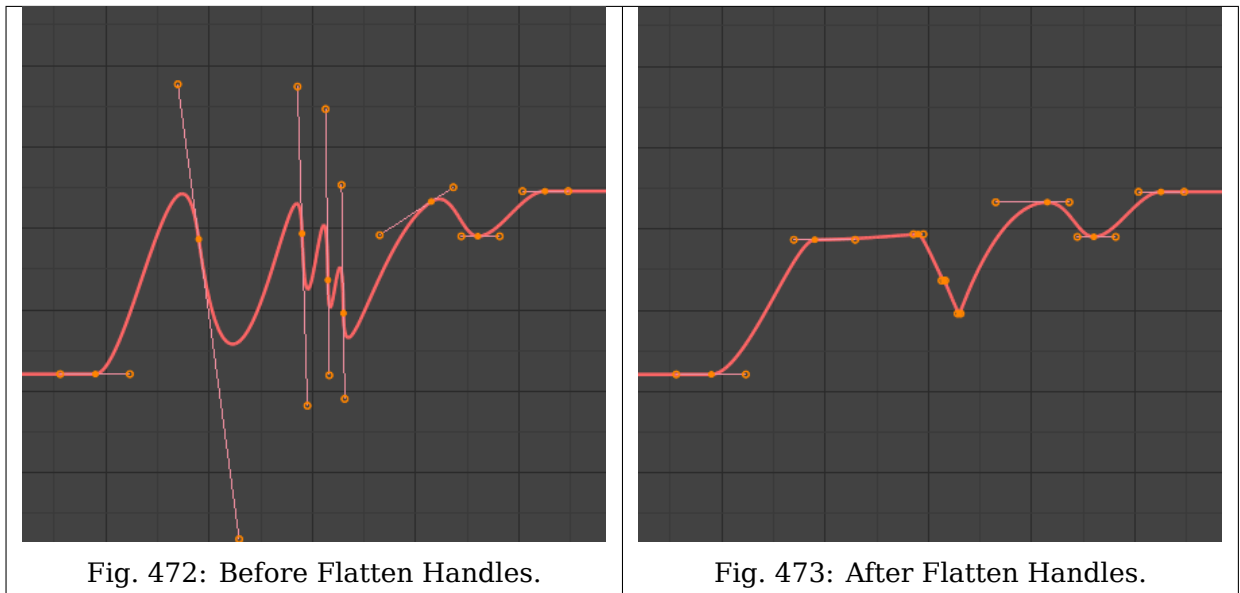


Fig. 472: Before Flatten Handles.

Fig. 473: After Flatten Handles.

**Cursor to Selected Ctrl-G** Places the cursor at the midpoint between selected keyframes.

**Cursor Value to Selection** Places the cursor value on the average value of selected keyframes.

## Mirror

---

### Reference

**Menu** *Key* → *Mirror*

**Hotkey** Ctrl-M

---

Selected keyframes can be mirrored over different properties using the *Mirror Keys* tool.

**By Times over Current Frame** Mirror horizontally over the playhead.

**By Values over Cursor Value** Mirror vertically over the 2D cursor.

**By Times over Time 0** Mirror horizontally over frame 0.

**By Values over Value 0** Mirror vertically over value 0.

**By Times over First Selected Marker** Mirror horizontally over the first selected marker.

## Insert Keyframes

---

### Reference

**Menu** *Key* → *Insert Keyframes*

**Hotkey** Ctrl-RMB, Shift-Ctrl-RMB

---

Inserts a keyframe to the active F-curve at the mouse position. The newly added keyframes will be selected, making it easier to quickly tweak the newly added keyframes. All previously selected keyframes are kept selected by using Shift-Ctrl-RMB.

---

## Add F-Curve Modifier

---

### Reference

**Menu** *Key* → *Add F-Curve Modifier*

**Hotkey** Shift-Ctrl-M

---

Opens a pop-up allowing you to add modifiers to the active F-curve. Settings for the *modifier* can be found in the *Sidebar* → *Modifiers* tab.

---

## Bake Sound to F-Curves

---

### Reference

**Menu** *Key* → *Bake Sound to F-Curves*

---

The *Bake Sound to F-Curves* operator takes a sound file and uses its sound wave to create the animation data.

**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.

---

## Jump to Keyframes

---

### Reference

**Menu** *Key* → *Jump to keyframes*

**Hotkey** Ctrl-G

---

Places the 2D cursor at the center of the selected keyframes.

---

## Copy/Paste

---

### Reference

**Menu** *Key* → *Copy Keyframes*, *Key* → *Paste Keyframes*

---

## Hotkey Ctrl-C, Ctrl-V

---

Use Ctrl-C to copy selected keyframes and Ctrl-V to paste the previously copied keyframes. During the paste action, the *Adjust Last Operation* panel provides some options in how the paste is applied.

### Offset

**No Offset** Pastes the keyframes in the location they were copied from.

**Frame Relative** Pastes the keyframe relative to the Playhead position based on the locations of the keyframes relative to the Playhead when they were copied.

**Frame Start** Pastes the keyframes with the first keyframe of the copied set placed at the Playhead.

**Frame End** Pastes the keyframes with the last keyframe of the copied set placed at the Playhead.

### Type

**Mix** Integrates the pasted keyframes in with existing keyframes only overwriting keyframes that share a frame.

**Overwrite All** Removes all previous keyframes and replaces them with the pasted keyframes.

**Overwrite Range** TODO

**Overwrite Entire Range** TODO

**Flipped** TODO

## Duplicate

---

### Reference

**Menu** *Key → Duplicate*

**Hotkey** Shift-D

---

Duplicates the selected keyframes. You can reposition them by moving the mouse.

## Delete Keyframes

---

### Reference

**Menu** *Key → Delete Keyframes*

**Hotkey** 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*

**Hotkey** V

---

Sets the *handle type* of the selected keyframes.

## Interpolation Mode

---

### Reference

**Menu** *Key* → *Interpolation Mode*

**Hotkey** T

---

Sets the *interpolation mode* between the selected keyframes.

## Easing Mode

---

### Reference

**Menu** *Key* → *Easing Mode*

**Hotkey** Ctrl-E

---

Sets the *easing mode* of the selected keyframes.

## Decimate

---

### Reference

**Menu** *Key* → *Decimate (Ratio)*

**Menu** *Key* → *Decimate (Allowed Change)*

---

The *Decimate* tool simplifies an F-curve by removing keyframes that influence the curve shape the least.

**Mode** Controls which method is used pick the number of keyframes to use.

**Ratio** Deletes a defined percentage of keyframes, the amount of keyframes to delete is define by the *Remove* property.

**Error Margin** Deletes keyframes which only allowing the F-curve to change by a defined amount. The amount of change is controlled by the *Max Error Margin* which controls how much the new decimated curve is allowed to deviate from the original.

## Clean Keyframes

---

### Reference

**Menu** *Key* → *Clean Keyframes*

**Hotkey** X

---

*Clean Keyframes* resets the keyframe tangents on selected keyframes to their auto-clamped shape, if they have been modified.

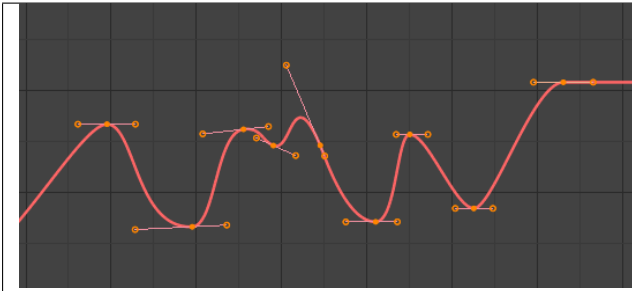


Fig. 474: F-Curve before cleaning.

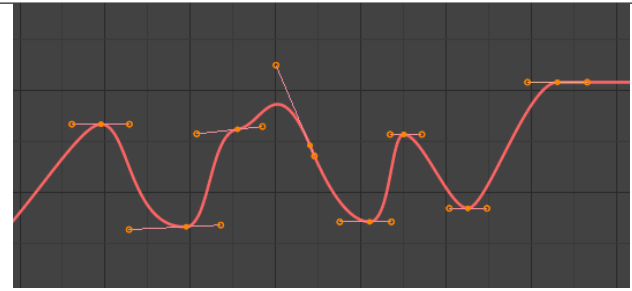


Fig. 475: F-Curve after cleaning.

## Clean Channels

### Reference

**Menu** *Key* → *Clean Channels*

**Hotkey** *X*

Acts like the *Clean Keyframes* tool but will also delete the channel itself if it is only left with a single keyframe containing the default property value and it's not being used by any generative F-curve modifiers or drivers.

**Note:** The modified curve left after the *Clean* tool is run is not the same as the original, so this tool is better used before doing custom editing of F-curves and after initial keyframe insertion, to get rid of any unwanted keyframes inserted while doing mass keyframe insertion (by selecting all bones and pressing *I* for instance).

## Smooth Keys

### Reference

**Menu** *Key* → *Smooth Keys*

**Hotkey** *Alt-0*

There is also an option to smooth the selected curves, but beware: its algorithm seems to be to divide by two the distance between each keyframe and the average linear value of the curve, without any setting, which gives quite a strong smoothing! Note that the first and last keys seem to be never modified by this tool.

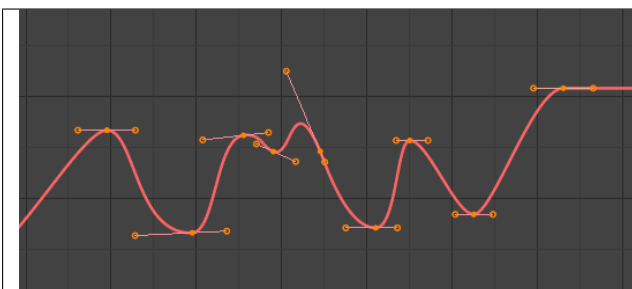


Fig. 476: F-Curve before smoothing.

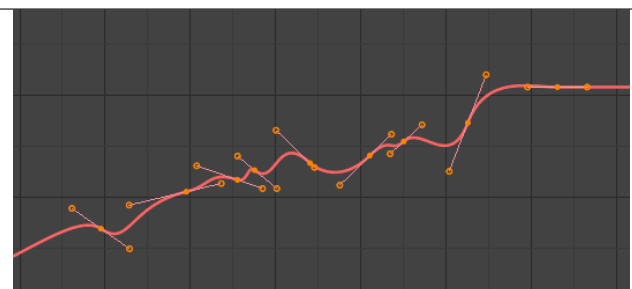


Fig. 477: F-Curve after smoothing.

## Sample Keyframes

### Reference

**Menu** *Key* → *Sample Keyframes*

**Hotkey** Shift-Alt-0

Sampling a set of keyframes replaces interpolated values with a new keyframe for each frame.

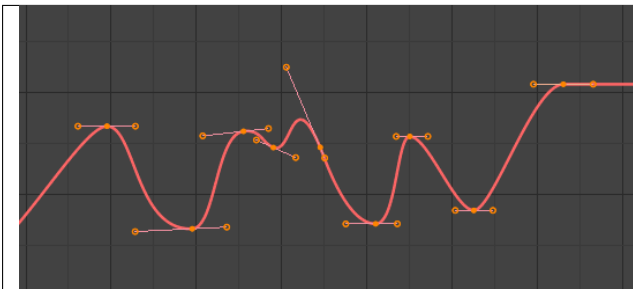


Fig. 478: F-Curve before sampling.

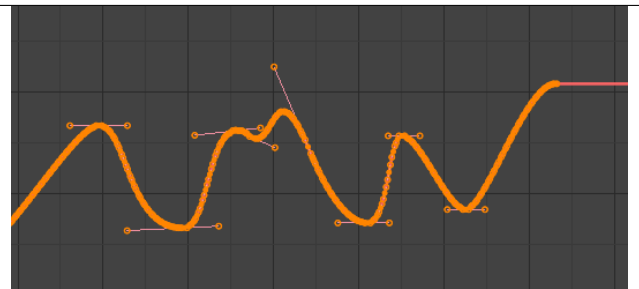


Fig. 479: F-Curve after sampling.

## Bake Curve

### Reference

**Menu** *Key* → *Bake Curve*

**Hotkey** Alt-C

Baking a curve replaces it with a set of sampled points, and removes the ability to edit the curve.

## Discontinuity (Euler) Filter

Todo.

## Properties

### Active F-Curve

### Reference

**Panel** *Sidebar region* → *F-Curve* → *Active F-Curve*

This panel displays properties for the active F-curve.

**Channel Name** ID Type + Channel name (X Location).

**RNA Path** RNA Path to property.

**RNA Array Index** Index to the specific property affected by the F-curve if applicable.

**Display Color** The method used to determine the color of the F-curve shown in the Graph editor.

**Auto Rainbow** Increment the hue of the F-curve color based on the channel index.

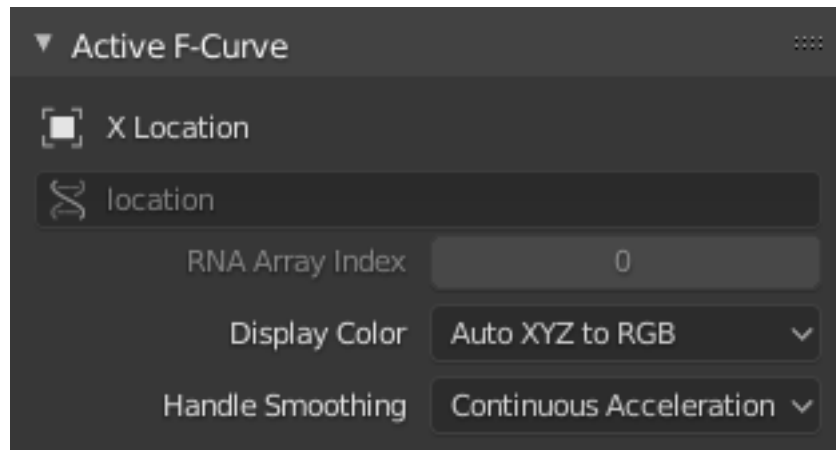


Fig. 480: Active F-Curve panel.

**Auto XYZ to RGB** For property sets like location XYZ, automatically set the set of colors to red, green, blue.

**User Defined** Define a custom color for the active F-curve.

**Handle Smoothing** Selects the method used to compute *automatic Bézier handles* (*Automatic*, *Auto Clamped*, *Vector*).

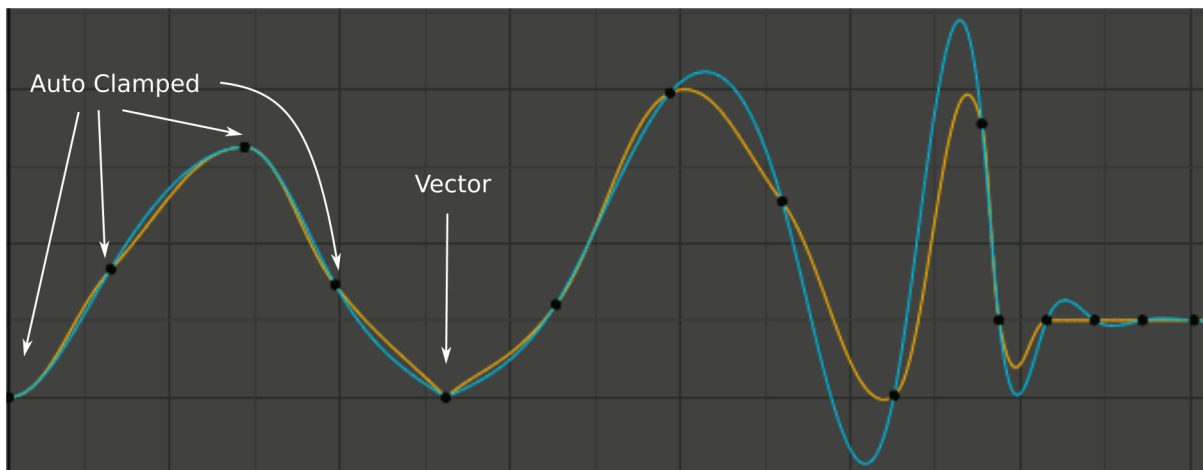


Fig. 481: Handle smoothing mode comparison. Yellow: *None*, Cyan: *Continuous Acceleration*. From left to right, four *Auto Clamped* keys, one *Vector*, and the rest are *Automatic*.

**None** Only directly adjacent key values are considered when computing the handles. Vector handles are pointed directly at the adjacent keyframes.

This older method is very simple and predictable, but it can only produce truly smooth curves in the most trivial cases. Note the kinks in the yellow curve around the keys located between the extremes, and near the Vector handles.

**Continuous Acceleration** A system of equations is solved in order to avoid or minimize jumps in acceleration at every keyframe. Vector handles are integrated into the curves as smooth transitions to imaginary straight lines beyond the keyframe.

This 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.

This mode also tends to overshoot and oscillate more with fully *Automatic* handles in some cases (see the right end of 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. This 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*

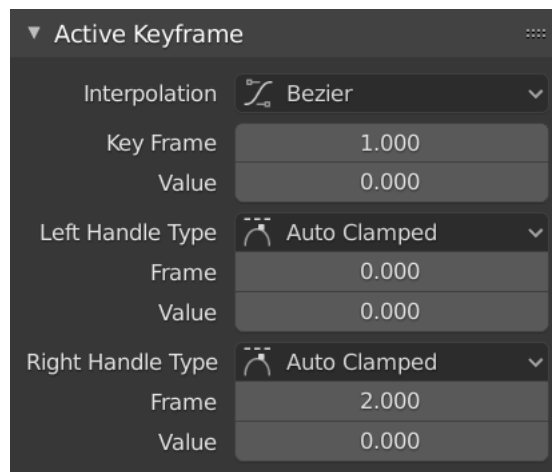


Fig. 482: Active Keyframe panel.

**Interpolation** Set the forward *Interpolation Mode* for the active keyframe.

**Key Value** Set the value for the active keyframe.

**Frame** Set the frame for the active keyframe.

**Left/Right Handle Frame/Value** Set the position of the left/right interpolation handle for the active keyframe.

**Type** See *Handle Type*.

## F-Curve Modifiers

### Reference

**Panel** *Sidebar region* → *Modifiers* → *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.



## Adding a Modifier

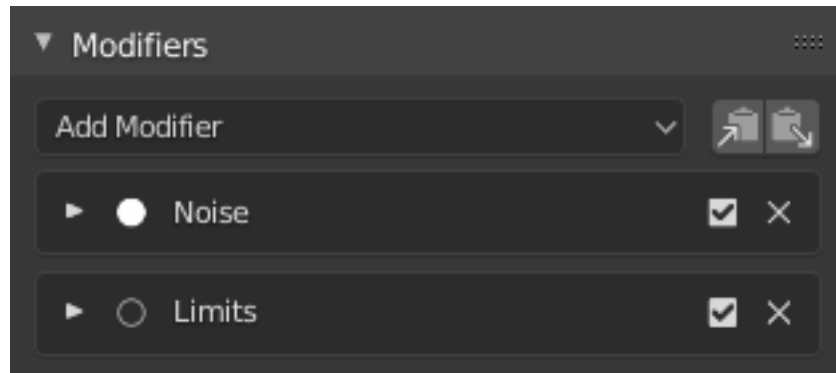


Fig. 483: Modifiers panel.

The F-curve modifier panel is located in the Sidebar region. Select a curve by selecting one of its curve points, or by selecting the channel list. Click on the *Add Modifier* menu to select a modifier.

## Types of Modifiers

### Generator Modifier

Generator creates a Factorized or Expanded Polynomial function. These are basic mathematical formulas that represent lines, parabolas, and other more complex curves, depending on the values used.

**Additive** This option causes the modifier to be added to the curve, instead of replacing it by default.

**Polynomial Order** Specify the order of the polynomial, or the highest power of X for this polynomial. (Number of coefficients: 1.)

Change the Coefficient values to change the shape of the curve.

**See also:**

[The Wikipedia Page](#) for more information on polynomials.

### 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:

- Sine
- Cosine
- Tangent
- Square Root
- Natural Logarithm
- Normalized Sine ( $\sin(x)/x$ )

**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

Allows you to adjust the overall shape of a curve with control points.

**Reference Value** Set the Y value the envelope is centered around.

**Min** Lower distance from Reference Value for 1:1 default influence.

**Max** Upper distance from Reference Value for 1:1 default influence.

**Add Point** Add a set of control points. They will be created at the current frame.

### Point

**Frame** Set the frame number for the control point.

**Min** Specifies the lower control point's position.

**Max** Specifies the upper control point's position.

## Cycles Modifier

Cycles allows you add cyclic motion to a curve that has two or more control points. The options can be set for before and after the curve.

### Cycle Mode

**Repeat Motion** Repeats the curve data, while maintaining their values each cycle.

**Repeat with Offset** Repeats the curve data, but offsets the value of the first point to the value of the last point each cycle.

**Repeat Mirrored** Each cycle the curve data is flipped across the X axis.

**Before, After Cycles** Set the number of times to cycle the data. A value of 0 cycles the data infinitely.

## 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 creating subtle or extreme randomness to animated movements, like camera shake.

### Blend Type

**Replace** Adds a -0.5 to 0.5 range noise function to the curve.

**Add** Adds a 0 to 1 range noise function to the curve.

**Subtract** Subtracts a 0 to 1 range noise function to the curve.

**Multiply** Multiplies a 0 to 1 range noise function to the curve.

**Scale** Adjust the overall size of the noise. Values further from 0 give less frequent noise.

**Strength** Adjusts the Y scaling of the noise function.

**Offset** Offsets the noise in time.

**Phase** Adjusts the random seed of the noise.

**Depth** Adjusts how detailed the noise function is.

### Limits Modifier

Limit curve values to specified X and Y ranges.

**Minimum, Maximum X** Cuts a curve off at these frames ranges, and sets their minimum value at those points.

**Minimum, Maximum Y** Truncates the curve values to a range.

### Stepped Interpolation Modifier

Gives the curve a stepped appearance by rounding values down within a certain range of frames.

**Step Size** Specify the number of frames to hold each frame.

**Offset** Reference number of frames before frames get held. Use to get hold for (1-3) vs (5-7) holding patterns.

**Use Start Frame** Restrict modifier to only act before its “end” frame.

**Use End Frame** Restrict modifier to only act after its “start” frame.

## 2.3.12 Drivers Editor

The Drivers Editor allows users to drive one property with another. See *Drivers* and *F-Curves*.

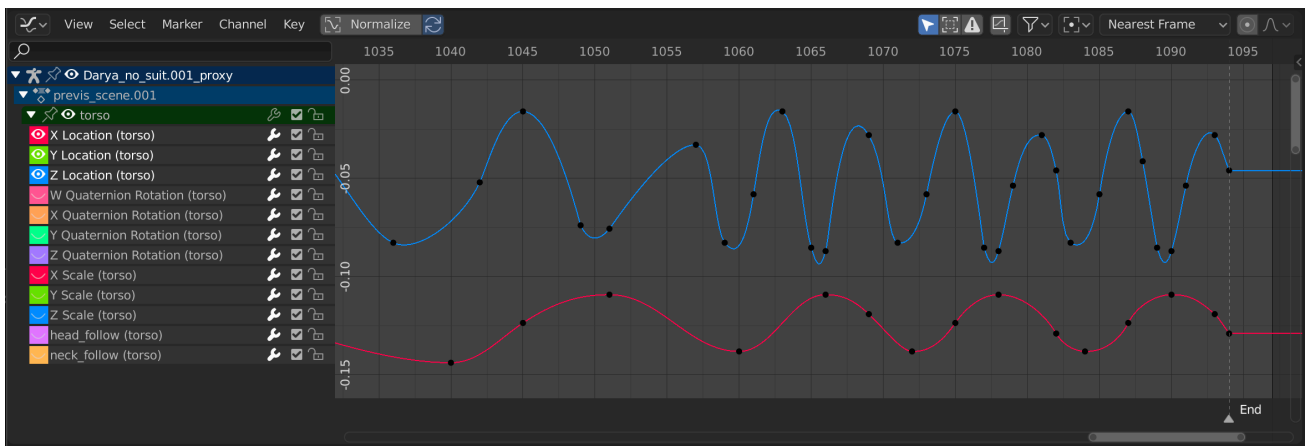


Fig. 484: The Drivers Editor.

### Main Region

The main view allows you to view and edit Driver F-curves. An F-curve has several key parts:

**Axes** The curve defines the relationship between two properties: The current (driven) property (Y axis) and the driver (X axis).

See *F-Curves*.

**Handles** Each point on the driver curve has a handle that helps determine the relationship between the two values. They can be selected and modified to change the shape of the curve. See *Handle Type* for more information.

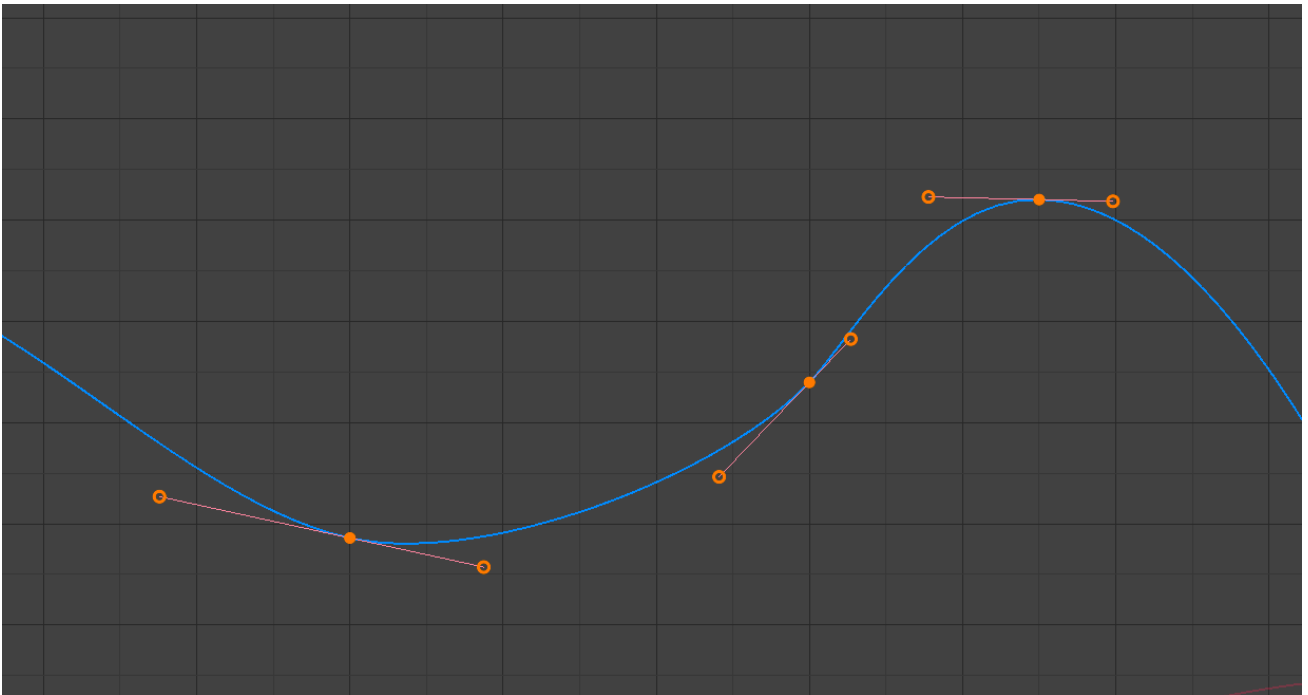


Fig. 485: A simple driver.

**See also:**

See *F-Curves* for more info.

**Navigation**

As with most editors, you can:

**Pan MMB** Pan the view vertically (values) or horizontally (time) with click and drag MMB.

**Zoom Wheel** Zoom in and out with the mouse wheel.

**Scale View Ctrl-MMB** Scale the view vertically or horizontally.

**Frame All Home** Fit the curve in the available space.

In addition, you can also use the scrollbars to pan and zoom the view.

**Header**

**View Controls**



Fig. 486: View controls.

**Normalize** Normalize curves so the maximum and minimum points equal 1.0 and -1.0 respectively.

**Auto** Automatically recalculate curve normalization on every curve edit. Disabling this setting may be useful to prevent curves from jumping after tweaking.

**Show Only Selected (mouse cursor icon)** Only include curves related to the selected objects and data.

**Show Hidden (dashed object icon)** Include curves from objects/bones that are not visible.

**Show Only Errors (warning triangle icon)** Only include curves and drivers that are disabled or have errors. Useful for debugging.

**Create Ghost Curves (square with curve icon)** Makes a visual indication in the background of the editor with a snapshot of the current state of the selected curves. This is useful to have a base for comparison on top of which to make edits.

**Filter (funnel icon)**

**Type Filter** Filter curves by property type.

**Sort Data-Blocks (az icon)** Object data-blocks appear in alphabetical order, so that it is easier to find where they occur (as well as helping to keep the animation of related objects together).

This option may affect the playback speed for heavy scenes.

## Curve Controls

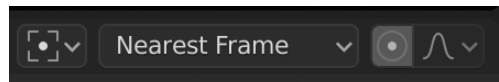


Fig. 487: Curve controls.

**Pivot Point** Pivot point for rotation.

**Bounding Box Center** Center of the selected curve handles.

**2D Cursor** Center of the *2D Cursor*. *Playhead + Cursor*.

**Individual Centers** Rotate the selected curve handles.

**Auto Snap** Auto snap the curve handles when editing.

- No Auto-Snap
- Frame Step
- Second Step
- Nearest Frame
- Nearest Second
- Nearest Marker

**Proportional Editing 0** See *Proportional Editing*.

## Sidebar Region

### Drivers Tab

See *Drivers Panel*.

## Modifiers Tab

See *F-Curve Modifiers*.

## View Properties Panel

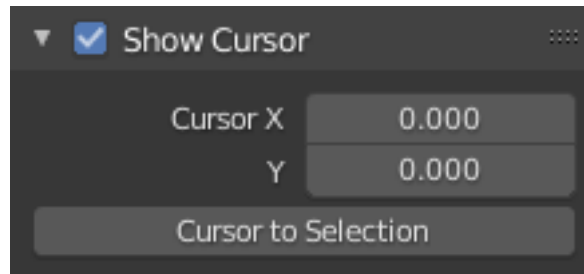


Fig. 488: View Properties panel.

**Show Cursor** Show the vertical *Cursor*.

**Cursor from Selection** Set the *2D cursor* to the center of the selected curve handles.

**Cursor X** *Time Cursor X* position.

**Cursor Y** Vertical *Cursor Y* position.

## 2.3.13 Nonlinear Animation

### Introduction

The NLA editor, short for NonLinear Animation, can manipulate and repurpose *Actions*, without the tedium of handling keyframes. It is often used to make broad, significant changes to a scene's animation, with relative ease. It can also repurpose, chain together a sequence of motions, and "layered" actions, which make it easier to organize, and version-control your animation.

### Header

#### View Menu

**Realtime Updates** When transforming NLA-strips, the changes to the animation are propagated to other views.

**Show Control F-Curves** Overlays a graph of the NLA-strip's influence on top of the strip.

**Show Markers** Shows the markers region. When disabled, the *Markers Menu* is also hidden and markers operators are not available in this editor.

**Show Local Markers** Shows action-local markers on the strip, this is useful when synchronizing time across strips.

**Set Preview Range P** Selecting a preview range by dragging in the NLA Editor.

**Clear Preview Range Alt-P** Unset the preview range

**Auto Select Preview Range Ctrl-Alt-P** Automatically select the preview range based on the range of keyframes.

#### See also:

See *Timeline's View Menu*.

## Select Menu

**All A** Select all NLA-strips.

**None Alt-A** Deselect all NLA-strips.

**Invert Ctrl-I** Invert the current selection of NLA-strips.

**Box Select B** Select NLA-strips by drawing a box. All NLA-strips that intersects the box will be added to the current selection.

**Border Axis Range Alt-B** Select NLA-strips by drawing a box. All NLA-strips that intersects the frames of the drawn box will be added to the current selection.

**Before Current Frame [** Select all NLA-strips before the current frame.

**After Current Frame ]** Select all NLA-strips after the current frame.

## Markers 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.

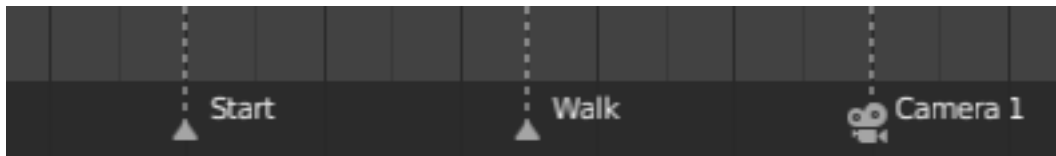


Fig. 489: Markers in animation editor.

For descriptions of the different marker tools, see *Editing Markers*.

## Edit Menu

### Transform

**Move** Move the selected NLA-strips in time or to different NLA-track.

**Extend E** Extend the selected NLA-strips.

**Scale S** Scale the selected NLA-strips.

### Snap

**Selection to Current Frame** Move the start of selected NLA-strips to the current frame.

**Selection to Nearest Frame** Move the start of the selected NLA-strips to the nearest frame.

**Selection to Nearest Second** Move the start of the selected NLA-strips to the nearest second.

**Selection to Nearest Marker** Move the start of the selected NLA-strips to the nearest marker.

**Duplicate Shift-D** Make a copy of the selected NLA-strips.

**Linked Duplicate Alt-D** Make a shallow copy of the selected NLA-strips.

**Split Strips Y** NLA-Split the selected strips into two NLA-strips. The split happens at the current frame.

**Delete Strips X** Delete selected NLA-Strips.

**Delete Tracks** Delete the selected NLA track and all strips that it contains.

**Toggle Muting H** Mute or unmute the selected NLA-strips. Muted NLA-strips will not influence the animation.

**Apply Scale Ctrl-A** Apply the scale of the selected NLA-strips to their referenced Actions.

**Clear Scale Alt-S** Reset the scaling of the selected NLA-strips.

**Sync Action Length** Synchronize the length of the action to the length used in the NLA-strip.

**Make Single User U** This tool ensures that none of the selected strips use an action which is also used by any other strips.

**Swap Strips Alt-F** Swap the order of the selected NLA-strips in their NLA-track.

**Move Strips Up PageUp** Move selected NLA-strips a track up if there is room.

**Move Strips Down PageDown** Move selected NLA-strips a track down if there is room.

### Track Ordering

**To Top** Move selected track to the top of the tracks.

**Up** Move selected track one track up.

**Down** Move selected track one track down.

**To Bottom** Move selected tracks to the bottom of the tracks.

**Remove Empty Animation Data** Remove Animation Data from selected objects when they don't contain any animation.

**Start Editing Stashed Action Shift-Tab** It will enter and exit Tweak Mode as usual, but will also make sure that the action can be edited in isolation (by flagging the NLA track that the action strip comes from as being "solo"). This is useful for editing stashed actions, without the rest of the NLA Stack interfering.

**Start Tweaking Strips Actions Tab** The contents of Action strips can be edited, but you must be in *Tweak Mode* to do so. The keyframes of the action can then be edited in the Dope Sheet.

### Add

**Add Action Strip Shift-A** Add an NLA-strip referencing an Action to the active track.

**Add Transition Shift-T** Add an NLA-strip to create a transition between a selection of two adjacent NLA-strips.

**Add Sound Strip Shift-K** Add an NLA-strip controlling when the speaker object plays its sound clip.

**Add Meta-Strip Ctrl-G** Group selected NLA-strips into a meta strip. A meta strip will group the selected NLA-strips of the same NLA-track.

**Remove Meta-Strip Ctrl-Alt-G** Ungroup selected Meta strips.

**Add Tracks** Add a new NLA-Track on top of the selected object.

**Add Track Above Selected** Add a new NLA-Track just above the selected NLA-track.

**Include Selected Objects** Let the selected objects appear in the NLA Editor. This is done by adding an empty animation data object to the selected object.

### Tracks

Tracks are the layering system of the NLA. At its most basic level, it can help organize strips. But it also layers motion much like an image editor layers pixels - the bottom layer first, to the top, last.

**Solo (star icon)** Toggling *Solo Track* causes only the selected tracks effects to be visible when animating. This is very useful for debugging complex animations.



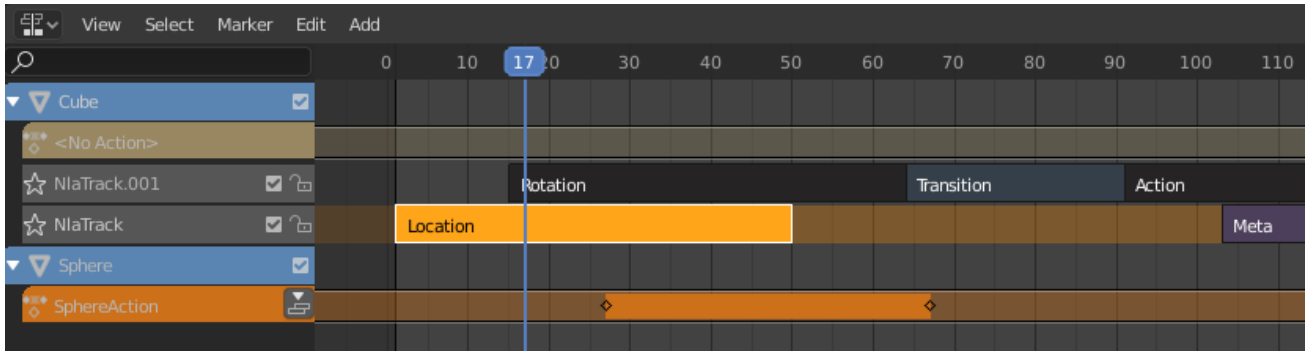


Fig. 490: NLA Tracks and Strips.

**Mute (speaker icon)** Keeps the track from having an effect on the animation. (Mute only applies when *Solo* is not used.) All strips in that track are shown as being muted (dashed outline).

**Lock (padlock icon)** Prevents changes from being made to this layer. This is useful, for example, if you want to select several strips and move them; but you want to keep a few tracks excluded from the change.

### Action Track

By default, the Action Editor automatically stores the keyframes you create through the 3D Viewport by storing them into an action based on the name as the object you are working on.

**Push Down Action (push down icon)** Turns the active action into a new NLA strip at the top of the NLA stack.



Push Down action button.

**Pin (pin icon)** If you try moving the strip, while in *Tweak Mode*, you will notice that the keys will go along with it. On occasion, you will prefer the keys to remain on their original frames, regardless of where the strip is. To do so, hit the *unpin* icon, next to the strip.

### Action Stashing

When creating a new action, if the existing action only has a single user (i.e. the current reference only), it will get “stashed” in the NLA stack. Action Stashing should prevent most cases actions getting lost.

The action “stashing” method works by storing otherwise unused/dormant actions in the NLA stack as strips in special muted NLA tracks. This way, the action is linked up to a particular datablock (i.e. to a specific object, or to a specific material/light/etc.) and will be preserved for later use.

### Deleting & Converting

If you decide that you no longer want a stashed action anymore, simply delete the corresponding NLA strip, then save and reload the file.

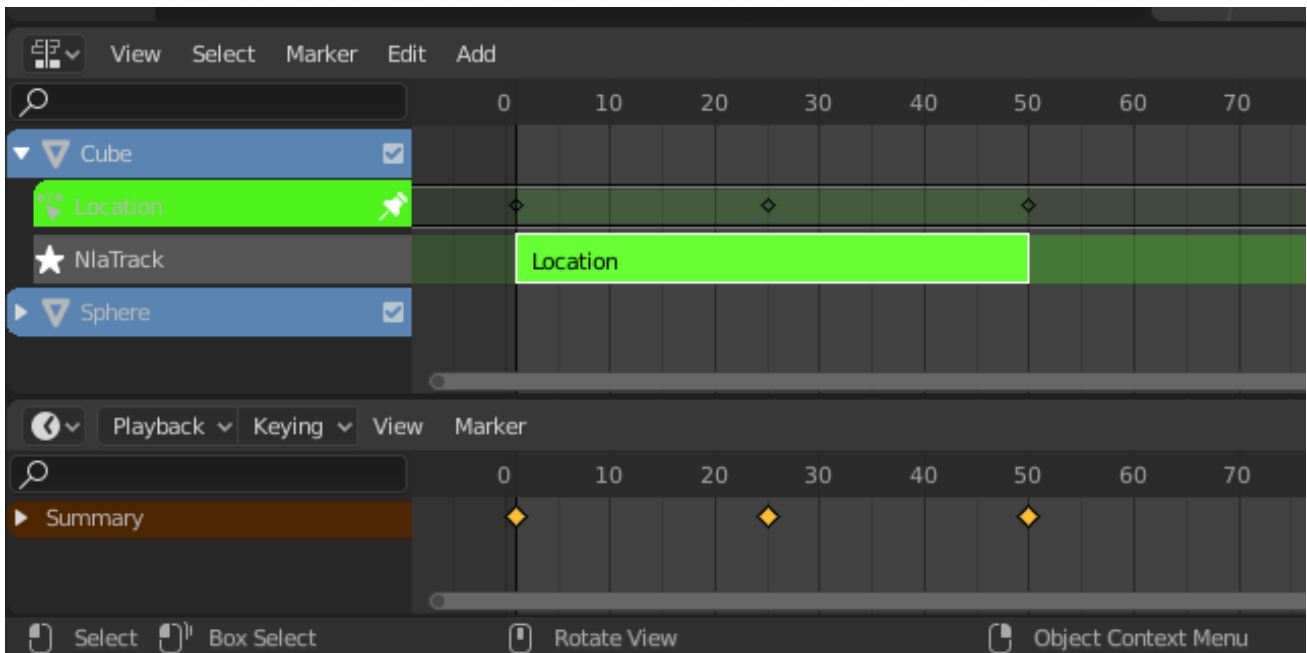


Fig. 491: NLA strip with pinned keys.

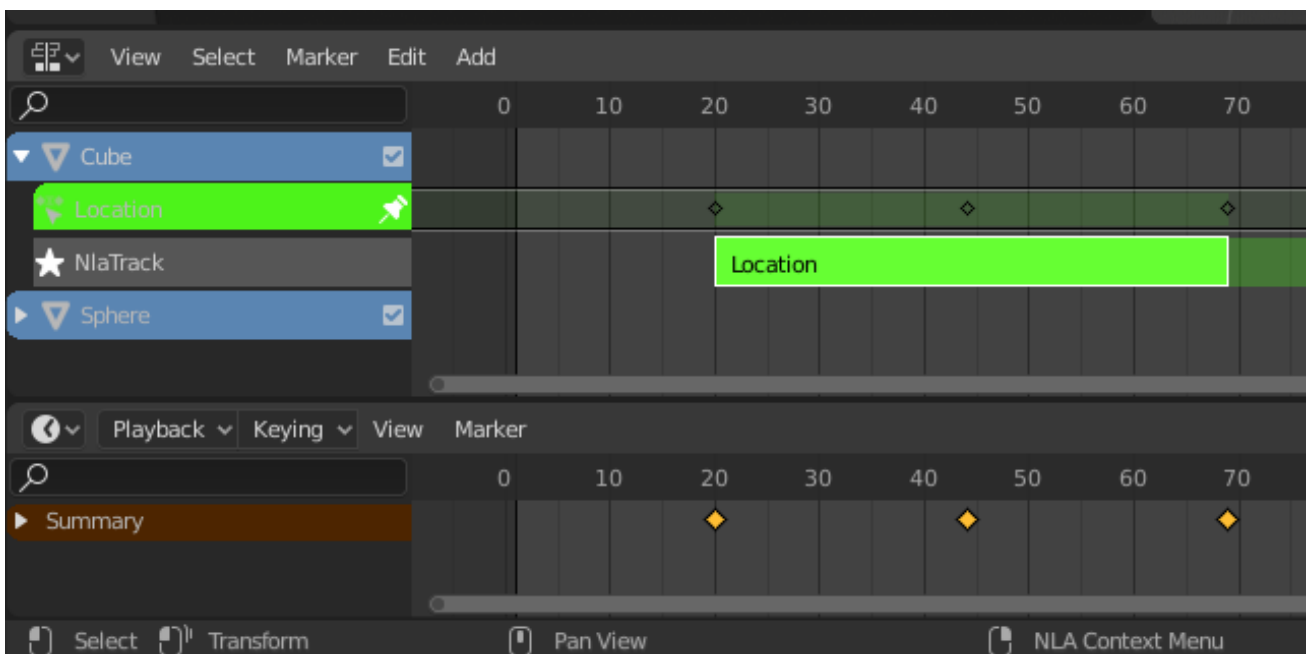


Fig. 492: Strip moved, notice the keys move with it.

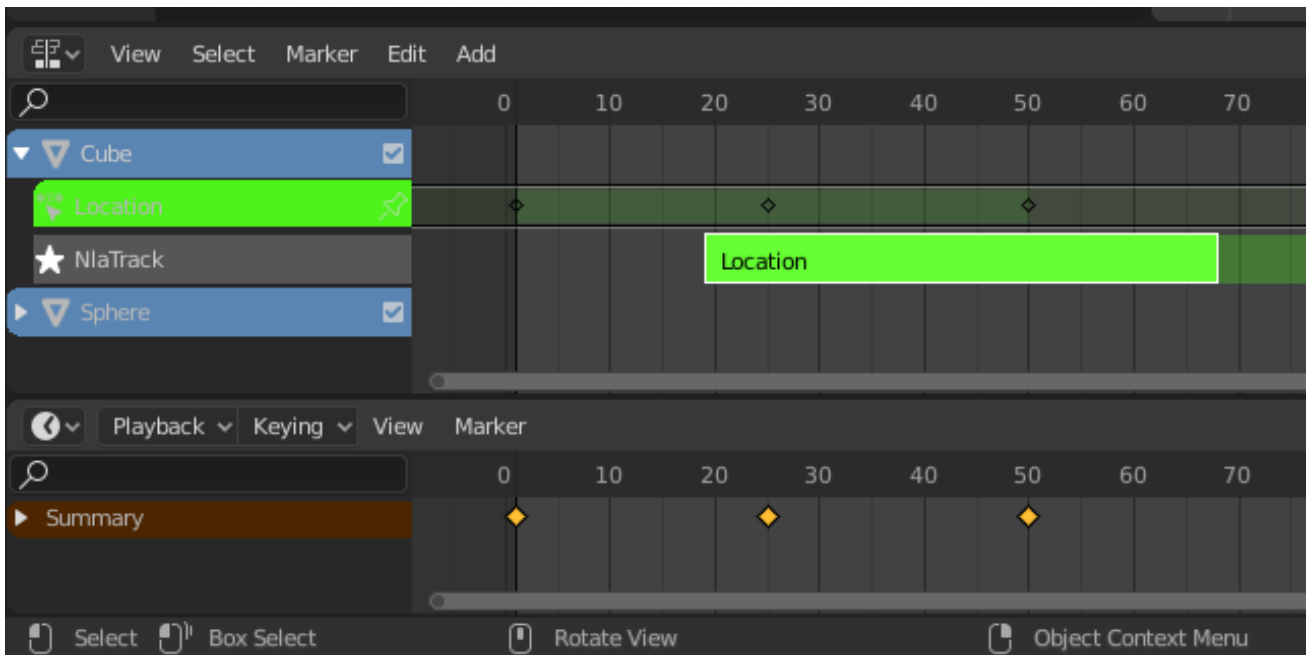


Fig. 493: The unpinned keys return to their original frames.

Also, note that since these are NLA strips, you can reuse these as normal NLA strips simply by un-muting (and renaming) the NLA track they live in. You may also want to move it above all the other stashed-action tracks.

## Strips

There are four kinds of strips: Action, Transition, Sound clip and Meta.

### Action Strips

Action Strips are a container of keyframe data of an action. Any action used by the NLA first must be turned into an Action strip. This is done so by clicking the *Push Down Action* button see above. Alternatively, you can go to *Add* → *Action Strip*.

---

**Note:** Action Strips only playback the keyframe data that fits into the length of the strip. This includes any *modifiers* that might extend keyframe data.

---

### Transition Strips

Transitions interpolate between Actions. They must be placed in between other strips. Select two or more strips on the same track, and go to *Add* → *Transition*.



Fig. 494: Transition Strip.

## Sound Clip Strips

Controls when a speaker plays a sound clip. *Add* → *Sound Clip*.

## Meta Strips

Meta strips group strips together as a whole, so you can move them as one. If you find yourself moving a lot of strips together, you can group them into a Meta strip. A Meta strip can be moved and duplicated like a normal strip.

### Reference

**Menu** *Add* → *Add Meta-Strips*

**Hotkey** Shift-G

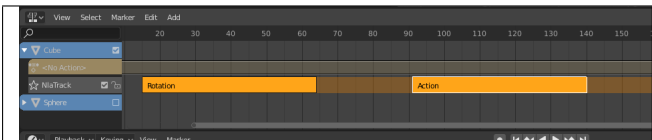


Fig. 495: Select two or more strips.

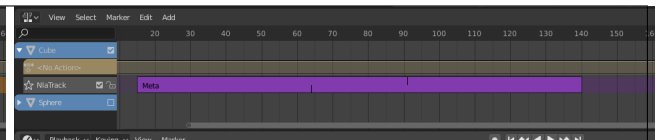


Fig. 496: Combine them into a meta strip.

A Meta strip still contains the underlying strips. You can ungroup a Meta strip.

### Reference

**Menu** *Add* → *Remove Meta-Strips*

**Hotkey** Alt-G

## Editing

### Transform

Todo.

### Snap

Todo.

### Duplicate

### Reference

**Menu** *Edit* → *Duplicate*

**Hotkey** Shift-D

Creates a new instance of the selected strips with a copy of the action.

## Linked Duplicate

### Reference

**Menu** *Edit* → *Linked Duplicate*

**Hotkey** Alt-D

The contents of one Action strip can be instanced multiple times. To instance another strip, select a strip, go to *Edit* → *Linked Duplicate*. It will use the same action as the selected strips.

Now, when any strip is tweaked, the others will change too. If a strip other than the original is tweaked, the original will turn to red.

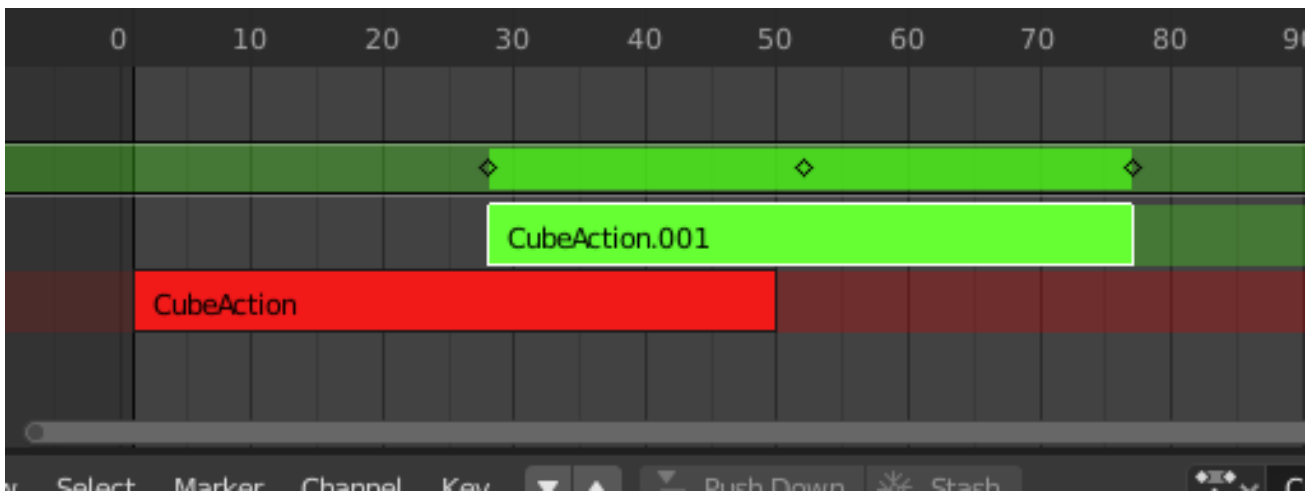


Fig. 497: Linked duplicated strip being edited.

### Split Strips

Todo.

### Delete Strips

Todo.

### Delete Tracks

Todo.

### Toggle Muting

Todo.

### Apply Scale

Todo.

### Clear Scale

Todo.

### Sync Action Length

Todo.

### Make Single User

---

#### Reference

**Menu** *Edit* → *Make Single User*

**Hotkey** U

---

This tool ensures that none of the selected strips use an action which is also used by any other strips.

---

**Note:** This does not recursively go inside meta strips.

---

### Swap Strips

Todo.

### Move Strips Up

Todo.

### Move Strips Down

Todo.

### Track Ordering

Todo.

### Remove Empty Animation Data

---

#### Reference

**Menu** *Edit* → *Remove Empty Animation Data*

---

This operator removes AnimData data-blocks (restricted to only those which are visible in the animation editor where it is run from) which are “empty” (i.e. that is, have no active action, drivers, and NLA tracks or strips).

It is sometimes possible to end up with a lot of data-blocks which have old and unused Animation Data containers still attached. This most commonly happens when doing motion graphics work

(i.e. when some linked-in objects may have previously been used to develop a set of reusable assets), and is particularly distracting in the NLA Editor.

## Start Editing Stashed Action

### Reference

**Menu** *Edit* → *Start Editing Stashed Action*

**Hotkey** Shift-Tab

It will enter and exit Tweak Mode as usual, but will also make sure that the action can be edited in isolation (by flagging the NLA track that the action strip comes from as being “solo”). This is useful for editing stashed actions, without the rest of the NLA Stack interfering.

## Start Tweaking Strips Action

### Reference

**Menu** *Edit* → *Start Tweaking Strips Action*

**Hotkey** Tab

The contents of Action strips can be edited, but you must be in *Tweak Mode* to do so. The keyframes of the action can then be edited in the Dope Sheet.

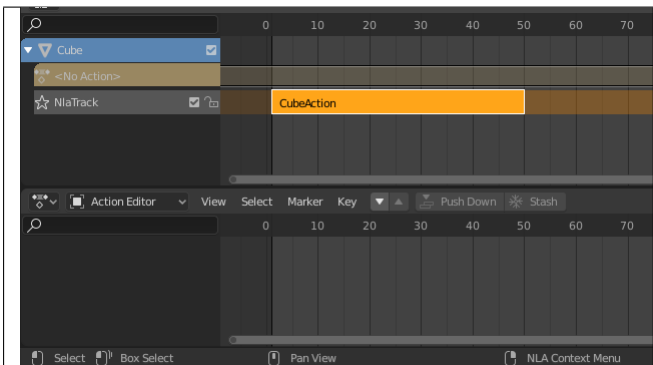


Fig. 498: Strip in NLA mode.

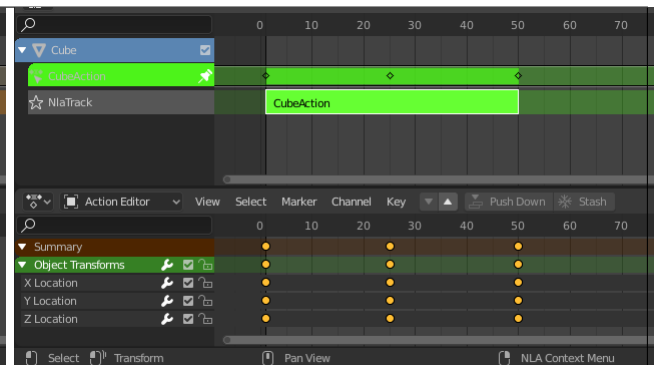


Fig. 499: Strip in Tweak mode.

When you finished editing the strip, simply go to *Edit* → *Tweaking Strips Action* or press Tab.

## Properties

### Edited Action

### Reference

**Panel** *Sidebar* → *Edited Action*

**Action** A *Data-Block* menu that allows you to edit actions shown in the action track. See also the *Action Editor's Action*.

**Extrapolation** Action to take for gaps past the strip extents.

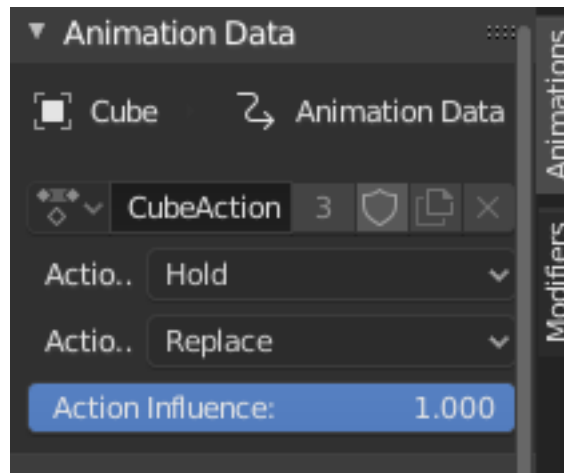


Fig. 500: Edited Action panel.

**Hold** Affects both sides of the strip. This should only be set on the very first strip. When the order of strips changes (for example by dragging them in the NLA editor), a strip that is marked as *Hold* is no longer the very first strip will automatically be set to *Hold Forward* by Blender.

**Hold Forward** Affects the region after the clip, only. This can be set on any strip.

**Nothing** Affects only the region of the strip itself. This can be set on any strip.

**Blending** Affects how the property values directly produced by the strip are combined with the result of evaluating the stack below. The bottom-most strip is blended on top of the default values of the properties.

**Replace** The top strip is linearly blended in with the accumulated result according to influence, completely overwriting it if influence is set to 100%.

**Multiply, Subtract, Add** The result of the strip is multiplied, subtracted, or added to the accumulated results, and then blended in according to influence.

$$result = mix(previous, previous(+ - *)value, influence)$$

**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)**  $result = previous * (value/default)^{influence}$

**Others**  $result = previous + (value - default) * influence$

This allows layering actions that can also be used as a standalone. Properties keyframed at their default values remain at default.

---

**Note:** Since this blending mode is based on using 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** Amount the active Action contributes to the result of the NLA stack.



## Strip

**Name** Name of the track which the strip currently belongs to.

**Mute** Toggles NLA strip evaluation, the strip outline will be dashed.

## Active Strip

### Reference

**Panel** *Sidebar* → *Strip* → *Active Strip*

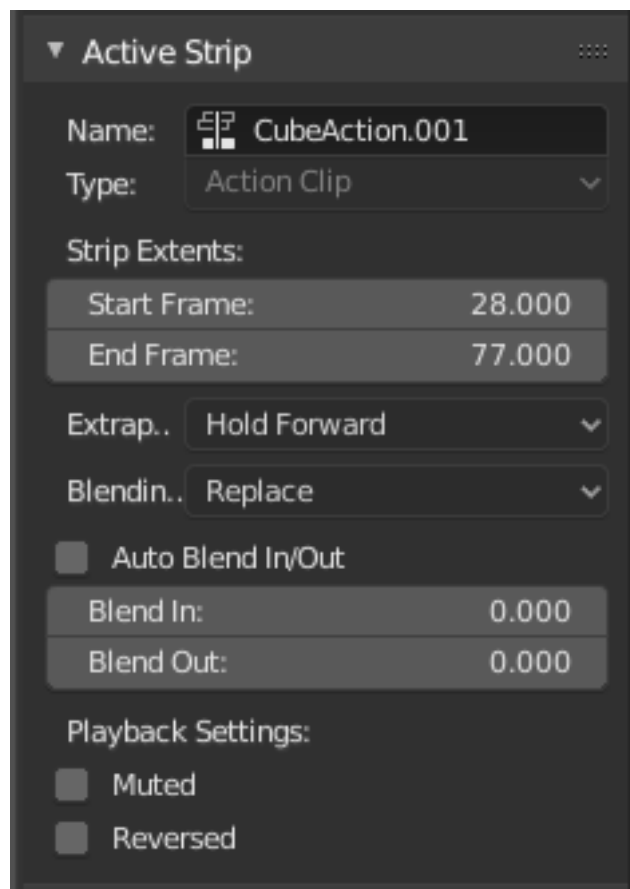


Fig. 501: Active Strip panel.

**Frame Start, End** The boundaries of the strip itself. Note that this will stretch the duration of the Action, it will not cause greater or fewer keyframes from the Actions to play (see below for that option).

**Extrapolation** See *Extrapolation*.

**Blending** See *Blending*.

**Blend In, Out** The first and last frame that represents when this strip will have full influence.

**Auto Blend In/Out** Creates a ramp starting at the overlap of the strips. The first strip has full control, and it ramps linearly giving the second strip full control by the end of the overlapping time period.

### Playback

**Reversed** Cause this strip to be played completely backwards.

**Cyclic Strip Time** Cycle the animated time within the action start and end.

### Animated Influence

Enabling alteration of the degree of influence this strip has as a keyframable value. If influence isn't animated, the strips will fade linearly, during the overlap. These can be found in the Dope Sheet or Graph Editors under the *NLA Control Curves* and look like group channels. They appear before all the groups or F-curves for that channel.

### Animated Strip Time

Same as *Animated Influence*, but with *Strip Time*.

### Action Clip

---

#### Reference

**Panel** *Sidebar region* → *Animations* → *Action Clip*

---

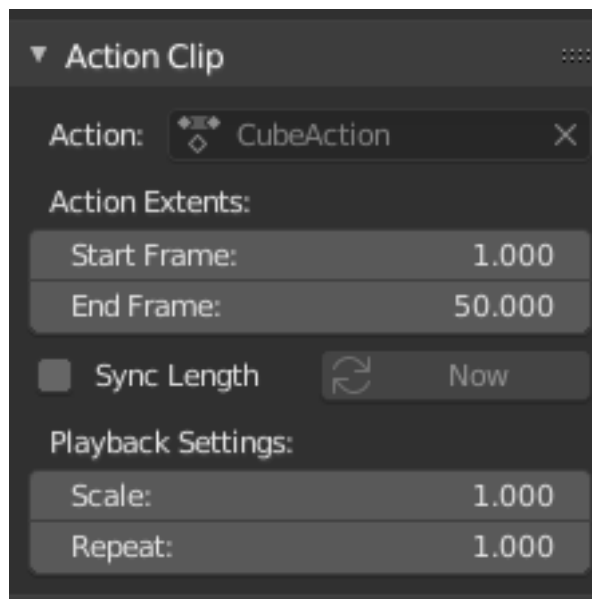


Fig. 502: Action Clip panel.

This represents the 'object data' of the strip. Much like the transform values of an object.

**Action** A reference to the Action contained within the strip. Can be changed to replace the current strip's value with another Action.

**Frame Start, End** How much of the Action to use.

For instance, it is common to set the first and last keyframe of an Action to be the same keyframes. The problem with this is if you loop the animation, there is a slight hitch where the same keyframes are played twice. To fix this, simply reduce the *End Frame*.

---

**Note:** If you select values that are above or below the actual keyframe count of the Action, then the *F-curve Extrapolation* will be applied.

---

**Sync Length** Causes the *Start* and *End Frames*, above, to be reset to the first and last keyframed frames of the Action.

**Now** The *Now* button causes the *Start* and *End Frames*, above, to be reset to the first and last keyframed frames of the Action.

**Playback Scale** Stretches strip, another way of increasing the *Strip Extents: End Frame*, above.

**Repeat** Also expands the strip, but by looping from the first keyframe and going forward.

## Modifiers

### Reference

**Panel** *Sidebar region* → *Modifiers* → *Modifiers*

Like its counterparts in graph and video editing, Modifiers can stack different combinations of effects for strips.

See *F-Curve Modifiers*.

## Scripting

### 2.3.14 Text Editor

Blender has a *Text Editor* among its editor types, accessible via the *Editor type* menu, or the shortcut Shift-F11.

#### 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.

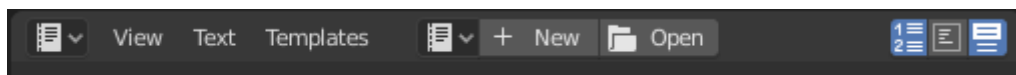


Fig. 503: Text header.

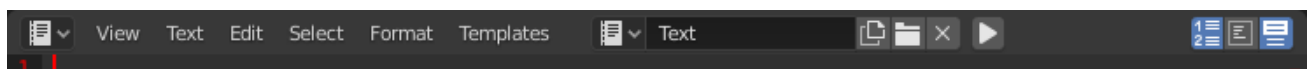


Fig. 504: Text header with a text loaded.

**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** Toggle display options.

Line Numbers, Word Wrap, 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 are too long to 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.

## 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 Text Block Alt-0.** 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).

**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 / file extension.

**Register** Registers the current text data-block as a module on loading (the text name must end with `.py`). Read more about the registration of Python modules in [API documentation](#).

**Live Edit** Todo.

**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 text clipboard.

**Copy Ctrl-C** Copies the marked text into the text 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

The *Text Editor* has some dedicated Python scripts, which are useful for writing tools, like a class/function/variable browser, completion...

Python, OpenShading Language

## 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:** Usages for the Text editor

The Text editor is handy also when you want to share your blend-files with others. The Text editor can be used to write in a README text explaining the contents of your blend-file. Be sure to keep it visible when saving!

---

## 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 upper-case and lower-case 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 right margin to help keep line length at a reasonable length when scripting. The width of the margin is specified in *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** Use *Tabs* or *Spaces* for indentations.

### Footer

The Text editor footer displays if the text is saved internal or external 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

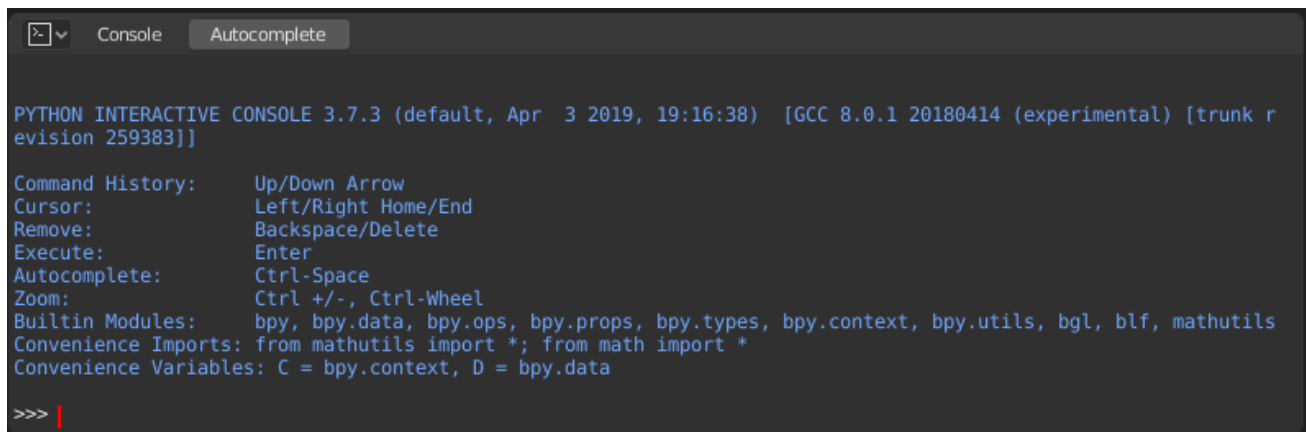
### Running Scripts

The most notable keystroke is **Alt-P** which makes the content of the buffer being parsed by the internal Python interpreter built into Blender. Before going on it is worth noticing that Blender comes with a fully functional Python interpreter built-in, and with a lots of Blender-specific modules, as described in the *Scripting & Extending Blender* section.

### 2.3.15 Python Console

The Python Console is a quick way to execute commands, with access to the entire Python API, command history and auto-complete. The command prompt is typical for Python 3.x, the interpreter is loaded and is ready to accept commands at the prompt `>>>`.

The Python Console is a good way to explore the possibilities of Blender built-in Python. The Python Console can be used to test small bits of Python code which can then be pasted into larger scripts.



```

PYTHON INTERACTIVE CONSOLE 3.7.3 (default, Apr  3 2019, 19:16:38) [GCC 8.0.1 20180414 (experimental) [trunk revision 259383]]

Command History:      Up/Down Arrow
Cursor:               Left/Right Home/End
Remove:               Backspace/Delete
Execute:              Enter
Autocomplete:        Ctrl-Space
Zoom:                 Ctrl +/-, Ctrl-Wheel
Builtin Modules:      bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports:  from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>>

```

Fig. 505: Python Console.

## Interface

### Header Menus

### View Menu

**Zoom In / Zoom Out** Increases/Decreases the font size of the console text.

**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.

**Move to Line End End** Moves the cursor to 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.

**Copy Ctrl-C** Copy the selection.

**Paste Ctrl-V** Paste 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 previous command as they appear in the command history.

**Forward in History Down** Changes the current command to next command as they appear in the command history.

**Autocomplete Tab** See *Auto Completion* for more information.

## Main View

### Key Bindings

- Left / Right - Cursor motion.
- Ctrl-Left / Ctrl-Right - Cursor motion, by word.
- 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 check what is loaded into the interpreter environment, type `dir()` at the prompt and execute it.



```

Console  Autocomplete
Execute:      Enter
Autocomplete: Ctrl-Space
Zoom:        Ctrl +/-, Ctrl-Wheel
Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> dir()
['C', 'Color', 'D', 'Euler', 'Matrix', 'Quaternion', 'Vector', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'bpy', 'bvhtree', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'geometry', 'help', 'hypot', 'inf', 'interpolate', 'isclose', 'isfinite', 'isinf', 'isnan', 'kdtree', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'noise', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> |

```

## Auto Completion

Now, type `bpy.` and then press `Tab` and you will see the Console auto-complete feature in action.

```

Console  Autocomplete
'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'geometry', 'help', 'hypot', 'inf', 'interpolate', 'isclose', 'isfinite', 'isinf', 'isnan', 'kdtree', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'noise', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> bpy.
  app
  context
  data
  msgbus
  ops
  path
  props
  types
  utils
>>> bpy.|

```

You will notice that a list of submodules inside of `bpy` appear. These modules encapsulate all that we can do with Blender Python API and are very powerful tools.

Lets list all the contents of `bpy.app` module.

Notice the green output above the prompt where you enabled auto-completion. What you see is the result of auto completion listing. In the above listing all are module attributed names, but if you see any name end with `()`, then that is a function.

We will make use of this a lot to help our learning the API faster. Now that you got a hang of this, lets proceed to investigate some of modules in `bpy`.

## Before Tinkering with the Modules

If you look at the 3D Viewport in the default Blender scene, you will notice three objects: Cube, Light and Camera.

- All objects exist in a context and there can be various modes under which they are operated upon.
- At any instance, only one object is active and there can be more than one selected object.
- All objects are data in the blend-file.
- There are operators/functions that create and modify these objects.

For all the scenarios listed above (not all were listed, mind you...) the `bpy` module provides functionality to access and modify data.

## Examples

### bpy.context

**Note:** For the commands below to show the proper output, make sure you have selected object(s) in the 3D Viewport.

```

[>] Console Autocomplete
mathutils
Convenience Imports: from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> bpy.context.mode
'OBJECT'

>>> bpy.context.active_object
bpy.data.objects['Cube']

>>> bpy.context.selected_objects
[bpy.data.objects['Cube'], bpy.data.objects['Light'], bpy.data.objects['Camera']]

>>>

```

**bpy.context.mode** Will print the current 3D Viewport mode (Object, Edit, Sculpt, etc.).

**bpy.context.object** or **bpy.context.active\_object** Will give you access to the active object in the 3D Viewport.

Change the X location to a value of 1:

```
bpy.context.object.location.x = 1
```

Move the object from previous X location by 0.5 unit:

```
bpy.context.object.location.x += 0.5
```

Change the X, Y, Z location:

```
bpy.context.object.location = (1, 2, 3)
```

Change only the X, Y components:

```
bpy.context.object.location.xy = (1, 2)
```

The data type of object's location:

```
type(bpy.context.object.location)
```

Now that is a lot of data that you have access to:

```
dir(bpy.context.object.location)
```

**bpy.context.selected\_objects** Will give access to a list of all selected objects.

Type this and then press Tab:

```
bpy.context.selected_objects
```

To print out the name of first object in the list:

```
bpy.context.selected_objects[0]
```

The complex one... But this prints a list of objects not including the active object:

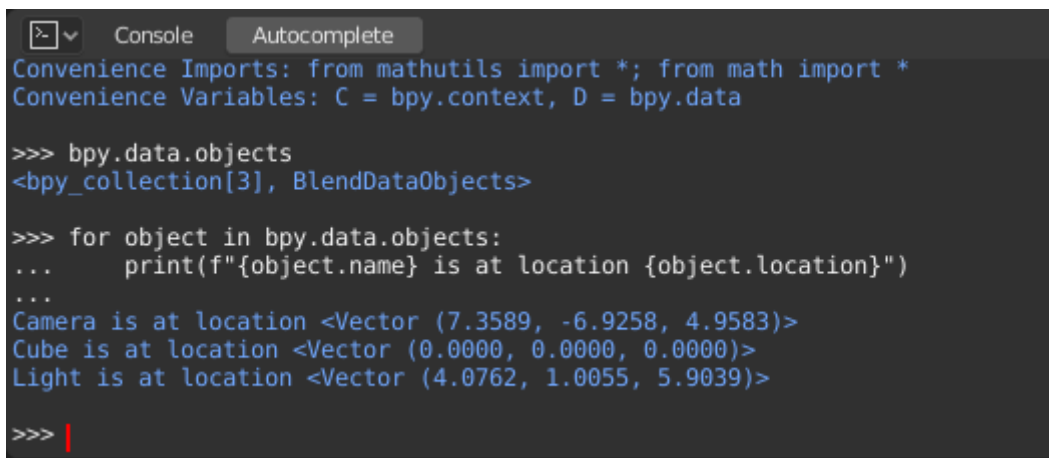
```
[obj for obj in bpy.context.selected_objects if obj != bpy.context.object]
```

## bpy.data

`bpy.data` has functions and attributes that give you access to all the data in the blend-file.

You can access following data in the current blend-file: objects, meshes, materials, textures, scenes, screens, sounds, scripts, etc.

That is a lot of data.



```

>>> bpy.data.objects
<bpy_collection[3], BlendDataObjects>

>>> for object in bpy.data.objects:
...     print(f"{object.name} is at location {object.location}")
...
Camera is at location <Vector (7.3589, -6.9258, 4.9583)>
Cube is at location <Vector (0.0000, 0.0000, 0.0000)>
Light is at location <Vector (4.0762, 1.0055, 5.9039)>

>>>

```

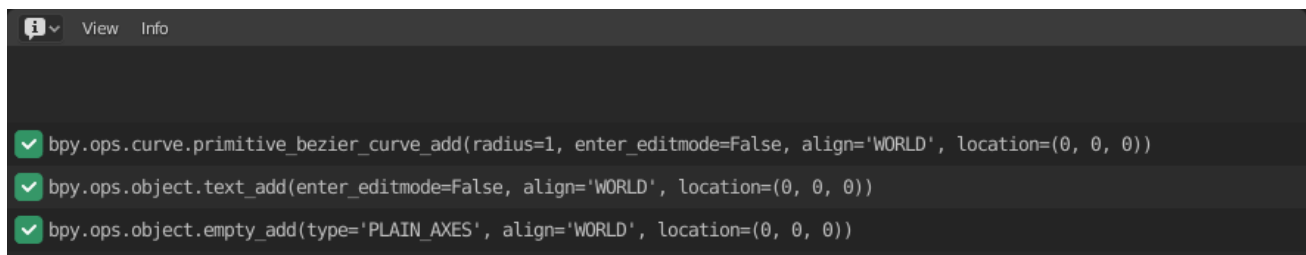
## bpy.ops

The tool system is built around the concept of operators. Operators are typically executed from buttons or menus but can be called directly from Python too.

See the [bpy.ops API documentation](#) for a list of all operators.

### 2.3.16 Info Editor

The Info editor logs executed operators, warnings and error messages. Logged reports can be selected by clicking on them. To select multiple reports, hold down Shift-LMB.



```

View Info

✓ bpy.ops.curve.primitive_bezier_curve_add(radius=1, enter_editmode=False, align='WORLD', location=(0, 0, 0))
✓ bpy.ops.object.text_add(enter_editmode=False, align='WORLD', location=(0, 0, 0))
✓ bpy.ops.object.empty_add(type='PLAIN_AXES', align='WORLD', location=(0, 0, 0))

```

Fig. 506: Info Editor.

## Interface

### Header Menus

#### View Menu

**Area** Area controls, see the *user interface* documentation for more information.

#### Info Menu

**Select All A** Selects the full log history.

**Deselect All Alt-A** Deselects any selected reports.

**Invert Selection Ctrl-I** Selects non-selected reports and deselects existing selection.

**Box Select B** Selects reports inside the defined box.

**Delete X** Removes selected reports from the log.

**Copy Ctrl-C** Copies selected reports to the clipboard.

## Data

### 2.3.17 Outliner

#### Introduction

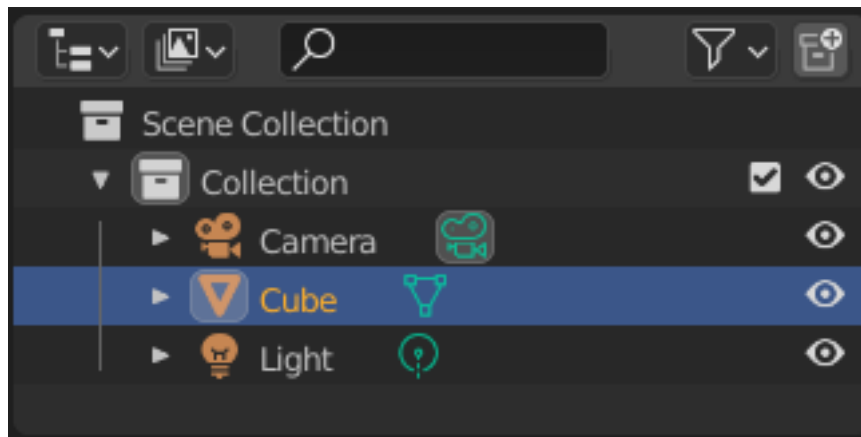


Fig. 507: The Outliner editor.

The *Outliner* is a list that organizes data in the blend-file, i.e. the scene data, Video Sequencer data, or anything that gets stored in a blend-file. The *Outliner* can be used to:

- View the data in the scene.
- Select and deselect objects in the scene.
- Hide or show an object in the scene.
- Enable or disable selection (to make an object “unselectable” in the 3D Viewport).
- Enable or disable the rendering of an object.
- Delete objects from the scene.
- Unlink data (equivalent to pressing the X button next to the name of a data-block).

- Manage collections in the scene.

Each row in the *Outliner* shows a data-block. You can LMB click the disclosure triangle to the left of a name to expand the current data-block and see what other data-blocks it contains. Holding Shift when clicking on the disclosure triangle will expand child data-blocks recursively. LMB Clicking and dragging along disclosure triangles will expand or collapse multiple data-blocks.

### Example

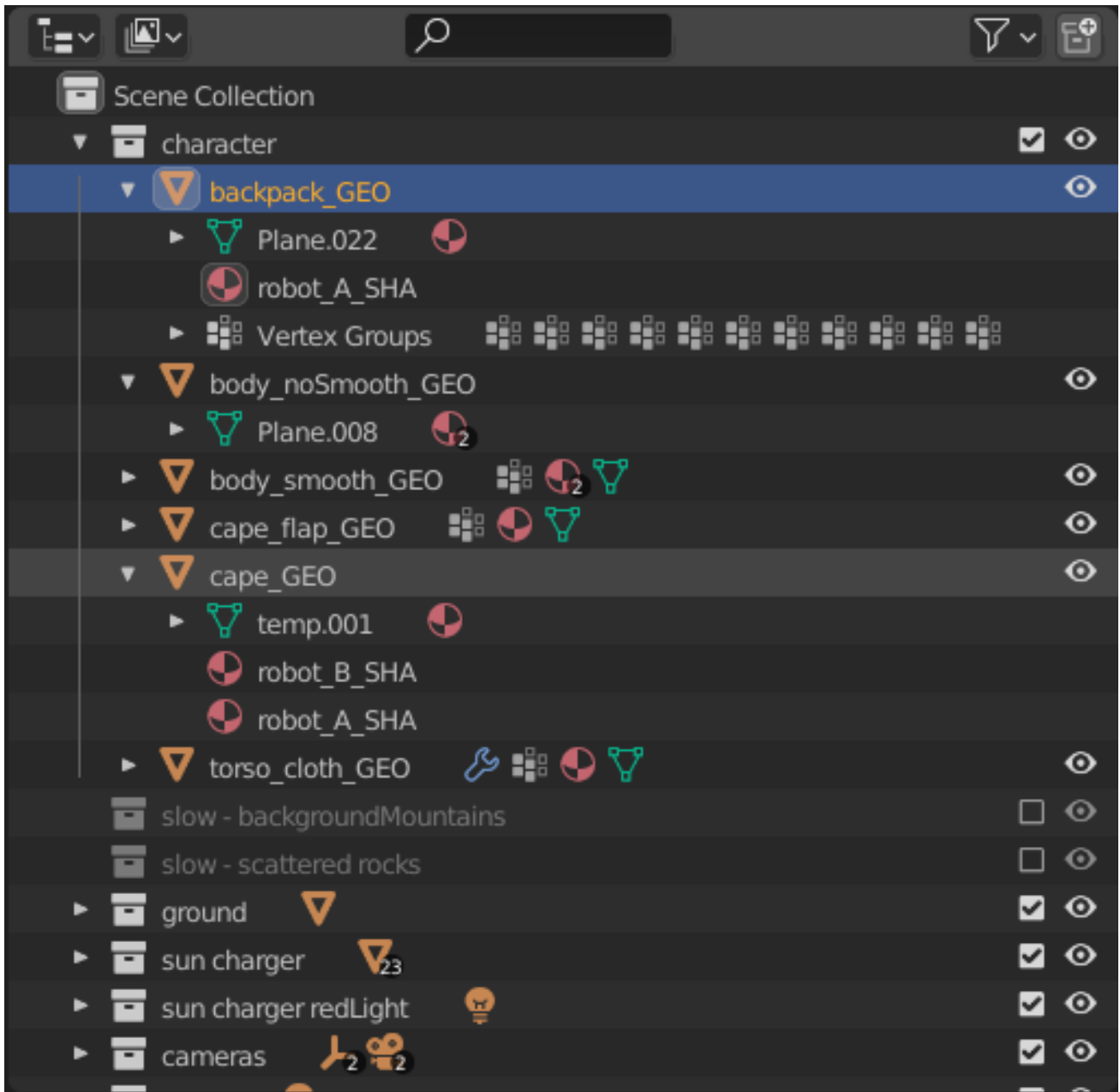


Fig. 508: The Outliner with different kinds of data.

## Interface

### Object Mode

The far left of the Outliner contains a region to toggle the current *Object Mode*. When an object is in a mode other than Object Mode, the mode icon will be displayed in this region. Any other objects that are valid to be added or swapped into the current mode display a dot. Clicking the dot icon will swap that object with the current active object. For modes that support *Multi-Object Editing*, Ctrl-LMB on the dot icon will add that object to the current mode. Clicking the mode icon next to the active object removes it or all other objects from the current mode depending if multiple object are in the same mode.

### Restriction Columns

The following toggles, in the right side of the *Outliner* editor, are available for collections, objects, bones, modifiers and constraints.

By default only the temporary viewport visibility is enabled. The other options can be enabled in the *Restriction Toggles* option in the Outliner *filter*.

- Holding Shift sets or unsets the value to all its child collections or objects.
- Holding Ctrl isolates the object or collection, so they are the only ones with its value set.

**Enable Collection (checkbox, collection only)** Exclude the collection from the view layer.

**Visibility (eye icon)** Toggles the visibility of the object or collection in the 3D Viewport. This is a file-local setting, and does not get imported when this data-block is linked into another blend-file. Objects hidden this way are still part of the *View Layer* and evaluated, so they still affect playback performance.

---

**Note:** The following options are hidden by default and need to be enabled in the Outliner Filter before they can be used.

---

**Selectability (mouse cursor icon)** 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.

**Global Viewport Visibility (screen icon)** This will still render the object/collection, but it will be ignored by all the viewports. Often used for collections with high-poly objects that need to be instanced in other files. Objects hidden this way are no longer part of the *View Layer*, are not evaluated, and such do not negatively affect playback performance.

**Rendering (camera icon)** This will still keep the object visible in the scene, but it will be ignored by the renderer. Usually used by support objects that help modeling and animation yet do not belong in the final images.

**Holdout (collection only)** Mask out objects in collection from view layer.

**Indirect Only (collection only)** Objects in these collections only contribute to indirect light - *Cycles only*.

## Header

### Display Mode

The editors header has a select menu that let you filter what the Outliner should show. It helps to narrow the list of objects so that you can find things quickly and easily.

**Scenes** Shows *everything* the *Outliner* can display (in all scenes, all view layers, etc.).

**View Layer** Shows all the collections and objects in the current view layer.

**Video Sequencer** Lists data, images and videos, that are used by the *Video Sequencer*.

**Blender File** Lists all data in the current blend-file.

**Data API** Lists every *data-block* along with any properties that they might have.

**Orphan Data** Lists *data-blocks* which 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 Fake User by clicking on cross/tick icon in the right side of the Outliner editor.

## Searching

You can search the view for data-blocks, by using Search field in the header of the *Outliner*, The *Filter* menu lets you toggle the following options:

- Case Sensitive Matches Only
- Complete Matches Only

## Filter

**Restriction Toggles** Set which *Restriction Columns* should be visible.

**Sort Alphabetically** Sort the entries alphabetically.

**Sync Selection** Sync Outliner selection to and from the *3D Viewport* and *Video Sequencer* editors. Disable to manage collections, object relations, and scene data without changing the selection state. Selection syncing is only available in Scenes, View Layer, and Video Sequencer display modes.

**Show Mode Column** Show the object mode toggling column in View Layer and Scenes display modes.

**Collections** List the objects and collections under the *collection hierarchy* of the scene. Objects may appear in more than one collection.

**Objects** List of all the objects, respecting the other filter options. Disabled only if you need an overview of the collections without the objects.

### Object State

**All** The default option, no restrictions.

**Visible** List only the objects visible in the viewports. The global and temporary visibility settings are taken into consideration.

**Invisible** List only the objects not visible in the viewports.

**Selected** Lists the object(s) that are currently selected in the 3D Viewport. See *selecting in the 3D Viewport* for more information.

**Active** Lists only the active (often last selected) object.

**Object Contents** List materials, modifiers, mesh data, ...

**Object Children** List the object children. If the *Collections* option is enabled, you will see the object children even if the children are not in the collection. However the Outliner shows them as a dashed line.

**Data-Block** Allows you to filter out certain data-blocks currently present in the scene.

## 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 selected collection.

**Filter ID Type (Orphan Data, Blender File)** Restrict the type of the data-blocks shown in the Outliner.

**Keying Sets (Data API)** Add/Remove selected data to the active *Keying Set*.

**Drivers** Add/Remove *Drivers* to the selected item.

**Purge (Orphan Data)** Remove all unused data-blocks from the file (cannot be undone).

## Selecting and Activating

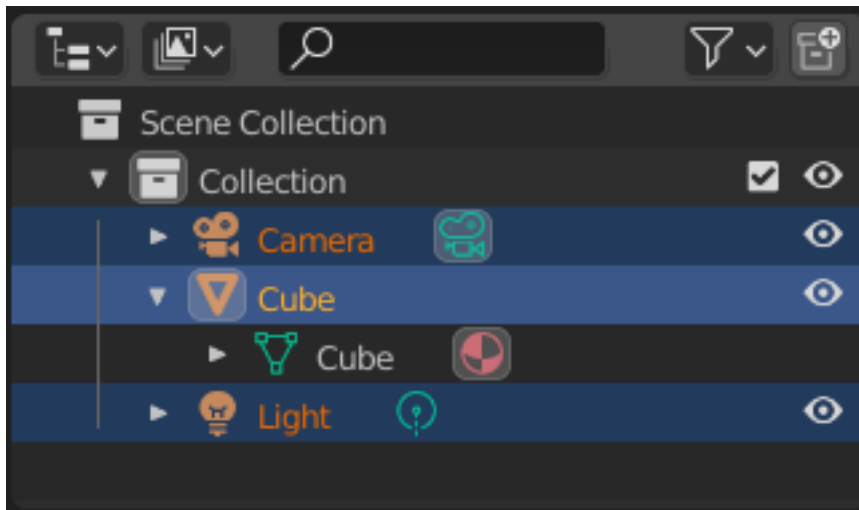


Fig. 509: Selected data-blocks with the Cube active.

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.

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**.

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.

## Editing

### Context Menu

Show the context menu for a data-block with RMB on the icon or name. Depending on the type of the preselected data-block(s), you will have all or part of the following options:

**Copy/Paste** Copy/pastes selected data-blocks.

**Delete** Deletes the selected data-block.

**Select, Select Hierarchy, Deselect** Add object to current selection without making it the active one.

### Collections

Collections are a way Blender uses to organize scenes. Collections contain objects and everything else in a scene. They can include collections themselves and are fully recursive.

#### See also:

Read more about *Collections*.

**New** Creates a new collection.

**Duplicate Collections** Recursively duplicates the collection including all child collections, objects, and object data.

**Duplicate Linked** Duplicate entire hierarchy keeping content linked with original.

**Delete Hierarchy** Deletes the collection and removes all its child objects or collections. It is important to note that this only deletes the collection, if child objects are part of another collection they will stay in the scene collection and their data-blocks will not be deleted from the blend-file.

**Instance to Scene** Creates a new *collection instance*.

**Visibility** Controls the collection's visibility in the 3D Viewport and the final render.

**Isolate** Hides all collections except the selected collection and any parent collections (if any exist).

**Show/Hide** Shows/Hides the selected collection from the *View Layer*.

**Show/Hide Inside** Shows/Hides all items that are a member of the selected collection, include child collections, from the *View Layer*.

**Enable/Disable in Viewports** Enables/disables drawing in the *View Layer*.

**Enable/Disable in Renders** Enables/disables visibility of the collection in renders.

**View Layer** Controls the collection's interactions with the *View Layer*.

**Disable/Enable in View Layer** Disables/Enables the collection from the view layer.

**Set Color Tag** Assigns or clears a collection's *color tag* for the selected collection.

## ID Data

**Unlink** To unlink a data-block from its "owner" (e.g. a material from its mesh).

**Make Local** To create a "local" duplicate of this data-block.

**Make Single User** This feature is not yet implemented.

**Delete** Deletes the selected data-block.

**Add Library Override** Add a local *override* of this linked data-block.

**Add Library Override Hierarchy** Add a local *override* of this linked data-block, and its hierarchy of dependencies.

**Convert Proxy to Override** Converts the selected *Proxy* data-block to an *override* and its hierarchy of dependencies. See also, *Converting Proxies to Library Override*.

**Reset Library Override** Reset this local *override* to its linked values.

**Reset Library Override Hierarchy** Reset this local *override* to its linked values, as well as its hierarchy of dependencies. This allows you to update local overrides when the relationship between data-blocks changed in the linked library data.

**Resync Library Override Hierarchy** Rebuilds the local *override* from its linked reference, as well as its hierarchy of dependencies.

**Delete Library Override Hierarchy** Deletes the local *override* (including its hierarchy of override dependencies) and relinks its users to the linked data-blocks.

**Remap Users** Remap Users of a data-block to another one (of same type of course). This means you can e.g. replace all usages of a material or texture by another one.

**Copy/Paste** Copy/pastes selected data-blocks.

**Add Fake User, Clear Fake User** Adds a "dummy" (fake) user so that the selected data-block always gets saved even if it has no users. The fake user can be removed with *Clear Fake User*.

**Rename F2** Renames the selected data-block.

**Select Linked** Selects the linked data, see *Select Linked* for more information.

## View

The view menu is part of the context menu and supported in all the Outliner elements.

**Show Active Period** Centers the Tree View to selected object.

**Show Hierarchy Home** To collapse all levels of the tree.

**Show/Hide One Level NumpadPlus/ NumpadMinus** Expand one level down in the tree or collapse one level using the keyboard shortcuts.

## Usage

### Relations Management

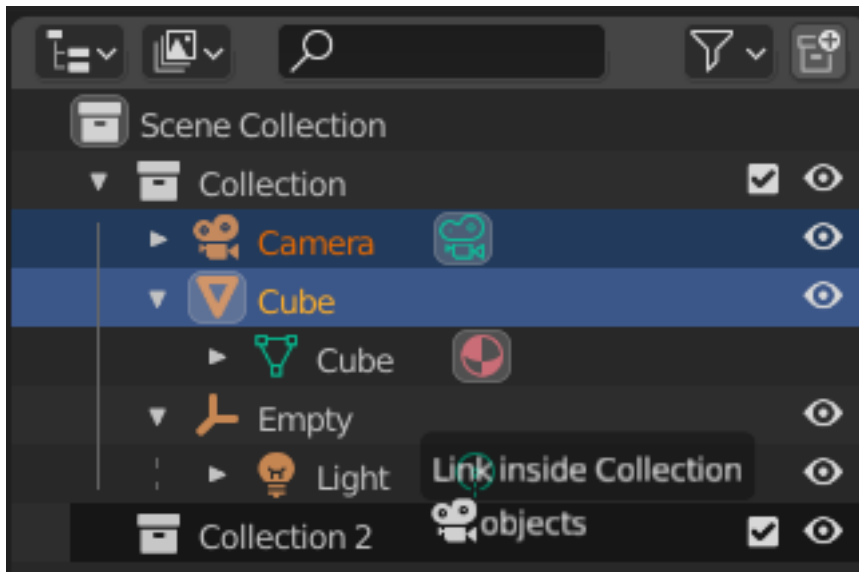


Fig. 510: Linking objects to a collection.

Data-blocks can be dragged and dropped to manage data relations in the Outliner. To begin a drag and drop, LMB click and drag from the name or icon of a data-block.

Objects can be moved to collections by dropping on the name or contents of a collection. To link an object to a collection, hold **Ctrl** while dropping. To set parent-child relations between objects, drop an object onto another object while holding **Shift**.

---

**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, & Visual Effects

You can manage *Modifiers*, *Constraints*, and *Visual Effects* from the Outliner in a couple ways:

- To change the order within the *stack* select the desired modifier and move it above or below other modifiers.
- To copy a single modifier to another select the modifier and drag it on top of the desired object.
- To copy the whole modifier stack to another object select the modifier icon and drag in to the desired object.

### Drag & Dropping to 3D Viewport

#### Objects & Object Data

Dragging object data-blocks from the Outliner to the *3D Viewport* creates a *duplicate* of the object. Dragging *object data* data-blocks from the Outliner to the 3D Viewport creates a *linked duplicate* of the object.

## 2.3.18 Properties

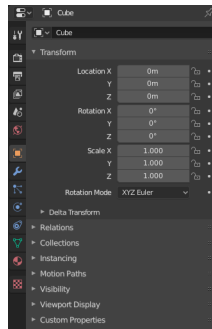


Fig. 511: The Properties, with *Object* properties shown.

The Properties shows and allows editing of many active data, including the active scene and object.

### Tabs

The Properties 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 View) and the current *workspace*.

### Scene

These tabs contain settings for the active scene.

- Render: *Eevee*, *Cycles* or *Workbench* settings
- *Output*
- *View Layer*
- *Scene*
- *World*

### 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

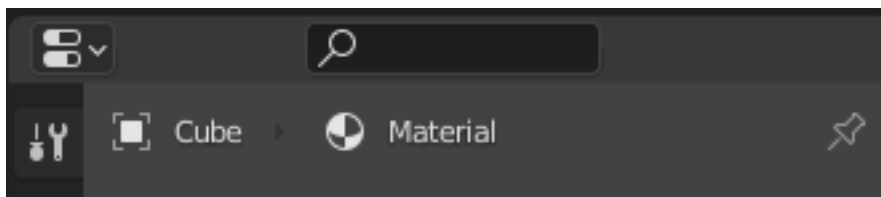


Fig. 512: The header of the Properties.

**Display Filter Ctrl-F** Allows you to search for a property by its name. The results will be highlighted with their 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.

### 2.3.19 File Browser

The File Browser is used in all the file-related operations. These include:

- Opening and saving blend-files.
- Browsing inside other blend-files, when appending or linking data-blocks, see *Linked Libraries*.
- Importing from/exporting to other file formats.
- Picking new locations for existing file paths (images, videos, fonts...).

The most common way to use this editor is through modal operators (like opening or saving a blend-file). It will appear maximized, waiting for the operation to complete, and then close and return to the former screen layout.

---

**Note:** You can always select several entries in the File Browser, the last selected one is considered as 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, other selected ones will just be ignored.

---

You can also keep the File Browser open, as any other editor type, to browse through the file system. The main purpose of this is to be able to drag-and-drop media files:

- Images into the *3D Viewport* (to set as background or apply as material texture).
- Media files into the *Video Editing*.

#### Header

**Navigation** Icon buttons for navigation of files.

**Left Arrow Backspace, Alt-Left** Move to previous folder (in navigation history).

**Right Arrow Shift-Backspace, Alt-Right** Move to next folder (in navigation history).

**Up Arrow P, Alt-Up** Move up to parent directory.

**Cycle Arrows R, NumpadPeriod** Refresh current folder.

**Create Directory I** Will ask you to confirm and create a new directory inside current one, scroll to it in the main view, and let you enter its name.

**Display Mode** Controls 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*.

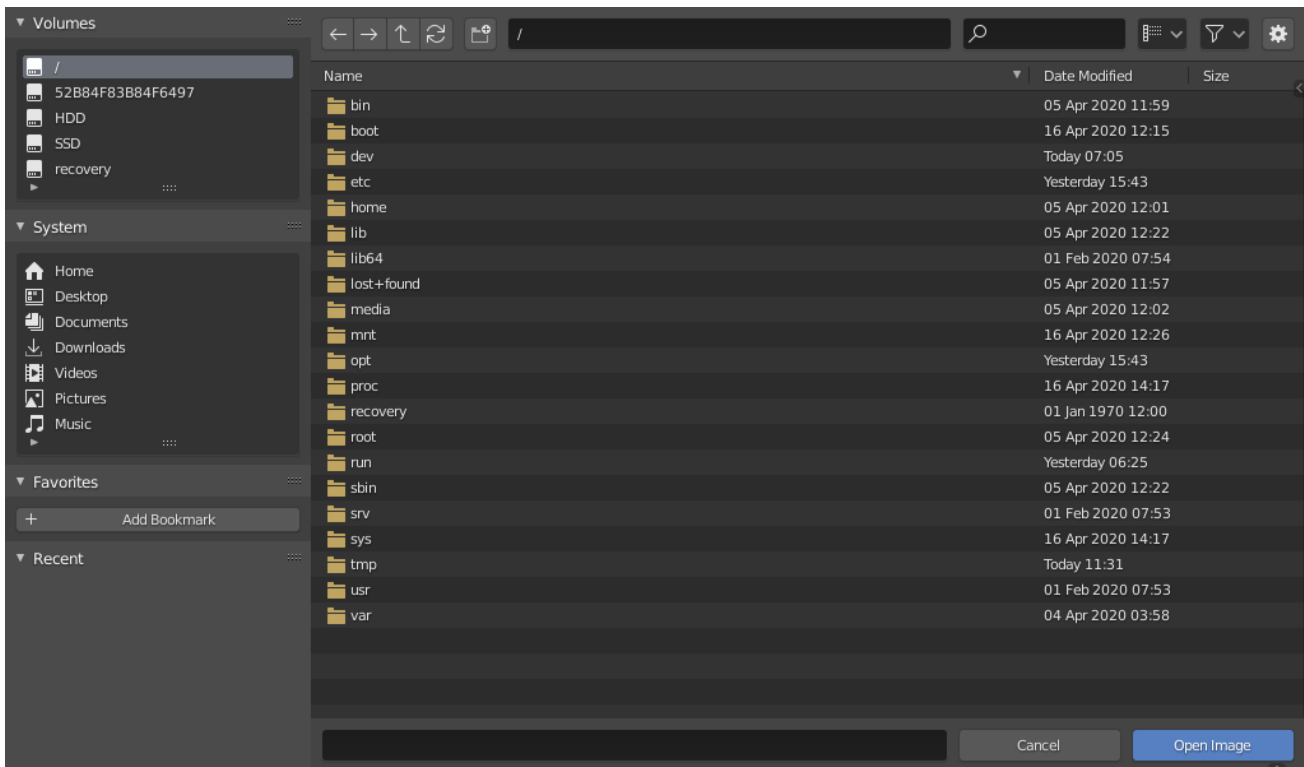


Fig. 513: The File Browser.

**Sorting** Sorts items by one of the four proposed methods (alphabetically, by types, by dates, or by size):

**Show Hidden H** Shows hidden files (starting with a `.`).

### View Menu

**Display Size** The size of the thumbnails, or the width of the columns.

**Recursion** The number of directory levels to show at once in a flat way.

- None (default, shows only the current directory content)
- Blend File (the whole content of a blend-file, only available when linking or appending data-blocks)
- One Levels
- Two Levels
- Three Levels

---

**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 will allow you to 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 (typically, it will be exponential, showing three levels at once may take three orders of magnitude more time to be fully listed).

## File Filtering

To the right of the header are the filtering options.

The first “funnel” button controls whether filtering is enabled or not.

**File Types** Filters files by categories, like folders, blend-files, images, etc.

**Data-Block Types** When appending or linking, you can also filter by data-block categories, like scenes, animations, materials, etc.

**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 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).

## File Region

**File Path** Text field for the current folder path. Tab will auto-complete an existing path. If you type a non-existing directory path, you will be prompted to create that new directory.

**File Name** Text field to edit the file name and extension. When saving, if the background is red, a file with same name already exists in the folder. Tab will auto-complete to existing names in the current directory.

**Increment Filename -, +** Removes/Decreases or adds/increases a trailing number to your file name (used e.g. to store different versions of a file).

**Confirm Return** The main button to validate the operation, which defines its name. Double-clicking on a non-directory item will have the same effect.

**Cancel Esc** Cancels the file selection (and the underlying operation), and closes the File Browser. Using the *Back to Previous* button in the *Topbar* will have the same effect.

## Toolbar

The left region is divided in two areas, Bookmarks on top, and the Operator panel at the bottom.

## Bookmarks

The top one displays different ways to quickly access some directories, in four *lists*. Clicking on one of the shortcut entries will immediately navigate to that folder.

**Volumes** Contains all OS-defined available volumes, e.g. drives or network mounts.

**System** Contains OS-defined common directories, like the main user folder...

**Favorites** Contains folders that you want to be able to access often without having to navigate to them in the File Browser. To the right of that list are buttons to perform basic management actions on your favorites, e.g. add/remove an entry, move it up or down in the list, etc.

**Recent** Contains recently accessed folders. The X button to the right allows you to fully erase 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 Panel

Shows the options of the calling operator. Besides 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*.

## Main Region

### Navigation

**Entering a Directory** A single LMB click on a directory enters 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 move it to the directory of the dropped file, and the file will be selected.

### Selection

**Select** Both LMB and RMB clicks work. Holding **Shift** will extend the items selection.

**(De)select All A** Toggles selecting all files.

**Dragging** Dragging with LMB starts a *box selection*.

### Arrow Keys

It is also possible to select/deselect files by “walking” through them using the arrow keys:

- Just using an arrow key, the next file in the chosen direction will be selected and all others deselected.
- Holding down **Shift** while doing this does not deselect anything so it extends to the selection, plus it allows to deselect files by navigating into a block of already selected ones (minimum two files in sequence).
- Holding down **Shift-Ctrl** further selects/deselects all files in between.

If no file is selected, the arrow key navigation selects the first or last file in the directory, depending on the arrow direction.

If you select a directory and hit **Return**, you will go into that directory (and highlighting ‘parent’ `..` entry will bring you up one level).

## File Management

**Delete Files** **Delete, X** Delete the currently selected files.

**Rename** **F2** Change the name for the currently selected file or directory.

**Warning:** Be careful, there is no way to undo those actions!

## Previews

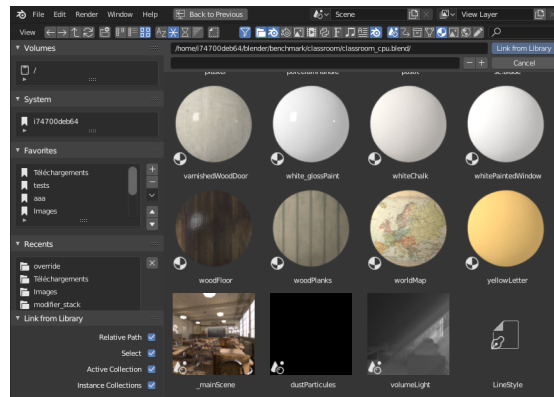


Fig. 514: The File Browser in *Thumbnail* mode.

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*

See *Blend-files Previews* for how to manage Blender data previews.

## 2.3.20 Preferences

### Introduction


#### Reference

**Menu** *Edit* → *Preferences...*

**Hotkey** F4, P

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.

### Managing Preferences

Default preferences are managed from the  menu in the preferences window.

The following items are available in this menu:

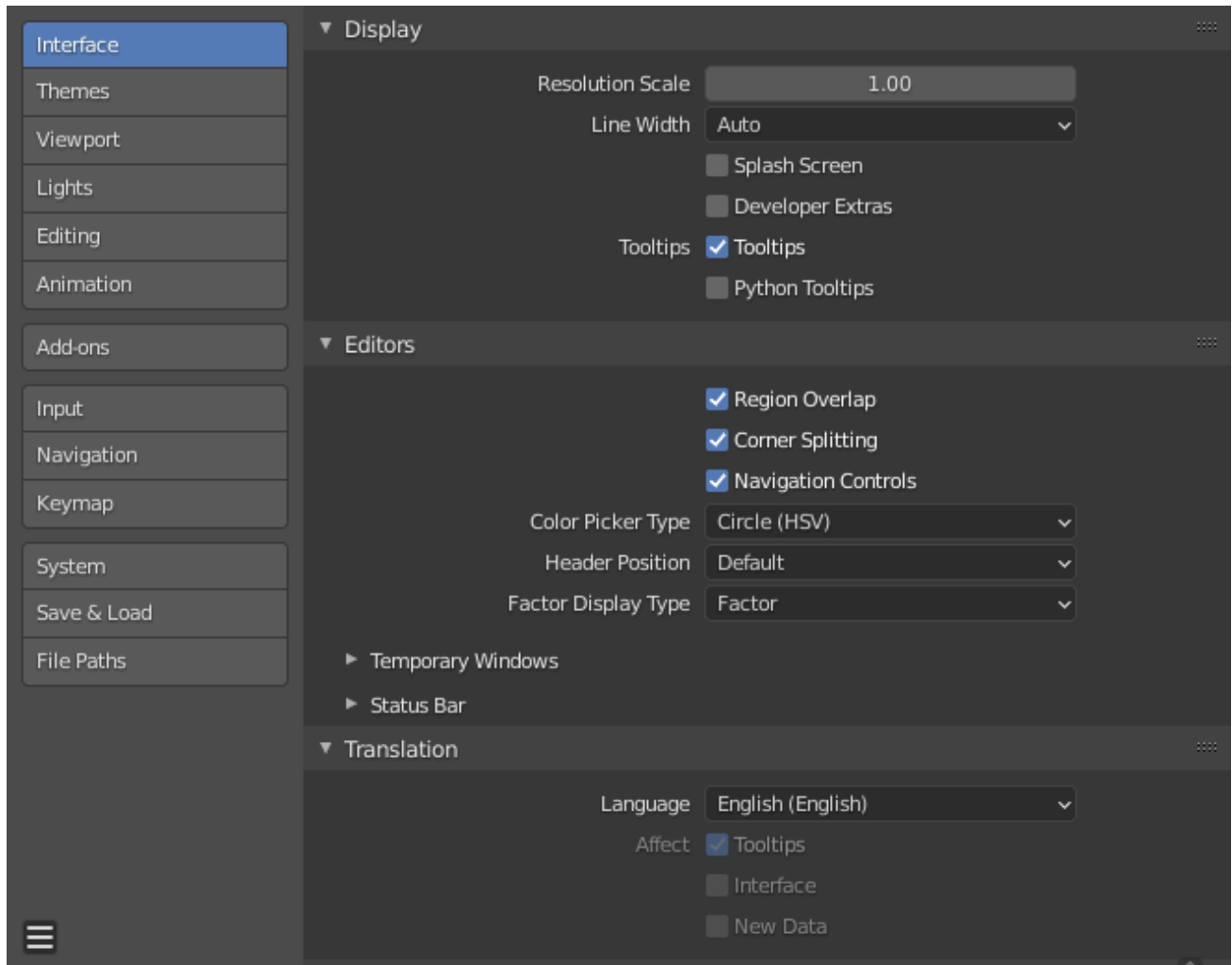


Fig. 515: Blender Preferences window.

**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 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, re-setting 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, Auto, 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:

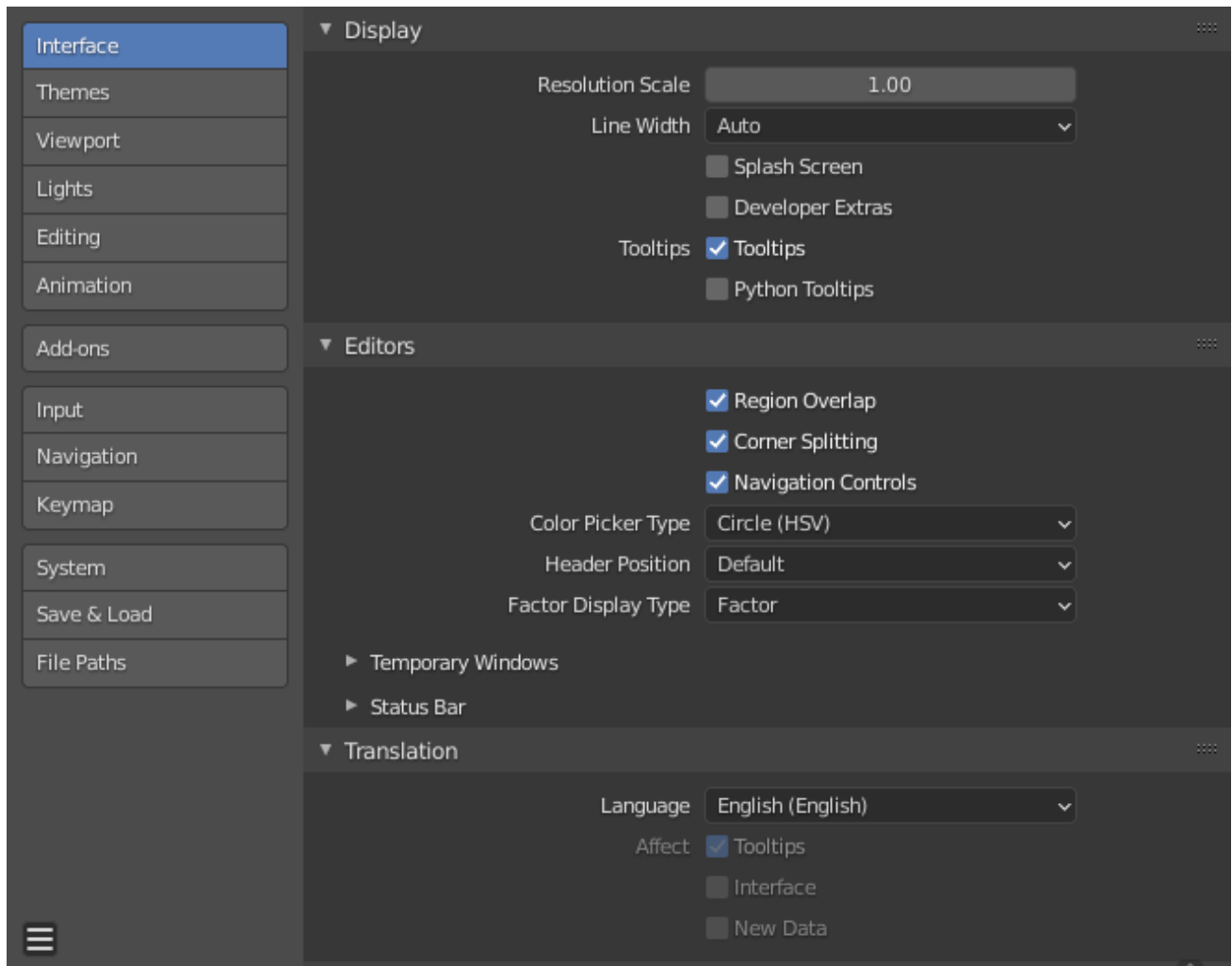
**Operator Search** See *Interface Searching*.

#### 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 View

**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

**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.

## Editors

**Region Overlap** This makes regions overlap the viewport. It means that the *Toolbar* and *Sidebar* regions, will be displayed overlapping the main area.

**Corner Splitting** Split and join by dragging from the corners. When disabled, you can use the context menu for area separators to perform these operations.

**Navigation Controls** Show navigation controls at top right of the area. This impacts the 3D Viewport as well as image spaces.

---

**Note:** If you're familiar with navigation key shortcuts, this can be disabled.

---

**Color Picker Type** Choose which type of *Color Space* you prefer. It will show when clicking LMB on any color field.

See the different color picker types at the *Color picker* page.

**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 Windows

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.

**Full Screen** 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:

**Full Screen** 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.

## Translation

**Language** The language used for translating the user interface (UI). The list is broken up into categories determining how complete the translations are.

### Affect

**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).

**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** Length of animation when opening Pie Menus.

**Tap Key Timeout** Keystrokes held longer than this will dismiss the menu on release (in 1/100ths of a second).

**Recenter Timeout** The window system tries to keep the pie menu within the window borders. Pie menus will use the initial mouse position as center for this amount of time, measured in 1/100ths of a second. This allows for fast dragged selections.

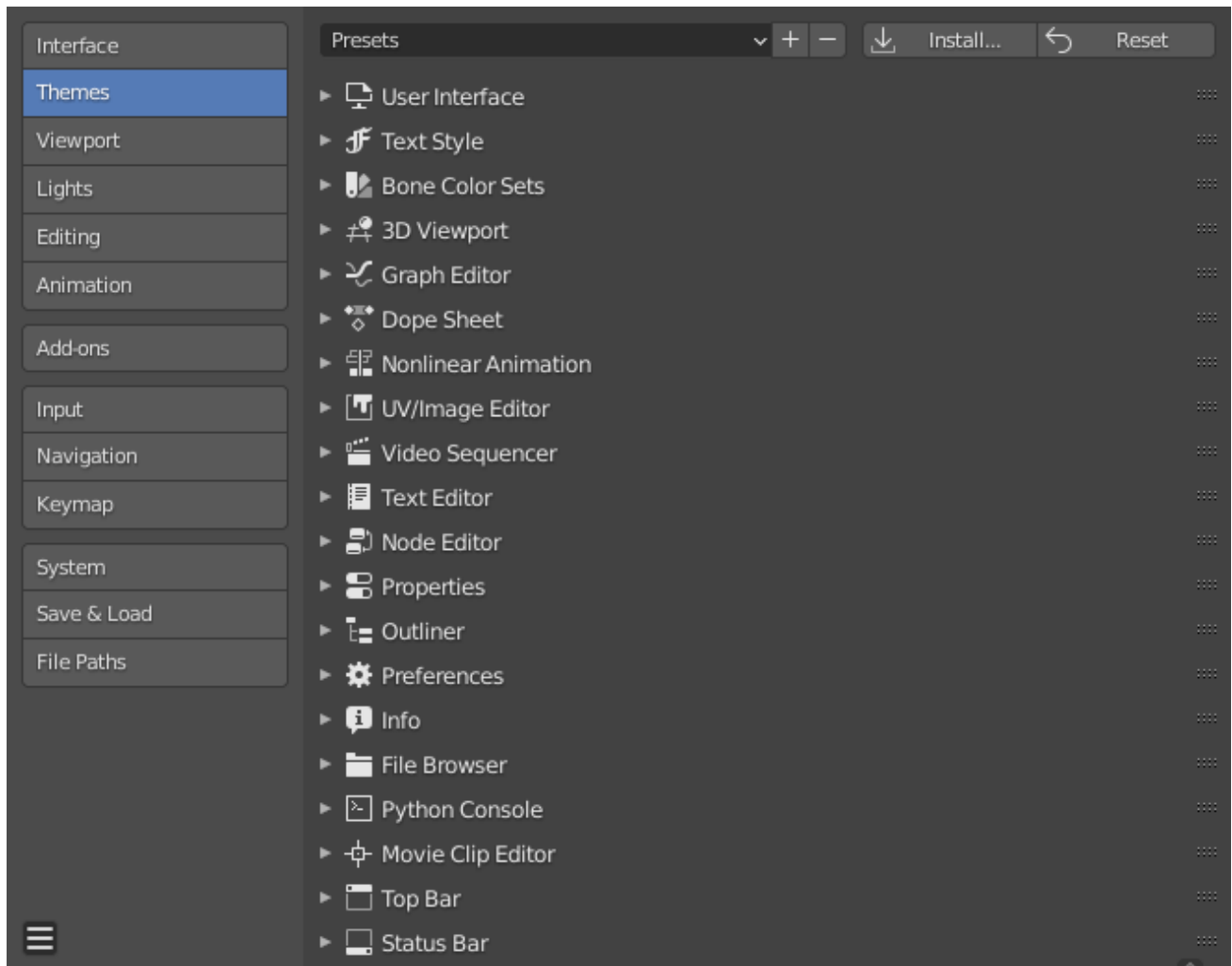
**Radius** The size of the Pie Menu set with the distance (in pixels) of the menu items from the center of the pie menu.

**Threshold** Distance from center before a selection can be made.

**Confirm Threshold** Distance threshold after which selection is made (zero disables).

## 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 View* or the *Graph Editor* can also be changed.

Themes use Blender's preset system to save a theme. This will save the theme to an XML file in the `./scripts/presets/interface_theme/` subdirectory of one of the *configuration directories*.

## Viewport

### Display

#### Show

**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 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



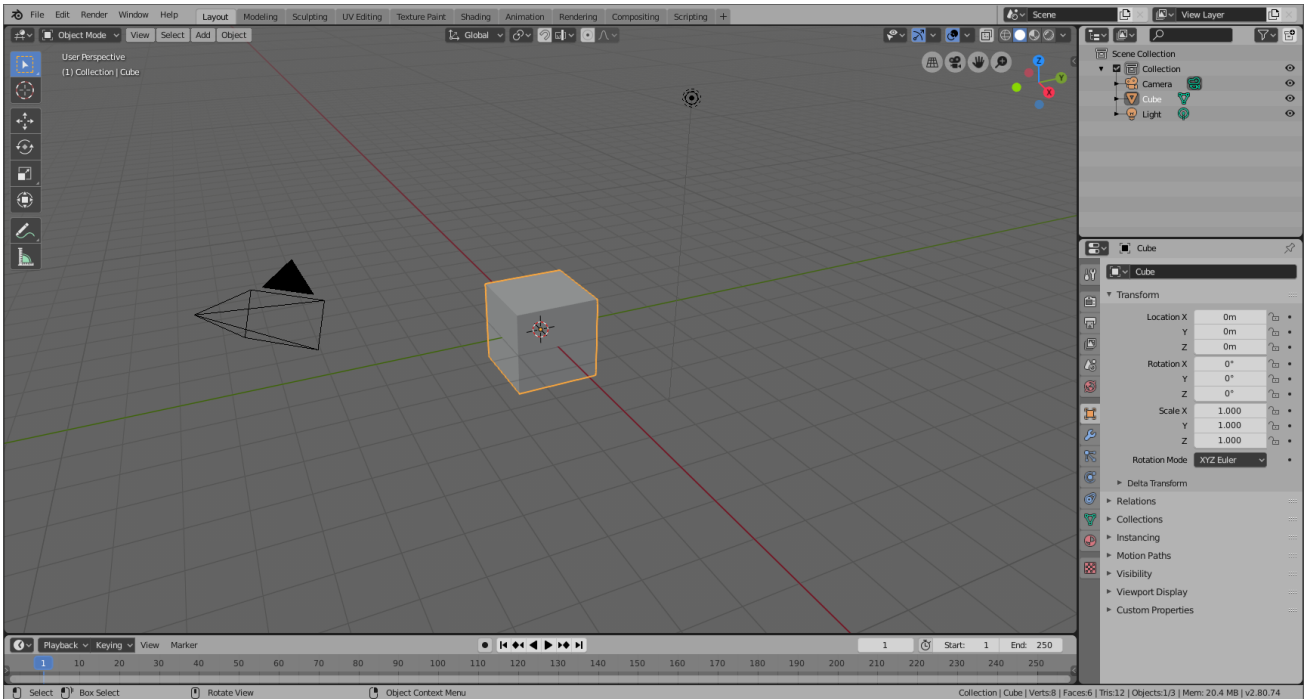


Fig. 516: Blender comes bundled with a small selection of themes. This is an example of the theme *Blender Light*.

cannot be reached.

**Gizmo Size** Diameter of the gizmo.

**HDRI Preview Size** Diameter of the HDRI sphere overlay.

### 3D Viewport Axis

**Interactive Navigation** Display the axis as an interactive gizmo.

**Click** Sets the viewport to display along this axis.

**Drag** Orbit the view.

**Simple Axis** Display simple, less intrusive axis in the viewport.

**Size** Size of the simple axis.

**Brightness** How vivid the colors of the simple axis are.

**Off** Disables the viewport axis.

### 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 and *Edit Mode*, since edges do not blend into other shaded regions.

**Edit Mode** Display smooth wire in Edit Mode, without this wires will be rendered aliased.

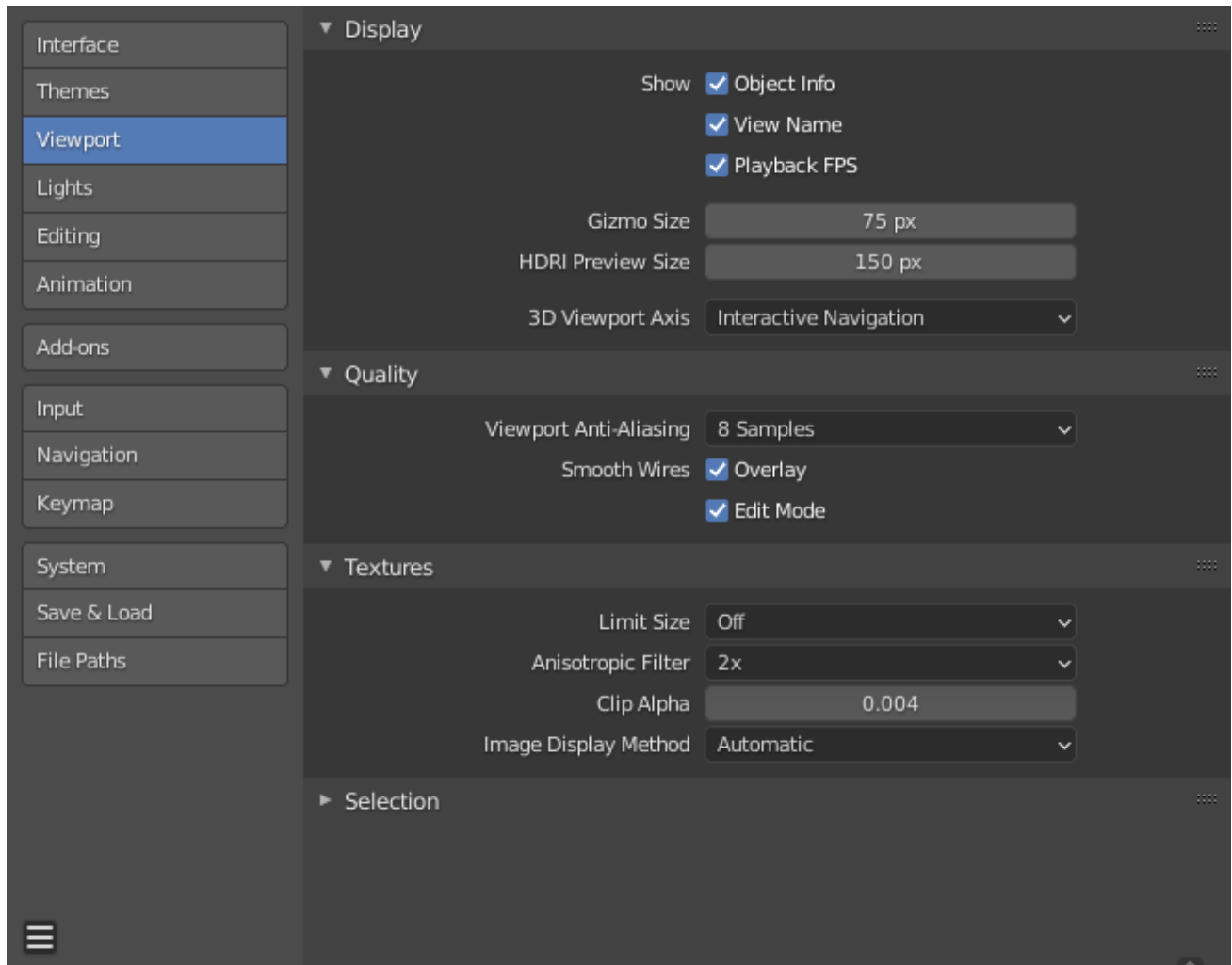


Fig. 517: Blender Preferences Viewport section.

## 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

**OpenGL 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.

## Lights

### 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.

### Light

**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.

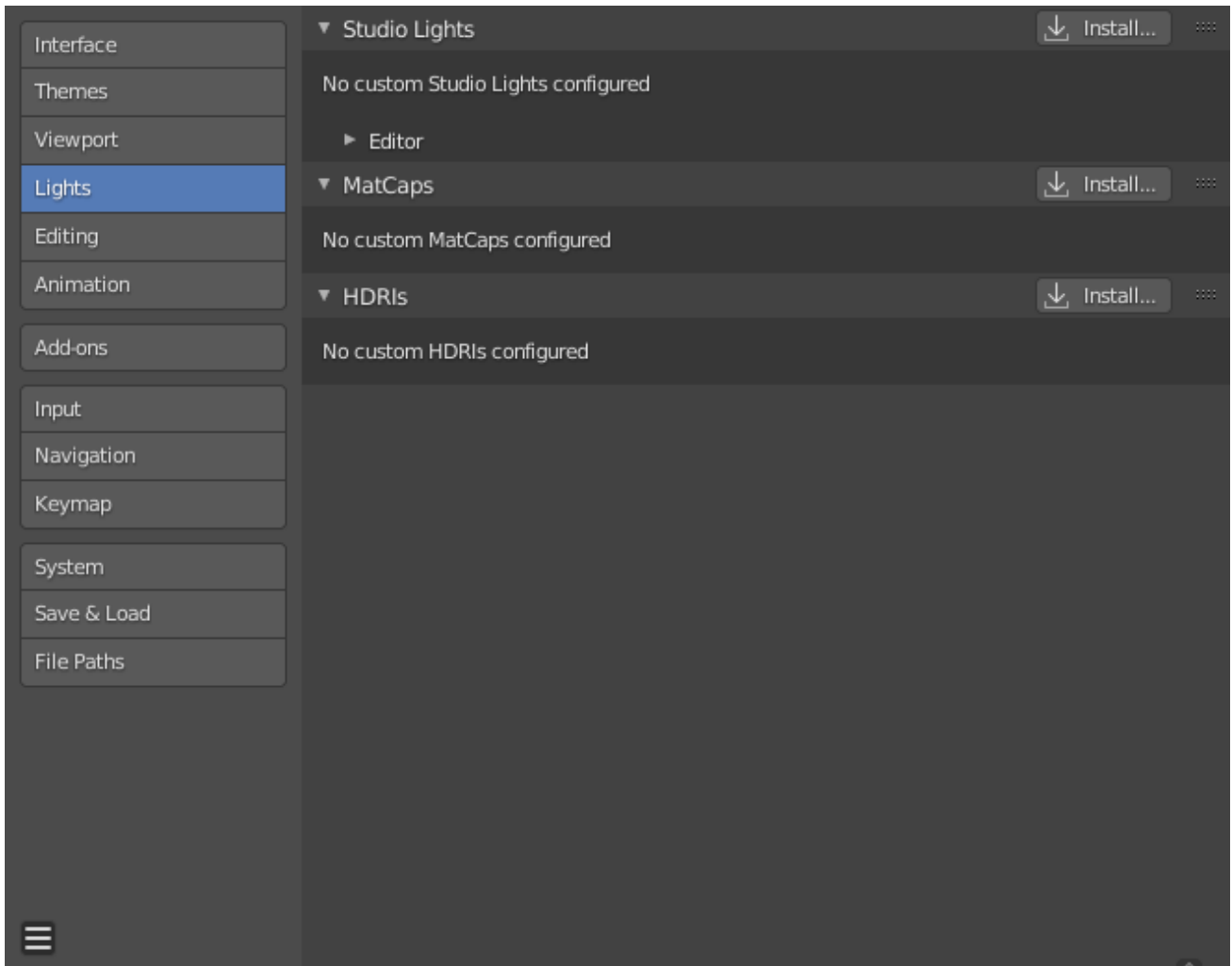


Fig. 518: Blender Preferences Lights section.

## MatCaps

This panel manages *MatCap* image files which can be 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.

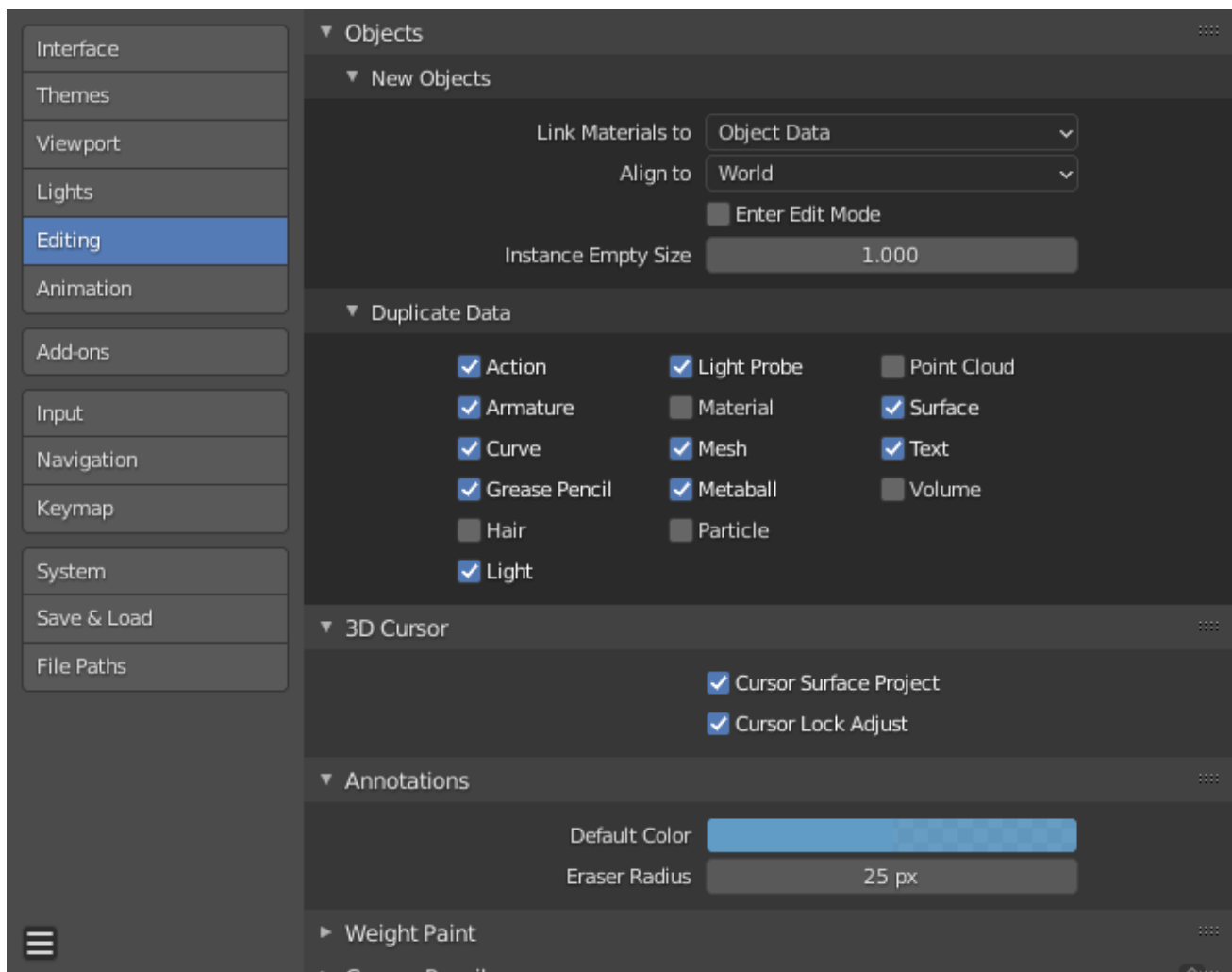
The diffuse pass is multiplied with the base color of the objects and the specular pass is added on top. MatCaps, that only have a diffuse pass tend to look very metallic, with a separate specular pass it is possible to simulate a wider variety of materials.

## 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 while the Object Data "ObData" data-block contains information about the mesh.

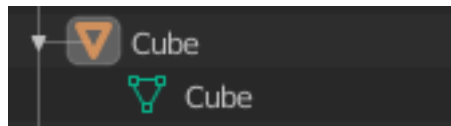


Fig. 519: Example for a mesh.

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.



Fig. 520: 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.

### Duplicate Data

The *Duplicate Data* 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 *Duplicate Data* 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.*

### Custom Weight Paint Range

*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

#### Distance

**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.*

### 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.

**Node Auto-offset Margin** Margin to use for *offsetting nodes*.

### Animation

The *Animation* section lets you manage settings related to *Animation*. This includes how editors look and also some different tools properties.

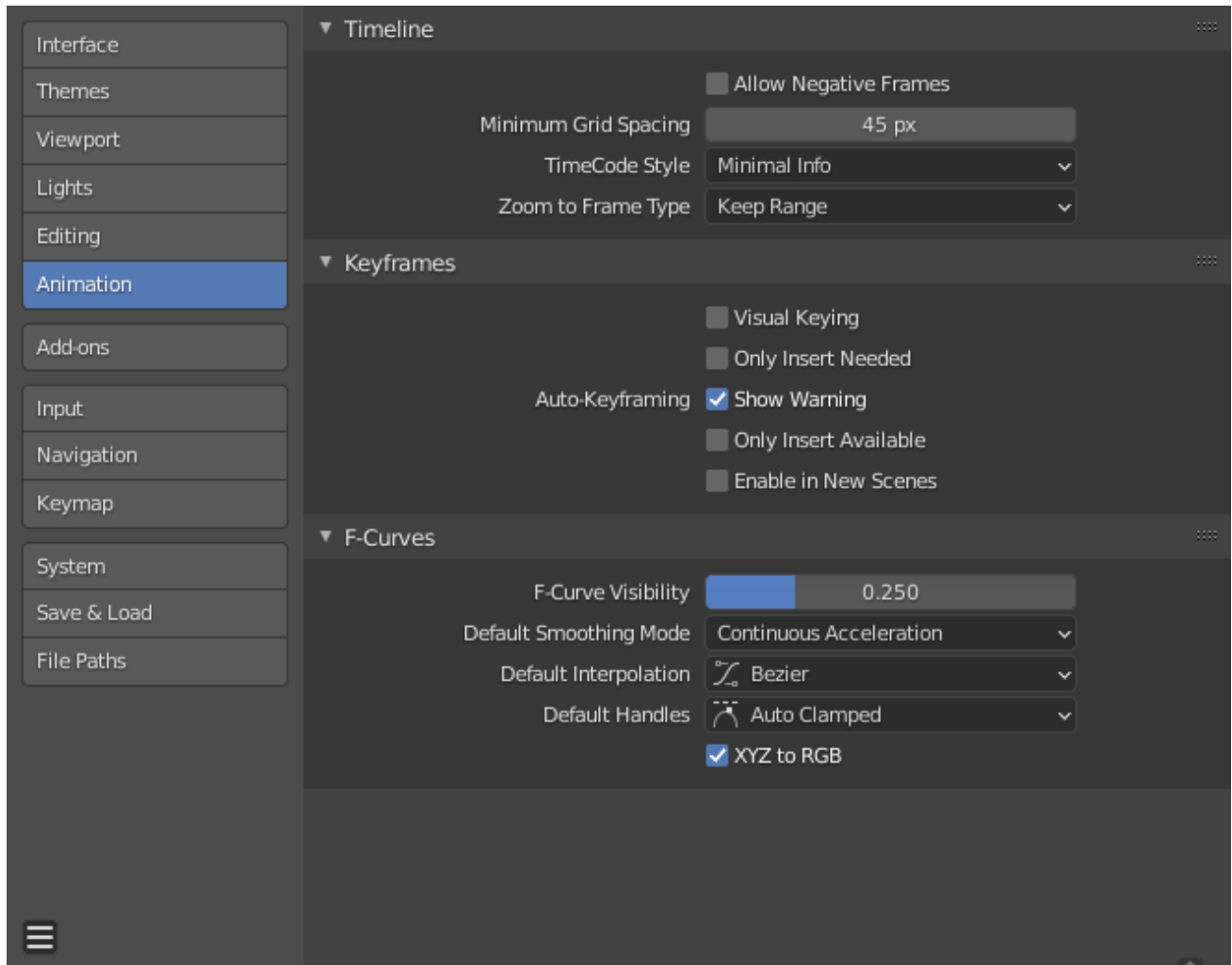


Fig. 521: Blender Preferences Animation section.



## 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. The format uses ‘+’ as a separator for sub-second frame numbers, with left and right truncation of the timecode as necessary.

**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.

**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.

**Only Insert Needed** This will only insert keyframes if the value of the property is different.

### Auto-Keyframing

**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.

**Enable in New Scenes** Enables *Auto Keyframe* by default for new scenes.

### See also:

Learn more about *Auto-Keyframing*.

## F-Curves

These settings control how *F-Curves* look and their default behavior.

**F-Curve Visibility** Opacity that unselected *F-Curves* stand out from 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.

## Add-ons

The *Add-ons* section lets you manage secondary scripts, called “Add-ons” that extends Blender’s functionality. In this section you can search, install, enable and disable Add-ons.

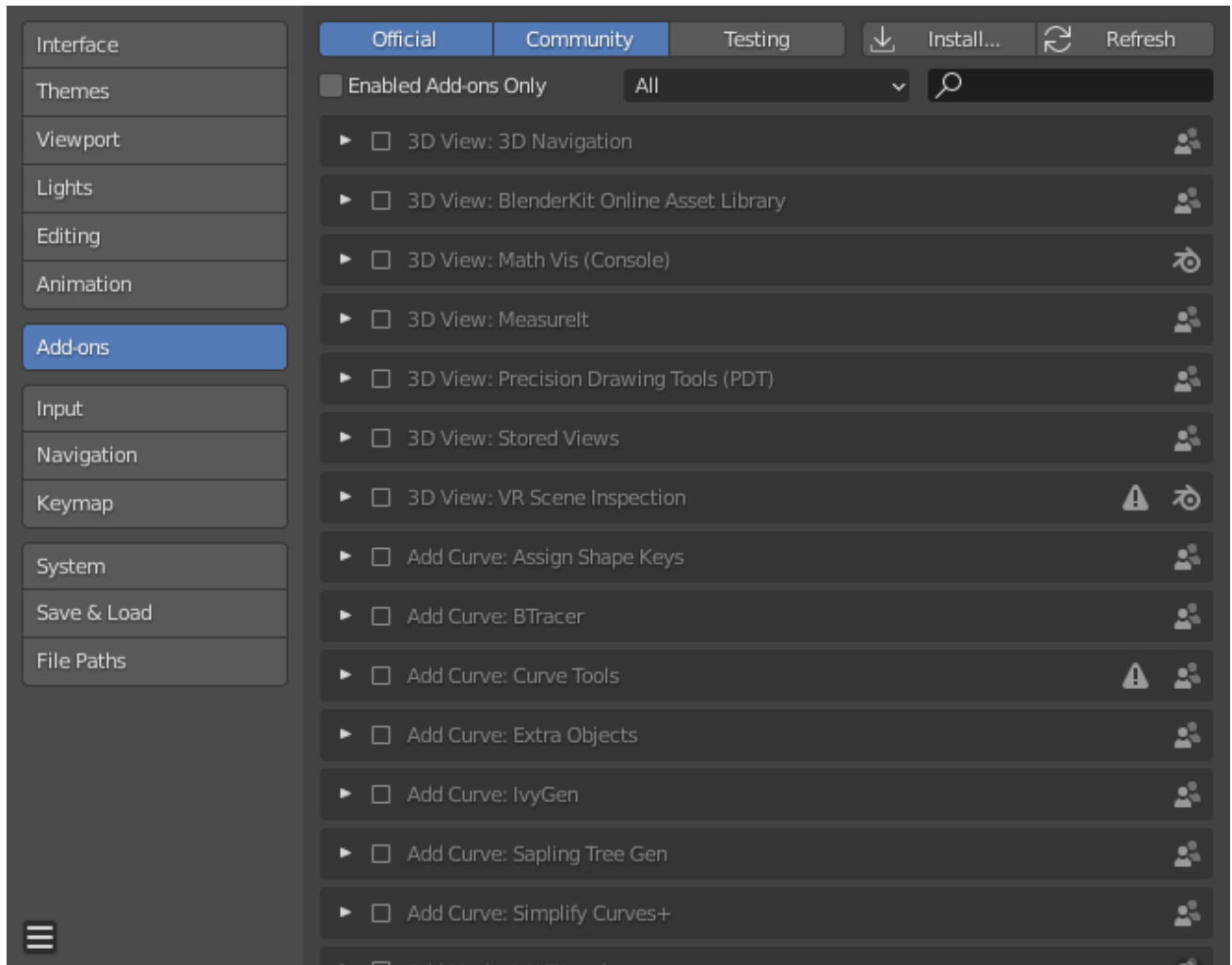


Fig. 522: Blender Preferences Add-ons section.

## Finding Add-ons

**Searching** Blender comes with some pre-installed Add-ons already, ready to be enabled. But you can also add your own, or any interesting ones you find on the web.

**Supported Level** Blender’s add-ons are split into two groups depending on who writes/supports them:

- **Official:** Add-ons that are written by Blender developers.
- **Community:** Add-ons that are written by people in the Blender community.

**Enabled Add-ons Only** Shows only enabled add-ons for the current *Category*.

**Category** Add-ons are divided into categories by what areas of Blender they affect.

## Enabling and Disabling

Enable and disable an add-on by checking or unchecking the box to the right of the add-on you chose, as shown in the figure below.

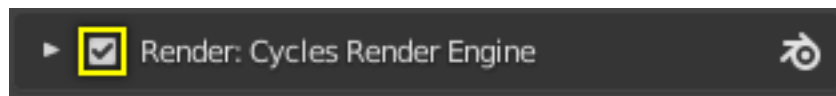


Fig. 523: Enabling an add-on.

The add-on functionality should be immediately available.

---

**Note:** Add-ons that activate or change multiple hotkeys have a special system of activation. For example, with the “UI: Pie Menu Official” add-on for each menu there’s a selection box to activate the menu and its hotkey.

---



---

**Tip:** If the Add-on does not activate when enabled, check the *Console window* for any errors that may have occurred.

---

## 3rd Party Add-ons

There are hundreds of add-ons that are not distributed with Blender and are developed by others. To add them to the list of other add-ons, they must be installed into Blender.

To install these, use the *Install...* button and use the File Browser to select the .zip or .py add-on file.

Now the add-on will be installed, however not automatically enabled. The search field will be set to the add-on’s name (to avoid having to look for it), Enable the add-on by checking the enable checkbox.

**Refresh** Scans the *Add-on Directory* for new add-ons.

---

**Tip:** User-Defined Add-on Path

You can also create a personal directory containing new add-ons and configure your files path in the *File Paths* section of the *Preferences*. To create a personal 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. Open the *File Paths* section of the *Preferences*.
4. Set the *Scripts* file path to point to your script directory (e.g. `my_scripts`).
5. Save the preferences and restart Blender for it to recognize the new add-on location.

Now when you install add-ons you can select the *Target Path* when installing 3rd party scripts. Blender will copy newly installed add-ons under the directory selected in your Preferences.

---

## 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.

## 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

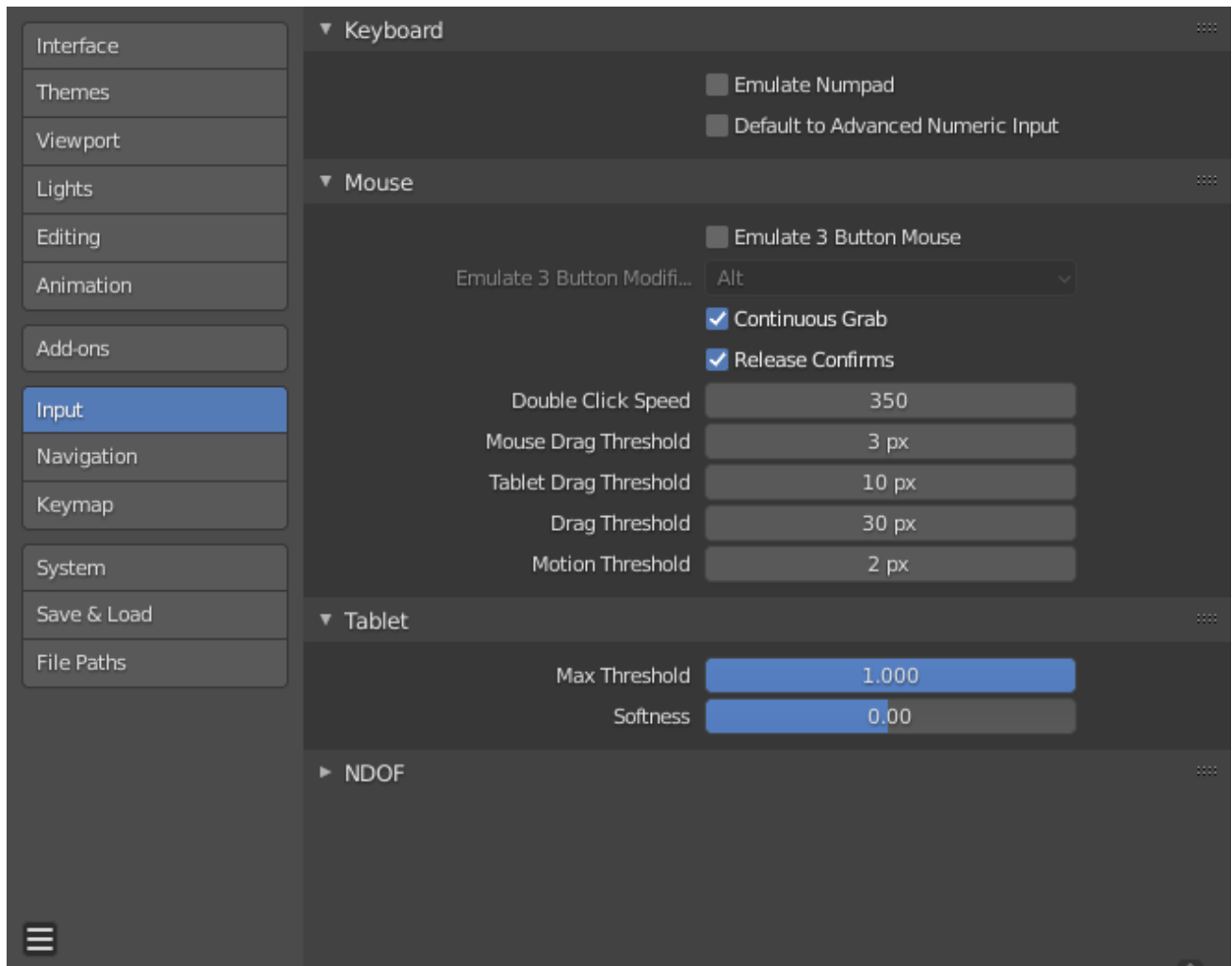
**Emulate 3 Button Mouse** Blender can be configured to work with pointing devices which do not have a MMB. The functionality of the three mouse buttons by holding `Alt-LMB`.

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 (*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.

**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.

---

## 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. Note, this requires restarting Blender for changes to take effect.

**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 a *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.

**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

### Orbit & Pan

**Orbit Method** Choose your 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.

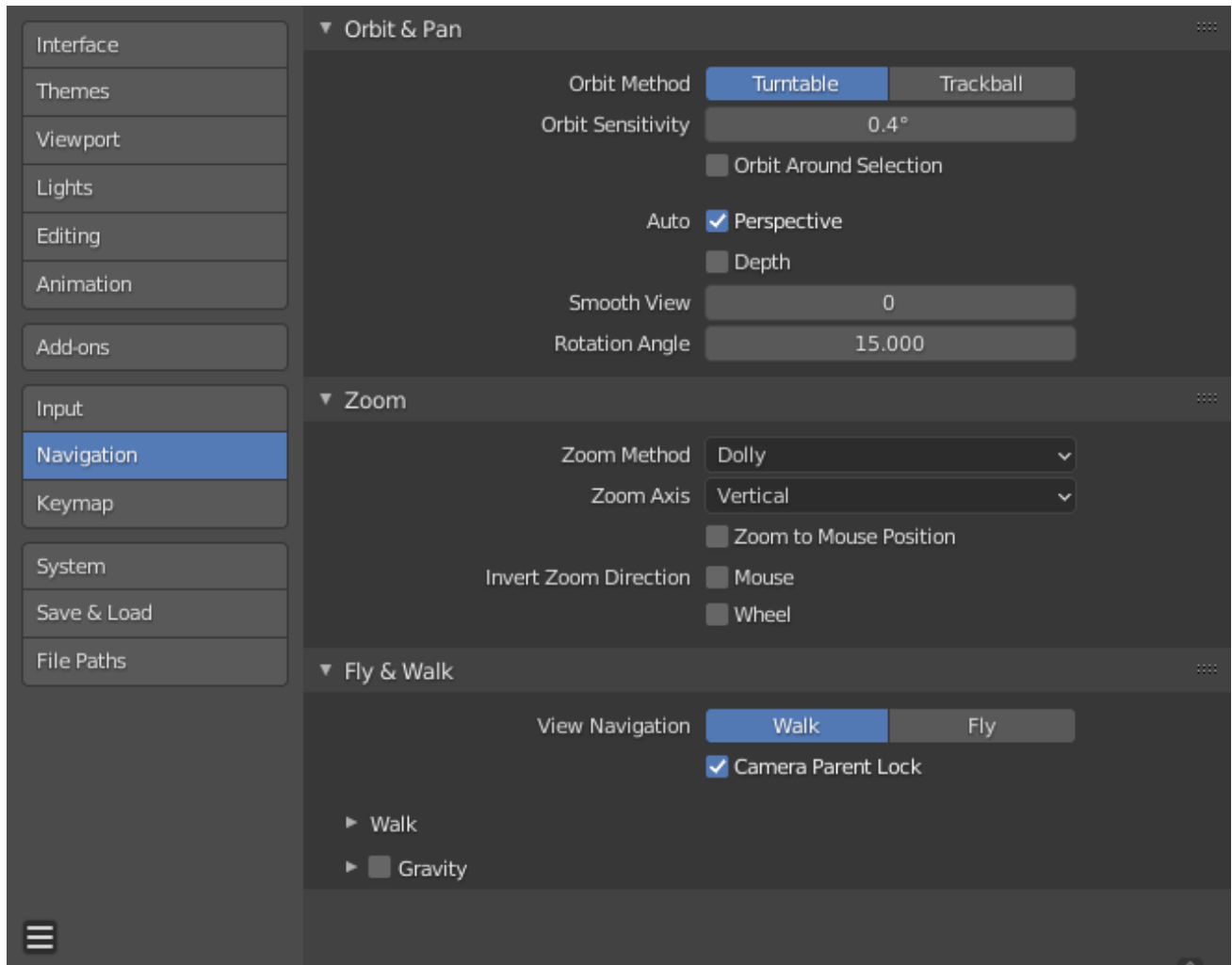


Fig. 524: Blender Preferences navigation section.



**Orbit Around Selection** The selection center becomes the rotation center of the viewport. When there is no selection the last selection will be used.

This uses the selected object (bounding box center), in Object Mode and select elements in edit/pose modes.

---

**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.

---

**Natural Trackpad Direction** Todo (macOS only).

### Auto

**Perspective** When enabled, the view switches to perspective when orbiting the view, setting axis views (Top, Side, Front, Back, etc.), sets the view to orthographic.

When disabled, orthographic/perspective mode needs to be changed 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.

**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*.

**Camera Parent Lock** When the camera is locked to the view, the root parent is transformed rather than the camera.

---

**Tip:** This is useful for camera rigs where you don't want to animate the camera directly.

---

## 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.

## Preset Management

**Keymap Presets** Select the keymap from a list of predefined keymaps.

**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.

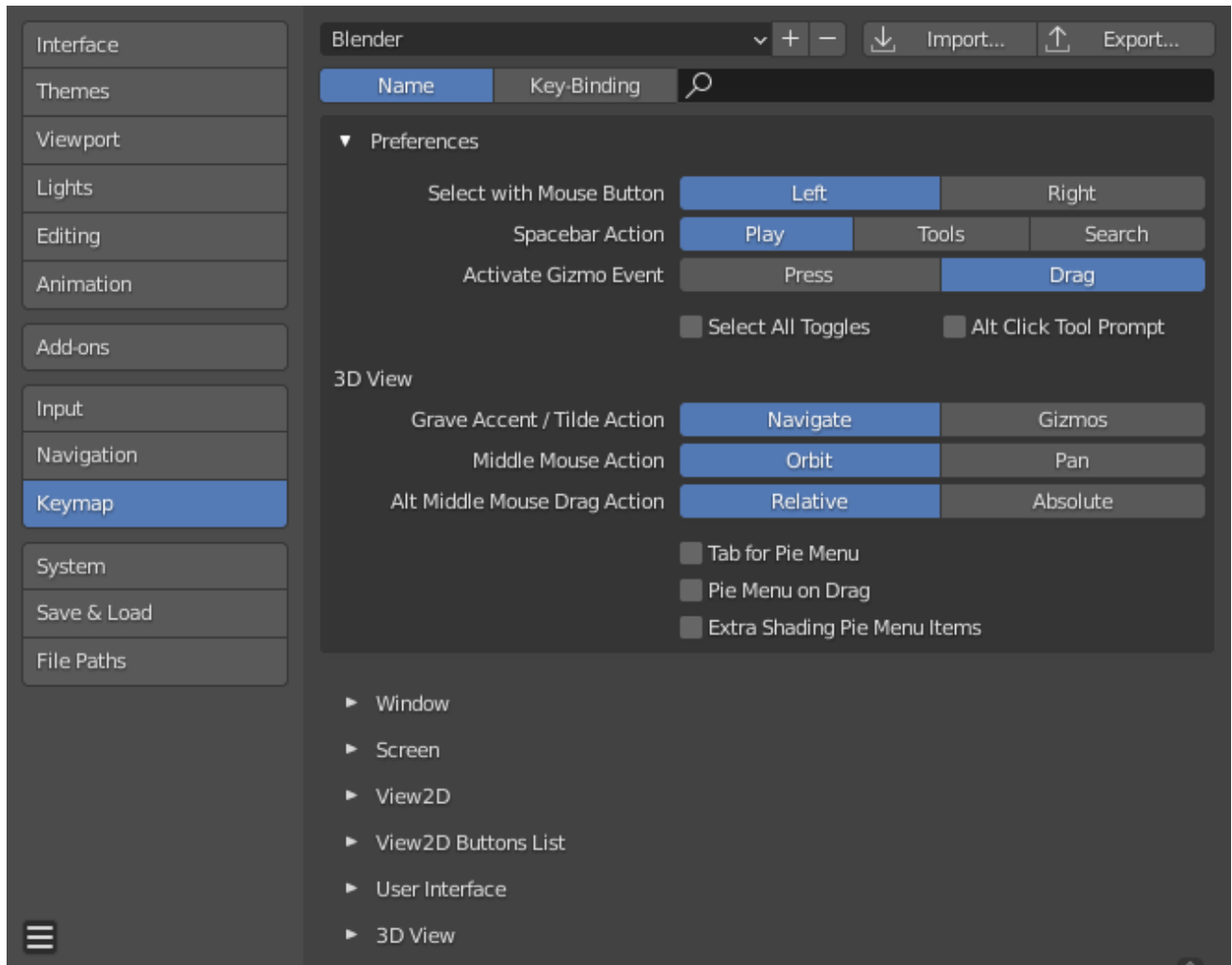


Fig. 525: Blender Preferences Keymap section.

## 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.

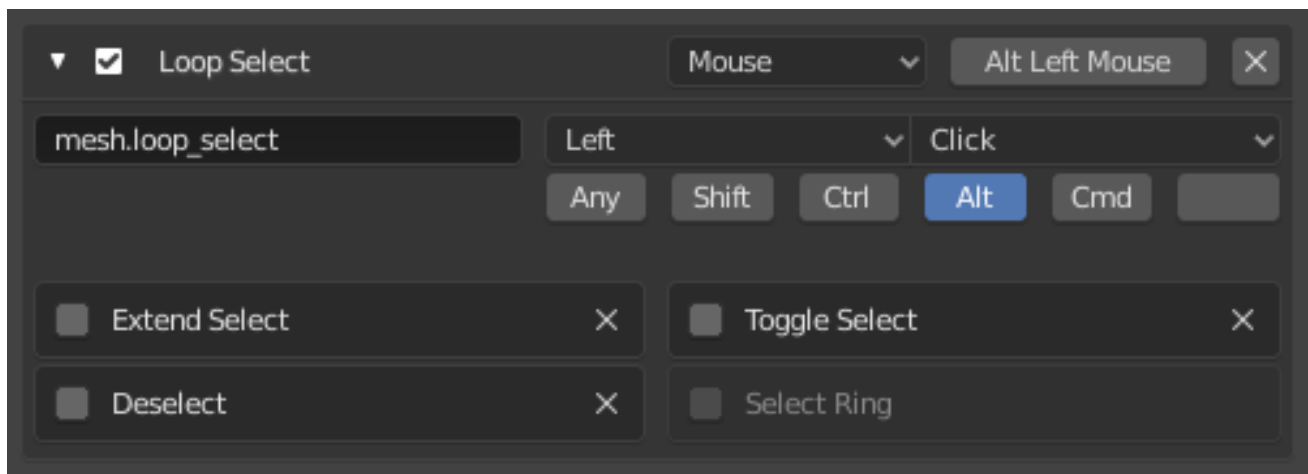


Fig. 526: Keymap editor.

## 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 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 show up or be corrected on startup.

### 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.

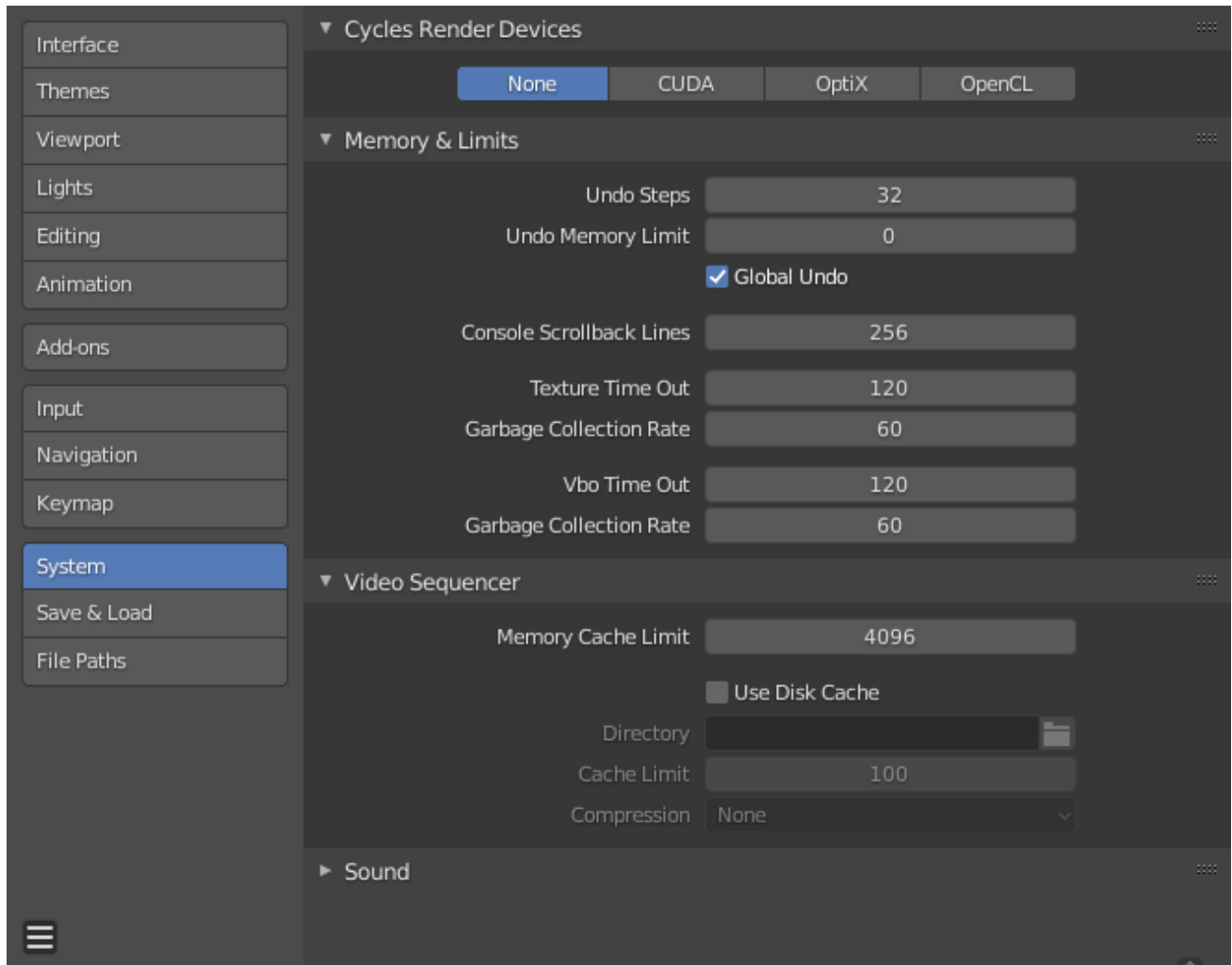


Fig. 527: Preferences System section.

**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 show up an option for rendering with Cycles.

**OptiX** If the system has a compatible Nvidia OptiX device, it will show up an option for rendering with Cycles.

**OpenCL** If the system has a compatible AMD OpenCL device, it will show up 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.

## 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 *Redo 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.

**Use Disk Cache** Writes cached strips to disk which can store a lot more than to use Disk Cache, this option must be enabled, the *Disk Cache Directory* and *Disk Cache Limit*, and then save or reopen existing blend-file.

**Disk Cache Directory** The location on disk to store the cache.

**Disk Cache Limit** Upper limit of the Video Sequencer's disk cache (in gigabytes), setting to zero disables disk cache.

**Disk Cache Compression Level** 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.

**See also:**

*Strip Proxy and Cache properties.*

## Sound

This panel contains the sound settings for live playback within Blender and are only available with *SDL* or *OpenAL*. 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 support (audio strips can still be loaded normally).

**SDL** Uses Simple Direct Media Layer API from [libsdl.org](http://libsdl.org) to render sounds directly to the sound device output. Very useful for sequencer strips editing.

**OpenAL** Provides buffered sound rendering with 3D/spatial support. Used for 3D source support by speaker objects.

**Channels** Sets the audio channel count.

**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

### Blend Files

#### Save

**Save Prompt** Asks for confirmation before closing or opening a new blend-file if the current file has unsaved changes.

**Save Preview Images** Previews of images and materials in the *File Browser* are created on demand. To save these previews into your blend-file, enable this option (at the cost of increasing the size of your blend-file).

#### Default to

**Relative Paths** Default value for *Relative Paths* when loading external files such as images, sounds, and linked libraries.

**Compress File** Default value for *Compress file* when saving blend files.

**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.

**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.



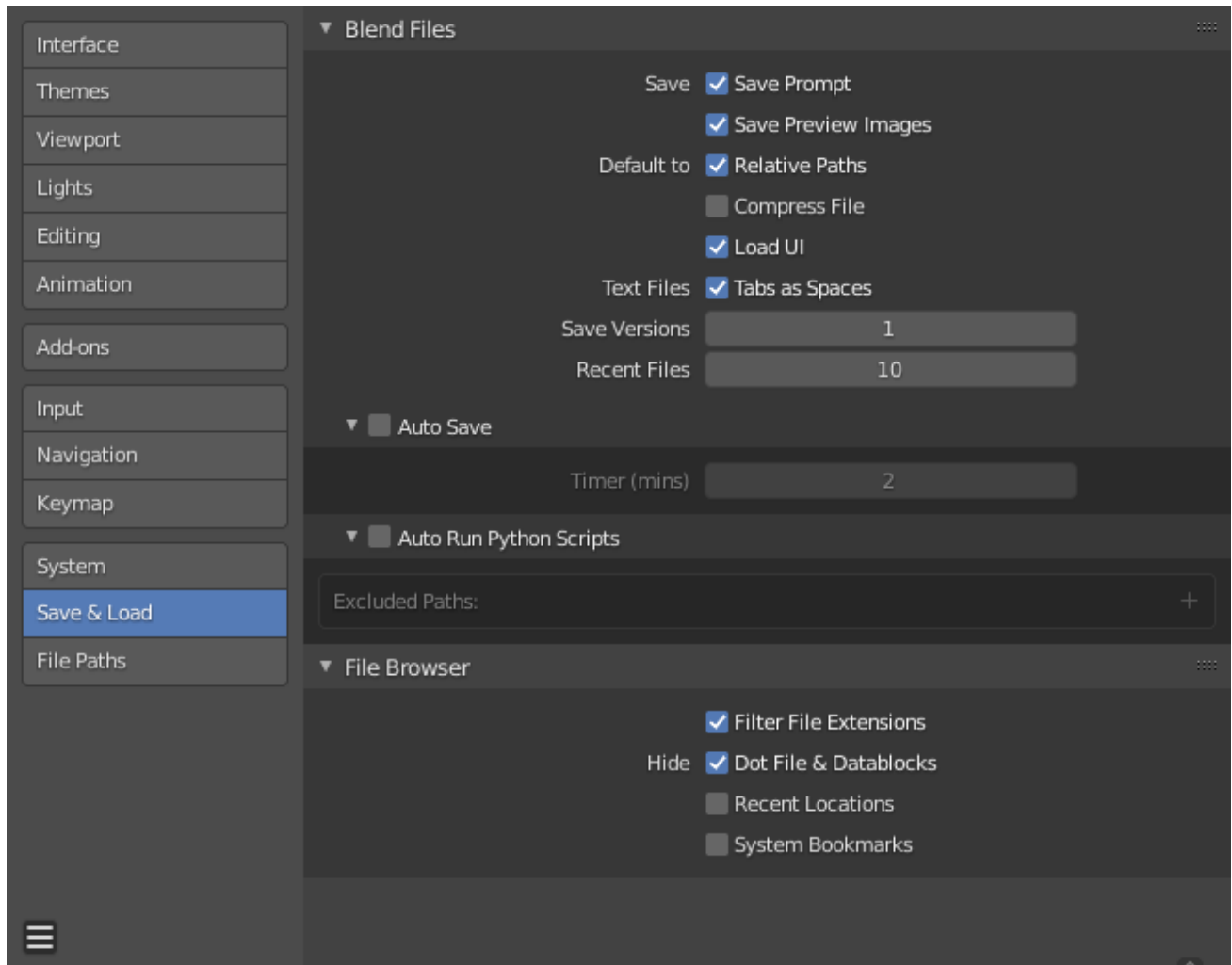


Fig. 528: Preferences Save/Load section.

\* **.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** 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).

### 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

**Filter File Extensions** 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.

#### Hide

**Dot File & Data-blocks** Hide files which start with `.` on File Browsers and ID selector.

---

**Hint:** Data-blocks beginning with a `.` can be selected by typing in the `.` characters. When explicitly written, the setting to hide these data-blocks is ignored.

---

**Recent Locations** Hide the *Recent* panel of the *File Browser* which displays recently accessed folders.

**System Bookmarks** Hide System Bookmarks in the *File Browser*.

### 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:

---

**Hint:** The default path `//` refers to the folder of the currently open blend-file (see *Relative Paths* for details).

---

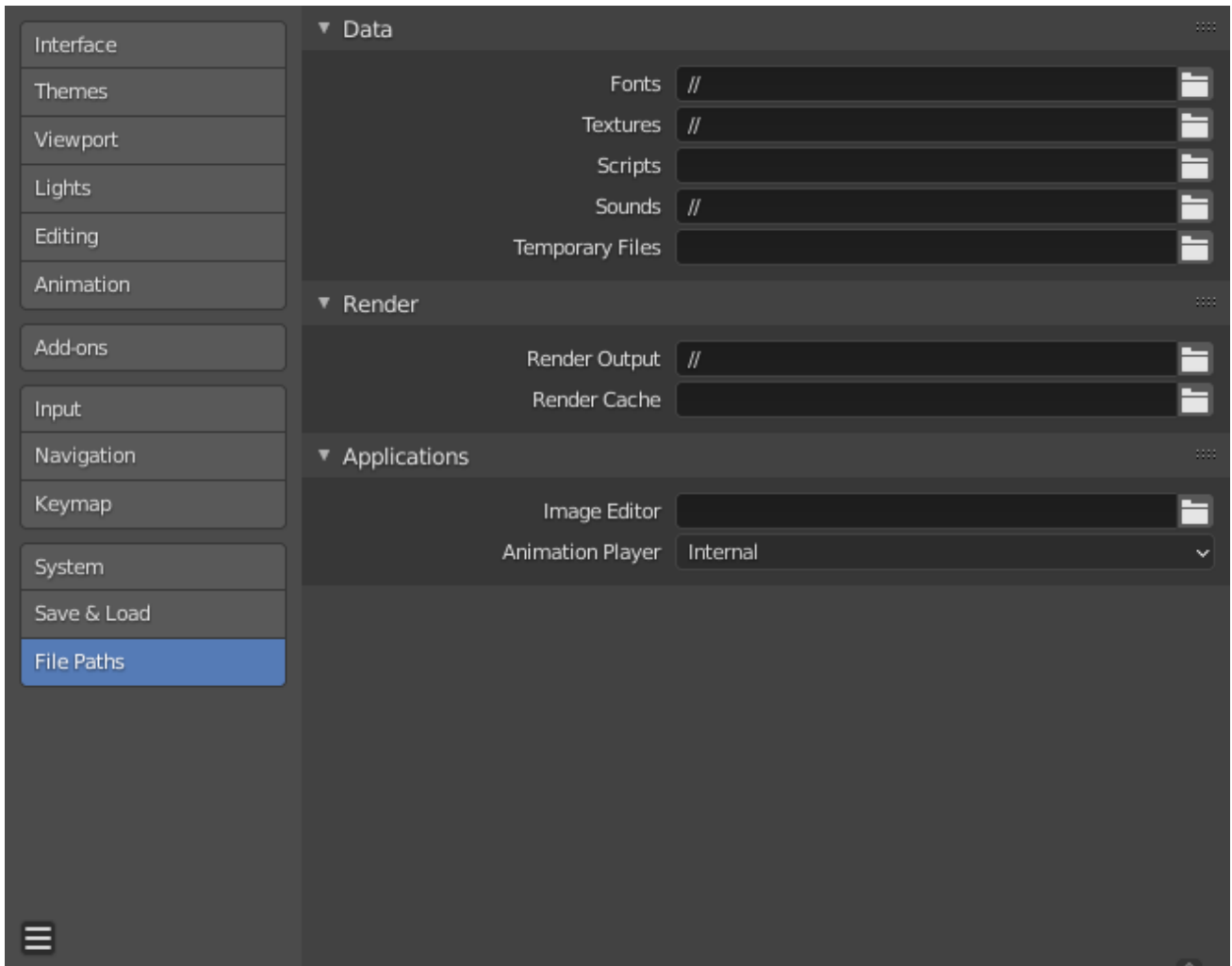


Fig. 529: Preferences File Paths section.

## Data

**Fonts** Default location to browse for *text object* font files.

**Textures** Default location to browse for image textures.

**Scripts** An additional location to search for Python scripts.

By default Blender looks in several directories (platform dependent) for scripts. By setting 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/** 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.

---

**Note:** You have to restart Blender for all changes to the users scripts to take effect.

---

**Sounds** Default location to browse for sound files.

**Temporary Files** The location where temporary files are stored, leave blank to use the systems temporary directory (see *Temporary Directory* for details).

## Render

**Render Output** Where rendered images/videos are saved.

**Render Cache** The location where cached render images are stored.

## 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.

## 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*.

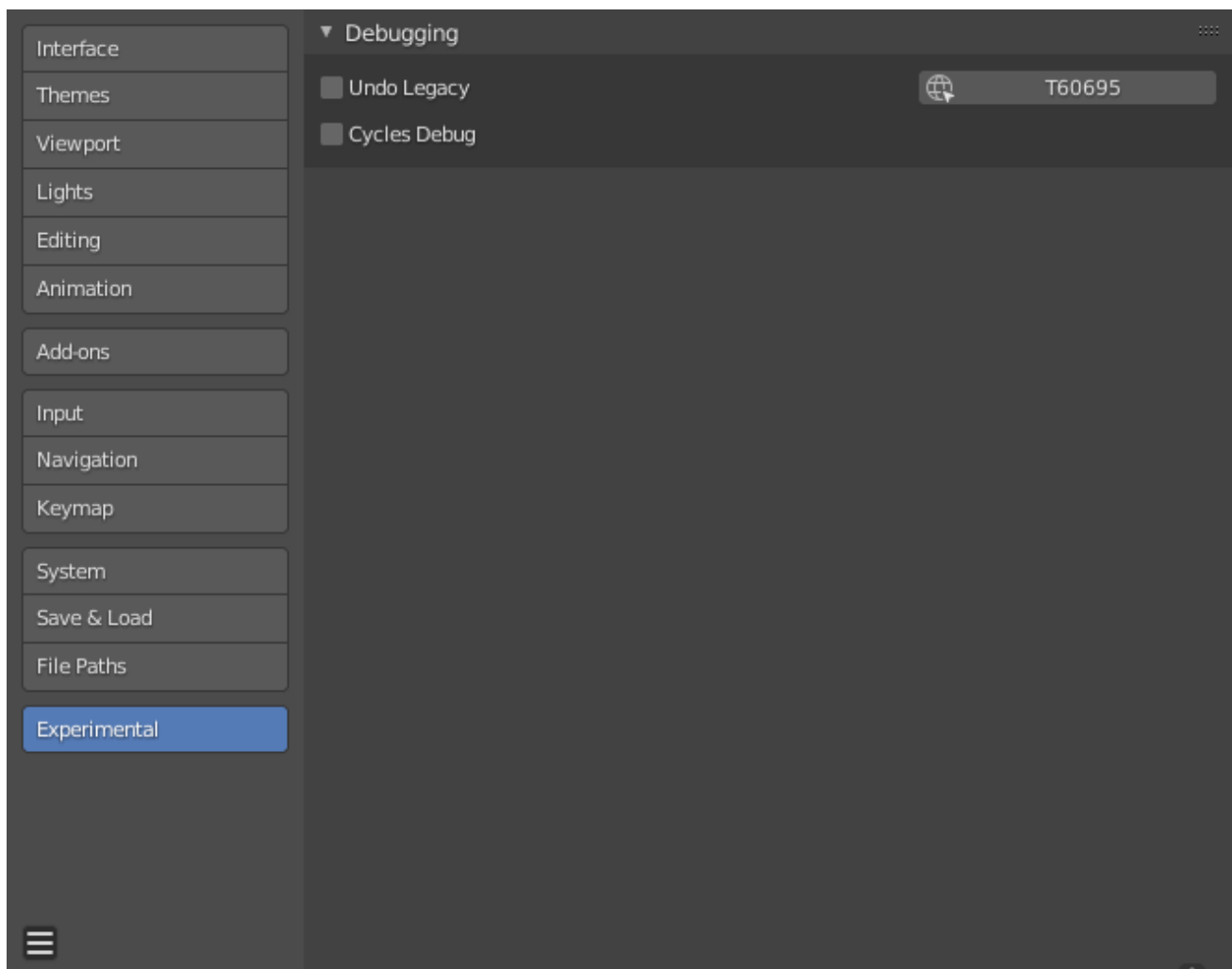


Fig. 530: Blender Preferences Experimental section.

## Debugging

**Cycles Debug** Show the Cycle’s rendering *Debug Panel*.