Blender 3D for Jobseekers

Learn professional 3D creation skills using Blender 3D

Laurie Annis

Blender 3D for Jobseekers

Learn professional 3D creation skills using Blender 3D

Laurie Annis

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-118

www.bpbonline.com

Dedicated to

Laurie Annis

For daring to believe in me, even when nobody else did

About the Author

Laurie Annis is an ethical entrepreneur and 3D/XR artist, designer, and developer with 10 years of experience converting her passion for games into a fun and sustaining career in 3D, preceded by over 20 years of experience in graphic design for social impact businesses.

Laurie's 3D skills helped her land her first dream job in 3D modeling and leading a team of 3D artists working on the multiplayer VR game Hypatia, available on Steam. Laurie is currently engaged in her next dream job as co-founder of the interactive narrative game AncestoriesXR.com, making indie games and stock assets freelance as Unreality3D.com, and founder of theAssetConnection.com, which helps connect Buyers and Creators of stock 3D models, textures, animations, effects, audio, and tools for 3D.

About the Reviewers

- **Oliver Villar** is a Spanish digital artist who has taught Blender since 2010 with tutorials, courses, and private classes through Blendtuts.com and at the University of Murcia. In 2016, he started **Blendtuts.es** to focus on teaching Blender to the Spanish community. He's the author of the book "**Learning Blender**", which has sold over 30k copies worldwide, and one of the organizers for several editions of "Blendiberia", the most important yearly national event for Blender users in Spain.

- **Arunkumar Bala Sundaram**, with 16 years of experience in CGI, is a good representation of a true Artist in many ways! He developed his skills in Jewellery design and Metal Fabrication, which involved intricate works and an eye for detail. Then he branched off into the CGI field, starting as a Web and Graphic designer after getting his Diploma in digital visual media from the Institute Of Multimedia Arts And Graphic Effects. He enhanced his skills to become a CG Generalist / Lighting Artist and compositing artist at Realworks studio. He contributed to the community by training some freshers for placement through his own Startup company. In 2017, he joined **House of Blue Beans**, a 100 % EOU product Visualization company, as Technical Lead and steadily grew to become a Functional Manager.

  Arun has continuously educated himself through formal degrees like BSc from IGNOU, attending CGI conferences and in-house leadership courses.

Acknowledgement

I wish to express immense gratitude to the original creator of Blender, Ton Roosendaal, for not simply creating and maintaining a tool that enabled me to develop marketable skills in the industry that I love, completely for free at a time when my family's finances were devastated, but also for inspiring a massive community of support that helped me help myself, and for proving that running a tech business ethically is not only possible but also sustainable and life-sustaining. It is because of your example that I am confident in doing business in this space conscientiously.

I also wish to express my gratitude to BPB for giving me the opportunity to write the book that has been burning in my heart since I first found Blender and started to hone the skills I needed to keep myself sane and my family afloat at one of the most difficult times in our lives, as a mom being relentlessly harassed by her children's abusive other parent. While Blender 3D for Job Seekers is a technical manual, it is still very personal work because the accessibility and power of this professional creative tool literally saved my and my children's lives and I consider this my opportunity to pay the favor forward.

I am eternally grateful to my children, who through their own powerfully loving, courageous, just, and kind natures, have achieved so much: freed themselves from their other parent's torture, worked incredibly hard to heal tremendous wounds, and graciously continue teaching me how to be a better person every day. Without you, none of this – the book, the professional work I enjoy so much, nor my deep sense of purpose in life – would exist. I am humbled and honored to be welcomed into your lives.

Likewise, without the extraordinary support of my mom, Annis Hopkins, none of us would be here today. I will never be able to repay your care and generous kindness to us all.

No acknowledgment of the existence of this work would be complete without more appreciation for Kevin, the love of my life than I can ever express. Without your loving presence for the past decade and a half, I would have lost the will to press forward many times over. You are my dream come true!

Preface

The opportunities for fulfilling, sustaining work in 3D have never been more plentiful than they are today, with everyone from solo and indie developers to mega-corporations employing the immense power of 3D to immerse and engage audiences in ways that 2D media cannot: by allowing viewers to experience art, entertainment, education, industry, medicine, advertising and more, from every perspective, inside and out.

Learning how to create in 3D is a daunting and lengthy process, no matter which software is used. Folks just starting out have an especially hard time choosing applications that are worthy of such great investments in effort and time that will serve them all the way from a beginner to a professional. This book is designed to equip new and aspiring 3D creators with the fundamentals of 3D creation and introduce the vast capabilities of the one industry-standard software which does all of that and happens to be free to use forever: Blender 3D.

Blender 3D for Job Seekers highlights the many facets of 3D creation, including modeling and sculpting, surface decoration, animation, effects, and preparing 3D creations for real-world markets, such as print, film, and games. Real-world exercises are designed to gradually acclimate readers to navigating and thinking in 3D space while inspiring further exploration through producing a marketable 3D asset suitable for games and interactive applications. Guidance on where to begin deeper exploration is provided with each core concept.

With the knowledge of how to get started creating in 3D, readers will be prepared to begin fulfilling creative work by making products that are in high demand in the vast, lucrative market of 3D.

Chapter 1: Features of Blender 3D - highlights the vast capabilities of Blender 3D for money-making content, from basic modeling to advanced skills, like special effects and physics simulations. Topics covered include ways that Blender is preferable to other 3D software, an overview of its core and advanced features, and ways to extend Blender's capabilities.

Chapter 2: Installation and Interface - walks the reader through the steps of downloading and installing Blender, opening Blender for the first time, and getting familiar with navigating the interface. Topics covered include getting started and saving progress, understanding the defaults, short exercises to get acclimated, working between versions and updating, and customizing the interface.

Chapter 3: General 3D Concepts - introduces general 3D concepts that apply to all 3D software, giving readers the necessary foundation to begin creating in 3D. Topics covered include the anatomy of 3D objects, getting oriented to working in 3D space, perception versus reality, exercise in creating 3D objects from scratch and understanding basic operations.

Chapter 4: Polygonal Modeling - introduces polygonal modeling: the essential starting point for creating 3D objects. Topics to be covered include the differences between various methods of 3D creation from poly modeling to sculpting, low and high poly, planning projects, topology, and edge flow, and an example project beginning to model a marketable 3D pedestal from reference.

Chapter 5: Poly Modeling Extras - explores supplemental tools and techniques for poly modeling. Topics to be covered include the full set of tools in the Edit Mode toolbar, editing 3D objects with modifiers, curves, and text, and enabling and working with add-on resources.

Chapter 6: 3D Sculpting - introduces 3D sculpting: a method of creating and editing 3D models with great detail and organic shapes. Topics to be covered include when to sculpt and how to plan the project, helpful workflows and introduction to sculpting tools, an exercise adding sculpted detail to the 3D pedestal, and working between high and low poly objects.

Chapter 7: 3D Surfaces - introduces the standard way to prepare a 3D model for surface details like color, roughness, metallicness, and fake height variation, as well as how to apply colors, images, and effects to the surface of 3D models, and how to edit their appearance. Topics to be covered include unwrapping UVs, texturing and how to edit textures, Materials, and Shaders, and exercises adding textures and transferring sculpted detail from the previous exercise onto the 3D pedestal model.

Chapter 8: 3D Animation - will get readers started animating 3D creations. Topics to be covered include an overview of the timeline, keyframing, and animatable properties, types of 3D animation from bone to mesh animation with example exercises, and an introduction to the advanced animation tools.

Chapter 9: Effects and Simulations - gets readers started exploring special effects and simulations in Blender, from fantasy effects to realistic water, fire, smoke, cloth, and even fur and hair. Topics to be covered include an introduction to the types of effects and simulations and their uses and exercises working with hair particles, modeling with physics, and creating a volumetric effect.

Chapter 10: Images and Video - teaches readers how to showcase their 3D models and animations in images and video, which can be used in portfolios, advertising, film, games, and an unlimited number of other creative projects. Topics to be covered include an overview of rendering, 3D cameras, lighting, shadows, reflections, and transparency, exercises in lighting and rendering the 3D pedestal, and an introduction to the advanced image and video editing capabilities of Blender.

Chapter 11: 3D in Production - readers will learn how to apply their 3D skills in Blender to earn money and become sought-after creators in their fields of interest. Topics to be covered include best practices, planning and optimization, pipelines and tools, working on teams, and finding a niche.

Coloured Images

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/g5yswbv

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

CHAPTER 1
Features of Blender 3D

Introduction

Welcome to **Blender 3D for Job Seekers**, and congratulations! We are about to embark on a journey of creative and financial freedom and empowerment. Before we begin navigating the application, we will discuss the many reasons why Blender 3D – referred to as **Blender** from this point forward – is preferable to other 3D creation software.

Structure

This chapter highlights the vast capabilities of Blender for fun, satisfying creation of sustaining, money-making content, from simple primitive 3D objects to special effects. Topics to be covered include:

- Competitors
- Standards
- Features
- Capabilities

- Capabilities
- Affordability
- Accessibility

- Performance
- User Friendliness
- Compatibility
- Extensibility

Objectives

Every industry imaginable uses and pays for 3D content. We have probably seen 3D employed by multiple industries all around us, just today, without even realizing it. While the average person may only imagine video games and film using 3D, 3D content creators also contribute to the fields of science, medicine, education, journalism, marketing, manufacturing, and innovation, among others.

The category of 3D creation software is cluttered with a wide range of options. To weigh each of these options as an aspiring 3D content creator, one must first consider the context. What do we want to create in 3D? What is even possible? After reading this chapter, we will understand when and why to choose Blender over other 3D creation software.

Alternatives to Blender

A quick internet search of "3D software" will reveal several major competitors who have paid to have their products appear at the top. The ubiquity of costly alternatives to Blender has little to do with the quality of other applications or their user reviews. Instead, that predominance reveals a malicious side of corporate efforts to mandate widespread dependence on inferior and prohibitively expensive products, throughout entire industries like computer-aided drafting. Fortunately, Blender's developers have come to the rescue.

Mission

Get the world's best 3D CG technology in the hands of artists as free/open source software."

-Blender.org

The founders of Blender seek to democratize the money-making power of 3D creation tools, by reducing as many barriers to Blender's use as possible. Instead of forcing out competition through exclusive agreements with bureaucratic institutions, Blender was designed to empower creators through its compatibility with other 3D creation software. Instead of developing proprietary file formats, for example, Blender allows the import and export of the widest variety possible, making Blender the go-to application for converting one 3D file format to another.

What 3D software does

The same internet search for "3D software" will reveal a few software packages that can do some part of what Blender is capable of overall, which highlights a significant way that Blender excels: in its flexibility. In general, all 3D software can be used to create basic 3D components – from the vertices, edges, and faces which make up all 3D geometry, to simple shapes like cubes, cylinders, and spheres – for combination into more complex structures as shown in *Figure 1.1*:

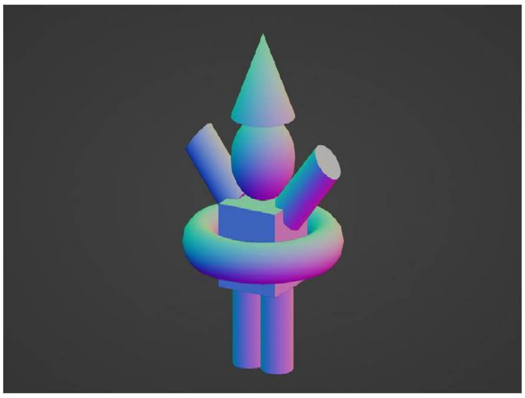


Figure 1.1: *Simple character made of 3D objects assembled in Blender*

Advanced 3D software often specializes in one area of 3D creation or another. **Zbrush**, for example, excels in the niche of high-resolution 3D sculpting. **Marvelous Designer** excels in the 3D cloth niche. Blender, on the other hand, performs both tasks and more adequately, in a way that is seamless with other 3D workflows (see *Figure 1.2*):

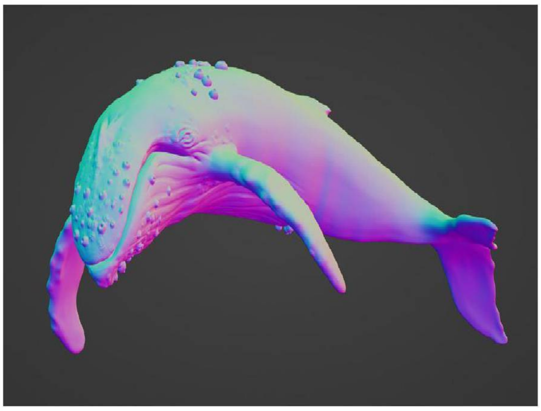


Figure 1.2: High-resolution 3D model sculpted and posed in Blender

Some of Blender's competitors offer similar varieties of more general functionality. **3DS Max** and **Maya**, for example, perform the same 3D modeling, 3D surface design, 3D animation, and rendering tasks that Blender does. Unlike other general-purpose

3D applications, however, Blender is not a haphazard assortment of functionality added on by developers who do not use and directly benefit from those features. Because Blender is developed by the very 3D creators who use Blender in production day in and day out, each of Blender's capabilities was conveniently crafted to complement the rest.

Above all, Blender is an industry standard suite of 3D creation tools. The term "industry standard" means that Blender can produce output that is not only fully functional for production, but also meets the highest standards of quality possible, wherever it is used in the creation of video games, interactive applications, film, and television, advertising, simulation, printing, digital archiving, education, science, and even medicine.

Blender is feature rich

Blender is primarily a 3D modeling software, but also much more. Blender comes loaded with every basic feature a 3D creation software might need. The most basic features of Blender include:

- 3D modeling and sculpting
- 3D model and scene editing
- 3D surface design (UVs, textures, materials, shaders)
- 3D animation (rigging and skinning, mesh morphing, keyframing)
- Rendering 2D images and videos of 3D objects and scenes

While 3D modeling alone is powerful functionality, Blender is also capable of many advanced features out of the box, like:

- 2D drawing and combined 2D+3D drawing
- 2D animation and combined 2D+3D animation
- Rendering 360 images and videos
- Recording, compositing, and editing images, videos, and audio
- SFX (volumetrics, particles, and motion tracking)

- Physics simulations (cloth, fluid, smoke and fire, forces)
- Procedural and parametric modeling (geometry nodes, modifier stack)
- Scripting and modding Blender itself
- Game development

Blender is capable

Blender's capabilities are vast, which means that when we learn how to use Blender, our capabilities will also become vast.

It is easy to identify the most common applications of 3D creations, enabled by Blender. Entertainment represents a trillion-dollar industry using 3D in every digital product from video games to film (both short and feature-length), as well as television. 3D is also used in the design and manufacture of toys, board games, and sports accessories. But entertainment uses only scratch the surface of 3D's power.

Science also employs 3D in every facet of the field: from illustration, education, and training to simulation and safety. From the far reaches of outer space to the depths of volcanic activity, 3D provides views into and interactions with the microscopic and macroscopic, real, and theoretical, in ways that just are not possible in the real world (yet).

Medicine is similarly served by 3D through illustration and education, as well as through medical treatments ranging from virtual experiences that cure or alleviate pain and trauma to the design and manufacture of assistive devices and prosthetics, from teeth to limbs.

Education also utilizes 3D to achieve more engaging and more thorough illustration of the known world through immersion and interaction, enabling embodiment that expands understanding, in addition to facilitating experimentation and exploration of the unknown.

Journalism employs 3D for archiving and documentation, ensuring the veracity of information, and communicating news and history in ways that increase empathy, by personalizing stories and immersing consumers in surprising and challenging topics.

Marketing with 3D generates virality and problem/solution awareness by giving customers a fuller perspective of products and services through affordable, accessible prototypes and visually stunning presentations.

Manufacturing enjoys the same benefits Science and Marketing do, through the affordable and safe design, testing, and additive and subtractive production of products, as well as worker training and automated assembly, all enabled by 3D.

Innovation is 3D. The amount of overlap between the uses of 3D in all the above industries speaks to the universal nature of 3D. The future is 3D, whether that means the realization of retro visions like holograms and holodeck experiences, or modern movements to build a digital twin for everything real within a shared metaverse.

Blender users the world over have contributed 3D creations to every single one of these industries, thanks to Blender's versatile capability.

"Free and Open Source

Blender is a public project hosted on blender.org, licensed as GNU GPL, owned by its contributors. For that reason Blender is Free and Open Source software, forever."

-Blender.org

Blender is affordable

One of the most compelling aspects of Blender is that it is absolutely, 100% free, forever; and not just free to learn, but also free to use commercially. This means there are no restrictions to using Blender to make money, and the developers of Blender take no cut of users' profits. If we have not yet run into the creative block of prohibitively expensive 3D creation software, we can consider ourselves lucky, because software comparable to Blender typically costs thousands of dollars per year for commercial use and upgrades.

It is not just the "sticker price" of competing 3D software that answers the question, "Why choose Blender?" Nor is it merely the capabilities of Blender compared to competitors that should be considered. Possibly the most compelling answer to this question is why those competitors exist, and at what cost. Frankly, the largest players in 3D software entered the market to make the corporations money, whereas Blender was created **by its users to empower other creators**.

Blender is accessible

The cost of using any 3D software package is not limited to the initial monetary outlay. There are additional costs like licensing to consider. The most popular competing 3D software packages do offer lower-priced educational licenses, but they restrict educational versions to non-commercial use. This is intentional, to trap users into spending more money through dependence. Blender, on the other hand, has only one license with no restrictions on commercial use, and the Blender Foundation charges no royalties no matter how much money we make using it.

There are other barriers to consider with competing 3D software, such as the availability and price of updates, training, supplemental content, and productivity tools like add-ons or plug-ins. Whereas competing 3D software must be updated yearly by creators who need to keep their skills relevant to the market – often at the same price as the original software – Blender is updated multiple times a year, always free of charge.

There are thousands of time and effort saving add-ons available for Blender, many for free and some paid. On the other hand, add-ons and plug-ins for comparable 3D software almost always cost a great deal more simply because their developers need

to charge more to recoup higher costs of development. The availability of Blender for free has nourished the growth of an unparalleled community of 3D creators offering free mutual support and very reasonably priced training and resources.

As the adage goes, time is money, and the time between learning and implementing new skills is also shorter for Blender than for comparable 3D software. Because we can learn Blender entirely on our own using free and affordable tutorials that are readily available online, we can get started creating money-making content with Blender today! No need for a college degree to acquire marketable 3D skills.

Vision

"Everyone should be free to create 3D CG content, with free technical and creative production means and free access to markets."

-Blender.org

Blender is also completely open-source, which means that anyone can change the code and modify Blender free of charge. We can develop Blender tools for ourselves or offer them for sale as well. When we sell anything we made with Blender – whether it be assets, scripts, tutorials, or tools – we are not restricted from profiting from our own work. As one might imagine, this freedom has opened the floodgates of Blender's developer community creativity.

Another benefit of Blender's accessibility is its unique license model, which will ensure its longevity. Users of paid and proprietary software often suffer when that software gets sold to another business and winds up fundamentally changed or even abandoned, but Blender's GNU General Public License model will ensure those tragedies never happen to Blender users.

Blender is performant

The specs for necessary hardware to run Blender are much lower than with other 3D applications (which can also equate to monetary savings). Although any 3D task can push the boundaries of computer hardware, Blender and Blender workflows can be easily adjusted to run well on minimal specs, making 3D creation possible even for youth, older adults, students, and borrowers or owners of low-end computers. While Blender users can enjoy plug-and-play drawing tablet functionality, all of Blender's operations can be performed with a mouse by default.

Blender is also a remarkably bug-free software. Even with years of use, we may never need to report a bug, but rest assured that Blender's developer community will be responsive if we do. Blender's exceptionally high-quality user experience is attributable to the fact that Blender's code is maintained by its avid community of users, and the fact that it is upgraded at no cost, multiple times each year.

Blender is user friendly

While creating in 3D space is challenging in and of itself, Blender is particularly user-friendly due to its logical layout and easily customized interface. In version 2.8, Blender's UI was overhauled by community developers, to better align it with the needs and preferences of everyday users. Every element, from icons and color schemes to button behavior and hotkey mapping was thoughtfully debated and carefully redesigned where needed. The resulting improvements have been widely celebrated since.

Every panel of Blender can be moved, duplicated, scaled, and even converted into a panel of another type and back again. Typical Windows File, Edit, Window and Help functions, plus a row of tabs that toggle between convenient layouts for the most common tasks appear across the very top of the main viewport. Hotkey presets are provided to match a variety of industry standard functions and can be easily remapped, while any available action can be added as a Quick Favorite bound to the Q key.

Blender's UI also employs progressive disclosure, which means the enabling of advanced features is conveniently accessed through the Preferences dialogue, without cluttering the view for more common tasks.

Blender is compatible

Blender is compatible with all other industry standard 3D creation software, which is an astonishing feat. Not only can we use Blender in place of expensive, exclusive software like 3DS Max, Maya, Zbrush, and 3D Coat, but we can also use Blender with all of those and more, in the same production pipeline.

For example, Blender is compatible with the entire suite of Adobe Creative Cloud software, such as *Photoshop, Illustrator, Premiere Pro, After Effects, Substance Painter,* *Substance Designer*, and the rest. All of the file formats that these apps have in common with Blender can be transferred back and forth seamlessly. Blender also allows the selection of *Photoshop*, *Krita*, *Gimp*, and any other image editing software as the preferred external tool.

Blender is also fully compatible with popular game development software, such as *Unity*, *Unreal*, *Godot*, and every other modern game engine. As previously mentioned, Blender allows for the import and export of all the commonly used (and some uncommon) 3D formats for web, mobile, desktop, and console games, viewers, and interactive applications: *FBX*, *OBJ*, *STL*, *GLTF*, and *USD* to name a few.

Blender is extensible

Finally, while Blender can be used to create an infinite variety of income sources all on its own, Blender's capabilities can be extended even further using both free and paid add-ons and plug-ins. Some of the most popular add-ons for Blender enable:

- Human character, hair, and clothing generation
- Humanoid and creature animation
- Nature prop generation
- Motion capture, animation, and editing
- Advanced simulations and SFX
- Architectural visualization and CAD
- Data visualization
- 3D printing preparation
- Advanced 3D editing (UV editing, retopology, LOD, optimization)
- Asset management and more

Since multiple web pages, articles, video tutorials, and entire marketplaces are devoted to the add-ons and plug-ins available for Blender, we will not address them in any great depth here. It is important to note, however, that Blender's extensibility is also superior to that of its competitors.

The relatively low cost and great abundance of extensions to Blender's core functionality truly set it apart as the 3D creator's software of choice. For this reason, hundreds of thousands, if not millions of freelancers and entire businesses thrive and owe their existence to Blender's welcoming, enthusiastic community, and ecosystem built on the foundation of democratizing tech built on the foundation of democratizing tech.

Conclusion

As we have seen, Blender is the obvious choice for beginners in 3D and seasoned professionals alike, when compared with its competitors. Blender easily meets industry standards with its versatile features, capabilities, and performance. No 3D creation software comes close to Blender's accessibility and user-friendliness. Finally, we have learned that Blender opens doors to other 3D creation tools rather than locking users in and offers limitless potential for creative expression and financial sustainability.

In the following chapter, we will install and open Blender for the first time, to get familiar with its interface and learn handy navigation shortcuts. In *Chapter 3, General*

3D Concepts we will delve into understanding 3D in general, and discover even more proof that Blender is the ultimate tool for creating in 3D.

Points to remember

- 3D is a trillion-dollar industry with uses in multiple lucrative fields.
- Popular and pricey does not equate to quality 3D creation software.
- Software licensing and subscription models impact our ability to make a living long-term.
- Blender makes learning and profiting from 3D skills accessible and affordable.

Questions

1. What are basic features of 3D creation software?
2. What are advanced features of 3D creation software?
3. What other 3D creation tools exist besides Blender?
4. What are the advantages to using free and open-source software?
5. Why is compatibility with other software important?

CHAPTER 2
Installation and Interface

Introduction

Now that we understand the benefits of using Blender for 3D creation, we'll start by installing the application and then learn to navigate the interface. Getting familiar with any new software package can be intimidating, but Blender is a bit different from most. Because Blender is developed and maintained *by* 3D creators *for* 3D creators, any time you wonder where or why a feature exists in Blender, you are sure to find a logical answer at your fingertips.

Structure

This chapter will walk the reader through the steps of downloading and installing Blender, opening Blender for the first time, and getting familiar with navigating and customizing the interface. Topics to be covered include:

- Getting started in Blender
- Versions and updating
- Navigating Blender's interface
- Saving your progress
- Backup features

- Customizing the interface and preferences
- Essential shortcuts and hotkeys
- Where to get help and expand your Blender knowledge

Objectives

After reading this chapter we will be comfortable installing, launching, and navigating Blender, then customizing it to our needs, locating every Blender tool at our disposal, and saving all our progress with confidence.

Getting started

To get started using Blender, first visit the Blender Foundation's website at **https://www.blender.org/**. A link to the downloads page appears in bold on the main page, above the fold. On the downloads page, the most recent version of Blender for your OS will be highlighted by a prominent button which you may click to begin your download. Alternatively, you may select a Windows, macOS, Linux, or another version of Blender from the dropdown list below that, or an LTS version from the lower link.

Following your download, a confirmation page will appear with an invitation to support Blender development. Contributing to these funds is voluntary, so feel free to sign up now or come back later when Blender inevitably becomes your favorite software.

Double click the file in your downloads folder, click **Next**, accept the License Agreement, and follow the installation wizard prompts. Blender does not start upon installation by default, so you'll navigate to the **Start menu**, **Launchpad**, or **desktop**, and click the icon to open the application.

Versions and updating

Note: In this chapter, we will refer to Blender version 3.1.2, however, the instructions herein will apply equally to any version 3.0 or newer. Future chapters will refer to the most recent version of Blender at the time of writing and will remain forward-compatible.

The number of Blender version options can be overwhelming at first, but it will be helpful to briefly look at the system in place for Blender releases to learn how to evaluate when it is time to upgrade, downgrade, double up, or stick with a version you already have installed.

As discussed in *Chapter 1, Features of Blender 3D*, Blender's free and open-source approach provides us with unique benefits including assurance that Blender will never be sold or abandoned. That means Blender updates will not stop so long as there are users using it.

This also means that multiple versions of Blender are available at any given time, and that content made for Blender, such as tutorials, assets, plug-ins, add-ons, and books like this one, may refer to features of a different version from the one that you are using.

As noted in this chapter, we refer to Blender version 3.1.2. Midway through this writing, Blender version 3.2 was released, and the impending release of Blender 3.3 in a few months' time was also announced. On top of that, Blender releases a **Long Term Support** (**LTS**) version every year, which receives important bug fixes and updates but often lags in terms of features.

This is normal, and as explained by Ton Roosendaal – original creator of Blender – is designed to, *"ensure that a project that started with an LTS version can be completed with the same version in a reasonable amount of time."*

For these reasons, the decision was made to tailor the illustrations and instructions contained in the chapters of this book to each new release, to show the way that Blender users typically work with an always-improving software.

Finding the right version

First, when a new version of Blender is announced, take a look at the release notes pages on *Blender.org* to find out what will be changing: **https://www.blender.org/download/releases/** Blender Foundation won't spam you or require a login, so to be notified of updates to Blender you may want to subscribe to one of the many online market places that promote Blender training materials, plug-ins / add-ons and assets.

Don't immediately switch to a newer version of Blender in mid-production, but feel free to download the latest version at any time. Newer versions of Blender will install in a directory next to the previous one, rather than overwriting it, allowing you to run multiple versions of Blender on your computer system at once. Windows Desktop and Start menu shortcuts will be updated to the newest version, but shortcuts pinned to the Taskbar will not, and unless you choose to delete the old version, both can coexist in your **Program Files** > **Blender Foundation folder** and be used interchangeably.

Timing updates

As Roosendaal and the Blender Foundation intended, you should consider timing any updates to coincide with the start and end of your Blender projects. When you are just starting out, this will likely be a very short cycle, such that you can complete a project before a new version is released. When you begin working on larger projects, it is advisable to hold off updating until completion, unless a new feature is essential to your product.

Working between versions

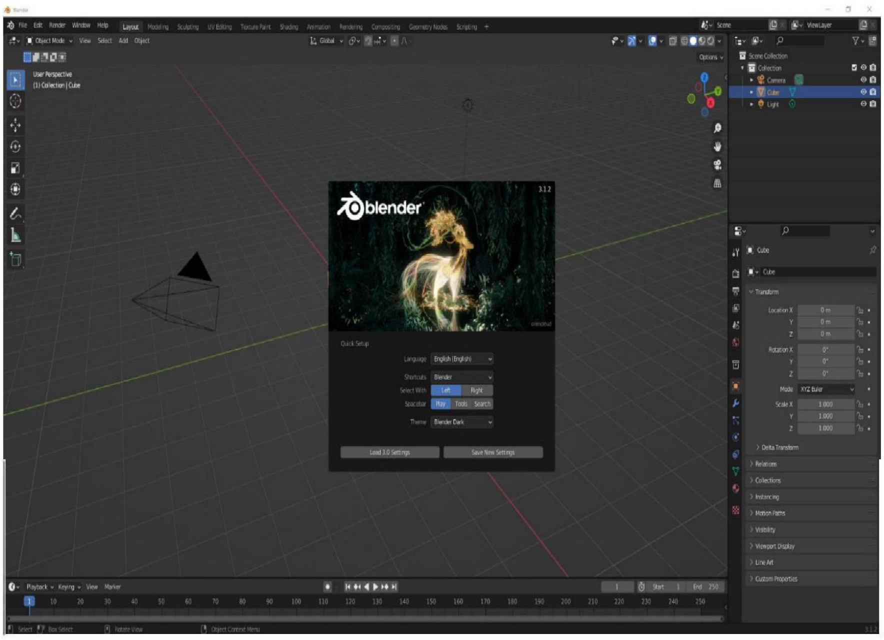
It's reassuring to know that when we need to, it's simple to work between multiple versions of Blender. Two different versions can be opened and run simultaneously, and **Ctrl** + **C** works to copy a selected object from one Blender instance and **Ctrl** + **V** works to paste it into another. Alternatively, data from one Blender file can be Linked or Appended to another BLEND file even if it is closed, via **File** > **Link**, or **File** > **Append**. Linking data maintains its relationship to the source file, so changes made externally affect that data in the linked file. Appending severs the relationship with the source file, so changes to either file do not impact the other (and doubling the space that data occupies on the hard drive).

To open two instances of Blender at once, navigate to where Blender is installed, usually in **C:/Program Files/Blender** Foundation on Windows machines. Inside this folder you will find Blender versions you have installed, in folders named to match their version number. Inside each numbered folder, locate the blender.exe, right-click, and select Pin to Start. When you click on the shortcut for each version, additional instances of Blender will open in the selected version.

To identify which version you are currently looking at, look in **Status Bar** at the very bottom right of the Blender application. If there are no numbers there, right-click that space and enable Blender Version.

Navigating the interface

A new installation of Blender will open with a Quick Setup overlay as shown in *Figure 2.1*. For 3D creators who are accustomed to other 3D software, the Shortcuts options can be adjusted to make working with Blender more familiar. Users of prior versions of Blender can also load settings saved in other versions here. Since this book assumes little or no prior experience with 3D software, we will start with the default settings.



Language: English (English)
Shortcuts: Blender
Select With: Left | Right
Spacebar: Play | Tools | Search
Theme: Blender Dark

Figure 2.1: *Blender version 3.1.2 interface upon first opening, with Quick Setup overlay*

Left-click anywhere within Blender to close the overlay, and you'll recognize common application menus in the upper left. The familiar File, Edit, Window and Help menus appear there, with Render – a term specific to 3D – in the middle.

The default layout

As shown in *Figure* 2.2, to the right of the topmost menus, you'll see several tabs, which reveal the same core functionality, laid out in more convenient ways for different tasks. This is called the Active Workspace, and functions similarly to Workspaces in other popular software such as Adobe Photoshop:

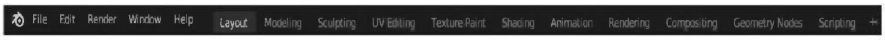


Figure 2.2: *Blender's topmost menus and Active Workspace tabs*

Tip: now is a good time to get accustomed to hovering your mouse cursor over any unfamiliar text, field, button, or icon in Blender. Blender includes detailed tooltips for almost every function, which is immensely useful for both novice and seasoned users alike.

Blender's default Active Workspace tab is labeled Layout. The Layout Workspace is a convenient starting point for basic 3D modeling and assembling objects within scenes. The largest panel within the Layout Workspace is set to display the **3D Viewport** by default. The **3D Viewport** is an all-purpose Editor Type, within which you can move, rotate, scale, and edit 3D objects.

To the upper right of the **3D Viewport** you'll see a text outline listing names of all the objects within the current scene, regardless of whether they are visible or not. This panel is called the Outliner. By default, the Outliner will show a Camera, Cube, and a Light nested within a Collection, further nested within a Scene Collection. The Outliner is an essential organization tool whose utility will become clearer over time.

Below the Outliner in the default view is a panel called Properties. Properties are essentially adjustable settings, so whenever you need to make a change to anything actively selected in the **3D Viewport** or Outliner – whether you're adjusting a tool or editing an object – you will likely be interacting with one of its Properties tabs.

Below the **3D Viewport** is the Timeline. The Timeline is where animation timing and length are managed, and animation key frames are recorded. In *Chapter 8, 3D Animation*, we will learn our way around the Timeline and its related Editor Types: **Dope Sheet**, and **Graph Editor**.

Note: An important feature of Blender is that every action, tool, and menu option is context-dependent. For that reason, it is important to pay attention to where your cursor is within the application, and which item or items are selected, to ensure that you are affecting exactly what you intended to affect.

Hidden away at the very bottom of the Blender application is the Status Bar which contains important contextual hints for almost every action and tool. When your mouse is within the **3D Viewport**, for example, the Status Bar shows actions that can be performed with the mouse: Select, Box Select, Rotate View, and Object Context Menu. Keep an eye on the Status Bar. While you may be tempted to forget that it exists, information there will be critical to multiple 3D creation and editing tasks we will begin to perform in *Chapter 3, General 3D Concepts.*

Exercise 2.1

In this first exercise, we will delete the default objects in the scene, add the Blender mascot Suzanne, and learn to zoom, orbit, and pan around the scene. These may seem like mundane steps at first, but practice with any software improves muscle memory, which will help navigating 3D space become like second nature to you. Through multiple mistakes you will find that it is best to practice on creations that are not too precious.

1. To begin, select the three default objects via the Outliner, by pressing and holding **Shift**, then clicking each object name with the left mouse button as shown in *Figure 2.3*.

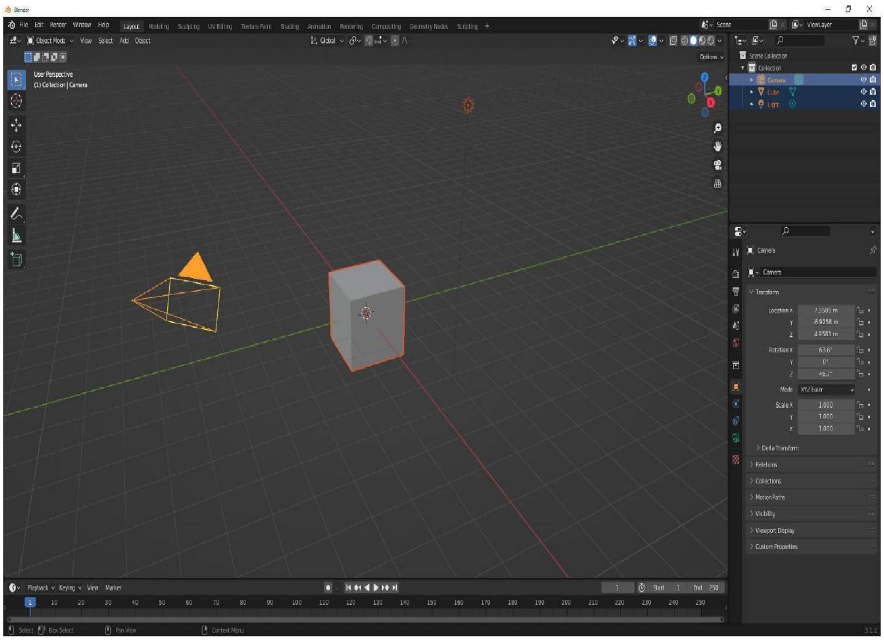


***Figure 2.3:** Three objects selected in the Outliner panel, Camera is outlined in yellow, Cube and Light in orange*

When **Shift** is pressed while selecting objects in the Outliner, every item between the first and last item selected will also be selected as expected. Notice the color of each of the items selected with the **Shift** key pressed: the **first** item selected with **Shift** will be highlighted in yellow, with the following selections highlighted in orange.

When **Ctrl** is pressed while selecting objects in the Outliner, individual items can be selected out of sequence, as expected. Notice the color of each of the items selected with the **Ctrl** key pressed: the **last** item selected with **Ctrl** will be highlighted in yellow, with the previous selections highlighted in orange.

Tip: The highlight color and order of selections will play an important role in many future exercises, so this is a good time to begin to associate the term Active with yellow and Selected with orange, as shown in Figure 2.4. Note too, that objects selected via the Outliner are highlighted in the same colors in the 3D Viewport.

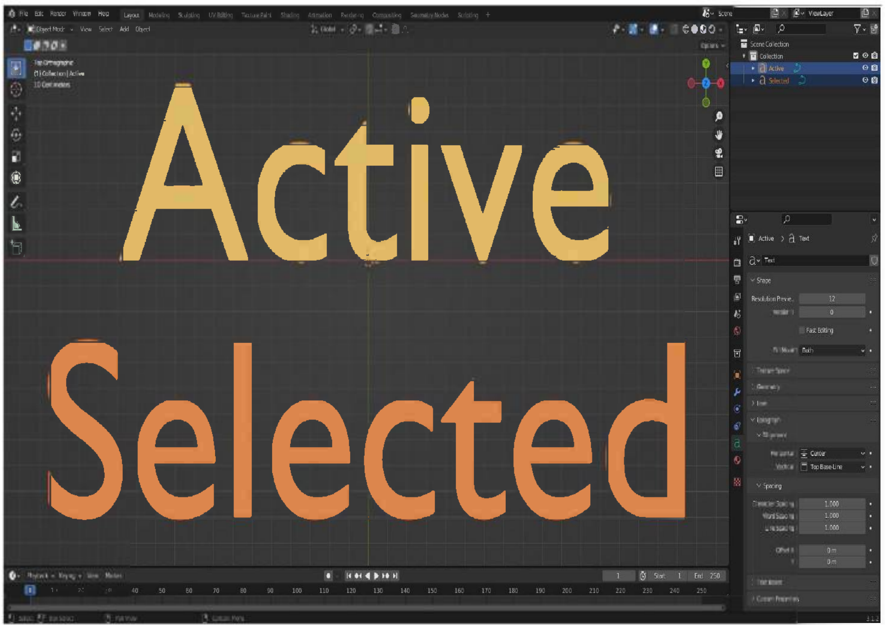


Figure 2.4: *Active objects are highlighted yellow, while Selected objects are highlighted orange*

2. Next, right-click the selected object names, and choose **Delete**, as shown in *Figure 2.5*.

Tip: Hotkeys for possible actions appear to the right of their text label. In this case, you can see that Delete is also bound to the X key. Also note that descriptive tooltips appear when your mouse cursor hovers over any of the available options.

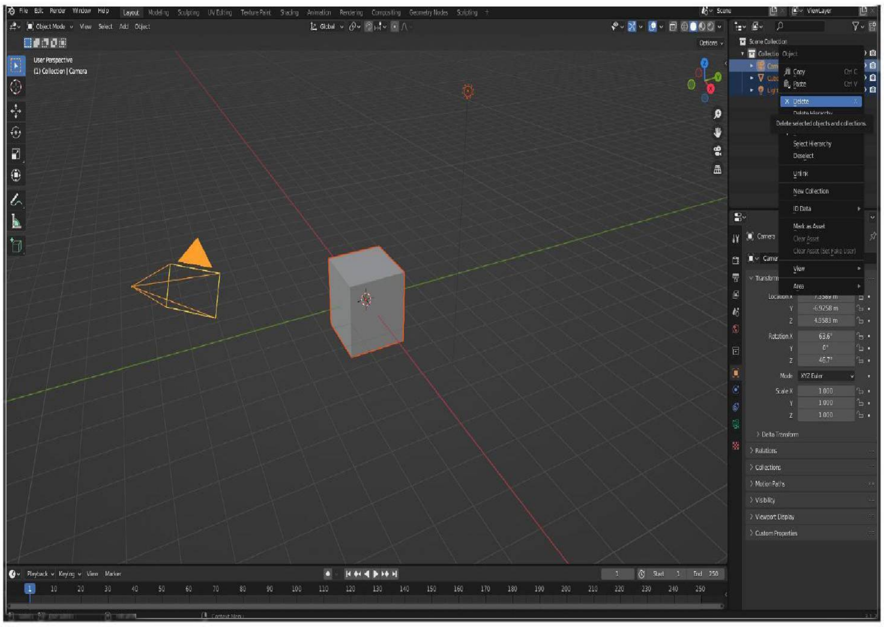


Figure 2.5: Right-clicking selected objects in the Outliner panel will open a menu

3. Once the default objects are deleted, hover your mouse cursor over the text in the upper left of the **3D Viewport**.

Notice that the **Add** option displays "Shortcut: **Shift A**" upon mouse-over.

Tip: Remember that shortcuts and hotkeys are context dependent, which means they perform different actions depending on what is selected and the location of your mouse cursor.

To use the Add shortcut, your mouse cursor must be within the bounds of a **3D Viewport** panel.

4. Click **Add** from the **menu** in the upper left of the **3D Viewport**, or **Shift + A** anywhere inside it, and select **Mesh** > **Monkey** from the menu, as shown in *Figures 2.6*:

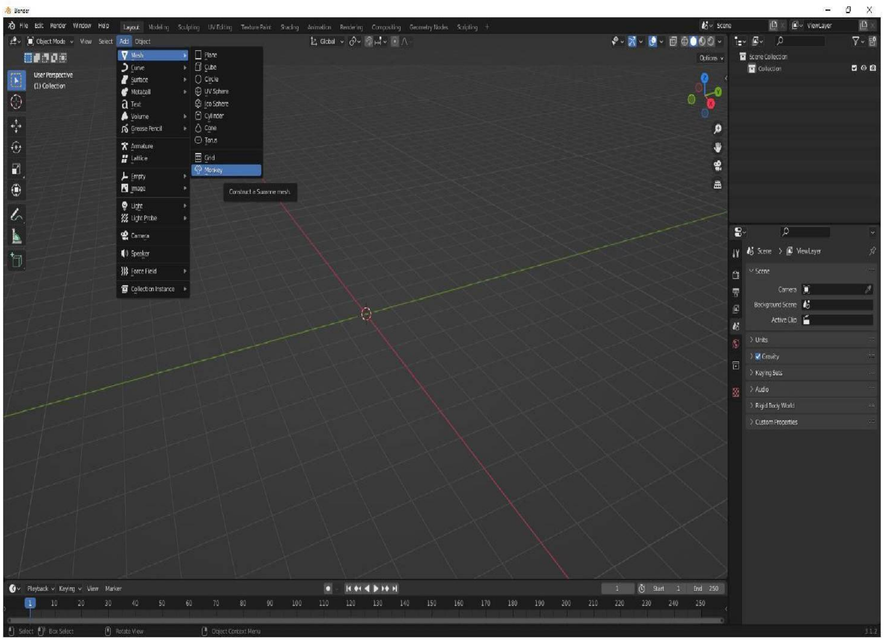


Figure 2.6: Selecting Add > Mesh > Monkey will add the 3D Blender mascot Suzanne to the scene

Figure 2.7 shows 3D object Suzanne, the Blender mascot:

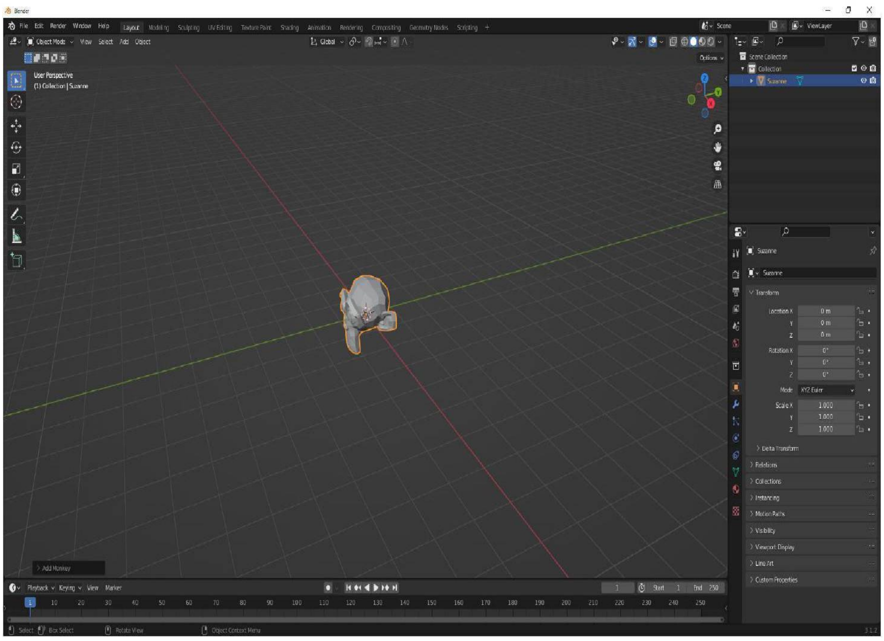


***Figure 2.7:** 3D object Suzanne, the Blender mascot*

View navigation

Now that we have a new 3D mesh in the scene, let's navigate around it.

To **zoom** the view, with your mouse cursor inside the **3D Viewport**, scroll your mouse wheel forward to zoom in, and backward to zoom back out.

To **orbit** the view around the selected object, press and hold the scroll wheel (middle mouse button), then drag the mouse around the screen.

To **pan** the view, first press **Shift**, and then press and hold the scroll wheel and drag the mouse around the screen.

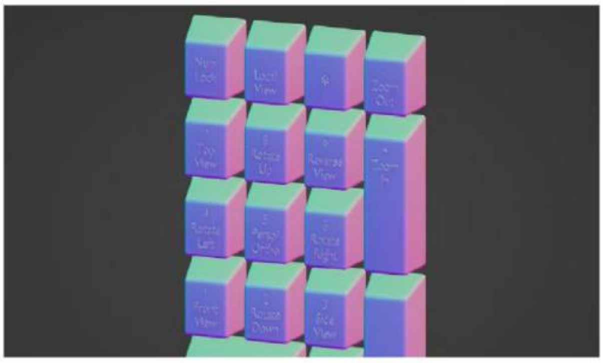
Note: At this point we are only manipulating the view, and not the object or the scene. It will be helpful to make this distinction in the next chapter when you start to manipulate the scale, rotation, and position (together, called transforms) of actual objects, as well as their sub-object components.

Once again, time is money, which means that wasted time is a waste of potential income. To increase the speed of your workflows, it will be helpful to utilize the keyboard number pad to navigate the view. There are corresponding menu options for all the following commands but using number pad keys saves steps, which allows you to stay in the flow of creation as you work. If you use a computer without a number pad, you may find it helpful to invest in a numeric keypad peripheral device.

From left to right in *Table 2.1*, as shown in *Figure 2.8*, the **viewport navigation** hotkeys are:

Shortcut	Hotkey
Lock Number Pad	Num Lock
Local View	Num /
N/A	Num *
Zoom Out	Num -
Top View	Num 7
Bottom View	Ctrl + Num 7
Rotate Up	Num 8
Reverse View	Num 9
Zoom In	Num +
Rotate Left	Num 4
Perspective / Orthographic	Num 5
Rotate Right	Num 6
Front View	Num 1
Rear View	Ctrl + Num 1
Rotate Down	Num 2
Right Side View	Num 3
Left Side View	Ctrl + Num 3
Camera View	Num 0
Frame Selected	Num .
Confirm	Num Enter

Table 2.1: *Viewport Navigation - Shortcuts and Hotkeys*



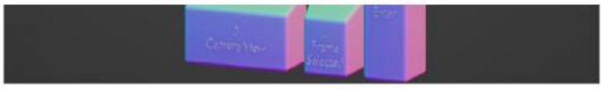


Figure 2.8: Number pad viewport navigation keys, labeled

Saving progress

At this point, we have not created anything new or remarkable, so it is not necessary to save our file. However, before you begin creating anything new in 3D, it is critical to know how to save your progress. Blender files are saved in the **BLEND** file format.

File saving in Blender is accessed through the topmost left menu. Under **File**, you can choose **Save**, **Save As**, or **Save Copy**. When you save a **BLEND** file for the first time, a file browser window will open for you to navigate to the preferred save location as shown in *Figure 2.9*. If your file has already been given a name and saved once, **Ctrl + S** or **Cmd + S** work too.

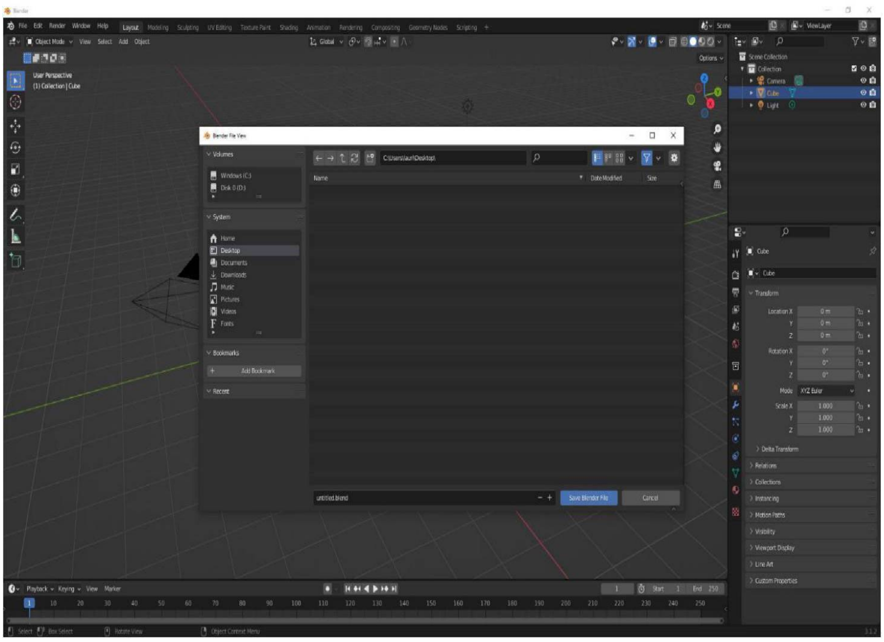


Figure 2.9: Navigating your local file system through Blender's File View

Save Copy is a unique feature which saves a separate version of your **BLEND** file in its current state, allowing you to continue working and changing the **open** file. When you save the open file again, the changes you made will not be applied to any copy you previously saved.

Note: The contents of Blender files can be accessed via older versions of Blender, although newer features of Blender will not work in older versions. Files saved in BLEND format can also be opened by several third-party programs, such as

in BLEND format can also be opened by several third-party programs, such as the game engine Unity or the 3D viewer Sketchfab, but not all 3D software can open BLEND files. Features specific to Blender will not work outside of Blender, even in software that can open BLEND files.

Save everything

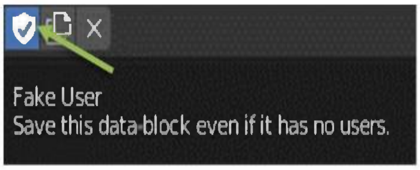
As with all creative software, it is imperative to save your work early and save often. To preserve *all* your progress though, it is important to know what data gets saved into a Blender file and what does not. That's right, **not everything you see on screen will be preserved whenever you save a 3D file**.

To understand why this is the case it is helpful to know that 3D software makes use of multiple kinds of data, to display and edit 3D objects. Some of this data is likely familiar, such as Images. Other kinds of data may require further explanation, such as Materials.

Some of this data is saved within a 3D file by default, while some of this data is kept outside of 3D software to keep it accessible through other applications (such as images, so they may be edited with image editing software). Such data is therefore "external" and must be saved separately.

For a chunk of data to be saved into a `BLEND` file, it must also be **used** within that file. This means that if you create data that can be transferred between objects such as an Animation, and then disconnect that data from any 3D object – through deletion of the animated object for example – so that the data block is no longer being "used" in the file, then that data will not be saved by default.

Fortunately, Blender does provide a way to save unused data as needed, by allowing us to mark certain blocks of data as having a **fake user** as shown in *Figure 2.10*. Additional use-cases for this feature will appear in later chapters.



Fake User
Save this data-block even if it has no users.

Figure 2.10: *The highlighted shield symbol indicates the associated data block will be saved with the BLEND file*

Packing to save

Images are one of the forms of data created with Blender or imported into Blender, which are "external" resources as described previously. This means that images must be saved separately from the `BLEND` file and will only remain referenced by Blender be saved separately from the **BLEND** file and will only remain referenced by Blender so long as neither the **BLEND** file nor the image file are moved from their containing folder. That is, unless you pack the external resource into the **BLEND** file.

Tip: If you ever open a BLEND file and find that objects have turned pink, that is a sign that external images have been moved, causing the file reference to be lost. To correct pink textures, reassign missing images manually and remember to move external resources with the BLEND file or pack resources before moving the file again. Instructions for assigning images to objects will be covered in Chapter 7, 3D Surfaces.

As shown in *Figure 2.11*, images made in Blender can be saved in a variety of popular formats including PNG and JPG.

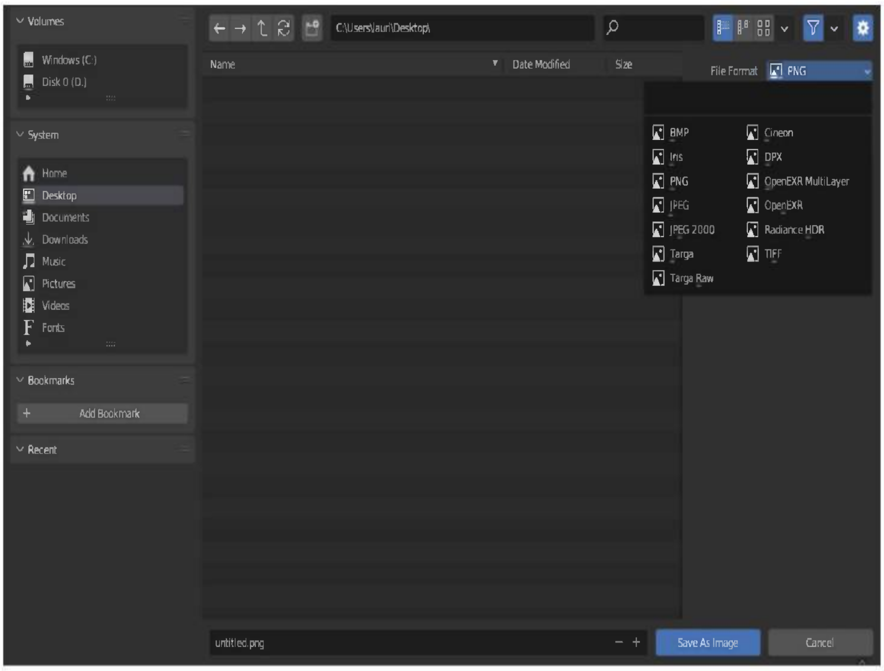


Figure 2.11: File Format options for saving an image made with Blender

To **pack** external resources like images into a **BLEND** file so you may share the **BLEND** file without losing references to the files, select **File** > **External Data** > **Automatically Pack Resources** (this will continue packing all resources as they are added to the file) or Pack Resources (this will only pack resources that are currently in use within the file) as shown in *Figure 2.12*:

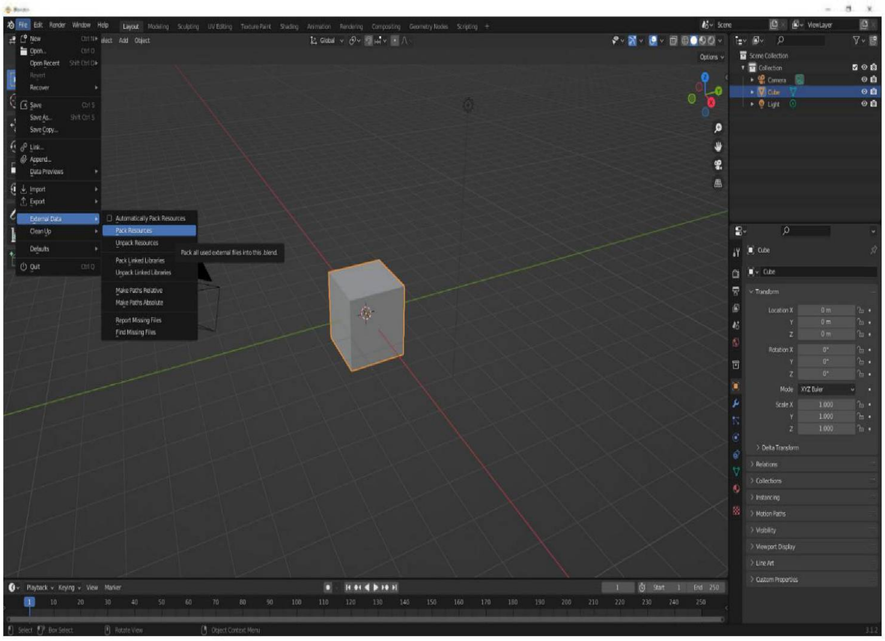


Figure 2.12: *Packing resources into the current BLEND file to maintain references*

Caution: the action of packing creates a duplicate of the packed file in your local file system which can seem inconsequential at first, but 3D files tend to be larger than common file formats and packing may lead to severe storage space problems when repeated.

Backup features

Blender provides multiple ways to back up our work.

Auto save

Blender's **Auto Save** feature is enabled by default in recent versions. Whether to leave this setting enabled or not is worth revisiting once you are familiar enough with Blender to start creating. Because 3D files are quite large due to their very nature of containing information in more than two dimensions, file saves (and undo histories) can easily accumulate to the point where they interrupt creative work by slowing the software.

For this reason, it is advisable to leave **Auto Save** enabled only until you develop an intuition about when your 3D work should be saved, and then disable it, when you are comfortable saving your files habitually. Enabling and disabling this feature is addressed later in this chapter in the section on editing Preferences.

Undo history

Another safety feature built into Blender is the Undo History which can be found in the topmost left menu under **Edit** > **Undo History**. It is helpful to look through your Undo History as shown in *Figure 2.13*, to understand the types of actions stored there.

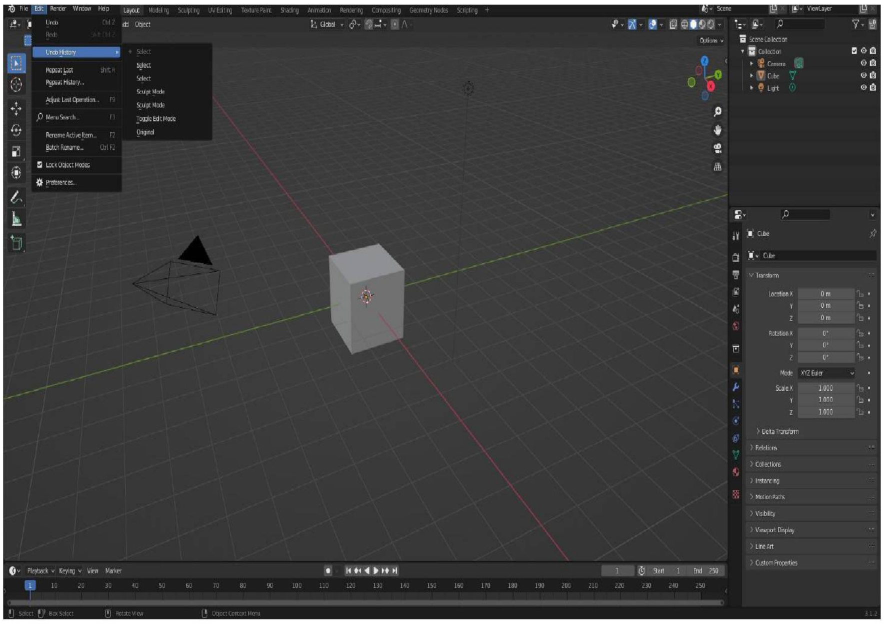


Figure 2.13: Viewing the current session's Undo History, located under Edit > Undo > Undo History

Ctrl + Z or **Cmd + Z** works to Undo actions one at a time, while **Ctrl + Shift + Z** will Redo actions. Keep in mind that Undo History is purged from memory when a Blender file is closed, even from a system crash. During a session, Blender keeps a maximum of 32 Undo steps in memory by default. This maximum can be increased or decreased through Blender's Preferences which will be discussed later in this chapter.

Like many actions in Blender, Undo is context-dependent. This means, for example, that when you switch from Object Mode to **Edit Mode**, that mode switch is recorded to your overall **Undo History**, such that **Ctrl + Z** will also reverse the toggling of modes. Similarly, changes to Properties are also recorded in the **Undo History**, like changes to **Brush settings**. This can be surprising to users of other creative software like Photoshop — in which changes to tool settings are independent of the action history — but is very convenient for experimenting with unfamiliar settings.

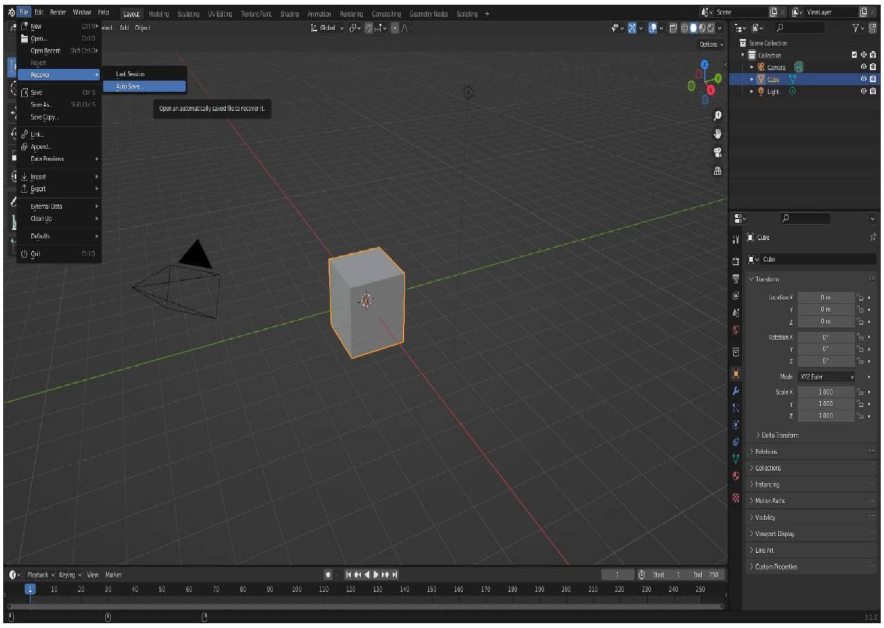
File reverting

If all else fails, and we find it necessary to restart from the last saved progress, Blender allows us to revert to a previously saved version of any **BLEND** file. File

Revert is accessed via the upper left **File menu** > **Revert**, and requires a confirmation click before the previously saved file will reload.

File recovery

In Blender, there are a few ways to recover lost work. In case of software or computer crashes or a file is corrupted, the **Auto Save file** can be accessed via **File** > **Recover** > **Auto Save** as shown in *Figure 2.14*:



Open an automatically saved file to recover it.

***Figure 2.14:** Accessing the Auto Save file via File > Recover*

In the case where you accidentally closed a **BLEND** file without saving it, you can reopen Blender, and choose Recover **Last Session** in the bottom left of the splash

screen, as shown in *Figure 2.15*, or choose the **Last Session** option from the **File** > **Recover menu** as shown in *Figure 2.14*:

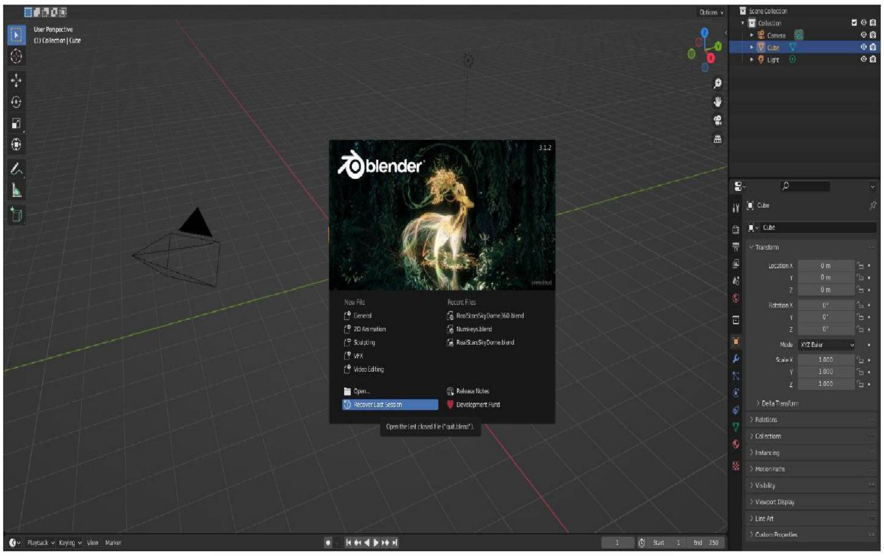


Figure 2.15: Recover Last Session via the splash screen

In the case where you accidentally saved over a **BLEND** file, you can navigate to the system folder where the original file is located, there will be a copy of the previous version of that file named BLEND1 which you can double click to open with Blender.

Within Blender, you can also access this file by selecting **File** > **Open**, navigating to the folder where your original file is located, and in the upper right Filters menu (with a funnel icon) enabling **Backup Blend Files** as shown in *Figure 2.16*. The **BLEND1** file containing the previous save can then be opened from there.

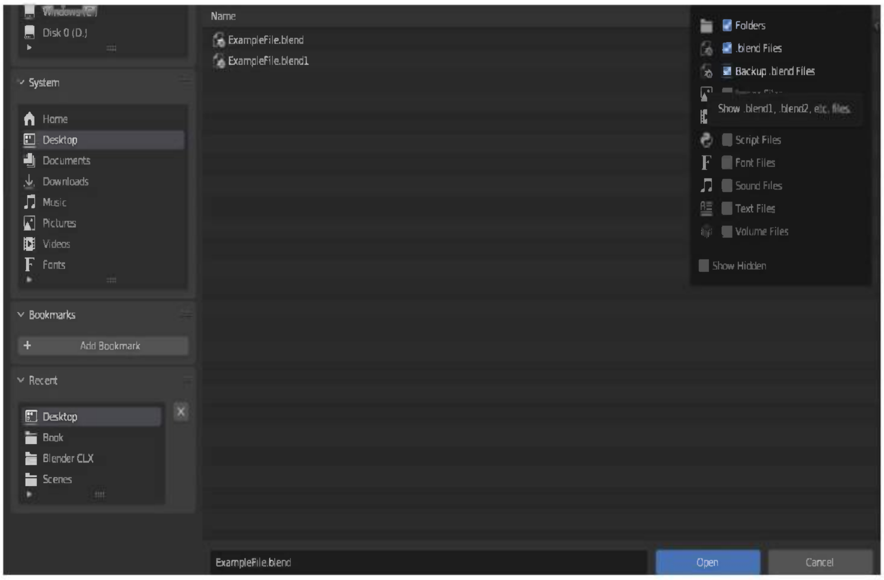


Figure 2.16: Revealing Backup .blend Files in the Blender File View

Customizing the interface

Every panel in Blender can be moved, split, maximized, resized, or converted into another panel type, called an **Editor Type**. The appearance of every panel can also be adjusted, menus and headers flipped between top and bottom, font sizes and colors adjusted, and so on.

The utility of these features will become clearer in later chapters, but for now, think of creating in 3D like using a workbench, table, or desktop. First, you gather the necessary tools and materials and arrange them in such a way that we have a clear central space to work, with our most frequently used items close at hand. As you work and our project progresses through different phases, we'll grab and use items, move things around, put some things away, and bring out new items as needed.

As described earlier, Active Workspace tabs provide several pre-set layouts that are convenient for common tasks, but Blender is as flexible as you are creative. In this second exercise, we will practice customizing the interface by splitting the **3D Viewport** into four equal panes, and then learn how to revert or save the custom layout.

Exercise 2.2

When we begin to create in three-dimensional space, one of our first discoveries will be that visible objects can easily hide or "occlude" other objects, while movement of the mouse and keypresses can have both visible and invisible results. For this reason, it is common for 3D creators to work with different views simultaneously.

1. From the default Layout view, right-click the top menu bar of the **3D Viewport** and select **Horizontal Split** as shown in *Figure 2.17*. A thin horizontal line will appear in the middle of the viewport, which you may move up and down freely until you click to confirm, but before clicking, press **Ctrl** or **Cmd** and notice how your mouse movements snap between one invisible increment and another.

2. Move the horizontal bar to the middle increment and left-click to confirm:

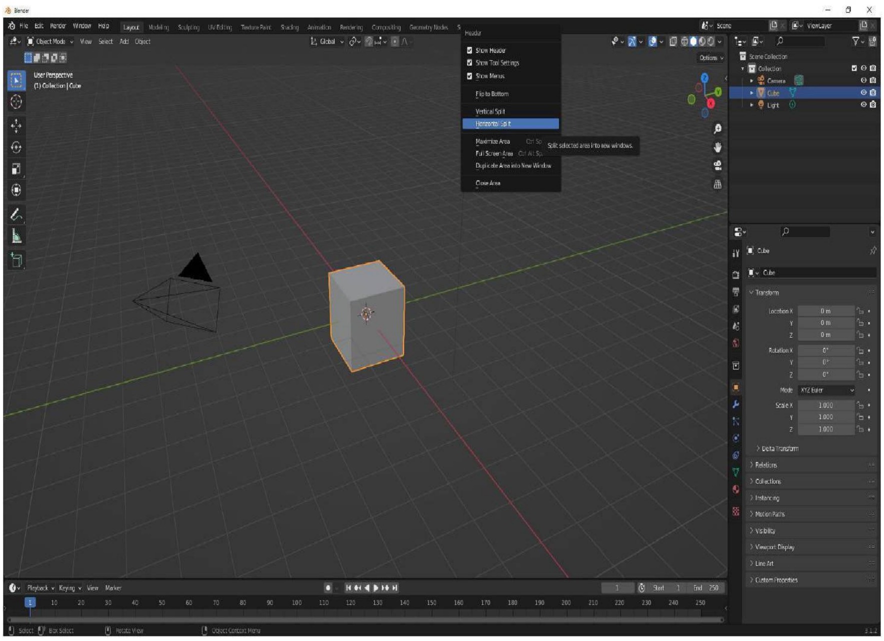


Figure 2.17: Selecting Horizontal Split from the 3D Viewport menu

Your **3D Viewport** is now split into two equal halves, upper and lower as shown in *Figure 2.18*. If you clicked too early, simply hover your mouse cursor over the split until the double arrow cursor appears, press **Ctrl**, and snap it up or down as needed:

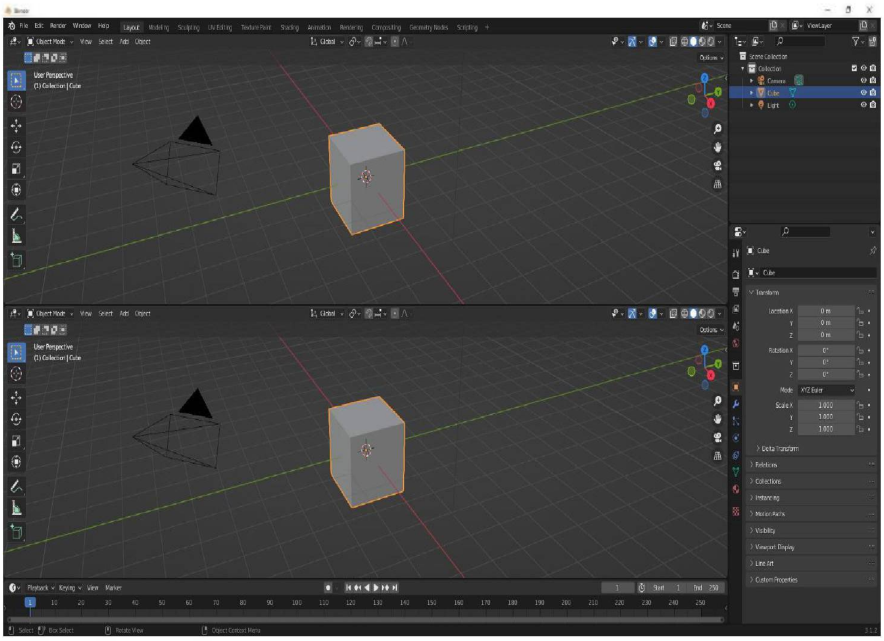


Figure 2.18: Splitting 3D Viewport into equal halves

Before moving on to splitting the view one more time, we will practice collapsing the newly created panel.

3. Hover your mouse cursor over the upper left or right corner of the bottom **3D Viewport**, until you see a plus-shaped cursor appear.
4. With the plus-shaped cursor, left-click and drag upward, just until the upper panel darkens.
5. Release the mouse cursor, and the lower pane will expand into the space where the upper pane existed.

Once again, it is helpful to note that we have not changed anything about our 3D objects or scene, we have only changed our view. Now that we know how to revert your changes, let's repeat the previous steps to split the panel horizontally again, as shown in *Figure 2.17*. Next, lets split the top and bottom **3D Viewports** vertically.

6. Right-click the menu bar above each **3D Viewport**, one at a time, and this time select **Vertical Split**. Press **Ctrl** and snap the vertical line to the middle of each **3D Viewport**. Now you should have what is commonly called a "quad view" as shown in *Figure 2.19*:

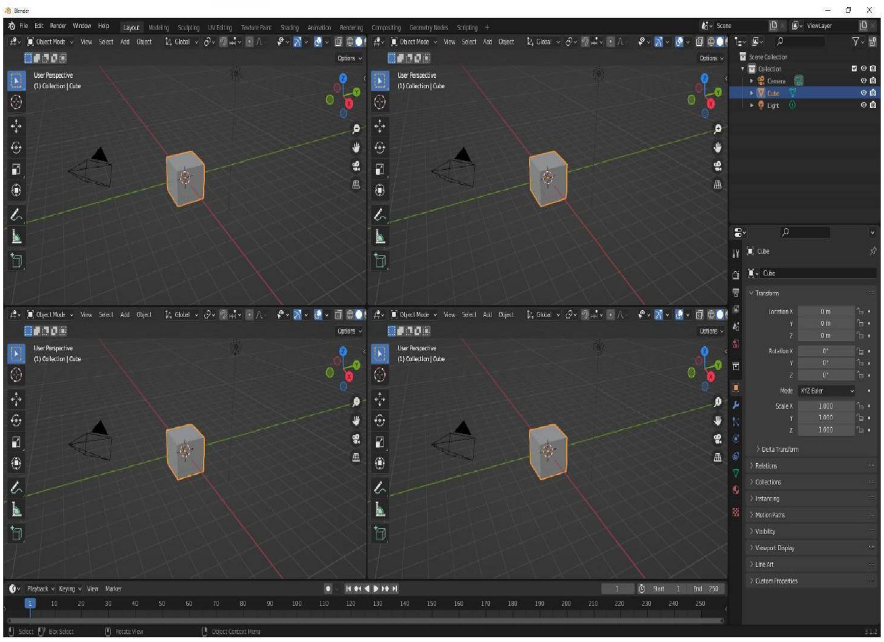


Figure 2.19: *Quad 3D Viewports*

The utility of a "quad view" will become more apparent when you begin polygonal modeling in *Chapter 4, Polygonal Modeling*. Of course, not all 3D creators use this layout, and nor must you . However, looking at a 3D model in the quad view will help illustrate an important concept of 3D creation that is difficult to describe in text is difficult to describe in text.

7. To see this concept for yourself, click the green transparent circle on the Gizmo in the upper left **3D Viewport** as shown in *Figure 2.20*, then the solid red circle on the upper right **3D Viewport** Gizmo, and finally click the solid blue circle on the lower left **3D Viewport** Gizmo.

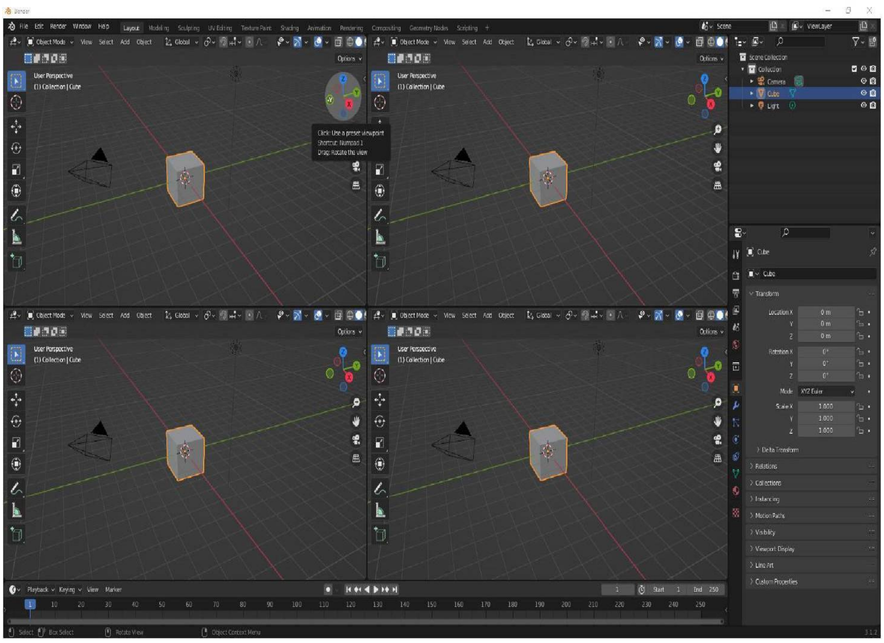


Figure 2.20 Changing the perspective shown in each 3D Viewport

If you still have a Suzanne in your scene from the earlier exercise, skip to the next paragraph.

8. If you started this exercise with a new project, the new views of a default Camera, Cube and Light may be unremarkable, so let's select all these objects again from the Outliner, and with your mouse cursor still over the Outliner, click the **Delete** or **X** key to remove them from the scene.

9. We'll employ these objects in *Chapter 10, Images and Video* instead. Now, with your mouse cursor over any of the four **3D Viewports**, press **Shift** + **A**, and select **Mesh** > **Monkey** from the menu.

Now, you can see a single Suzanne model from the Front, Right, Top, and Perspective views simultaneously, as shown in *Figure 2.21*.

10. With your mouse cursor over any of the **3D Viewports** and Suzanne still selected, press **G**, then **Y**.

11. Move your mouse around, and notice what happens in each of the **3D Viewports**. In three of the views, you will see Suzanne move, but in the fourth, she will appear motionless. This lack of motion is an illusion.

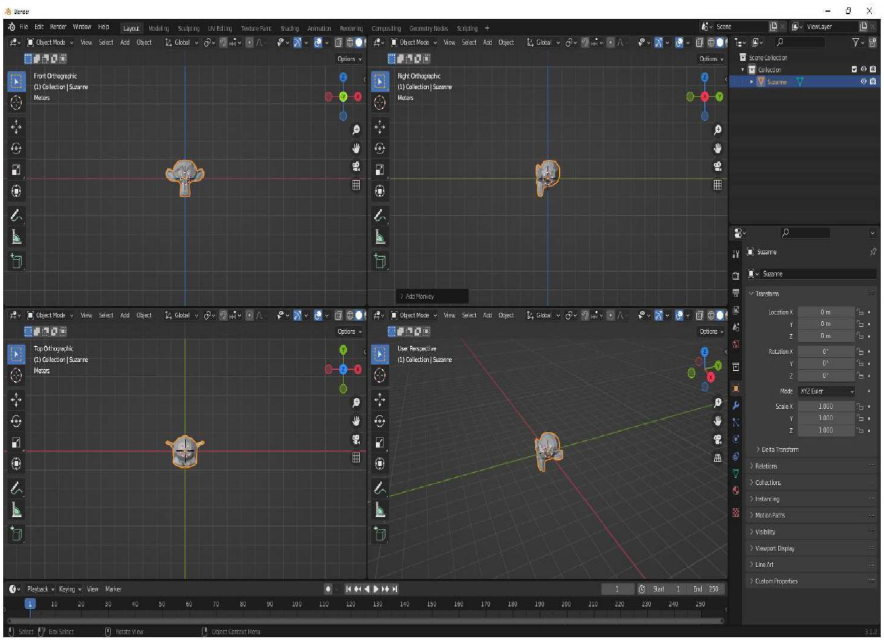


Figure 2.21: Simultaneous views of Suzanne in Front, Right, Top and Perspective

The cause of this illusion is the difference in perspectives shown in each of the **3D Viewports**. In the upper left corners of each of the four **3D Viewports** there are text descriptions of the perspective shown within that panel. In the upper left **3D Viewport**, where you clicked the transparent green circle on the Gizmo in the upper right, the view description reads Front Orthographic.

Below the view description is more information we will use in later chapters, including the frame number of the current animation in parenthesis, the name of the currently Active Collection, the name of the Active selected model, and the current unit of measure.

Perspective

The bottom right **3D Viewport** description reads User Perspective instead of Orthographic. Technically, Perspective and Orthographic are both "perspectives," but the term Perspective is used to differentiate the distorted perspective that mimics the way human eyes and camera lenses "see" the world, and the mathematically accurate Orthographic perspective. The Perspective mode is not actually a single perspective, but rather Perspective refers to any amount of distortion applied to the view.

Many people feel more comfortable working in Perspective view, at least at first, because it feels more natural to the human eye. Unfortunately, relying on the distorted Perspective view can ruin your 3D creations by misleading your eye.

1. To illustrate this problem, open the **Sidebar menu** using the **N** hotkey, select the **View tab** from the right-hand side, and set the View's Focal Length to 15mm by clicking in the text area which reads 50mm, typing 15, then pressing the **Enter** key.

   At first, nothing is visibly amiss other than Suzanne appearing to shrink.

2. Next let's frame the still-selected Suzanne in the view by pressing the number pad period key as shown in *Figure 2.22*. Now the distortion of **Perspective view** is on full display.

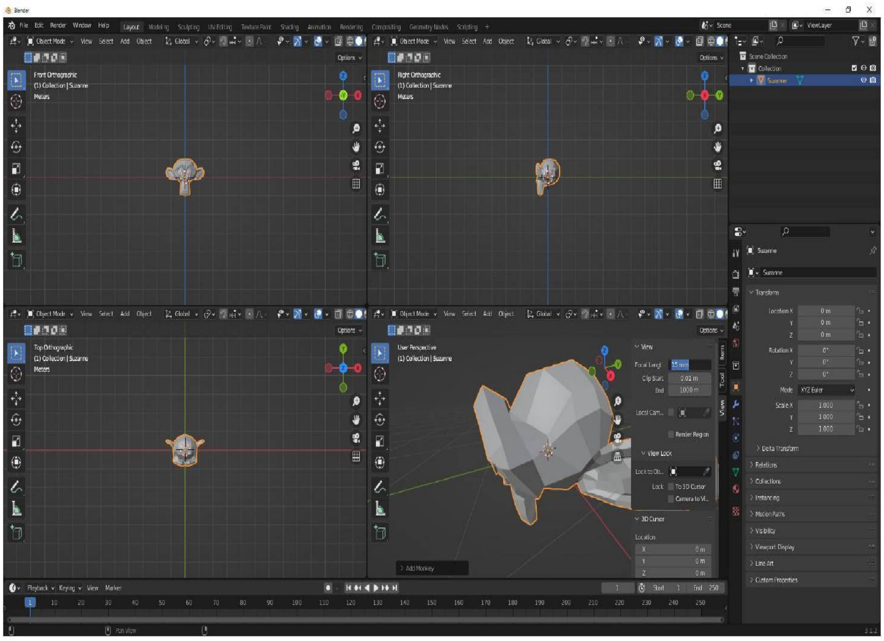


Figure 2.22: *Bottom Right 3D Viewport, Suzanne framed in User Perspective view mode, with 15mm Focal Length*

To wrap up this exercise, lets save this custom layout, and then learn how to revert to the default again.

3. At the top of the Blender application, click the plus-sign at the right end of the **Workspace** tabs, and choose **Duplicate Current** as shown in *Figure 2.23*:

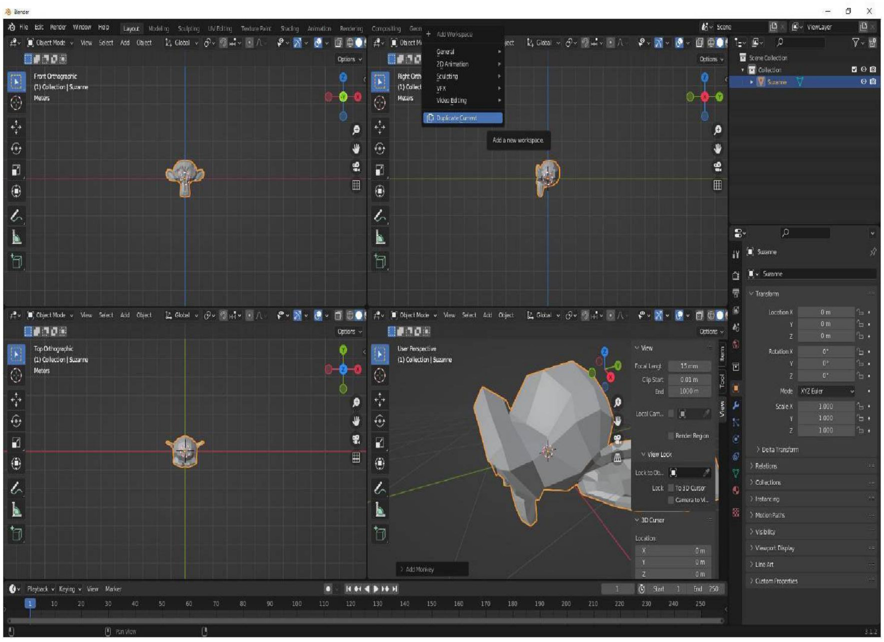


Figure 2.23: *Creating a New Workspace from the current layout*

This makes a copy of the **Layout** tab in its current state appear to the right of Layout, appended in Blender's unique numbering format, "Layout.001". These changes have not been saved to the original Layout tab, which will revert the next time Blender is closed and reopened, unless you save the file. The new tab can be renamed by double clicking on the text and typing over the existing text.

4. Right-click the **tab** label, to see the options for changing this tab's place in the tab group order.

If you would rather have Blender open with this entire file as-is – Suzanne alone, in a quad view – under **File** > **Defaults**, select **Save Startup File** as shown in *Figure 2.24*. If we close and reopen Blender now, the new file will begin in this configuration. If we don't want Blender to open this way from now on, we can simply navigate to **File** > **Defaults**, and this time choose **Load Factory Settings.**

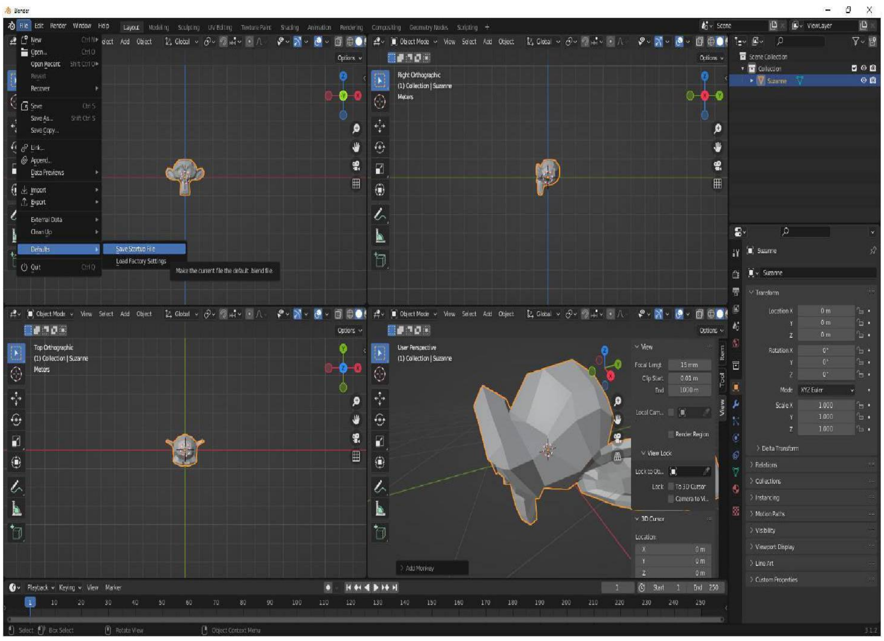


Figure 2.24: Saving the current layout to the Startup file

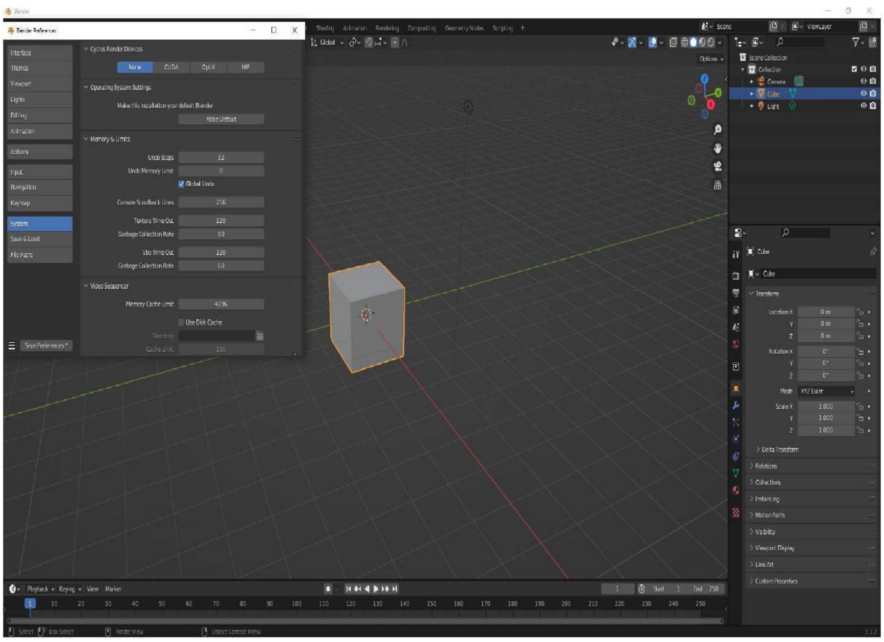
Unfortunately, this will remove the new **Workspace** tab we just made, but if we close the file without saving, the previously saved Startup File will appear again. Factory Settings will only become the default Startup File again if we choose **File** > **Defaults** > **Load Factory Settings** and then choose **File** > **Defaults** > **Save Startup File**.

The way to preserve the original **Layout** tab along with a custom one is to duplicate the **Factory Setting** tab first, make changes to the layout in the duplicate tab, then choose **Save Startup File** from **File** > **Defaults**.

Editing preferences

The extent to which we can edit Blender's Preferences is too vast for an introductory section to cover in full, but here we will highlight the types of changes that can be made there, so you will know where to go whenever you discover a need to modify

Blender's Preferences. Blender Preferences are accessed via **Edit** > **Preferences**, which opens a modal window as shown in *Figure 2.25*:


Cycles Render Devices: None [✓], CUDA, OptiX, HIP

Operating System Settings: Make this installation your default Blender — Make Default

Memory & Limits

Undo Steps	32
Undo Memory Limit	0
Global Undo	✓
Console Scrollback Lines	256
Texture Time Out	120
Garbage Collection Rate	60
Vbo Time Out	120
Garbage Collection Rate	60

Video Sequencer

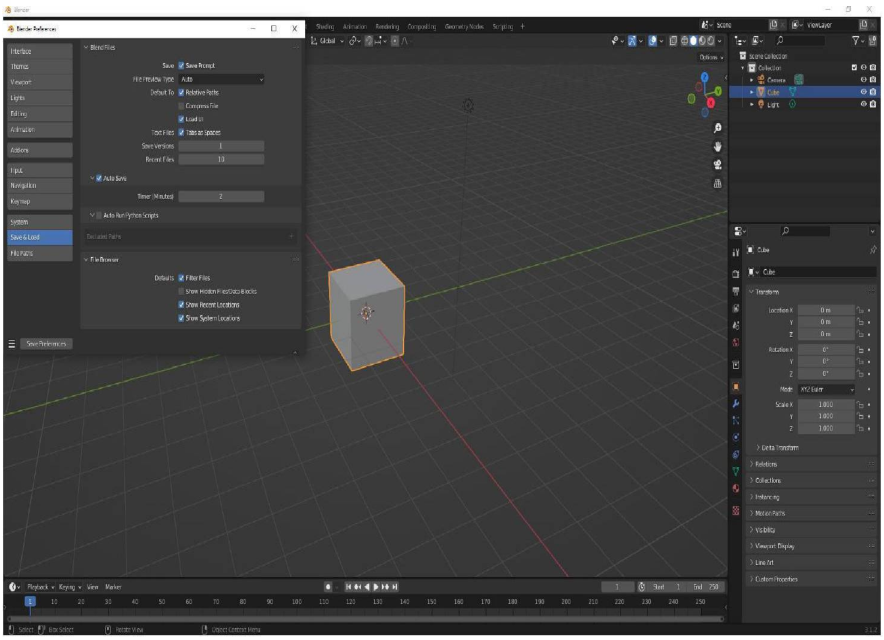
Memory Cache Limit	4096
Use Disk Cache	☐
Directory	
Cache Limit	100

Figure 2.25: Opening the Preferences modal window via Edit > Preferences

As mentioned earlier, increasing the maximum **Undo Steps** can be accessed via the **Preferences** window. Under **System** in the left-hand side as shown in *Figure 2.25*, the number of **Undo Steps** is listed under the third group, Memory & Limits.

Tip: To change any numerical value, anywhere in Blender, click once in the text field, begin typing, and press Enter to confirm.

Also mentioned earlier, **Auto Save** can be disabled and re-enabled through the **Preferences** panel. To access the **Auto Save** toggle, in the **Preferences** window, select **Save & Load** from the lower left-hand side as shown in *Figure 2.26*, and check or uncheck the box to the left of **Auto Save**. A highlighted checkmark means the feature is enabled.



Blend Files

Save ✓ Save Prompt

File Preview Type Auto

Default To ✓ Relative Paths

☐ Compress File

✓ Load UI

Text Files ✓ Tabs as Spaces

Save Versions 1

Recent Files 10

✓ Auto Save

Timer (Minutes) 2

☐ Auto Run Python Scripts

Excluded Paths

File Browser

Defaults ✓ Filter Files

☐ Show Hidden Files/Data-Blocks

✓ Show Recent Locations

✓ Show System Locations

Figure 2.26: Toggling Auto Save on or off

Essential shortcuts and hotkeys

In each upcoming chapter, we will add new tables of Shortcuts and Hotkeys based on what we have learned. In this chapter, we have learned the following Shortcuts and Hotkeys:

Shortcut	Hotkey
Select	Left mouse click
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc

Table 2.2: Essential Shortcuts and Hotkeys

You can also access menu options by assigning them your own Shortcuts. Right-click any menu option to set it as a Quick favorite accessed via the **Q** key or Assign Shortcut to assign the menu option to a key of your choosing. Every single key can be remapped individually via **Edit** > **Preferences** > **Keymap**, but this is an advanced option that likely won't be useful until later.

Getting help

The first resources to reach for when you need more help than this book can provide are the Blender Foundation Support and Documentation pages. At **https://www.blender.org/support/**, the Blender Foundation provides basic Documentation in the form of a User Manual, as well as the advanced documents Python API Reference, and Blender for Developers. Also provided on the Blender Support page are links to free Tutorials, Community Support links, an FAQ, and Blender's Report a Bug forum.

Aside from the support directly from Blender Foundation, there are thousands of popular community resources available online, both free and paid. Remember that while free support and tutorials are an integral part of the Blender ecosystem and democratizing tech, the people offering support are doing laborious work. So, if you have the means to pay for tutorials or training, please do so!

A simple internet search preceded with the term Blender, along with the version number 3, filtered to the past year, should bring up the most useful results.

Conclusion

We now have a firm understanding of how to get started and navigate our way around Blender. Rest assured that we have barely scratched the surface, but don't let the sheer number of options both on-screen and hidden behind menus alarm you. If at any point you lose your way, you are now equipped to revert to default settings, and reliably save or restore lost work. In the following chapter, we will begin to apply this knowledge and create our first 3D model from scratch, to become familiar with the anatomy of a 3D object and understand 3D modeling in general.

Points to remember

- Practicing Hotkeys will help you develop muscle memory, reducing the time it takes to complete a task, Hotkey combos for Shortcuts appear to the right of their matching menu options
- Right-click any menu option to set it as a Quick favorite accessed via the **Q** key, or Assign Shortcut to assign the menu option to a key of your choosing
- Refer to Blender official Documentation and User Manual at **https://docs.**

- Refer to Blender official Documentation and User Manual at **https://docs.blender.org/** for a refresher, troubleshooting, or to learn more about the interface and common tasks

Questions

1. What are the mouse and keyboard commands to zoom, orbit, and pan the view?
2. What is the most recent version of Blender?
3. What are two ways to save your BLEND files?
4. What happens to my BLEND file if my computer crashes?

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

CHAPTER 3
General 3D Concepts

Introduction

Now that we have learned how to install and navigate Blender, we will get hands-on experience with general concepts that are useful for any 3D creation. As we saw in *Chapter 2, Installation and Interface*, working in 3D space is a step more complicated than working in 2D space as with documents and images, but you do not need to be intimidated by that added complexity. The third dimension is just that: one more, of two familiar dimensions.

If you have ever picked up a pencil, you can already create it in 2D. Connect three or more lines together, and you begin to make 2D shapes. You have likely already created in 3D as well if you have ever handled paper, fabric, or clay; prepared food; crafted with textiles; carved into wood or stone; dug in sand or soil; or built a structure

Structure

This chapter introduces the general 3D concepts that apply to all 3D software, giving readers the necessary foundation to begin creating in 3D. Topics to be covered include:

- The anatomy of 3D objects: coordinates, vertices, edges, faces, and origins

- Orienting yourself in 3D space for fast, accurate 3D creation
- Exercise: creating a new 3D object from vertices, edges, and faces
- Understanding transforms: position, rotation, and scale
- Tools of the trade: gizmos, parameters, hotkeys, shortcuts, and toolbars

Objectives

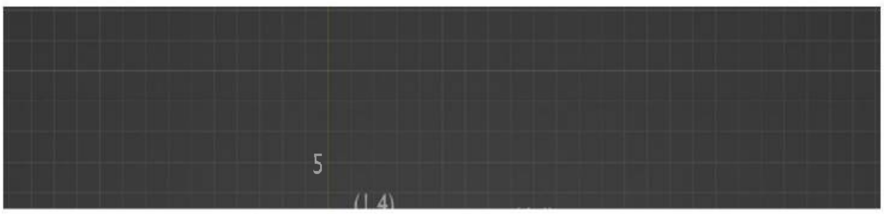
After reading this chapter you will have a thorough understanding of general 3D concepts and will be ready to begin modeling your own designs with Blender.

Anatomy of 3D Objects

You may have heard some of the terminologies we will use here in math lessons. Math courses tend to delay discussing the third dimension, not because 3D math is much more difficult, but because it comes later in a sequence of logical steps. Rest assured, the same basic addition, subtraction, multiplication, and division that works in 2D, also works in 3D.

Note: Advanced math can be helpful to 3D creators but is not necessary for creating in 3D. Simply keeping the additional dimension in mind will help us avoid many common mistakes and pitfalls.

If you have studied Geometry in the context of learning math, you may already be familiar with axes and coordinates, and how they can map the extents of 2D shapes when connected, such as the triangle shown in *Figure 3.1*:



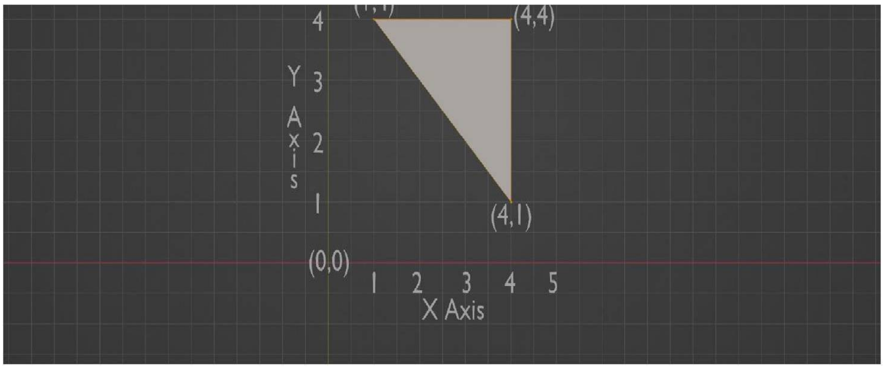


Figure 3.1: Mock graph of a 2D shape, with x and y coordinates, labeled

Essentially, coordinates are an address to the locations of points where two or more lines intersect on a two-dimensional grid. Coordinates are written in parenthesis with the x value listed first, followed by a comma and then the y value as shown in *Figure 3.1*. The point at (0,0) is called the origin. Coordinates are useful for recording and communicating the size, position, and outline of basic shapes such as points, lines, triangles, rectangles, and even circles.

In essence, this is what 3D software does: It stores data such as the coordinates of all the points, lines, and shapes that make up a 3D object, along with information about the intended appearance of the surface – such as which side is facing outward – and allows that data to be saved and shared. In this way, 3D is merely an extension of 2D shapes into one additional dimension as shown in *Figure 3.2*:

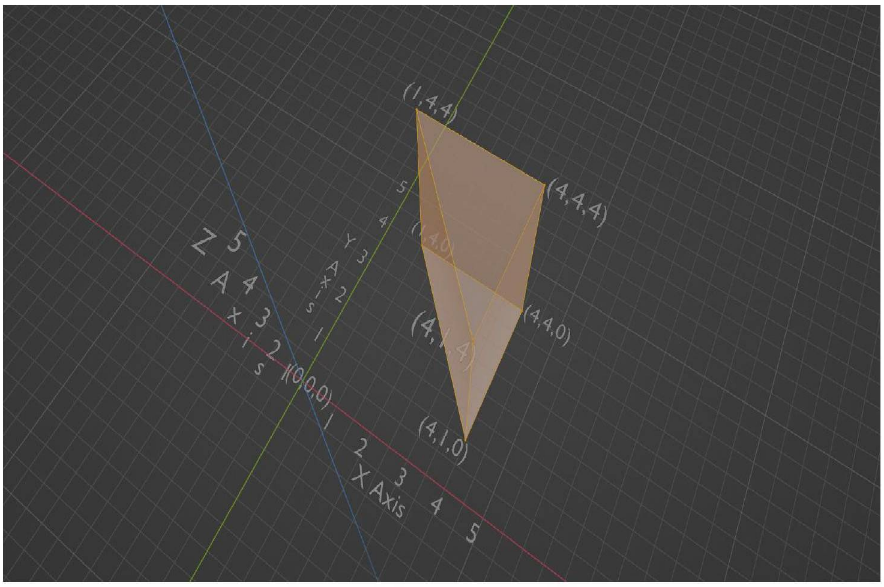


Figure 3.2: Mock graph of the previously shown 2D triangle extended into a 3D shape, with x, y, and z coordinates labeled

The surface of a 3D object, as shown in *Figure 3.3*, is composed of vertices (points), connected by edges (lines), filled with faces (shapes):

connected by edges (lines), filled with faces (shapes).

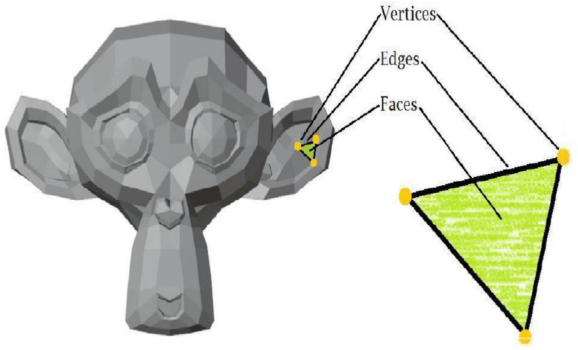
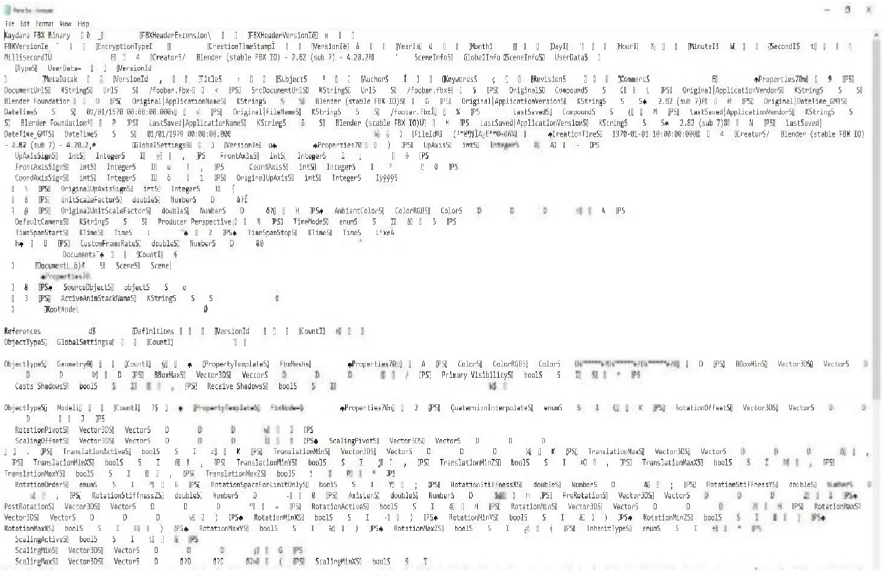


Figure 3.3: A 3D object with vertices, edges, and faces labeled

Note: Simpler 3D software may not expose these sub-object components to direct editing, but all 3D objects are made up of vertices, edges, and faces behind the scenes.

As touched, the coordinates of Vertices, Edges, and Faces are not the only data that 3D objects are composed of. On top of their surface, 3D software also displays color, to mimic the casting of light in a variety of ways that we will learn more about in *Chapter 7, 3D Surfaces*. In *Figure 3.3*, the 3D Suzanne model is rendered with a solid grey surface, with simulated lighting in a "flat shaded" mode, in a scene with a white background.

At the most basic level, all 3D objects are just collections of the preceding kinds of data, in a format that is readable by 3D software. To illustrate this point, observe that is even possible to open a 3D file with a text editor as shown in *Figure 3.4* and find some recognizable terms, although not in a useful format for many humans:

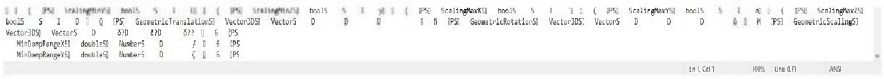


```
Kaydara FBX Binary  FBXHeaderExtension\  FBXHeaderVersionI  x
FBXVersionI  EncryptionTypeI  CreationTimeStamp  VersionI  YearI  MonthI  DayI  HourI  MinuteI  SecondI  t
MillisecondIU  4  Creator5/  Blender (stable FBX IO) - 2.82 (sub 7) - 4.20.2  SceneInfoS  GlobalInfo SceneInfoS  UserDataS
  TypeS  UserData-  VersionId
  MetaData  VersionId  ,  TitleS  SubjectS  AuthorS  KeywordsS  RevisionS  CommentS  Properties70  9  PS
DocumentUrlS  KStringS  UrlS  S  /foobar.fbx  <  PS  SrcDocumentUrlS  KStringS  UrlS  S  /foobar.fbx  $  PS  OriginalS  CompoundS  S  PS  Original|ApplicationVendorS  KStringS  S  S
Blender Foundation  0  PS  Original|ApplicationNameS  KStringS  S  S  Blender (stable FBX IO)  G  PS  Original|ApplicationVersionS  KStringS  S  S  2.82 (sub 7)  M  PS  Original|DateTime_GMTS
DateTimeS  S  S  01/01/1970 00:00:00.000  PS  Original|FileNameS  KStringS  S  S  /foobar.fbx  %  PS  LastSavedS  CompoundS  S  M  PS  LastSaved|ApplicationVendorS  KStringS  S
S  Blender Foundation  P  PS  LastSaved|ApplicationNameS  KStringS  S  S  Blender (stable FBX IO)  H  PS  LastSaved|ApplicationVersionS  KStringS  S  S  2.82 (sub 7)  N  PS  LastSaved|
DateTime_GMTS  DateTimeS  S  S  01/01/1970 00:00:00.000  FileIdR  CreationTimeS  1970-01-01 10:00:00:000  4  CreatorS/  Blender (stable FBX IO)
- 2.82 (sub 7) - 4.20.2  GlobalSettings  VersionI  Properties70  PS  UpAxisS  intS  IntegerS  PS
  UpAxisSignS  intS  IntegerS  PS  FrontAxisS  intS  IntegerS  I  PS
  FrontAxisSignS  intS  IntegerS  u  PS  CoordAxisS  intS  IntegerS  I  0  PS
  CoordAxisSignS  intS  IntegerS  1  PS  OriginalUpAxisS  intS  IntegerS  Iÿÿÿÿ5
  5  PS  OriginalUpAxisSignS  intS  IntegerS  {
  8  PS  UnitScaleFactorS  doubleS  NumberS  D
  @  PS  OriginalUnitScaleFactorS  doubleS  NumberS  D  H  PS  AmbientColorS  ColorRGBS  ColorS  D  D  D  4  PS
  DefaultCameraS  KStringS  S  S  Producer Perspective  %  PS  TimeModeS  enumS  S  3  PS
  TimeSpanStartS  KTimeS  TimeS  2  PS  TimeSpanStopS  KTimeS  TimeS
  h  8  PS  CustomFrameRateS  doubleS  NumberS  D
        Documents  CountI  §
      Document  SceneS  Scene
        Properties70
  8  PS  SourceObjectS  objectS  S
  3  PS  ActiveAnimStackNameS  KStringS  S  S
        RootNode

References  d$  Definitions  VersionId  CountI
ObjectTypeS  GlobalSettings  CountI

ObjectTypeS  Geometry  CountI  PropertyTemplateS  FbxMeshS  Properties70  A  PS  ColorS  ColorRGBS  ColorS  D  PS  BBoxMinS  Vector3DS  VectorS
    D  D  D  PS  BBoxMaxS  Vector3DS  VectorS  D  D  D  PS  Primary VisibilityS  boolS  S  PS
  Casts ShadowsS  boolS  S  ,  PS  Receive ShadowsS  boolS  S

ObjectTypeS  Model  CountI  PropertyTemplateS  FbxNodeS  Properties70  2  PS  QuaternionInterpolateS  enumS  S  I  K  PS  RotationOffsetS  Vector3DS  VectorS  D  D
    D  PS
  RotationPivotS  Vector3DS  VectorS  D  D  D  PS
  ScalingOffsetS  Vector3DS  VectorS  D  D  D  PS  ScalingPivotS  Vector3DS  VectorS  D  D  D
  .  PS  TranslationActiveS  boolS  S  I  K  PS  TranslationMinS  Vector3DS  VectorS  D  D  D  K  PS  TranslationMaxS  Vector3DS  VectorS  D  D  D  ,
 PS  TranslationMinXS  boolS  S  I  ,  PS  TranslationMinYS  boolS  S  I  ,  PS  TranslationMinZS  boolS  S  I  ,  PS  TranslationMaxXS  boolS  S  I  ,  PS
TranslationMaxYS  boolS  S  I  ,  PS  TranslationMaxZS  boolS  S  I  PS
  RotationOrderS  enumS  S  I  6  PS  RotationSpaceForLimitOnlyS  boolS  S  I  ;  PS  RotationStiffnessXS  doubleS  NumberS  D  ;  PS  RotationStiffnessYS  doubleS  NumberS  D
  ;  PS  RotationStiffnessZS  doubleS  NumberS  D  0  PS  AxisLenS  doubleS  NumberS  D  H  PS  PreRotationS  Vector3DS  VectorS  D  D  D  PS
PostRotationS  Vector3DS  VectorS  D  D  D  +  PS  RotationActiveS  boolS  S  I  H  PS  RotationMinS  Vector3DS  VectorS  D  D  D  H  PS  RotationMaxS
Vector3DS  VectorS  D  D  D  )  PS  RotationMinXS  boolS  S  I  )  PS  RotationMinYS  boolS  S  I  )  PS  RotationMinZS  boolS  S  I  PS
RotationMaxXS  boolS  S  I  )  PS  RotationMaxYS  boolS  S  I  )  PS  RotationMaxZS  boolS  S  I  (  PS  InheritTypeS  enumS  S  I  PS
  ScalingActiveS  boolS  S  I  PS
  ScalingMinS  Vector3DS  VectorS  D  D  D  G  PS
  ScalingMaxS  Vector3DS  VectorS  D  D  D  (  PS  ScalingMinXS  boolS  S  I
```



```
             [PS  ScalingMinYS  boolS  S        [PS  ScalingMinZS  boolS  S        [PS  ScalingMaxXS  boolS  S  I        [PS  ScalingMaxYS  boolS  S        [PS  ScalingMaxZS
boolS   S   I   D     Q   [PS  GeometricTranslationS  Vector3DS  VectorS  D    D    D      N  [PS  GeometricRotationS  Vector3DS  VectorS  D    D    D      M  [PS  GeometricScalingS
Vector3DS  VectorS  D   δ?D   ε?D   δ??     6  [PS
  MinDampRangeXS  doubleS  NumberS  D     ƒ     6  [PS
  MinDampRangeYS  doubleS  NumberS  D     Ç     6  [PS
                                                                                          Ln 1, Col 1      100%   Unix (LF)      ANSI
```

Figure 3.4: A 3D planar object in FBX format, opened with a text editor

Getting Oriented in 3D Space

When learning to navigate 3D space it is helpful to remember that every digital object is merely a trick of the eye. Digital photos, for example, are not actually the flattened physical objects depicted within them, rather, they are tiny squares of colored light appearing so close together that the human eye perceives the pattern as something recognizable.

Every digital object in 3D software is similarly a trick of the eye. Sure, we can take a 3D design, and build a physical object based on that design or even have a computer 3D print it into existence, but the original digital object is still just data that a computer reads and displays on the screen, preferably in a way that the 3D creator intended.

This "trick of the eye" concept is helpful to the process of translating what you imagine in your mind, into something visible on a screen. Earlier we asserted that even circles can be formed from a series of points connected by lines. That may seem like a strange claim, but remember, up until very recently every digital shape has been displayed as tiny dots of light that only connected end to end to create the illusion of contiguous lines, both straight and curved.

In that format, digital circles or roundness in general can be thought of as a series of straight lines, each drawn at an angle from the previous one. Shapes constructed with larger numbers of smaller segments appear more smoothly rounded than those constructed with smaller numbers of larger segments, as illustrated in *Figure 3.5*:

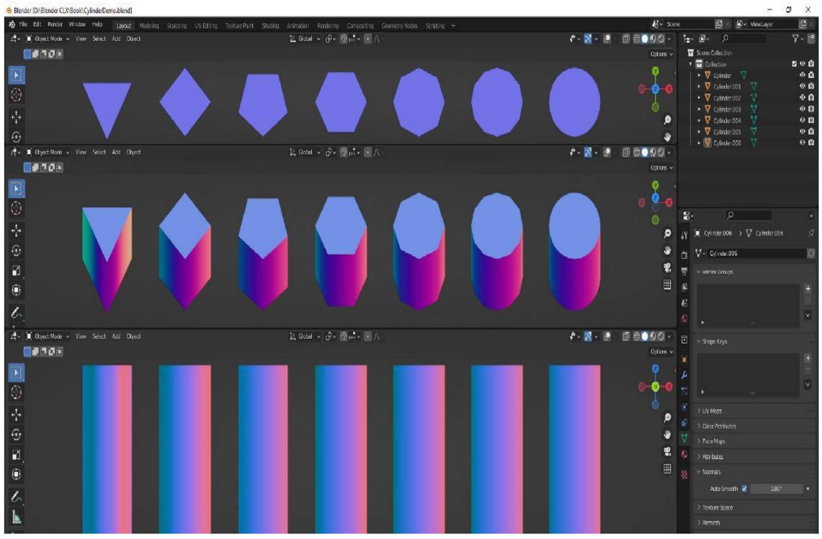


Figure 3.5: Seven columns appearing rounded, each made of straight lines, seen from three viewpoints: top, 45 degrees, and front

All of this is to say that fooling the eye is a vital component of displaying 3D objects, so keeping that concept in the forefront of our minds will help us become thoroughly comfortable creating in 3D.

Perspective again

Another concept first introduced in *Chapter 2, Installation and Interface,* and further illustrated is that perspective is important when working in 3D because what you

see is not always what you get. Here, we are referring to both "perspective" by its dictionary definition as well as the technical term, a viewpoint used in 3D software.

The reason we practiced opening multiple **3D Viewports** at once in *Chapter 2, Installation and Interface* was to prepare us for the reality that shifting our attention back and forth, from one dimension to another as we work, is incredibly helpful to avoid accidentally making changes to our project that are hidden, and to develop our power to fool the eye with intention.

It is completely up to you whether to use a "quad" 3D view, a split view, or simply zoom, orbit, and pan while toggling between Perspective and Orthographic views as needed, but whatever method you choose, it is highly advisable to make it a habit to check for hidden mistakes before your undo history is filled.

Remember from *Chapter 2, Installation and Interface* that the Undo History in Blender records 32 steps by default, including mode switches and tool parameter changes. This limit can be increased via **Edit** > **Preferences** > **System** > **Memory and Limits**. No matter which 3D software you use, it is helpful to familiarize yourself with how and how often you can undo mistakes.

Scene statistics

In a similar vein, we can reduce the odds of making costly errors by keeping an eye on information about the whole scene, and not just the 3D objects you are working with. **Scene statistics** are numbers, like how many objects, vertices, edges, and faces are not just visible on screen, but also impact the file size and speed of the 3D software at any given time.

It is helpful to locate this information before beginning work in any 3D software. In

Blender, the display of scene statistics can be enabled by right-clicking the bottom right Status Bar and selecting Scene Statistics, or via the Overlays menu appearing as two overlapping circles in the upper right icon cluster of the **3D Viewport**, which will enable scene statistics to appear in the upper left.

Tip: The reason to keep an eye on scene statistics is that very large numbers of 3D components on screen at once can crash the application or cause it to perform very slowly. Editing 3D models can have the effect of dividing a single shape into millions of objects as quickly as you can move a slider. The more 3D objects on the screen, the harder your computer must work. For this reason, when adjusting settings in any 3D software, it is very important to proceed slowly.

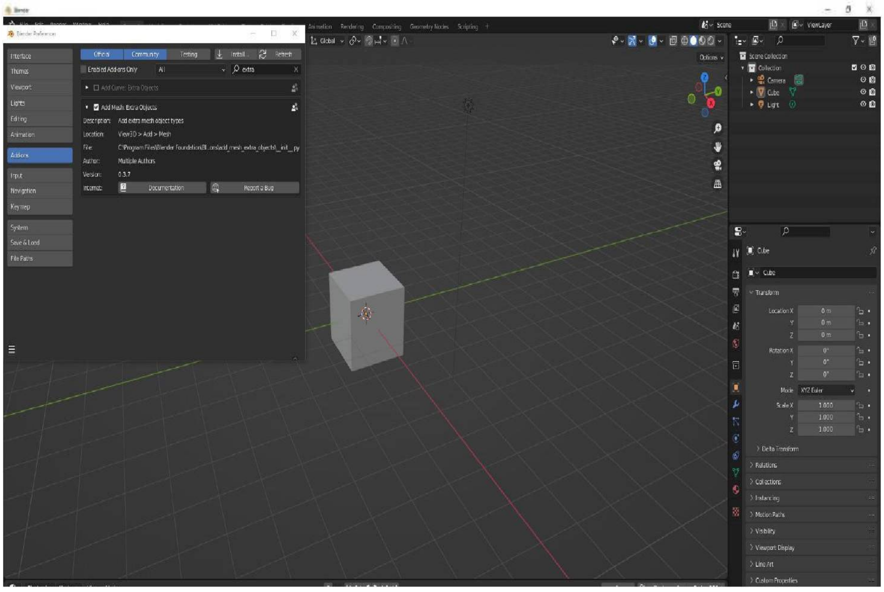
Exercise 3.1

In this exercise, we will create our first new 3D model beginning with a single vertex in Blender. To begin, we will enable one of Blender's extra options which are disabled by default, to keep the menus and workspace free from clutter. The option we will

enable can be found under **Edit** > **Preferences** > **Add-ons**. In the right-hand Blender Preferences window, a search box appears near the top, on the right, just below the Install and Refresh buttons.

1. In this search box, type "extra" and press the Enter key. Two add-ons will appear below: Add Curve: Extra Objects and Add Mesh: Extra Objects. Clicking in the empty check box to the left of the Add Mesh: Extra Objects options will enable this menu option in the **3D Viewport.**

2. Click the white triangle to the left of the text to see a description of this Extra feature, and the location where it now appears in Blender's menu, as shown in *Figure 3.6*:



☐ Enabled Add-ons Only   All   extra

▸ ☐ Add Curve: Extra Objects

▾ ☑ Add Mesh: Extra Objects

Description:	Add extra mesh object types
Location:	View3D > Add > Mesh
File:	C:\Program Files\Blender Foundation\Bl...ons\add_mesh_extra_objects\__init__.py
Author:	Multiple Authors
Version:	0.3.7

Figure 3.6: Add Mesh: Extra Objects add-on enabled via Edit > Preferences > Add-ons

3. Close the Blender Preferences window using the X button in the upper right corner.

   By default, changes to Blender's preferences are saved automatically when we save our BLEND files and will persist even when we close and reopen the application.

4. If we do not plan to save this file, we can choose **Save Preferences** via the "hamburger" (three lines) menu icon in the bottom left corner of the **Preferences** window.

   The Auto Save Preferences option can be disabled or re-enabled from this menu as well.

5. If you have any objects in the scene, either from opening a new instance of Blender or from previous exercises, hover your mouse cursor in the **3D Viewport**, left-click anywhere away from 3D objects (this will clear your current selection), press the A key to select all, then press the **Delete key**, or press the **X** key and choose **Delete**.

   Tip: The preceding sequence of left-clicking away from 3D objects to clear the selection, and then pressing A to select all objects may not technically constitute a Shortcut, but it is advisable to get into the habit of first clearing your selection this way, to avoid accidentally performing an action like deletion on something you forgot that you selected moments ago, that may now be hidden or off-screen.

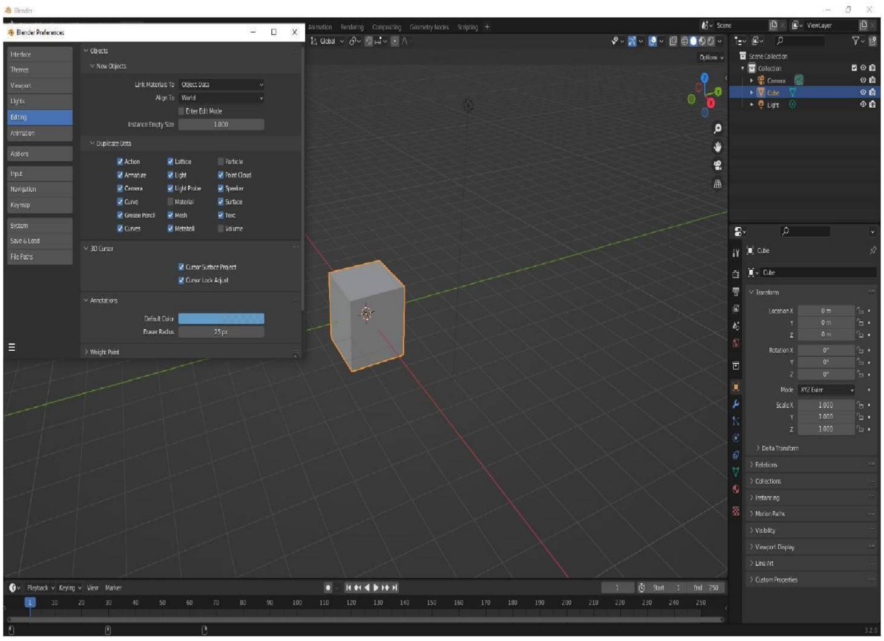
6. With the scene clear, click Add at the top left of the **3D Viewport** or press Shift + A with your mouse cursor inside the **3D Viewport**, then select **Mesh** > **Single Vert** > **Add Single Vert.**

Note: This new object is named Vert, as we can see in the Outliner. This is just a descriptive default name and can be changed at any time by double-clicking on the text in the Outliner and typing a new name, then press Enter. If we make a duplicate of an object with the same name, the new object will have the suffix .001 appended. The .001 suffix will be incremented to .002 for the next duplicate, and so on.

Modes, and editing 3D objects

Congratulations! It may not seem like much, but we have begun creating a new 3D object from a single vertex. To edit this or any 3D object in Blender, we must first verify that we are in **Edit Mode.**

Note: The Single Vert object is unique, in that when we add it to a scene, Edit Mode is automatically enabled for us (likely because a single vertex is almost invisible in Object Mode.) If you prefer to enter Edit Mode automatically whenever you add any new object, the setting can be enabled via Edit > Preferences > Editing > Objects > New Objects as shown in *Figure 3.7.*



Blender Preferences

Interface, Themes, Viewport, Lights, Editing, Animation, Add-ons, Input, Navigation, Keymap, System, Save & Load, File Paths

Objects

New Objects

Link Materials To: Object Data

Align To: World

☐ Enter Edit Mode

Instance Empty Size: 1.000

Duplicate Data

✓ Action	✓ Lattice	☐ Particle
✓ Armature	✓ Light	✓ Point Cloud
✓ Camera	✓ Light Probe	✓ Speaker
✓ Curve	☐ Material	✓ Surface
✓ Grease Pencil	✓ Mesh	✓ Text
✓ Curves	✓ Metaball	☐ Volume

3D Cursor

✓ Cursor Surface Project

✓ Cursor Lock Adjust

Annotations

Default Color

Eraser Radius: 25 px

Weight Paint

Figure 3.7: Enabling Enter Edit Mode upon adding New Objects

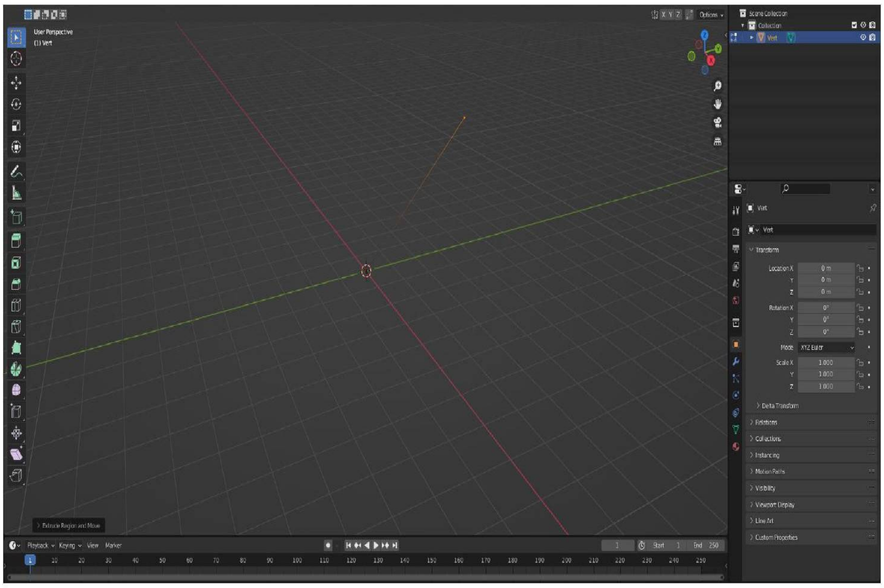
Operation Modes are an important feature of Blender and many 3D applications. Changing modes enable us to switch between related sets of tasks and tools – such as painting or sculpting – and allows Hotkeys to be reused based on the current mode.

As noted, in Blender, adding most new 3D objects will not enable **Edit Mode** by default. To toggle between **Edit Mode** and Object Mode in Blender, select the desired default. To toggle between **Edit Mode** and Object Mode in Blender, select the desired mode from the drop-down menu in the upper left of the **3D Viewport,** or press the **Tab** key with your mouse cursor inside the **3D Viewport**. When in **Edit Mode**, you will see a different set of tools and options appear from those available in **Object Mode**.

- **Edit Mode** is for directly modifying an object's sub-components: vertices, edges, and faces.
- **Object Mode** is for modifying whole objects within a scene.

7. With **Edit Mode** enabled, let's make the first change to our new 3D object. If the single vertex is still highlighted in yellow, press the E key for Extrude, and move your mouse around the view. (If the vertex wasn't highlighted, simply left-click to select it first.)

You should see a yellow line extending between your cursor and the first vertex as shown in *Figure 3.8*. Left-click and release anywhere, and you have created an Edge, connecting two Vertices. This is the same as a line, drawn between two points, as discussed earlier in the chapter.



Vert

Transform
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
Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Line Art
Custom Properties

Figure 3.8: *A new Vertex, extruded from the original Single Vertex, connected by a single Edge*

Next, clear your selection again by left-clicking anywhere in the 3D Viewport, away from any 3D object, then press **A** to select all. Press the **E** key again, move your cursor to extrude the Edge, and left-click anywhere to confirm. This creates a new Face as shown in *Figure 3.9*:

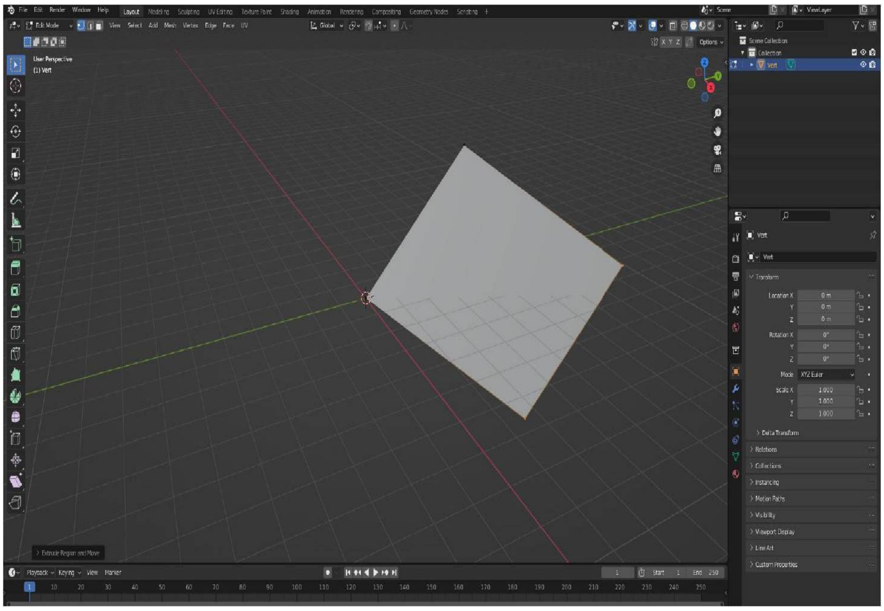


Figure 3.9: *A new Face, extruded from an Edge*

Clear your selection by clicking away from the 3D object again, select all by pressing **A**, press **E** to extrude again, and left-click to confirm once more. You may need to scroll the mouse wheel backward to zoom out and click **+** drag the middle mouse button to orbit your view, to get a good look at the resulting cube as shown in *Figure 3.10*:

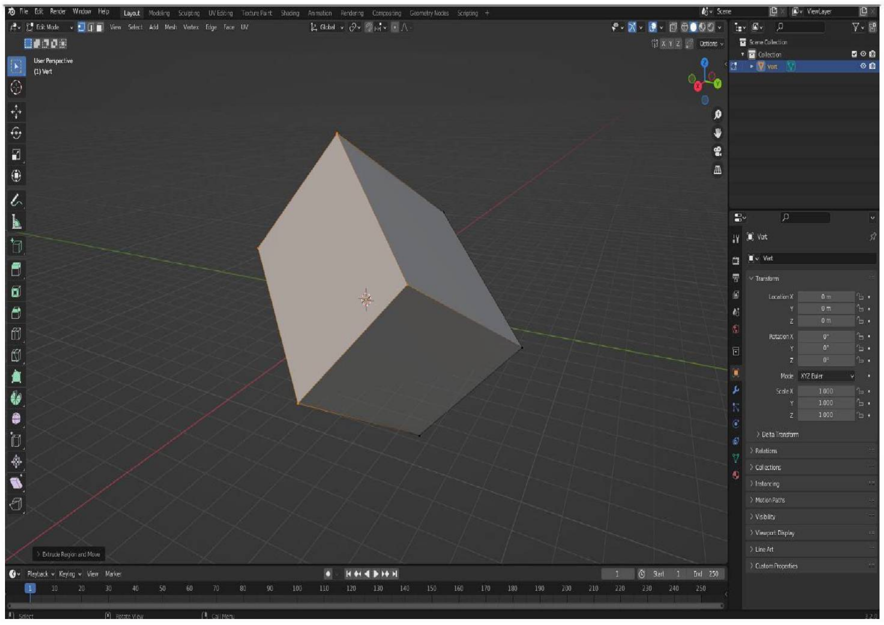


Figure 3.10: *A new 3D cube, extruded from a Face*

Primitive Objects

To create our new 3D object, we began with a basic component upon which to build it. The most basic component of 3D models is the humble vertex. From a single vertex, we can begin to create and edit 3D objects. From one vertex we can extrude an edge. In 3D space, this edge can be extruded at any distance and in any direction. Similarly, from an edge, we can extrude a face as far as we want in whichever direction we want. Multiple faces together form the surface of a 3D object.

Often, beginning with a slightly more structured shape will help to save us many steps. It is a waste of time to "reinvent the wheel," as the saying goes. For this reason, the ability to start from basic premade objects – called primitives – is a common feature of all 3D software. As shown in *Figure 3.11*, default primitive objects typically include cubes, cylinders, cones, spheres made from four-sided planes, spheres made

from triangles, torus shapes, and often, one special object with a complex enough shape to make for a universal example such as Blender's Suzanne.

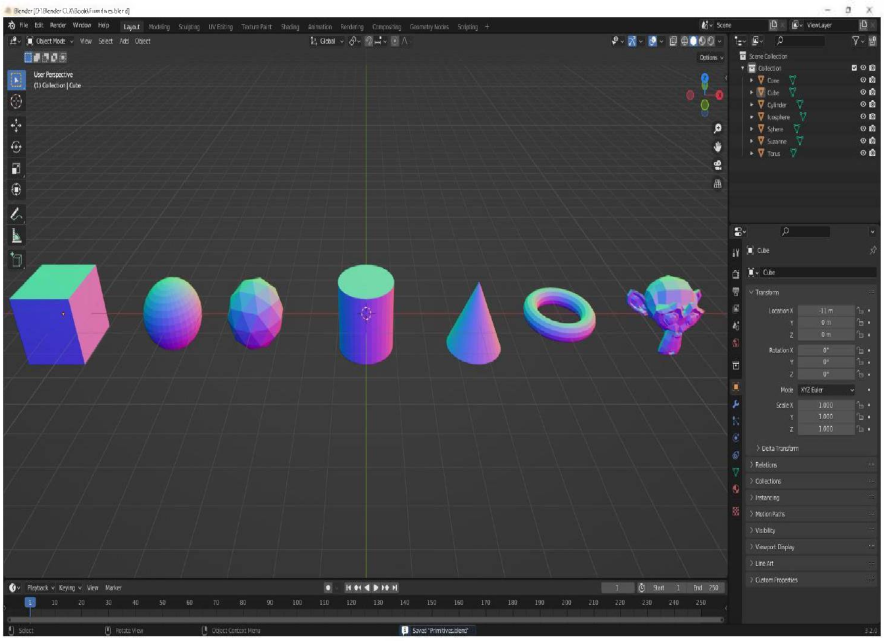


Figure 3.11: Primitive 3D Objects from left to right: Cube, UV Sphere, Icosphere, Cylinder, Cone, Torus, Suzanne

Primitive objects are included in 3D software because many everyday objects share some aspect or another of one or more primitive shapes. The torus primitive, for example, is a literal wheel that saves us the trouble of reinventing it. Reflecting on the example simple character made from primitive objects in *Chapter 1, Features of Blender 3D*, we can see that a cylinder can serve as the beginning of an arm or leg, a sphere the foundation for a head, a cone for a hat, and so forth. In *Chapter 4, Polygonal Modeling* we will explore multiple ways, to begin with primitive objects and develop them into more complex and interesting designs.

Editing normals

Depending on which direction you moved your mouse after pressing the **E** key in the earlier exercise, you may have created an inverted cube without realizing it. This means that the cube's surface may be pointing inward rather than outward.

Think of 3D objects as being like garments, in that there is a specific side that is intended to face outward. Like garments, 3D objects can be turned inside out. As you can see, this is not always a problem or even visible, but in many situations, an inverted 3D object will vanish from view completely. This brings us to another important feature in the anatomy of 3D objects: normals.

In 3D, the term "normal" refers to the direction a given component of a 3D object is pointing, whether that is a vertex, an edge, or a face. Even though a rectangular face looks like a piece of paper with two distinct sides (technically six: one front, one back, and four that are imperceptibly thin), in 3D software, a single face has only one side. The side of a face that is always visible is typically the side that it normally is facing, otherwise known as its normal direction.

Because of the way that 3D software displays the surface of 3D objects, it may not always be obvious whether the normals of a 3D object are facing outward or inward. Under the hood, though, all 3D software keeps track. Note that a single 3D object can have some faces pointing inward, and some pointing outward at the same time, for practical reasons we will clarify in future chapters. A simple extrude operation like we performed previously, however, will only result in all faces turned inward, or all faces turned outward. Much like a garment, the direction you pulled the free edge will dictate whether the whole object winds up outside out, or inside out.

The fact that – depending on the display mode – you may not be able to tell which side is facing out and which is facing in is more evidence of the concept introduced earlier, that what we see on screen is not necessarily as it appears in 3D.

It is possible and sometimes advantageous to render 3D objects in such a way that both sides of any given face are visible at once, but this too is merely a trick of the eye. Either the renderer has been set to calculate and display the nonexistent side as is the case in the preceding exercise, or the 3D creator has duplicated faces and flipped the duplicates to face in the opposite direction, giving the illusion of a two-sided object.

In most 3D applications, normals facing different directions are highlighted by a difference in shading. In Blender, there are multiple methods for verifying which direction the faces of a 3D object are pointing, regardless of the viewport rendering mode.

In the upper right-hand corner of the **3D Viewport,** there is a cluster of icons that control the view displays. As indicated earlier, the third icon from the left within this right-hand cluster, appearing as two overlapping circles, controls Overlays. To display nonexistent, inward pointing faces (also called back-faces) as red, and

outward pointing faces as blue, mouse-click the icon to expand the menu and near the middle, enable **Face Orientation** as shown in *Figure 3.12*:

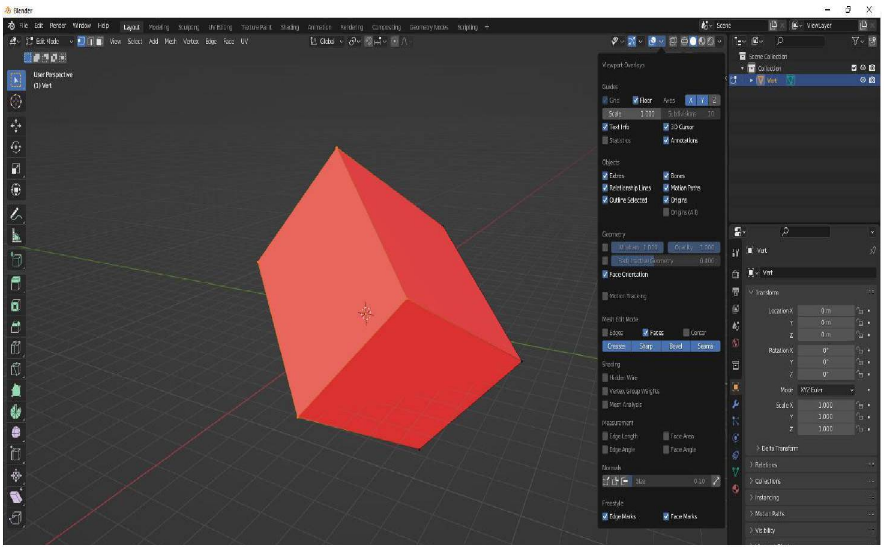


Figure 3.12: A cube with inverted (inward facing) face normals

In Blender, another way to visualize the direction an object's normals are facing is also accessed through the **Overlays menu**, nearer the bottom left. Click one or multiple of the icons appearing as a square with lines protruding out from it, to enable the overlay. The left-most of these 3 icons enables the overlay display for Vertex Normals, the next is for Split Normals (which we will explain in future chapters), and the third displays [face] Normals.

Note: In the default Blender theme, a blue highlight indicates that a feature is enabled.

When any of these normal displays are enabled, a short blue line will be displayed protruding in the direction of the sub-objects' normals. If the indicator is too small to see, you can increase its length in the Size field to the right of the same area of the **Overlays menu**. However, if your 3D object is currently inverted, you may not see these indicators at all because they are facing inside the cube. To flip all the normals of a 3D object at once, in **Edit Mode**, press **Alt + N** and choose **Flip**, as shown in *Figure 3.13*:

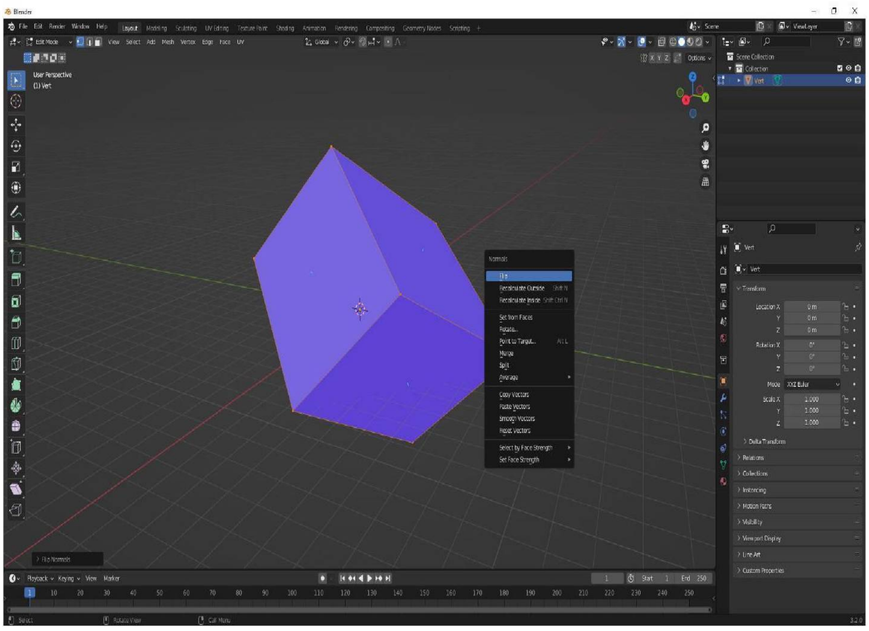


Figure 3.13: A cube with correct, outward facing normals, and the face normal visualization showing after a Flip operation

Enabling backface culling

Yet another way to check that normals are facing the intended direction in Blender is to disable the rendering of the nonexistent sides of faces, also called "back faces." This brings up another key concept to working in 3D called **culling**.

In 3D, culling is the computer temporarily hiding things from view. Hiding some things from view when they do not need to be seen can be extremely helpful, making it easier to understand what we see on screen, but can also be deceptive like most other tricks of the eye. Culling is a behavior that can be turned on or off as needed in 3D software, so it is helpful to familiarize yourself with what is being culled or hidden by default, and when, in any 3D application.

In Blender, to hide "back faces" that are nonexistent and therefore not visible (or "rendered") in many circumstances, open the drop-down in the farthest upper right-

hand corner of the **3D Viewport** and enable backface culling, about halfway down, as shown in *Figure 3.14*:

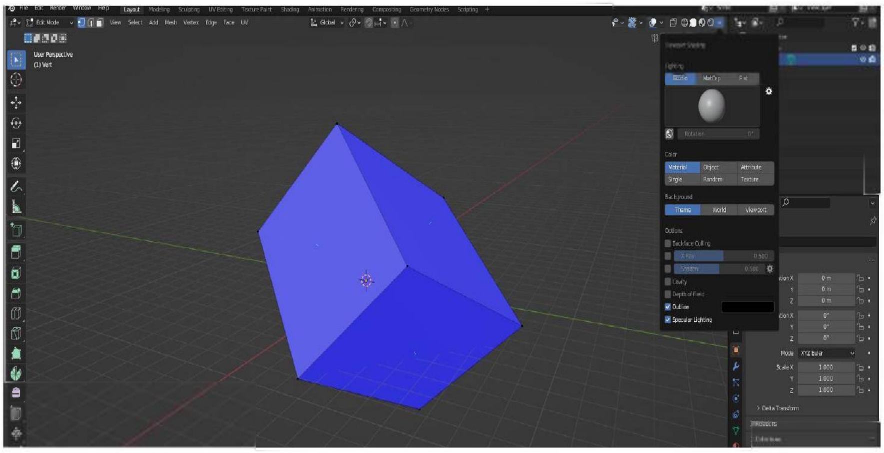


Figure 3.14: Toggle for Backface Culling in the 3D Viewport

To see Backface Culling in full effect, first verify that you are in **Edit mode** (indicated in the upper left corner of the **3D Viewport**). Then, switch to Face Select mode, either by pressing the 3 key in the upper row of the keyboard, or by selecting the Face Select icon that looks like a grey cube with one white side in the upper left of the **3D Viewport,** to the right of the **Mode menu**. Finally, left-click the middle of any of the front faces, and delete it. Now you should be able to see that the shape's inner faces, or "back faces" are rendered transparent, as shown in *Figure 3.15*:

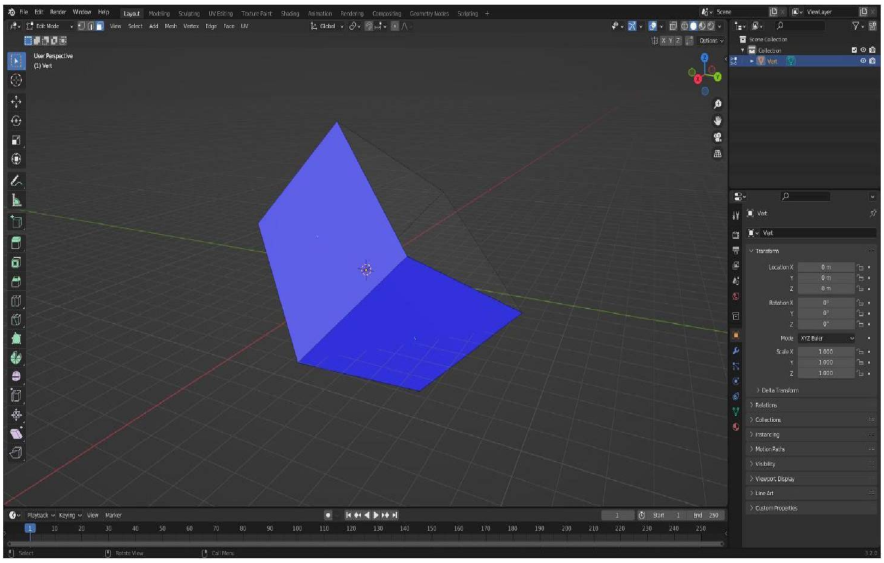


Figure 3.15: Toggle for Backface Culling in the 3D Viewport

In general, it won't be necessary to have all these normal displays enabled at once, so feel free to disable any of them until you discover a need to see them. Like most Overlays, normal displays will remain as they were set when you last saved the file.

Transforms

While we first discussed zooming, orbiting, and panning the 3D view in *Chapter 2, Installation and Interface*, now we will dive into modifying 3D objects themselves. Modifying whole 3D objects or their sub-component vertices, edges, and faces, involves a universal concept called Transforms.

In 3D, the term Transform is both a verb and a noun, simultaneously referring to the **action** of Moving, Rotating, and Scaling a 3D object, and to the **amount** that a 3D object has been Moved, Rotated, or Scaled from a previous state. The reason we make this distinction is so we can understand common instructions, like "zero out all transforms," or "apply all transforms," which we will explore in just a moment.

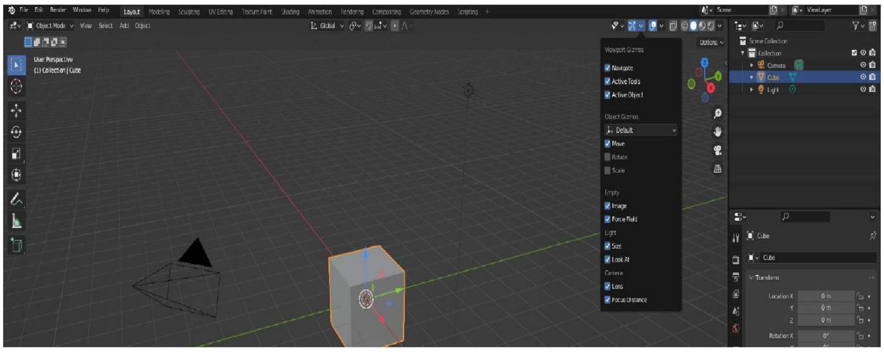
Transforming with gizmos

In 3D software there are multiple ways to change object transforms. One of the more straightforward approaches to moving, rotating, and scaling in 3D is to use a visual overlay called a gizmo as a sort of 3D handle. A gizmo is not part of a 3D object but acts as a visual stand-in for operations you can perform on 3D objects, with symbols indicating the available directions of movement, rotation, or scale.

Tip: Very commonly, gizmo handles are color coded red, green, and blue to match the display of the three axes of three-dimensional space: x, y, and z. If you find it easy to remember the abbreviation RGB, it might help to associate the colors Red, Green, and Blue with the dimensions x, y, and z listed in alphabetical order.

In Blender, the display of gizmos can be controlled via the 3D Viewport's upper right-hand cluster of icons, second icon from the left that looks like a quarter circle with an arrow through it as shown in *Figure 3.16*. The appearance of gizmos can also

be modified in Blender via the drop-down just above the Move, Rotate, and Scale gizmo options. Please refer to the following figure:


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

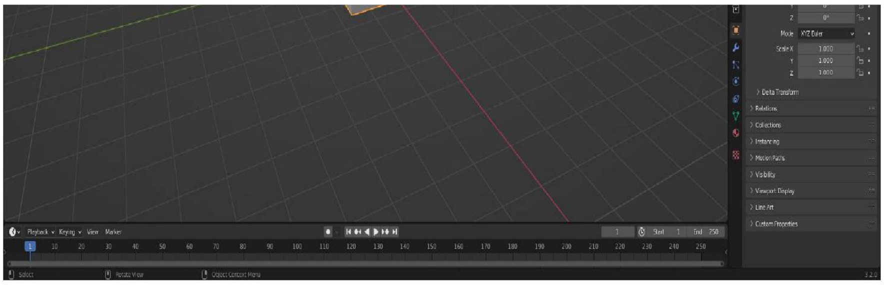


Figure 3.16: Enabling the Move Gizmo

The typical way to use a gizmo, is to click and drag one of its handles, which are commonly displayed as arrows for Move, circles for Rotate, and small cubes for Scale.

Clicking and dragging the ends of a **Move** gizmo will move the corresponding object in that direction only – which is called "constrained" movement – while clicking and dragging the center will allow movement in all directions.

Rotate gizmos behave slightly differently, in that the red, green, and blue circles constrain the rotation to the corresponding x, y, and z axes, while the white circle tends to constrain the rotation to the view plane. This means that if you zoom, orbit, or pan your view, the transformed object will rotate in a different direction than it did prior to the view change.

Like the Move gizmo, clicking and dragging the ends of a **Scale** gizmo will scale the object in that dimension only, while clicking and dragging the center of the gizmo will scale the object in all the dimensions at once.

Gizmos are especially convenient for new 3D creators because they enable click + drag operation and leave no doubt about what to expect. However, gizmos can clutter the view and often require extra clicks to use, so seasoned 3D creators tend to prefer using Hotkeys instead.

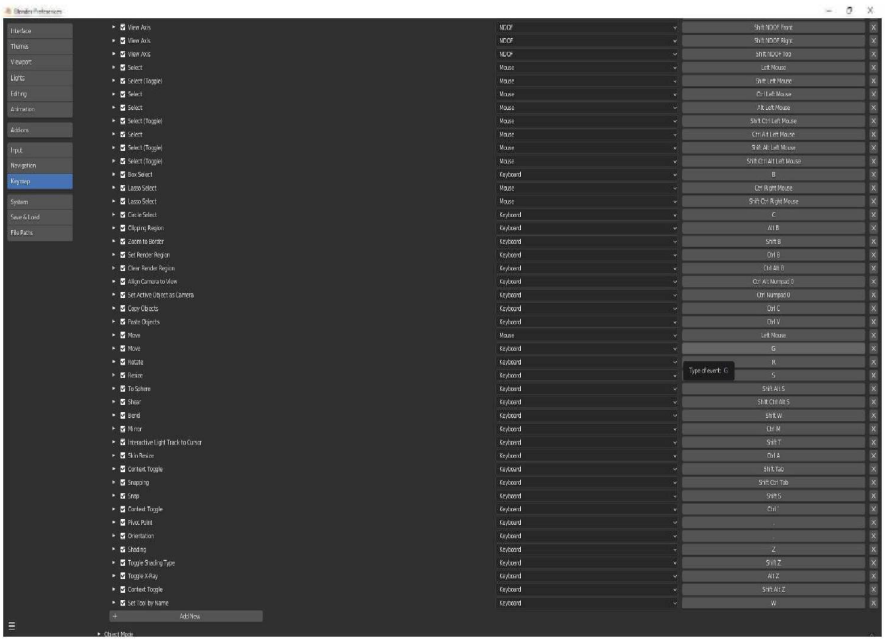
Transforming with hotkeys

Hotkeys for transform Shortcuts vary between different 3D applications, but it is universal to include keys for Move, Rotate, and Scale. In Blender, the Hotkeys for Move, Rotate, and Scale transforms are **G** (for "grab"), **R** (for "rotate"), and **S** (for "scale"). To **confirm** a transform that was initiated with a Hotkey, left-click with the mouse. To **cancel** a transform, right-click instead.

Tip: It can be helpful to get in the habit of testing a transform by first pressing the Hotkey, moving the mouse cursor around to see the result, and then right-clicking to cancel that action if the desired result is not achieved.

In Blender, the axes of movement, rotation and scale are all linked by default, meaning that if you press **S** to scale, the object will be scaled equally in all three dimensions. To get around this, you can press a second key to modify the first key's behavior. For example, to move only on the x axis, press **G** and then **X** before moving the mouse. To rotate only in the y axis, press **R** and then **Y**. Press S and then Z to scale only in the z axis, and so-forth.

While it is inadvisable to modify the Hotkey combinations for any software before you are familiar with the bindings for keys you may prefer to use, it is helpful to know that we can always view and change Hotkey bindings in Blender via **Edit** > **Preferences** > **Keymap** > **3D View** > **3D View** (**Global**) as shown in *Figure 3.17*:



View Axis	NDOF	Shift NDOF Front
View Axis	NDOF	Shift NDOF Right
View Axis	NDOF	Shift NDOF Top
Select	Mouse	Left Mouse
Select (Toggle)	Mouse	Shift Left Mouse
Select	Mouse	Ctrl Left Mouse
Select	Mouse	Alt Left Mouse
Select (Toggle)	Mouse	Shift Ctrl Left Mouse
Select	Mouse	Ctrl Alt Left Mouse
Select (Toggle)	Mouse	Shift Alt Left Mouse
Select (Toggle)	Mouse	Shift Ctrl Alt Left Mouse
Box Select	Keyboard	B
Lasso Select	Mouse	Ctrl Right Mouse
Lasso Select	Mouse	Shift Ctrl Right Mouse
Circle Select	Keyboard	C
Clipping Region	Keyboard	Alt B
Zoom to Border	Keyboard	Shift B
Set Render Region	Keyboard	Ctrl B
Clear Render Region	Keyboard	Ctrl Alt B
Align Camera to View	Keyboard	Ctrl Alt Numpad 0
Set Active Object as Camera	Keyboard	Ctrl Numpad 0
Copy Objects	Keyboard	Ctrl C
Paste Objects	Keyboard	Ctrl V
Move	Mouse	Left Mouse
Move	Keyboard	G
Rotate	Keyboard	R
Resize	Keyboard	S
To Sphere	Keyboard	Shift Alt S
Shear	Keyboard	Shift Ctrl Alt S
Bend	Keyboard	Shift W
Mirror	Keyboard	Ctrl M
Interactive Light Track to Cursor	Keyboard	Shift T
Skin Resize	Keyboard	Ctrl A
Context Toggle	Keyboard	Shift Tab
Snapping	Keyboard	Shift Ctrl Tab
Snap	Keyboard	Shift S
Context Toggle	Keyboard	Ctrl '
Pivot Point	Keyboard	.
Orientation	Keyboard	,
Shading	Keyboard	Z
Toggle Shading Type	Keyboard	Shift Z
Toggle X-Ray	Keyboard	Alt Z
Context Toggle	Keyboard	Shift Alt Z
Set Tool by Name	Keyboard	W

Figure 3.17: *Maximized Preferences showing the 3D View keymapping options with the G for Move key binding highlighted*

Transforming with parameters

Another way to change transforms in 3D is to directly edit numerical characteristics – like measurements or increments – otherwise known as **parameters**. In 3D, parameters for transforms occur in sets of x, y, and z, corresponding to the three axes of three-dimensional space. As shown in *Figure 3.18*, transforms for the selected object or sub-object component (vertex, edge, or face) can be typed directly into parameter fields appearing in the top tab of the **Properties panel** (in the bottom right of the default **Layout view**), and via the **Item tab** of the **Side Bar menu** accessed via the

N key.

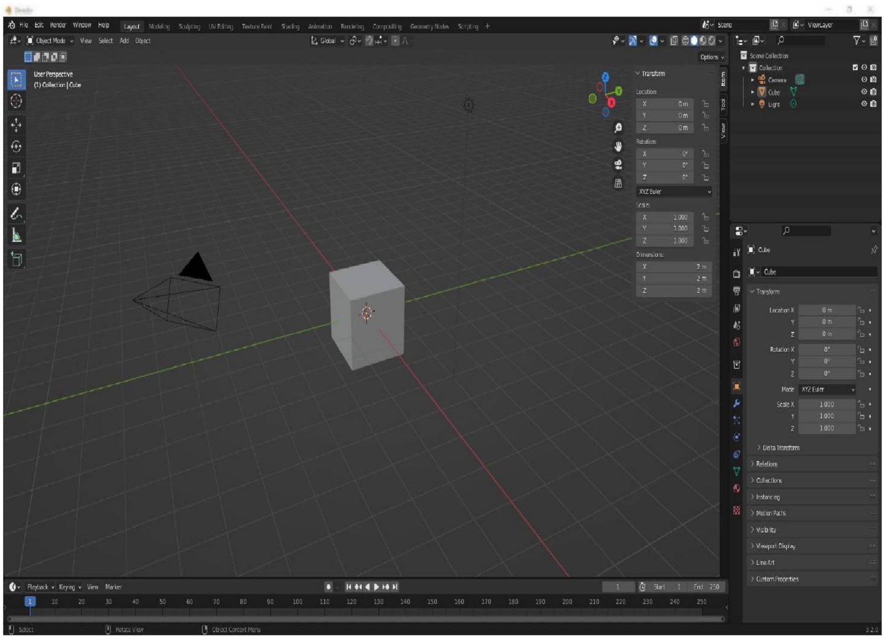


Figure 3.18: Two areas where parameters can be edited directly: in the Side Bar > Item menu and the Properties panel

In both the **Properties panel** and the **Side Bar menu**, the Location, Rotation, and Scale of each dimension (x, y, and z) of a selected object can be edited by typing a value directly into the corresponding field. In the **Side Bar menu**, we can also edit the selected object's overall Dimensions.

In this context, Scale refers to the object's original state, such that a value of 1 in all three of the x, y, and z axes mean the object is currently its original size. Typing positive numbers into any of the object's Scale parameters will scale the selected object up, that many times its original scale in that dimension. Typing a negative number (preceded by the minus sign) will scale the object by that amount in the opposite direction.

Tip: Typing -1 into any Scale axis is a handy way to mirror the object across that axis.

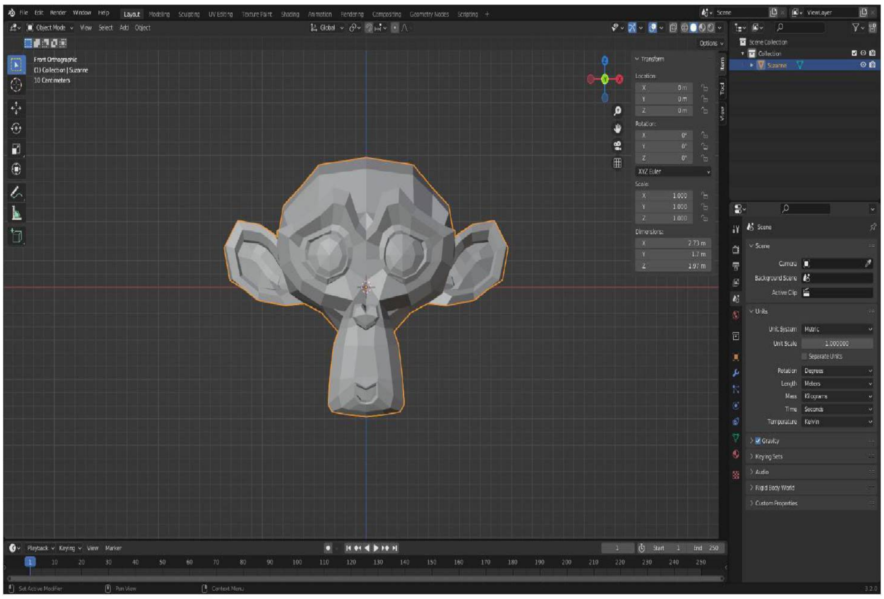
Dimensions, on the other hand, refer to an object's measurements relative to the scene's unit of measure. A default cube at a Scale of 1 in Blender's default unit of measure is 2 meters wide, by 2 meters deep, by 2 meters tall; while Suzanne measures 2.73 meters wide, by 1.7 meters deep, by 1.97 meters tall at a Scale of 1.

Tip: In Blender, Dimension parameters can be entered as text, to indicate the

intended unit of measure. Typing a value followed by a unit of measure such as cm, centimeters, ft, feet, in, or inches into a parameter field, for example, will be translated correctly into the default unit of measure automatically.

Real-world measurements like these are very important when creating 3D objects for 3D printing, or for objects to appear life-sized in games and interactive applications. Dimensions can also affect the behavior of operations that are based on relative distance or size, such as setting one object's distance from another. The utility and interaction of Dimensions will become more apparent, the deeper into 3D creation we go.

Working between 3D applications – such as when exporting 3D objects from modeling software for import into printing software or game engines – can get confusing because mismatches between two applications' base unit scales are common. In Blender, the default unit of measure is meters, but Blender allows us to change the base unit scale to match our needs, via the **Properties panel**, in the **Scene Properties tab** with the icon appearing as a collection of shapes, under **Units** as shown in *Figure 3.19*:


Units

Unit System: Metric

Unit Scale: 1.000000

Separate Units

Rotation: Degrees

Length: Meters

Mass: Kilograms

Time: Seconds

Temperature: Kelvin

Dimensions: X 2.73 m, Y 1.7 m, Z 1.97 m

***Figure 3.19:** Unit scale options located in Scene Properties tab of the Properties panel*

Note: When we change the base unit of measurement in a 3D application, we are not changing the dimensions of the 3D objects themselves.

Another area where transforms can be changed via parameters in Blender is through contextual menus that appear in the bottom left of the **3D Viewport** for every move, rotate, and scale operation, as well as other operations we will address in greater detail through later chapters. Whenever we use Hotkeys to move an object, a new menu will appear in the bottom left of the **3D Viewport** with contextual options. In Blender, we can continue to adjust the settings that appear in this menu until we have either deselected the object or performed another operation with it.

Combining hotkeys with parameters

Transforming with parameters can also be combined with Hotkeys. For example, if we press the S key, to scale an object, we can also type in a number and then either left-click with the mouse or press the **Enter key**, to indicate the amount we want to scale it. Typing S, 3, Enter will scale the object up three times its starting size. Rotation works the same way, but in degrees. So, if we type R, 180, and Enter, the object will be rotated by 180 degrees.

In Blender we can also type in decimal values using the keyboard period key, to transform by fractions. Typing S, .5, Enter, for example, will scale the object down by half, S, .25, Enter by a quarter, and so forth. Negative values can also be typed in to invert the direction of the transform operation.

Remember that the x, y, and z axes of movement, rotation and scale are linked when transforming with Hotkeys, so typing S, 3, Enter, will scale the entire selected object three times its starting size in all three dimensions at once. However, just as we can press a second key to modify the first key's behavior, we can string together a series of Hotkeys to isolate transforms in one dimension. For example, typing S, X, 3, Enter will scale the selected object to three times its original size in the x axis only.

To exclude a dimension from the transform operation, press the Shift key before typing the dimension to exclude. For example, typing S, Shift, X, 3, and Enter will scale the selected object to three times its original size in the y and z axes only.

Undoing transforms

As we discovered when flipping the normals of our 3D object earlier in this chapter, the Alt key used in combination with other keys can serve as a way of undoing certain operations in Blender. To undo whole object transforms, resetting the object to its original state, we can of course press **Ctrl + Z**. If a transform was performed long ago, though, it may no longer appear in our Undo History. The good news is, so long as the whole object's transforms have not been applied, object transforms

can be reverted to their original states by using the **Alt key** in combination with a transform Hotkey. More on applying transforms in a bit.

In Object Mode, undoing movement to reset an object to its original location, for example, can be accomplished by pressing **Alt + G.** Rotation can be reset to zero with **Alt + R,** and unapplied scale changes can be undone by pressing **Alt + S**. Undoing transforms this way is one method of what we call zeroing out transforms. Another way to "zero out" transforms is by literally typing the number 0 into parameter fields for location and rotation, or 1 for scale.

Note: Up until now, we had been learning about transforms that can be applied both to whole objects in Object Mode, as well as to sub-components (vertices, edges, and faces) in Edit Mode. It is very important to note that beyond Ctrl + Z and the Undo History, transforms can only be reverted to an original state at the object level, because sub-object components share a single origin point with the whole object. Therefore, instructions for reverting transforms to their original states apply to the whole object in Object Mode only.

Origins and applying transforms

Earlier we specified that transforms can only be reverted to their original states so long as they have not yet been "applied." **Applying** a transform essentially gives the whole object a brand-new original location, rotation, or scale, which is also known as its **origin**. The origin point of an object is very important to the way operations on its transforms behave and can almost be considered a basic component of all 3D objects along with their vertices, edges, and faces.

An object's origin is sometimes, but not always at its center, because as we just discovered, applying a transform can change the object's origin. But the origin is the center or "pivot point" of any transform operations performed on the object. For example, if we move an object 1 meter along an axis, and apply the transforms there, the object's origin point will now be 1 meter away from the world origin in that dimension. Then, when we go to transform the object again, the transform operation will be offset so that the object will move, rotate, or scale from that offset, instead of from the object's center.

As we can see, applying transforms has a cascading effect. When creating in 3D we may find this effect to be desirable. For example, if our object is a 3D building, we may want to change its origin to the base, so that when we increase its scale, it will grow upward and to the sides, instead of downward into the ground.

Thankfully, applying transforms requires a few steps, so it is easy to do with intention and difficult to do by accident. In Blender, we can apply location, rotation, and scale transforms individually or apply all transforms at once. In **Object Mode**, pressing **Ctrl + A** will open the Apply menu, where individual transforms or all transforms

can be applied. The Apply menu can also be accessed from the upper left hand of the **3D Viewport** under **Object** > **Apply**.

After applying transforms to an object that was previously moved, rotated, or scaled, the Location, Rotation, and Scale transforms will be what we call "zeroed out," and the new location, rotation, and/or scale will have become the object's permanent origin. Now, **Alt + G**, **Alt + R**, and **Alt + S** will revert the object to this state, rather than the previous one.

In Blender, an object's origin is displayed as a tiny orange dot (or a yellow dot, if the selected object is active) as shown in *Figure 3.20*. In an unedited primitive object, the origin starts out located in the center, or more precisely, in the center of where a cube's center would be if the cube bounded the farthest vertices of the 3D object from the world's center, also called the World Origin. Please refer to the following figure:

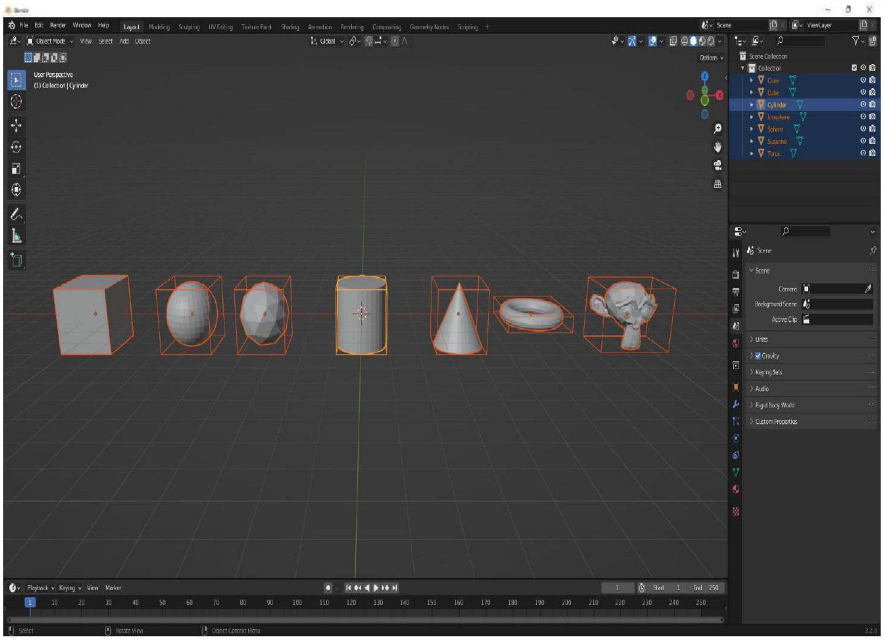

File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
User Perspective
(1) Collection | Cylinder
Scene Collection
Collection
Cone
Cube
Cylinder
Icosphere
Sphere
Suzanne
Torus
Scene
Camera
Background Scene
Active Clip
Units
Gravity
Keying Sets
Audio
Rigid Body World
Custom Properties
Playback Keying View Marker
Start 1 End 250

Figure 3.20: *Primitive objects selected with orange origin points shown*

In all 3D applications, the world **origin** lies at (0,0,0,): the intersection of x, y, and z axes on a three-dimensional grid. The world origin is important because the view, and every new 3D object begins there. Otherwise, 3D objects could start an infinite distance apart in 3D space and 3D scenes would be impossible to navigate. The world origin also serves as an important reference point for all other points in 3D space.

Blender's 3D cursor

All transforms are applied to an object relative to this world origin unless an alternate point (also called a "pivot point") is specified for the operation. In Blender, a special gizmo or overlay appearing as a red and white dotted circle with a grey plus sign in the middle, called the **3D Cursor** also appears at the world origin by default. Like other gizmos, Blender's 3D Cursor can be moved and used for a variety of purposes, but unlike transform gizmos, the 3D Cursor operates independently from any 3D object.

A common use for the 3D Cursor is as a temporary origin point, or pivot point, that replaces the object's true origin during transform operations where the object's origin point should be retained. In Blender, the Transform Pivot Point options appear in the upper center of the **3D Viewport** drop-down. By default – appearing as a chain link icon – the Transform Pivot Point will be set to the Median Point of the selected object or objects.

To transform an object with the 3D Cursor as a temporary Pivot Point so that the object can be easily and accurately returned to its original location, rotation, and scale in what is called a "non-destructive workflow," we can select the 3D Cursor from the Transform Pivot. Then we can enable movement of the 3D Cursor through the **Tool bar menu**, which can be toggled on or off by pressing the T key.

Press the T key if the **Tool bar menu** is closed, and look for the red and white dotted circle with a grey plus sign in the middle and click it to enable freehand placement of the 3D Cursor. When this 3D Cursor tool is active, you can left-click anywhere in the scene to move the 3D Cursor to the location you clicked. Disable freehand placement of the 3D Cursor by enabling the **Select mode** again, at the top of the upper left corner of the **Tool bar**.

More precise positioning of the 3D Cursor is also possible and often desired. The 3D Cursor can be automatically placed – in both **Object Mode** and **Edit Mode**, at the origin of any object, or at the location of a sub-object component, or even between multiple selected objects or sub-objects – by pressing **Shift + S** and selecting Cursor to Selected from the radial menu that appears, as shown in *Figure 3.21*. Be careful not to confuse Cursor to Selected with Selection to Cursor, which will move the

selected object to the 3D Cursor, another useful operation we will employ in *Chapter*

4, Polygonal Modeling.

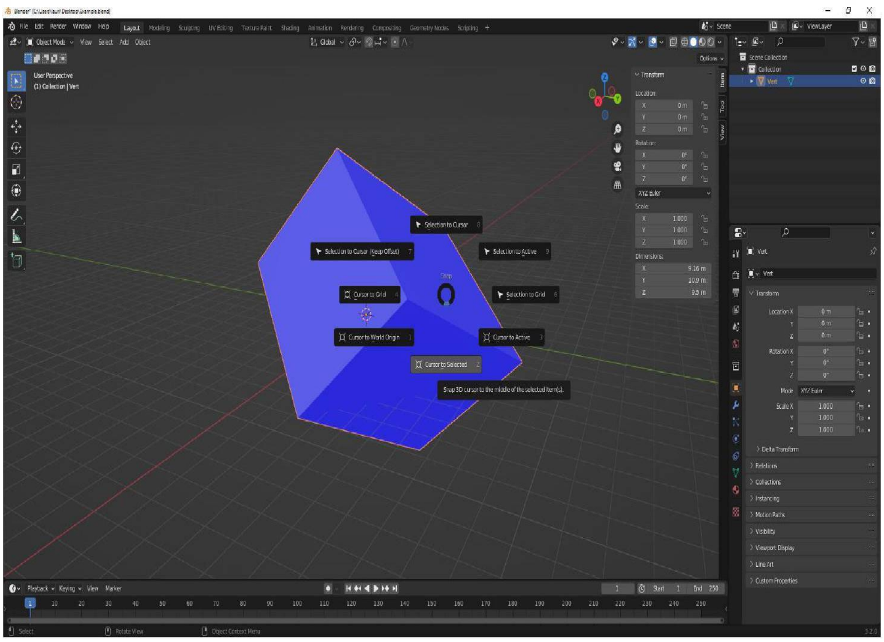


Figure 3.21: Radial 3D Cursor menu accessed via Shift + S, with Cursor to Selected highlighted in lighter grey

Notice the number key Shortcuts displayed to the right of each of the options in this radial menu. Using the Hotkeys for these Shortcuts allows for quick confirmation of the desired operation, without having to move the mouse. Some 3D creators prefer to memorize and use the Hotkeys for these Shortcuts, rather than finding the locations of their frequently used options along the radial menu.

Transforming with tool bar tools

Now that we have toggled the Tool bar menu off and on, we can explore more ways to transform 3D objects available there. The Tool bar menu is another area that is context-dependent. In **Object Mode**, the **Tool bar menu** contains Selection tools, the 3D Cursor placement tool, Move, Rotate, and Scale tools that operate exactly as the gizmos do, a Transform tool which is essential all the gizmos rolled into one, and 3 more tools we will explore in later chapters. In **Edit Mode**, the **Tool bar menu** contains all the preceding, plus even more editing tools that we will explore in *Chapter 4, Polygonal Modeling.*

Selection

To Transform 3D objects and their sub-components, you must first know how to select those components. In Blender, we have already practiced direct selection quite

a bit, by left-clicking. Also revealed in the Tool bar menu by pressing the T key, are a bit, by left clicking. Also revealed in the Tool bar menu by pressing the T key, are more selection options. If we press and hold the top tool in the **Tool bar menu**, we will also find Tweak, Select **Box**, Select **Circle**, and Select **Lasso**.

The default selection mode is Select **Box**, also known as Box Select. Box Select works by left-clicking and dragging a dotted rectangle over the view plane, inside of which everything visible and fully encompassed by the box will be selected. From any other selection mode, Box Select can be activated by pressing the B key within the **3D Viewport**. Using any active select mode with the Ctrl key pressed will **deselect**. Using Shift with any selection mode will add to the previous selection.

The second selection mode available in the Tool bar menu is called Tweak, which is essentially direct selection that also allows freehand movement of the selected object or sub-object component in all axes via click and drag. Clicking anywhere in the **3D Viewport** away from any 3D object deselects all.

Select Circle, or Circle Select, enables a small, dotted circle that follows the mouse cursor, and works like a paintbrush by left-clicking and dragging across objects, which will select anything visible and fully encompassed by the circle's radius while the left mouse button is depressed. The radius of Circle Select can be decreased or increased by rolling the scroll wheel forward or backward. From any other selection mode, Circle Select can be activated by pressing the C key within the **3D Viewport**. Clicking with the middle mouse while Circle Select is active deselects.

Select Lasso enables drawing a freehand dotted line that loops back onto itself upon release of the left mouse button and selects anything visible within and fully encompassed by the bounds of the loop. The Lasso Select tool is most helpful for making precise selections among complex groups of objects. From any other selection mode, Lasso Select can be activated by pressing Ctrl + right mouse click and drag. Shift + Ctrl + right mouse click and drag will deselect.

Mode dependent selection

It is important to note that some selection behaviors are mode-dependent. Earlier we specified that Box, Circle, and Lasso Select methods select anything visible within and fully encompassed by the bounds of the selection type. This is entirely true of **Edit Mode** selections; however, Object Mode selection is a bit more flexible. In **Edit Mode**, if we draw a selection around a vertex but not the entire neighboring edges, the neighboring edges will not be selected even if the selection crosses over them. Likewise, if an object or sub-object component is hidden behind other geometry in **Edit Mode**, it will not be selected, even if it really exists within the area of selection. In **Object Mode**, however, a selection must only cross over an object for it to be selected, even if it is hidden by another object.

In the same way that gizmos can obscure the view of our work, geometry itself – as we refer to objects and sub-object components – can also obscure important views of our work, making it unavailable for selection and otherwise un-editable.

Conclusion

In this chapter, we have learned the common anatomy of all 3D objects and begun exploring the most basic ways to create and edit them. We now have a firm understanding of the transforms that are used to work with 3D geometry and have explored several Shortcuts and Tools that will help us begin to translate designs from our mind, into visible 3D objects to be interacted with and shared.

In *Chapter 4, Polygonal Modeling*, we will begin to practice adjusting the visibility of different objects through Show and Hide Shortcuts and menu options.

Essential shortcuts and hotkeys

In this chapter we used the following Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left mouse click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G

Show/Hide Side Bar	N
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Extrude (in Edit Mode)	E

***Table 3.1:** Essential Shortcuts and Hotkeys*

Points to remember

- Every digital object is a trick of the eye
- Make a habit of checking for hidden mistakes before your undo history is filled
- Normals is the direction a vertex, edge, or face is facing
- Transforms are both the movements, rotations, and scale that have been applied to an object, and the act of moving, rotating, and scaling.

Questions

1. How do we flip the normals of an inside-out object?
2. Why would we want to zero out transforms?
3. Why would we want to apply transforms?
4. What is the default selection mode?

CHAPTER 4
Polygonal Modeling

Introduction

With a firm foundation of general concepts necessary for 3D creation, we will build on our knowledge through hands-on experience, by planning and beginning to create a practical 3D object. Individual 3D objects such as the one we will create here are an essential part of a thriving and growing worldwide market for 3D assets.

Just like print media relies on a mix of stock and contracted or commissioned photography, 3D products like games and film rely heavily on both à la carte purchases of stock 3D models, and agreements with skilled 3D creators to custom model, texture, animate, and render 3D objects to the buyer's specifications.

Structure

This chapter introduces polygonal modeling with Blender: the essential starting point for creating 3D objects. Topics to be covered include:

- Triangles, Quads, and Ngons
- Methods of 3D creation
- Where to start 3D modeling
- Topology and edge flow

- Exercise: poly modeling a pedestal

Objectives

After reading this chapter, we will understand when and why to begin 3D creation with the poly modeling method and have a firm understanding of the tools and techniques to get started in Blender. We will have gained hands-on experience applying our creativity to model and begin customizing a marketable, practical, and decorative 3D object. By learning the differences in the theory behind different methods for 3D creation, we will understand how to plan early to save significant time and effort, saving money in the process.

Triangles, quads, and ngons

Polygonal modeling – known as "poly modeling" for short – refers to modeling by directly editing a 3D object's sub-object components, together called polygons. Polygons in this context are the shapes we learned about in *Chapter 3, General 3D Concepts*: one or more faces, bounded by three or more edges, connecting three or more vertices. A three-sided polygon is a triangle, a four-sided polygon is a rectangle – also known as a quad – and any polygon with more than four sides is called an ngon.

In 3D, all polygons – whether they are three-sided, four-sided, or n-sided – are made of triangles even when we can't see the edges between them on screen. That's because 3D software effectively ignores those invisible edges in computations where it is helpful to do so. However, in terms of the anatomy of 3D objects – coordinates of points where lines intersect filled with planes – all four-sided polygons are in fact two reflected triangles sharing an invisible diagonal edge, which may connect at either of the opposing corners.

Examining a quad from top-down as in *Figure 4.1*, this invisible diagonal edge may either connect the upper left vertex to the lower right vertex, or the lower left to the upper right. These are the two options for the "triangulation" of any given quad (four-sided face) of a 3D object.

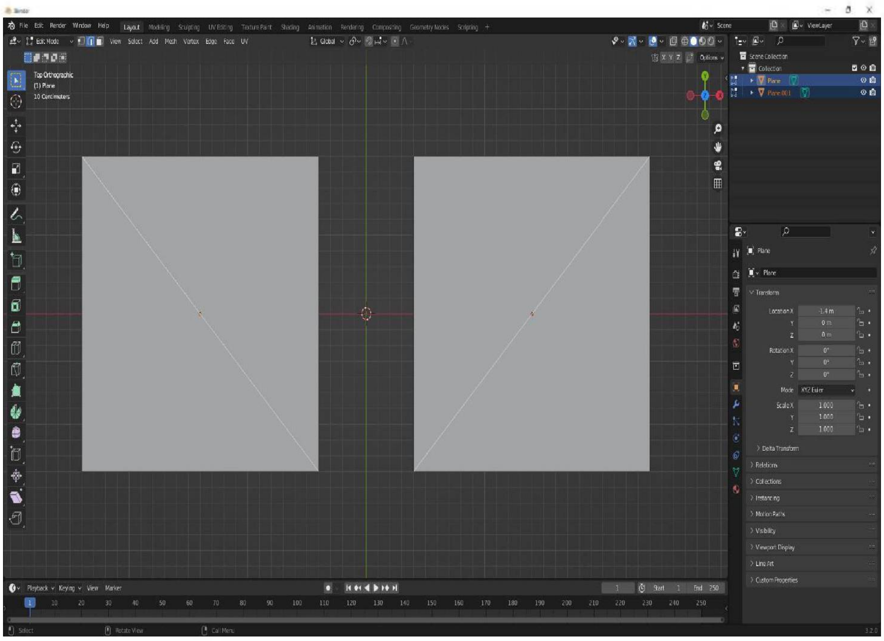


Figure 4.1: *Left quad triangulated from upper left to lower right vertices, and the right quad triangulated from lower left to the upper right*

Ngons have potentially infinite sides and can therefore be triangulated in an infinite number of directions. The pairs of vertices which are connected invisibly this way will dictate the directions in which any polygon will bend when its vertices are moved, as shown in *Figure 4.2*:

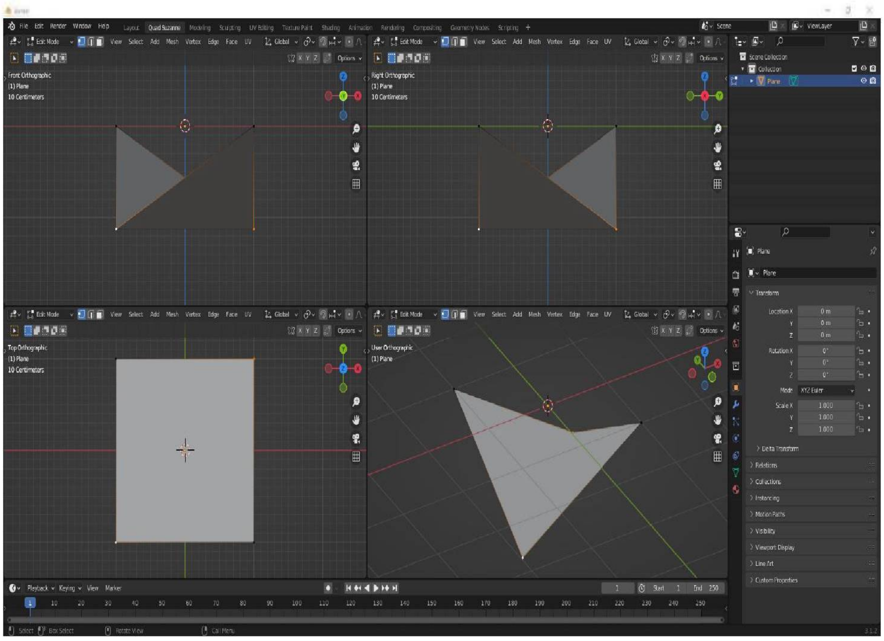


Figure 4.2: *A quad, with opposite corner vertices translated in the z axis, bending along the invisible edge in between*

The same quad as shown in *Figure 4.2*, with the same two vertices (highlighted) in the same exact position, bends along the opposite diagonal when the invisible edge triangulating the quad has been rotated, as shown in *Figure 4.3*. This time the surprising effect is not a trick of the eye and can be recreated with a square of paper held by four hands and folded across a diagonal. Whichever direction the fold is oriented will determine whether the paper folds up or down. Please refer to the following figure:

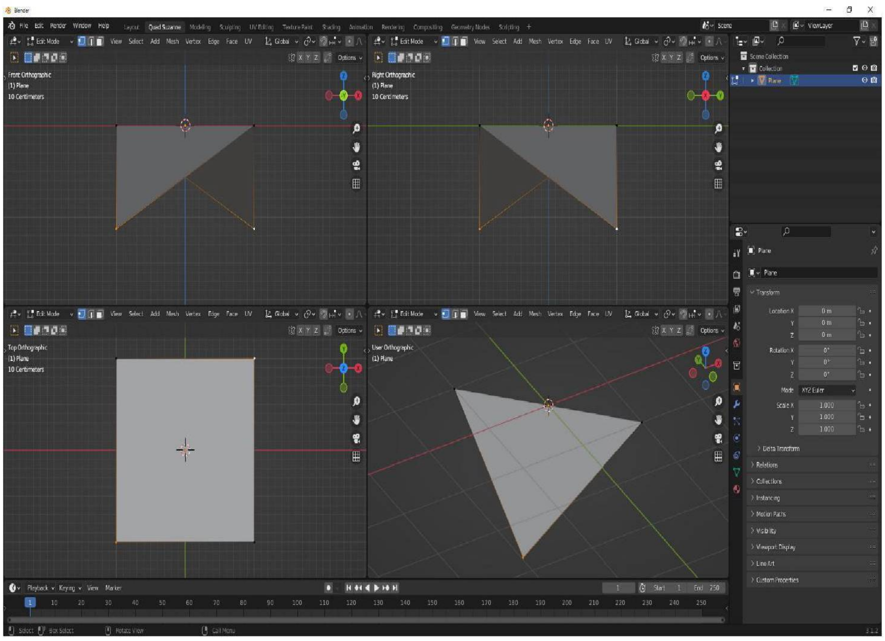


Figure 4.3: *A quad with the same opposite corner vertices translated as above, bending along the opposite invisible edge*

The initial direction in which these invisible edges are rotated is determined at the creation of every new polygon but can also be manually or automatically changed. It is rare to need to adjust the rotation of these edges face-by-face early on in 3D creation, but for future reference: from **Edit Mode**, with **Edge select mode** enabled, Rotate Edge CW (clockwise) or Rotate Edge CCW (counterclockwise) can be accessed through the **Edge menu** in the upper left of the 3D Viewport or via the **Ctrl + E** shortcut, with the mouse cursor over the 3D Viewport.

Pros and cons of triangulation

If you are interested in 3D designing for game development, you may have heard that triangulation of 3D objects should always be avoided, or that every polygon should be a quad. While triangulation of 3D game objects is not always a problem, it can be. But the situations when and the reasons why are commonly misunderstood. As we've just discovered, all 3D objects are already triangulated, it's just that the

As we've just discovered, all 3D objects are already triangulated, it's just that the connecting edges are not always being displayed or calculated for editing purposes.

When the direction of those invisible triangulating edges is unplanned, a 3D object's surface can look very different from what was intended, especially when the object is deformed through animation or displayed with certain surface treatments as we will see in Exercise 4.1. Whether or not the underlying triangulation is visible and calculated can also significantly impact the speed of certain poly modeling tasks, which we will explore later in this chapter. For now, notice how visible triangulation cuts a jagged and asymmetrical path across rows and columns of connected quads as shown in *Figure 4.4*.

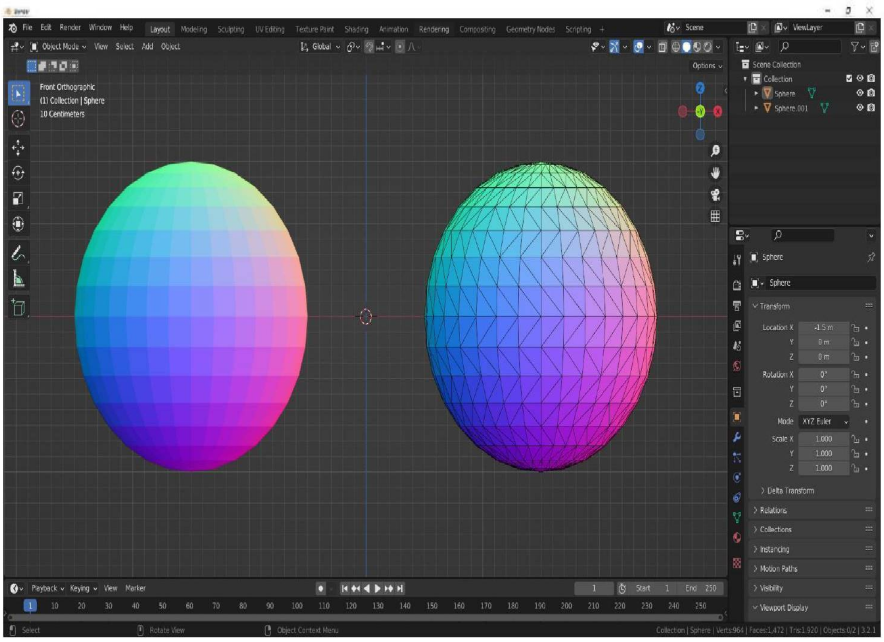


Figure 4.4: *Two spheres constructed from quads, with default triangulation visible on the right*

Most often, the rotation of the unseen edges of any given 3D object is inconsequential, especially in simpler designs, and under common display situations. Some 3D surface treatments such as those we will discuss in *Chapter 7, 3D Surfaces* can also hide unwanted effects of unintended triangulation. Once we begin working with more convoluted shapes that curve and bend in 3D space, the situations where unseen triangulation matters will become more apparent. For now, this serves as another reminder that digital objects are all merely a trick of the eye.

Methods of 3D creation

The most common methods for creating in 3D are poly modeling, sculpting, and **computer-aided drafting** (CAD.) CAD is a more mathematically precise method of 3D creation from parameters that we won't discuss in this book, as it is typically reserved for situations where precision is critical to safety and stability, such as in architecture, manufacturing, and scientific applications. While Blender excels at

hard-surface modeling where technical constraints are more flexible, such as in product visualization – CAD software is better suited to digital engineering tasks than Blender. Fortunately, Blender excels at both poly modeling and sculpting, of either hard surface or organic objects.

Hard surface versus organic

3D creation is often subdivided into the hard surface and organic categories to help separate the skills and tools that are faster and more precise for the creation of each type of 3D object and its intended use. 3D creators often choose to specialize in either hard surface modeling or organic, and many employment opportunities specify a preference.

Hard surface, in 3D, means appearing to be made through mechanized or computer-aided means, such as 3D models of cars and buildings. Machine and computer made objects tend to have a more rigid and regular appearance than objects made by living beings. Even objects which only exist in fantasy may be considered hard surfaces if they appear too mathematically precise to be made by an imperfect, living creature. Representations of living organisms, if made to look carved, printed, or generated through an algorithm, such as humanoid robots or a plastic flower, can still be considered hard surfaces under this definition. Tools and techniques which produce precisely smooth, sharp, and regular shapes are preferable for hard surface 3D creation.

Organic can refer literally to plant and animal life but doesn't exclusively refer to living things. In the context of 3D, organic means anything appearing to be affected by natural forces like gravity, growth, erosion, wind, and water, or anything made directly by living beings. Organic objects have irregularity, asymmetry, wear, and accidental imperfections. By this definition, 3D rocks are considered organic rather than hard surfaces. A 3D bag created to look machine sewn from machine woven fabric but slouching with bulges and wrinkles as if affected by gravity or placement by a human hand can also be considered organic. Tools and techniques which allow for randomness and free-hand creation are preferable for organic 3D objects.

Note: While computer generated noise patterns and mathematical randomness can resemble or enhance an organic appearance, the human mind often finds something uncanny about the artificial regularity created by a machine and will reject it as unrealistic. Believable organic results are often easier to achieve manually because of subtle, accidental irregularities produced by the touch of a living being.

Some 3D objects aren't clearly hard surfaces or organic, or the skill and tools for both must be employed in their creation. A crumbling building, for example, or a zipper sewn to a lumpy coat can each be considered a mixture of both. Hard surface skills and tools would be required for the underlying structures, and organic skills and tools would be required to weather and warp them realistically.

Poly modeling versus sculpting

While poly modeling involves directly editing a 3D object's sub-object components, 3D sculpting – like its real-life counterpart – involves adding to and subtracting from the surface and volume of an object, as well as pushing and pulling it like clay. 3D sculpting can also include changing an object's overall shape in ways that don't correlate to real-world sculpting at all, such as inflating, subdividing and collapsing, remeshing, or using simulated forces like gravity, collision, wind, and fluid on both soft and rigid bodies.

Sculpting in 3D is typically performed with a specialized set of tools, with many interchangeable parts often referred to as brushes even though they have qualities unlike a real brush, such as carve, pinch, inflate, and nudge, which we will begin to explore in *Chapter 6, 3D Sculpting*. But first, we will create a base mesh upon which to sculpt later, like a lump of clay in the overall form of our desired result, via poly modeling.

Low poly versus high poly

With any given 3D object, it is important to remain aware of the total polygon count and make thoughtful choices about when to add and remove them. This is one reason we explored how to find scene statistics in *Chapter 3, General 3D Concepts*. As a reminder, scene statistics, and individual sub-object component counts can be enabled to display in the bottom-most menu bar on the right-hand side in Blender via right-click. Scene statistics can also be enabled to appear in the upper left corner of the 3D Viewport via the Overlays menu in the upper right corner, as shown in *Figure 4.5*:

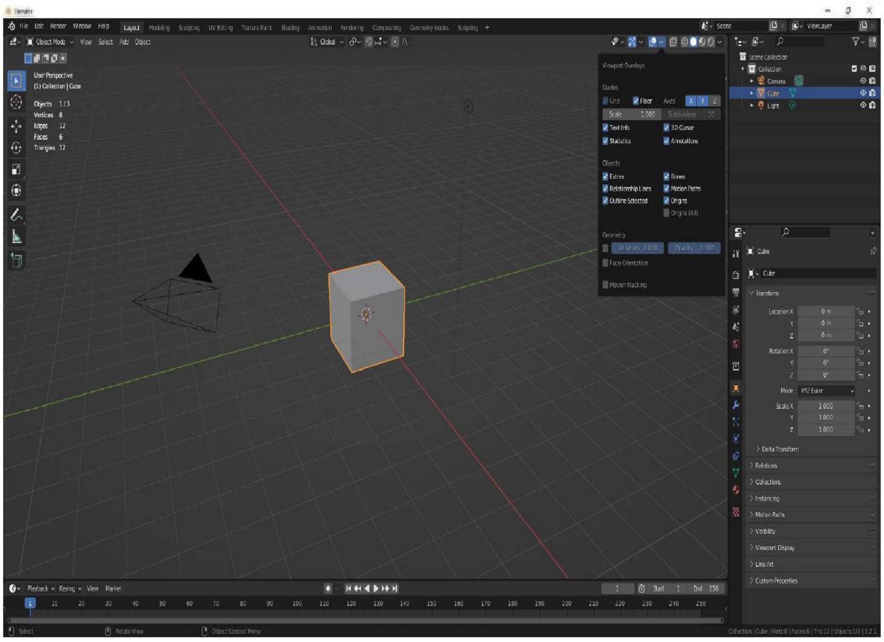


Figure 4.5: *Enabling Scene Statistics in the 3D Viewport via Overlays menu*

While poly modeling is typically used to create objects with a lower or controlled number of polygons, and while some sculpting techniques rely on or create higher polygon counts, the distinction between high poly and low poly is not strictly dictated by the methods through which a 3D object was made. Whether a 3D object counts as low or high poly is entirely dependent on whether the finished object consists of few or many polygons, relative to the whole project it is used within.

It is important to keep track of poly counts both while planning the desired result and throughout multiple stages of the creation process, but the terms low poly, high poly, and even medium poly all refer to wide ranges of poly counts that only matter in context. 3D productions will often establish a budget or maximum limit for how many polygons any individual 3D object should have, although setting these limits is tricky and must account for factors like the prominence and importance of specific objects in relation to others.

High poly 3D objects are most appropriate for use where high resolution smoothness, sharpness, and detail are needed, such as in 3D printing, or product renders when the object will never be displayed or moved in real time. Objects given higher poly count allowances in video game development are sometimes called "hero" pieces because their placement or importance to the whole product justifies their consuming a larger portion of the overall budget. It is important to note though, that even allowances for higher poly counts don't justify excess, a concept we will address again in *Chapter 11, Using 3D Objects in Production*.

Low poly, on the other hand, implies a certain level of optimization. An object accurately described as low poly means that only the number of polygons necessary to the object's form and function are used and no more, making that object performant for its intended use, such as in real time applications like games and interactive applications. Unfortunately, the label "low poly" is often misapplied to medium and high poly 3D objects that only appear to have fewer polygons because of a faceted or flat-shaded surface treatment.

Tip: When marketing 3D creations for sale, it is very important to truthfully represent the number of polygons a given 3D object contains rather than labeling an asset low poly based solely on the style. High poly objects in a product with a low polygon budget, such as Virtual Reality, can literally break a game, make players of a game physically ill, or crash a customer's computer.

Complicating low and high poly considerations is the fact that – as we will see in *Chapters 5, 3D Sculpting*, and *Chapter 6, 3D Surfaces* – low and high poly objects are often used together in the creation of one 3D object. A common scenario is that high poly objects are used for small, high-density detail, to be transferred onto the surface of a low poly duplicate object through a process called "baking normal maps" which we will explore in *Chapter 7, 3D Surfaces*. This is an essential way that 3D creators can fake computationally demanding high-resolution detail while achieving adequate real time performance.

Low to high, large to small

Through poly modeling, it is time consuming but possible, to achieve quite complex results. However, as we experienced in *Chapter 3, General 3D Concepts*, poly modeling can also begin with the least complex components. This simplicity is one of the key benefits of poly modeling.

In introductory art classes it is explained that the human brain naturally condenses chaotic input from daily life into simple forms or symbols, to help us make sense of otherwise overwhelming complexity. Almost universally, a child instructed to draw a person will begin with a stick figure, just as their human ancestors did millennia ago. For this reason, it is naturally easier to begin creative efforts from bite-sized starting points and build from there.

As with all other art forms, 3D creation also benefits from a thoughtful approach

As with all other art forms, 3D creation also benefits from a thoughtful approach of gradually increased complexity from low to medium, then high, and from a few larger forms to more medium forms, with too many small details. For more on this concept, popular professional 3D artist Neil Blevins explains and illustrates various rules of three for composition in the many art lessons on his website **http://www.neilblevins.com/**.

As we saw in *Figure 4.1*, a humanoid figure can be conceived of as a collection of primitive shapes. Sometimes, a basic shape is a result we are seeking to create. The default cube in *Figure 4.6*, makes a striking composition all on its own, for example. However, most creative compositions call for further development and refinement. Please refer to the following figure:

Figure 4.6: *Blender's infamous default cube, destined to be deleted in perpetuity*

Planned complexity

In *Chapter 3, General 3D Concepts* we learned that primitive objects can be a helpful shortcut for beginning a complex shape. A face may be sculpted from a primitive sphere for example. In this chapter, however, we will discover that there is sometimes a stage of complexity in between that gets us closer to higher complexity when desired, through fewer steps. Once again, it is important to consider our time as creators as valuable and follow time-saving workflows whenever possible.

For this reason, it is advisable to plan for the desired complexity of the finished work and then think backward through mid-levels of complexity, before choosing a low complexity starting point. Imagining the finished creation in our mind is one helpful way to proceed. Creating a 2D sketch is another. Creating a 3D mockup out of primitive shapes – called a grey box – is yet another helpful way to experiment and discover where the high, medium and low levels of complexity exist naturally in your planned 3D creation.

Collecting multiple references to work from is an even more advanced way to plan a 3D project. Taking photos in the real world, or screenshots from the web, and compiling them into a collage – along with any 2D sketches and screenshots of 3D mockups – is a great way to gather reference material while enhancing our mental image of the desired result. Including markups, like circles and arrows, and text to emphasize areas of importance as shown in *Figure 4.7*, is also extremely useful to this planning process:

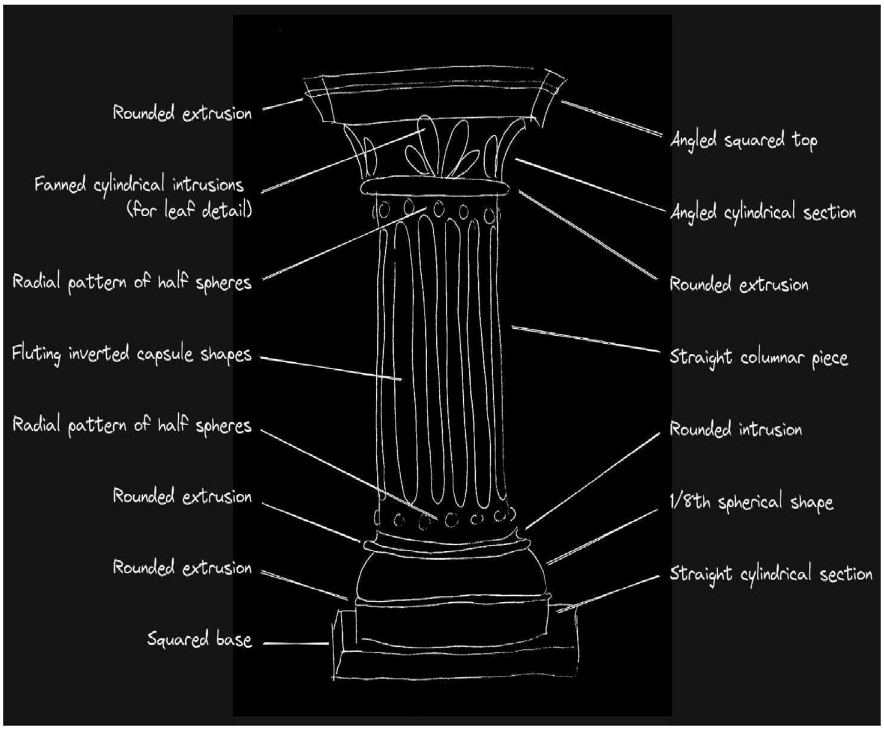

Rounded extrusion
Fanned cylindrical intrusions (for leaf detail)
Radial pattern of half spheres
Fluting inverted capsule shapes
Radial pattern of half spheres
Rounded extrusion
Rounded extrusion
Squared base
Angled squared top
Angled cylindrical section
Rounded extrusion
Straight columnar piece
Rounded intrusion
1/8th spherical shape
Straight cylindrical section

Figure 4.7: *Hand-drawn sketch of pedestal concept, with annotations made with Excalidraw.com virtual whiteboard*

Reference images also help to communicate a shared vision to clients or members of a team. Working from reference is not just convenient, it is also necessary to ensure performance of our 3D creations as intended, and to minimize wasted time and money in the production of 3D products like games, film, and 3D prints. For this reason, we will revisit working from reference in *Chapter 11, Using 3D Objects in Production*.

Performance considerations are an important reason to start from a thoughtfully chosen level of complexity; not just the performance of an end-product, but also hardware and software performance during production. As we will also explore further in *Chapter 11, Using 3D Objects in Production*, the speed of editing and rendering are both critical concerns for digital products where development time makes or breaks the success of a launch, or where customers will interact with 3D objects in real-time on a variety of computers and devices.

Where to start

We'll begin our exploration of poly modeling by imagining a low-to-medium poly 3D pedestal you might like to create for display on the web or in product renders as shown in *Figure 4.7*. 3D pedestals, pillars, columns, or plinths commonly appear in 3D environments, serving both practical and decorative purposes. Pedestals have a recognizable, simple shape, while allowing for infinite variety of creative customizations.

At this point, it might be tempting to wonder: if we can sculpt a pedestal from a sphere without ever touching the sphere's sub-object components, then why would we ever start with anything less? The simple answer is that a sphere has many vertices, edges, and faces (512, in the default UV sphere shown in *Figure 4.4*) that are extraneous to the shape of a pedestal which we would need to remove later through a process called "retopology," to make the resulting object performant for popular uses like games and interactive applications.

Therefore, an ideal midpoint of complexity for beginning to model a pedestal – a shape which could potentially include multiple curves, twists, extrusions, and intrusions – exists somewhere between the simplicity of a single vertex and the complexity of a sphere. In Exercise 4.1, we will begin poly modeling our pedestal from a cylinder because it most closely resembles the final form while saving us the effort of making a perfectly circular columnar shape by hand.

Another reason to start modeling from a polygon or primitive rather than sculpting is to easily achieve this precision. Modeling a set of 3D walls, floors, ceilings, doors, and windows to be snapped together into scenery, for example, requires consistency between the units of measurement. While a pedestal can be sculpted to resemble many organic shapes, many of its features are also self-similar. Starting from precise shapes like columns, cubes, spheres and curves makes maintaining that precision simpler and faster.

Topology and edge flow

In *Chapter 3, General 3D Concepts*, we performed the first of many possible poly modeling operations when we extruded a vertex, then an edge, and finally a plane, resulting in a cube. Extrusion – one of the primary tools of poly modeling – also allows us to plan and create faces, edges, and vertices in a grid configuration, by extending new faces from previous faces, and new groups of connected faces from previous groups.

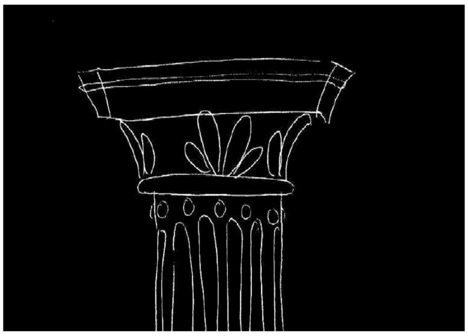
In the same way that we can select entire rows and columns of a spreadsheet or a table through the click of a button, creating a 3D model with a grid-like surface allows us to rapidly select and edit entire rows and columns of vertices, edges, and polygons at once. If you've ever edited a spreadsheet, you know the tedium

of selecting every cell individually, which is why one of the first shortcuts we tend to seek is how to select more than one at once. Creating models with the aim of maintaining a seamless flow of edges in neat rows and columns over as much of the surface as possible is a valuable method of optimizing both our workflows, and the shapes we work with.

Another reason to begin with poly modeling is for optimization. Poly modeling allows us to start creating a 3D object from an optimal shape and continue building on that in an optimal way. Poly modeling techniques like extruding, collapsing, subdividing, and dissolving allow for iteration through removal of the unneeded while refining the necessary, whereas sculpting can change a 3D object in ways that cannot be undone. This has what is called a "destructive" effect on the underlying structure of a 3D object, called its "topology." Poly modeling ensures precise control over the creation, removal, and placement of every polygon, every step of the way.

Exercise 4.1

We'll begin this exercise by first opening a new instance of Blender, then selecting and deleting the default Camera, Cube and Light objects. Then we will place an image from our hard drive into the BLEND file to serve as a reference. To follow this exercise using the same design, you may want to take a digital photo of the sketch provided in *Figure 4.8*, but many of the concepts will apply to your own designs as well.



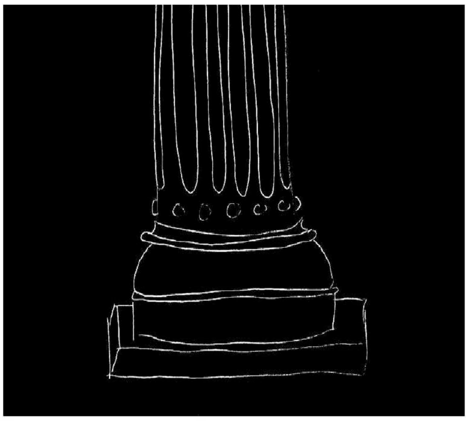


Figure 4.8: *Hand drawn sketch of pedestal concept*

1. To set up a reference image, we'll enter Front Orthographic view by pressing the number pad 1 key, or by clicking the green circle on the 3D Viewport gizmo with a negative X in the center.
2. From Object Mode, we will press **Shift + A** with our mouse inside the 3D Viewport, and select **Image** | **Reference**, to open Blender's File View.
3. Navigate to a reference image – changing the **Display Mode** to Thumbnail if necessary – and select, then load the desired image as shown in *Figure 4.9*:

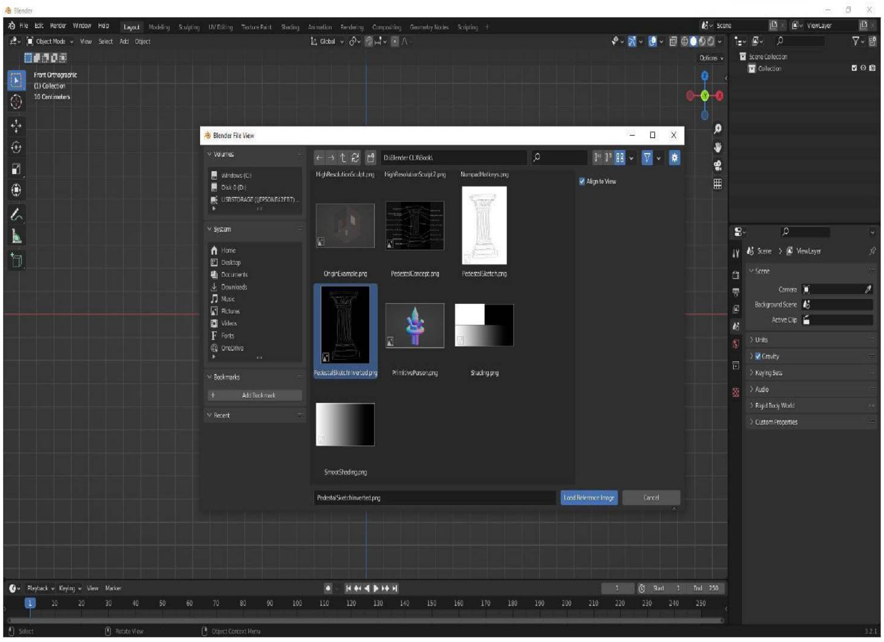


Figure 4.9: Navigating through Blender's File View to locate the desired reference image, using Thumbnail Display Mode

Note: At the time of this writing, Blender version 3.2.1 has just been released, and is the version that will be shown in the following figures. Refer to Chapter 2, Installation and Interface, for guidance about when and why to upgrade to the latest version of Blender.

Depending on the dimensions of this image, labeled Empty in the Outliner, its imported scale may be too large or too small. Just like any other 3D object though, it can be infinitely re-positioned, rotated, and re-sized to suit our needs. To create our 3D object in real world dimensions we should scale any reference image so that the object depicted within is also real-world scale. If we want the pedestal to stand about 4 feet or 1.2 meters tall, for example, we should scale the reference to 1.2 meters tall as well. However, since there is space in the preceding image and the following object depicted, you can't refer to the image size alone. Another object scaled to real world units of measurement will help with that, so now we'll add our first primitive shape.

4. With our mouse cursor inside the 3D Viewport, we'll press **Shift + A**, and select **Mesh** | **Cylinder** to add a new cylindrical primitive shape to the scene. Press the number pad period key, to frame the selected object in the view.

   Notice the contextual menu which appears in the bottom left area of the 3D Viewport. If you accidentally clicked away and this menu has disappeared, in the top-most upper left menu of Blender, select **Edit** | **Adjust Last Operation** (or press the **F9** key) to bring it back.

   From here, we can adjust the starting number of edges in our cylinder, as well as its starting radius and depth.

5. The cylinder's radius to the reference image once we correct the image scale, so leave that alone for a moment.

   When creating objects with performance in mind – such as for displaying on the web – it is desirable to start with the minimum shape necessary to create a recognizable silhouette. In 3D, as we saw in *Figure 4.5*, the illusion of curvature in 3D objects such as a cylinder can be created from a very minimal number of sides. In game development, a minimum of 6 sides is commonly used for curved objects that will be seen from first person perspective. At 32 sides, the default cylinder is much more complex than we need, so we will first adjust the parameters defining this object before continuing to edit it.

6. In the Add Cylinder contextual menu, change the Number of Vertices to 12 – by typing 12 into the open parameter field.

This is the actual number of vertices in the circumference, not the total, and will result in a cylinder made up of 12 faces. 12 faces provide a more than adequate appearance of roundness, but is still easily reduced to 6 later, by removing every other edge. The resulting shape appears more faceted than rounded at first, so next we will change how the surface is shaded.

Shading and smoothing

In real life, humans perceive the shape of visible objects through light bouncing off surfaces and back onto receptive parts of the eyes which send signals to the brain. Faking the appearance of reflected light is one way that optical illusions like Escher's impossible structures are achieved. Art lessons often introduce shading in these terms.

In addition to helping create the impression of a surface, the perceived distance of that surface from the viewer can be adjusted through varying its brightness or darkness. In 3D "shading" refers to the simulated behavior of light reflecting off 3D surfaces displayed through variation in the brightness and darkness displayed on each visible face of a 3D object. In 3D modeling software, there are two primary shading models, smooth and flat.

- **Flat shading** is, as we can observe in our 12-sided cylinder, the true structure of a 3D model – which is multiple connected, but flat, surfaces – is displayed more realistically, but not always as desired.
- **Smooth shading** is, where edges between any two faces marked smooth are artificially blurred. Instead of an abrupt delineation between surfaces facing different directions, a gradient is displayed across the common edges.

Looking back at *Chapter 3, General 3D Concepts*, we can begin to understand how these two modes of shading are achieved. As we discovered, the normals of a 3D object's faces, edges, and vertices dictate which directions those components **appear** to be pointing. The appearance of smooth shading between one face and another is achieved by blending the directional information provided by neighboring normals, resulting in a gradient between light and shadow that adjacent edges marked smooth appear to be receiving as shown in *Figure 4.10*.

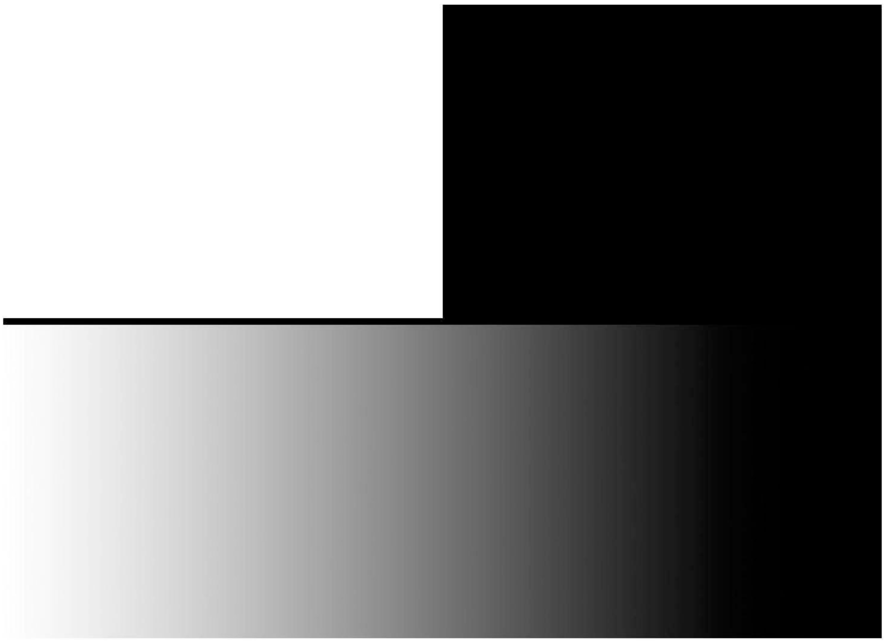


Figure 4.10: *Solid light and dark represent flat shaded areas, gradient below represents smooth shading*

1. To enable a smooth, round-appearing shading on our 12-sided cylinder, with the Cylinder selected we will select Object from the upper left menu in the 3D Viewport and choose Shade Smooth as shown in *Figure 4.11*:

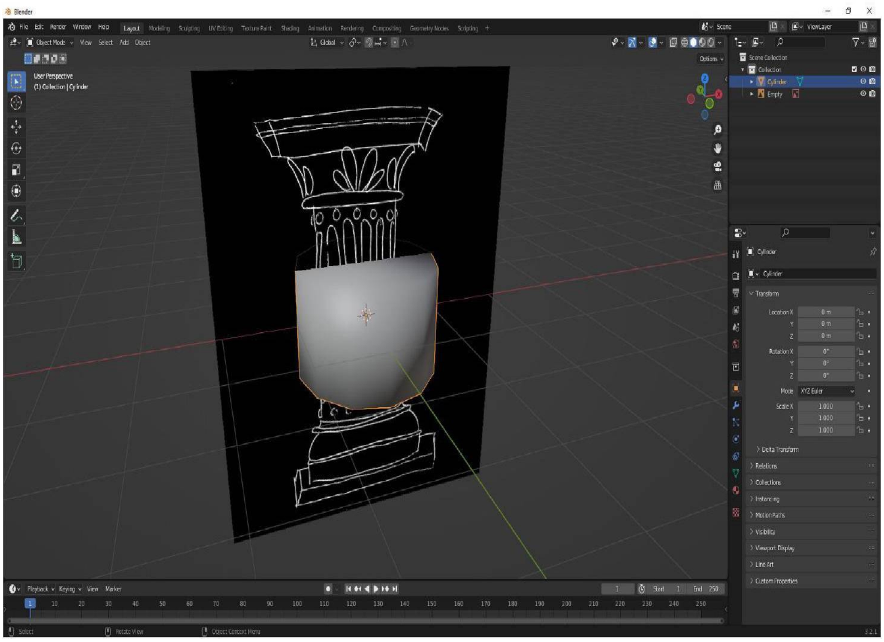


Figure 4.11: A completely smooth shaded, 12-sided cylinder

If we orbit the view by pressing the middle mouse button and moving the mouse, we can see that the entire cylinder appears smooth, but that isn't quite right. Now the upper and lower edges of the cylinder also appear smooth where they should appear flat, the way cut ends of a log would. To correct this, we'll need to enter **Edit Mode** and apply markings to delineate where we want the shading to appear flat or "sharp", separate from the side edges that we want to appear smooth. In 3D this is commonly referred to as assigning "smoothing" groups.

2. Enter **Edit Mode** by pressing the **Tab** key. From here, we will switch from the default **Vertex select mode** to Edge select by pressing 2 on the keyboard.

   This is where we'll see how working with untriangulated quad faces can come in handy, as it will allow us to select entire loops of edges – referred to as "edge loops" – all at once.

   In much the same way that we can select entire rows and columns of a spreadsheet or a table through the click of a button, creating a 3D model with a grid-like surface allow us to rapidly select and edit entire rows and columns of vertices, edges, and polygons.

3. In Edge select mode, press and hold **Alt**, and then left mouse click in the middle of one of the edges in the loop encircling the top face, and the whole loop of edges will be selected. Activate the Edge menu again with **Ctrl + E** and this time select Mark Sharp from near the bottom of the menu.

and this time select Mark Sharp from near the bottom of the menu.

4. Press and hold **Alt** again, and this time left mouse click on an edge in the bottom edge loop. Repeat the previous step again, marking the bottom edge loop sharp, via the Edge menu, or by pressing **Shift** + **R** to repeat the previous action from history.

Note: If you haven't changed the object's creation settings, you will notice that the top and bottom faces of this cylinder are currently 12-sided ngons. It's okay to leave them as-is, for now.

5. Exit **Edit Mode** by pressing the **Tab** key, and it will appear as if nothing has changed. For the Mark Sharp effect to take hold, we also need to enable Auto Smooth, which allows for groups of edges to display as either smooth or sharp independently, based on markings applied to the group's bounding edges. Auto Smooth is accessed through the Properties panel in the bottom right of the default Layout workspace, under the Object Data Properties tab appearing as a triangle with tiny circles at the corners.

   In this tab, expand the Normals section mid-way down, click the box to enable Auto Smooth, and then increase the amount to 180 degrees as shown in *Figure 4.12*. This amount is the face angle threshold under which Auto Smooth will take effect.

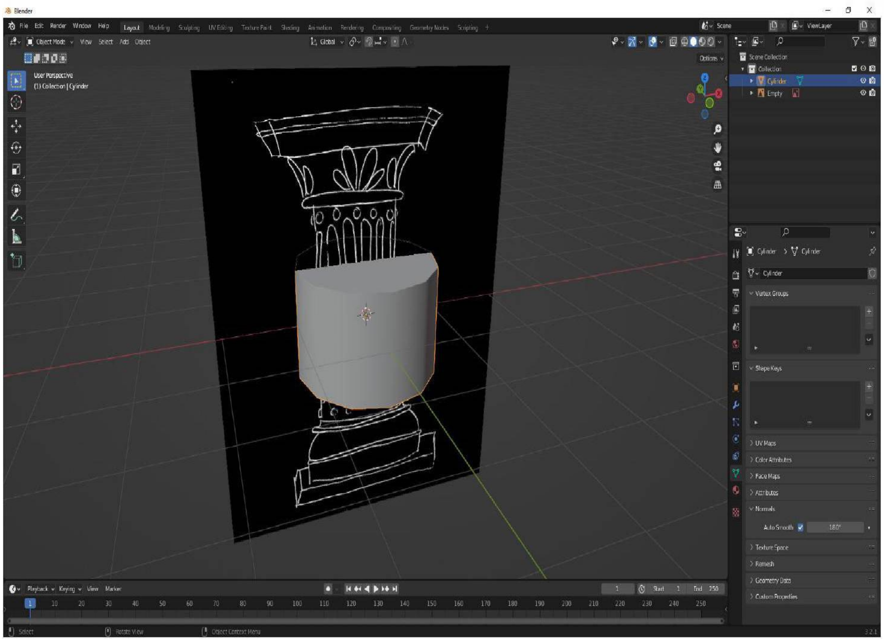


Figure 4.12: *Auto Smooth enabled via the Normals section inside the Object Data Properties tab of the Properties panel*

Before continuing, let's go back and scale the reference image so that the sketched pedestal matches the height of our cylinder, which we set earlier to be the total height of the pedestal: 1.2 meters.

6. We could simply select the Empty object and press **S** to scale, but the cylinder would be in the way. Instead, let's temporarily enter the Wireframe Viewport Shading mode, by pressing the icon appearing as a spherical grid in the upper right menu of the 3D Viewport.

   Wireframe mode draws only the edges of 3D objects, while special objects like our reference image remain visible. If necessary, scroll the mouse wheel to zoom, or press the number pad period key to frame the reference image in the view, and press number pad key 1 to enter front Orthographic view.

7. Now, by pressing the **S** key, scale the whole image so that the pedestal reference fits within the top and bottom of the cylinder's wire frame.

Tip: To make edges and wires a little bit easier to see, navigate to Edit | Preferences | Interface, and under Display increase the Line Width option to Thick.

   With the reference image properly scaled so that the pedestal pictured within it is 1.2 meters tall, we can ignore the cylinder for a moment.

8. Press **G + Z** and position the reference so that the base of the pictured pedestal is at the same level as the red and green grid lines, which mark the world x and y axes. It is perfectly alright for the previous transforms to be merely approximated since we're creating an object from our imagination. In this situation, getting close to real world scale is good enough.

Beginning to add detail

Let us now add details by implementing the following steps:

1. Set the **Viewport Shading mode** back to Solid by pressing the upper right hand menu icon appearing as a solid sphere. Now that we have a column appearing smooth and sharp where desired, let's begin to add some more details from our design.

   In Blender, we can easily combine multiple primitives, in either **Edit Mode** or in **Object Mode**, for different effects.

2. For this exercise, let's make sure we're in Object Mode, to learn how to combine separate objects. To add a square base to the cylinder, lets first add a Cube to the scene, by pressing **Shift + A** | **Mesh** | **Cube** with our cursor in the 3D Viewport.

If we selected both the cube and cylinder at once, we could enter **Edit Mode** and still edit the Cube and Cylinder separately, but we wouldn't be able to bridge or merge parts of the separate objects together.

3. To allow the connecting of sub-object parts of these two objects (while leaving the cylinder's Auto Smooth enabled), first select the cylinder from the Outliner panel in the upper right, and then press Shift, and select the Cube. Then, with the Cylinder highlighted in yellow (Active) and the Cube highlighted in orange (Selected) press **Ctrl + J**.

   Now, in Edit Mode, we'll be able to connect parts of the two primitives, making them a single object.

4. This is a good time to rename the Cylinder to something more descriptive like Pedestal, and the Empty to Reference, by double clicking and typing in each object's name field in the Outliner. This is also a good point to save our file.

Tip: Save early and save often! Saving early makes it easier to habitually press Ctrl + S to save every time that significant progress would be lost to a sudden crash or power outage.

   The cube is currently engulfing the whole cylinder, but we can easily correct this.

5. First, with the Pedestal object selected we'll return to **Edit Mode** by pressing the **Tab key** and zoom out a bit by rolling the middle mouse wheel back, to give ourselves some room to work. In Edit mode, we can see that newly added sub-object component are selected at creation, along with any sub-object component that was already selected. This is a handy feature that we can use intentionally later, so that obscured geometry can be selected more easily.

In this case, we probably still have an edge of the cylinder selected, so if we were to move the cube around, part of the cylinder would move with it. Go ahead and check, by pressing G, then moving the mouse around. Right-click to cancel out of the Move operation, or if you accidentally left-clicked and confirmed, press **Ctrl + Z** to undo. Click away from any mesh inside the 3D Viewport, to clear the selection.

Selecting linked sub-objects

What we want now is to edit the cube without moving the cylinder. Even though they were joined via **Ctrl + J**, at this point, the cylinder and the cube are not "linked" by any geometry. This means that they do not share any vertices, edges, or faces in common, even though their components may overlap in space.

1. To select only the cube, be sure to have cleared your selection, then hover the mouse over the cube anywhere that the cylinder is not also under the cursor, then press the L key. Now when we move the cube, the cylinder stays put.

Selecting linked sub-object parts of a 3D object this way will come in handy in many future situations. The contextual menu that appears in the lower left of the 3D Viewport (or if you have clicked away, reopen this menu from Edit | Adjust Last Operation, or by pressing F9) is the Select Linked menu and offers alternate options for what Blender will consider linked when using the L key for this selection method.

By default, this action selects all geometry connected to the component currently under the mouse cursor. The Normal Delimit option will select linked faces whose normals are matching the inside or outside pointing direction of the face under the mouse cursor. This can be useful for selecting only faces whose normals are facing the opposite way from what is intended so they can be flipped via **Alt + N**. The Sharp Delimit option will select linked faces within the boundaries of edges marked sharp. The other Delimit options will be useful when we begin exploring Materials, Seams, and UVs in *Chapter 7, 3D Surfaces*. To reset the Select Linked options to default behavior, Shift click the highlighted option to disable it.

Tip: Contextual menus like the preceding one will appear whenever an operation is performed that can still be modified. The default operation will be performed until these settings have been changed. Once the settings are changed, the last setting will be repeated when the operation is performed again.

Snapping

Now is a good time to plan out the next few steps. To match the reference, we'll want to:

- Raise the cube's bottom face to sit on top of the world grid
- Lower the cube's top face in the z axis to match
- Shrink the cube's width and depth in the x and y axes
- Move the cylinder's bottom face to sit on top of the cube
- Shrink the cylinder's width and depth in the x and y axes
- Lower the cylinder's top face in the z axis

Performing steps in a logical order is helpful when one action may impact another. In our case, some of the steps are interchangeable, however "snapping" to the world grid is easier to do with a default cube because of its uniform scale of 2 meters in all axes.

1. So first, let's move the cube up in the z axis, so that it is resting on the grid like a floor. With only the cube selected, press G, then Z (to constrain movement to the z axis only), then press Ctrl to enable snapping, and move the mouse

up until the cube snaps with its bottom face at the same height of the world grid as shown in *Figure 4.13*:

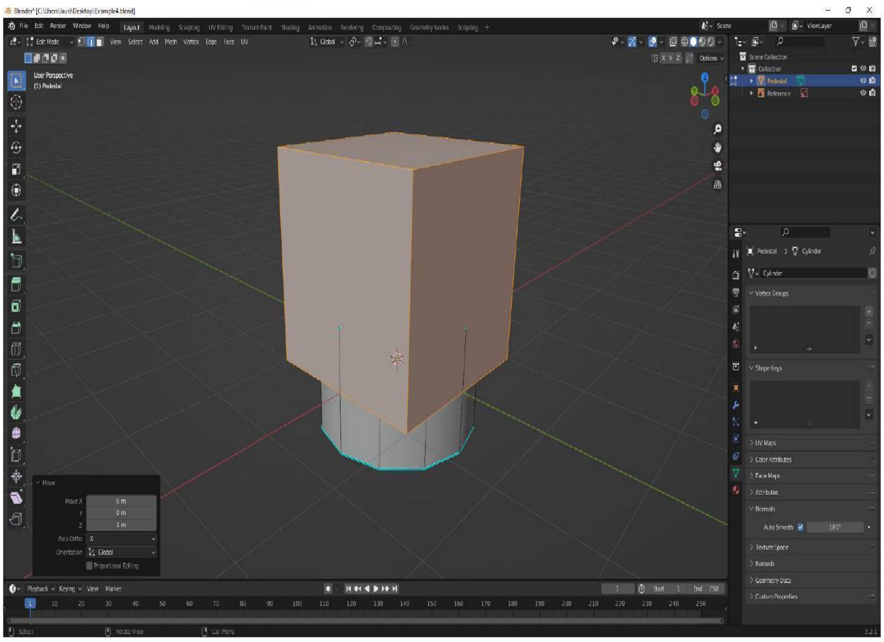

Move
Move X 0 m
Y 0 m
Z 1 m
Axis Ortho X
Orientation Global
Proportional Editing

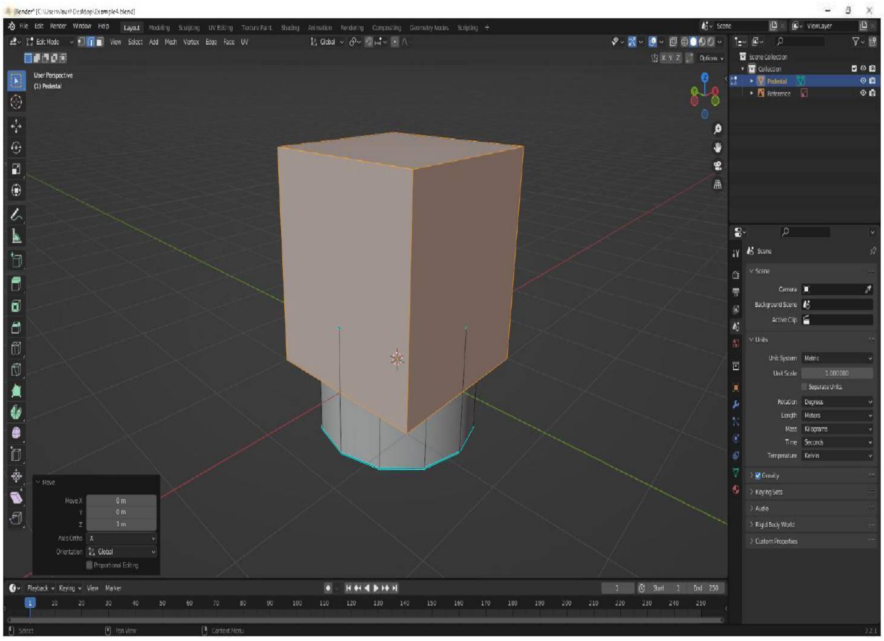
Figure 4.13: *Cube sub-object, snapped 1m in the z axis so its bottom face rests on the world grid*

Note again, the contextual menu appearing in the lower left, allowing further adjustment of the Move operation we just performed. This menu also shows that we moved the selected geometry 1 meter in the z axis positive (or up axis, in Blender). By using the Ctrl key to modify the Move transform operation, we temporarily enabled snapping to the default element – in this case, Increment – based on the file's unit of measure which is set to meters by default.

Along with Increments, there are other elements that can be snapped to, such as vertices, edges, and faces, which will come in handy very shortly. Snapping can be enabled temporarily during transforms as we just saw by combing a transform hotkey with the Ctrl key. Alternately, snapping can be toggled to stay on via the Snap To icon in the middle top of the 3D Viewport, appearing as a U-shaped magnet. The element to snap to can be switched through the dropdown menu just to the right of the Snap To icon.

It is also possible to change the Increment snapping behavior by changing the file's unit of measure, from meters to centimeters or kilometers for example. For rotation, the file's units can be left at degrees or changed to radians. It is also possible to switch completely from Metric to Imperial units, but it is advisable to leave the defaults alone until a need arises for such a far-reaching change. Even when the default is set to Metric, Blender will convert Imperial units when we type them in. Typing "3 inches" into a transform

parameter field for example, will automatically be converted to 0.0762 m. Alternate units of measure can be found in the Scene Properties tab of the Properties panel, appearing as a collection of objects, under the Units section, as shown in *Figure 4.14*:


Move
Move X: 0 m
Y: 0 m
Z: 1 m
Axis Ortho: X
Orientation: Global
Proportional Editing

Scene
Camera
Background Scene
Active Clip
Units
Unit System: Metric
Unit Scale: 1.000000
Separate Units
Rotation: Degrees
Length: Meters
Mass: Kilograms
Time: Seconds
Temperature: Kelvin
Gravity ✓
Keying Sets
Audio
Rigid Body World
Custom Properties

Figure 4.14: *Location of Scene Properties, Units, and alternate Unit Systems*

For the next step – shrinking the cube height – we'll leave snapping set to Increment, and this time select just the top face of the cube.

2. To begin, first switch to **Face Select mode** by pressing the **3** key, or by clicking the Face Select icon in the upper left of the 3D Viewport, and then left-click select the top face. This time, press **G**, then **Z**, then press and hold **Ctrl** + **Shift**, and notice how the Shift modifier has made the snapping Increments smaller and easier to manage.

Tip: Pressing Shift during an operation with Snapping enabled not only switches to a smaller increment, but also conveniently slows down the movement in the viewport, almost like down-shifting gears in a vehicle. This can be extremely helpful when attempting precise transforms, at extremes of scale, or with the view zoomed very far in or very far out.

3. In the middle of this or any transform operation, peek at the upper left menu of the 3D Viewport, and you will see that the text and icons have been momentarily replaced with a readout of the current transform amount,

as shown in *Figure 4.15*. Making a habit of checking this readout during transform operations will be incredibly helpful from now on.

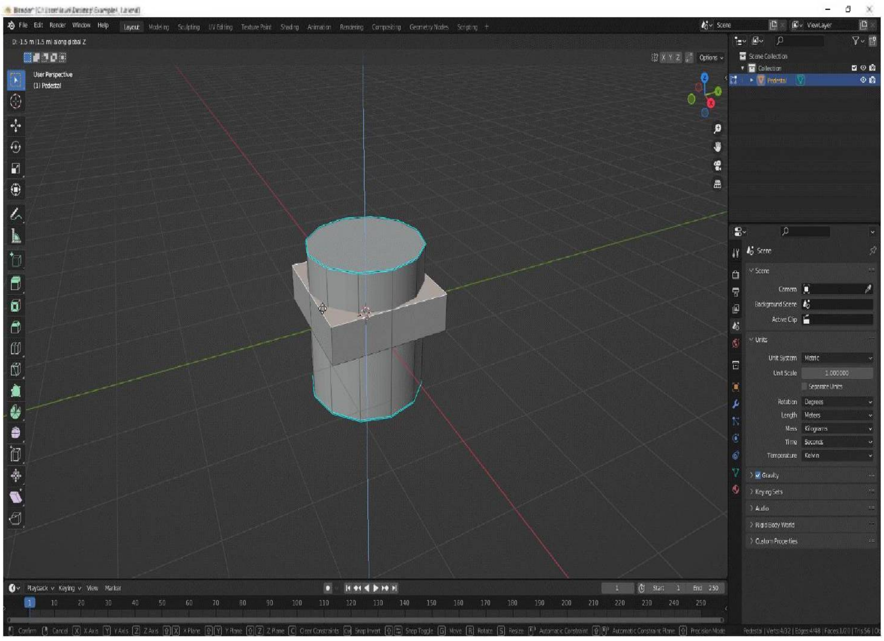

File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
D: -1.5 m (1.5 m) along global Z
User Perspective
(1) Pedestal

Figure 4.15: Upper left menu area temporarily replaced by readout of the current transform amount

4. In this case, lets left-click to confirm when the top face of the cube is lowered by exactly 1.7 meters (or -1.7) in the z axis. That may seem arbitrary, but that makes the cube height a fraction of the width that closely matches the reference. In a moment we will scale everything to match the reference.

   If pressing **G**, then **Z** then holding **Ctrl** + **Shift** is a difficult maneuver, remember that we can also move the face any amount, left-click to confirm, then adjust the transform amount via the contextual menu that appears in the lower left of the 3D Viewport. Typing -1.7 into the Z field will perform the same operation.

Tip: Getting close to the desired result by eye and then editing the values to perfection this way is another valuable time-saving approach.

Now that the rudimentary pedestal base is in place, let's move the cylinder up so that it sits flush atop the cube.

5. To begin, clear the current selection by clicking within the 3D Viewport away from any 3D object, and then hover the mouse over the cylinder anywhere that the cursor is not also over the cube, and then press **L**. We could move the whole cylinder incrementally until it's at the correct height just as we did with the cube shape, but this is a good time to practice snapping one sub-object component to another.

object component to another.

This time, we'll snap the cylinder's bottom face to the bottom face of the cube, moving the whole cylinder up in the process.

6. In the **Snap To** menu accessed in the middle menu of the 3D Viewport via the icon appearing as a ruler, select **Face** this time, and notice that new options appear in the lower half of the menu. From here, select **Active** from the **Snap With** section, and verify that Project onto Self is enabled, as shown in *Figure 4.16*:

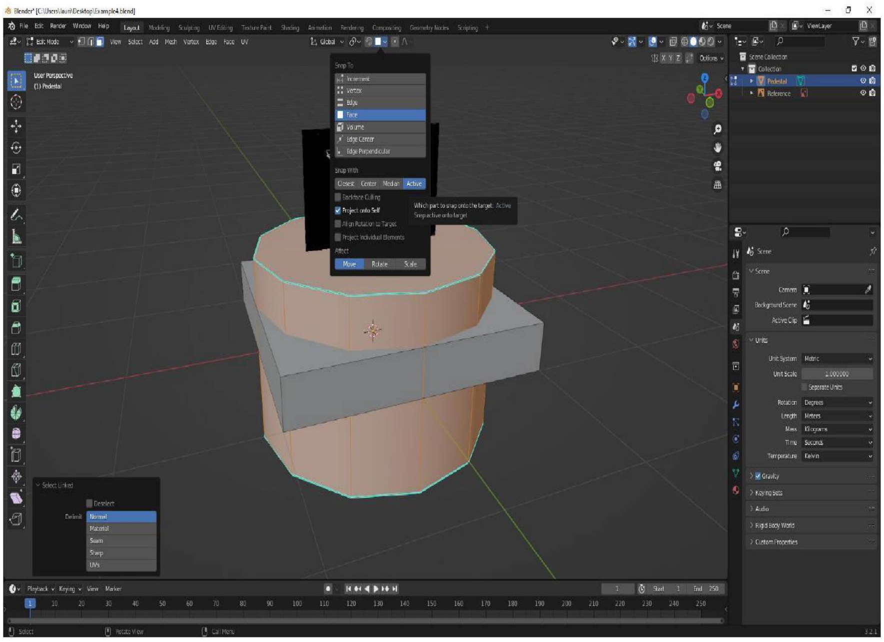


Figure 4.16: *Face snapping selected via the Snap To menu, with Snap With set to Active*

7. Now we'll rely on the distinction between Active and Selected, that we discussed in *Chapter 2, Installation and Interface*, to perform the desired snapping operation. First, let's get a better look. With the whole cylinder still selected, press, and hold the middle mouse button then move the mouse upward, to orbit the view downward until the bottom of the cylinder is visible. We can press and hold **Shift** while pressing and holding the middle mouse button and move the mouse to pan the view if needed as well. Release the mouse button when the bottom face is clearly visible.

8. Next, press **Shift**, and then left-click the bottom face of the cylinder, which will first deselect it, then left-click again to re-select the bottom face.

Tip: In any selection of multiple objects, one object (or sub-object component, in Edit Mode) will be highlighted in a lighter color than the rest, which indicates the last selection, which is the Active object. When no object is Active, operations on the Active object will not work. Re-selecting the intended Active object may

be necessary in this scenario.

Although it's hard to see, this series of steps has made the last selected sub-object component our current Active object, as shown by its slightly lighter highlight from the other selected components. This is the point where our selected objects will snap onto whichever element Snap To is currently set.

9. By pressing the middle mouse button, orbit the view back to the top, scroll back to zoom out if needed, then press **G** then **Z**, then hold **Ctrl** to activate snapping again, and move the cylinder upward until it snaps to the top face of the cube as shown in *Figure 4.17*. Left-click to confirm the operation:

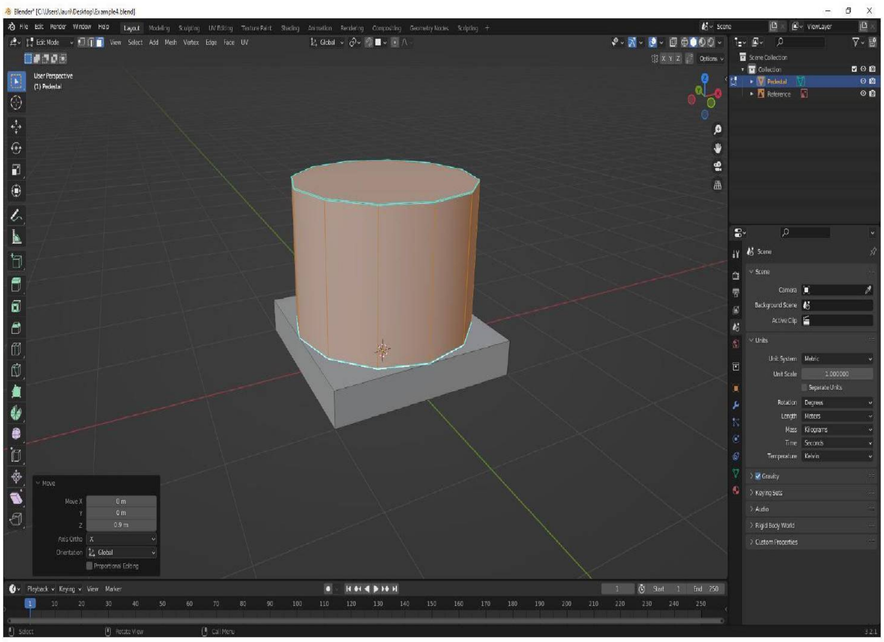


***Figure 4.17:** With the cylinder's bottom face set to Active, cylinder snapped on top of the cube base*

For a moment during this operation, you may have seen a small orange circle appear near the mouse cursor. As you move the mouse cursor around while snapping is active, this circle will jump to the nearest available snapping point, as an indicator of where the Active component will snap onto the target if confirmed at the current mouse position. This can get a little confusing when transforming freehand, but using the X, Y, or Z keys to constrain movement to one axis makes the snapping behavior easier to predict.

Hiding and unhiding geometry

Since the top and bottom of this cylinder will never be seen in the pedestal's final form, now is a good time to simply remove those faces.

1. Removing the top face is as simple as left-clicking to select, then pressing Delete or X, and choosing Faces. The bottom face is partly obscured though,

which gives us an opportunity to practice hiding and unhiding geometry to make accessing obscured components easier.

With the top face removed, we might assume that selecting the bottom face of the cylinder is as simple as rotating the view and clicking inside it, but this is where the face normal direction becomes important. Because the cube's face normal is pointing upward in the scene, and the cylinder's bottom face is pointing downward, if we try to select through the cylinder to its bottom face, we will only wind up selecting the cube's top face. This may be frustrating at first, but it is desirable behavior, confirming that the pedestal's face normals are all pointing outside so far, as intended.

To directly select the cylinder's bottom face, we'll need to temporarily hide the geometry obstructing it from view. There are multiple methods available to help us achieve this, besides switching to Wireframe as we did earlier. Another way is to enable a see-through mode called X-Ray, which as the tooltip suggests, allows selection through obstructions.

2. Enable this **transparent display mode** via the X-Ray icon in the upper right hand menu bar of the 3D Viewport, appearing as a cube with back edges visible as shown in *Figure 4.18*, or press **Alt + Z** as the tooltip suggests:

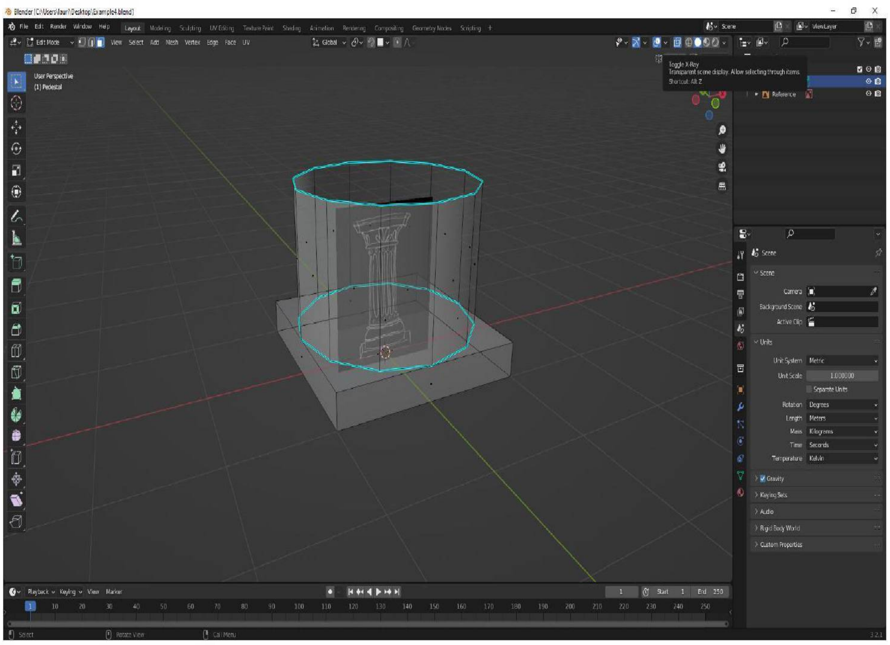


Figure 4.18: *Transparent scene display mode toggled via the X-Ray icon in the upper right 3D Viewport*

In this mode, with Face select still enabled, we can see a tiny black dot in the

In this mode, with Face Select still enabled, we can see a tiny black dot in the center of every visible face. These dots are a bit like an origin point for faces, in that it is always located at a point between the face's vertices. These dots can serve as markers, allowing us to distinguish between two offset faces that overlap in the view.

3. For now, the Reference image obstructs the view, so let's hide it temporarily too, by clicking the eye icon to the right of its name in the Outliner panel.

Note: Whole objects can be hidden through the Outliner panel by clicking the eye icon to the right of their name in either Edit or Object Mode. On the other hand, hiding and unhiding sub-object geometry via hotkeys H and Alt + H is mode dependent. Geometry hidden in Edit Mode, will still be visible in Object Mode, and vice versa.

4. With X-Ray mode enabled, look for the tiny dot at the center of the cylinder's bottom face, and left-click it to select the bottom face, while avoiding the centers of other faces. Be sure that the selected face is the bottom of the cylinder and not the top face of the cube, by checking that the highlight does not extend outside the boundary of the cylinder's edges (if it does, simply click the same spot again and the selection should switch), then delete the face.

   In more complicated shapes, many layers of geometry can make selecting the intended component difficult, even in X-Ray mode. In such a scenario, it might be preferable to temporarily hide entire sections of a mesh. In the preceding example, we could have selected the whole cube or the whole cylinder via Select Linked (**L** key), and with the cube selected, pressed **H** to hide it, or with the cylinder selected, pressed **Shift + H** to hide the inverse of the selection. We could also use one of the other selection methods introduced in *Chapter 3, General 3D Concepts*: box, circle, or lasso, but they might inadvertently select part of the mesh we want to keep visible.

5. Once finished with our selection and deletion operations, we can unhide any hidden sub-object components by pressing **Alt + H**. Now that unneeded faces of the cylinder have been deleted, we can return to solid shaded mode by toggling **X-Ray mode** off in the upper right of the 3D Viewport, or by pressing **Alt + Z**. Let's also unhide the Reference image by clicking the eye icon to the right of its name in the Outliner.

Transforming with the 3D cursor

Next, lets scale both these objects to the proper width and depth, but leave adjusting the overall height for later. Since our cube is currently sitting on top of the world grid like a floor, we can use the world center as the Transform Pivot Point for this operation. To do this, we'll perform the following series of steps:

- Ensure that the 3D Cursor is at the center of the world grid

- Set our Transform Pivot Point to the 3D Cursor
- Scale all the objects relative to this point

Remember that the 3D Cursor is the red and white dotted circle with a grey plus sign in the middle first introduced in *Chapter 3, General 3D Concepts*. By default, the 3D Cursor appears at the center of the 3D scene, also known as the world origin, but it is not difficult to accidentally move it.

1. To verify that the 3D Cursor is currently at the center of the world, with the mouse cursor inside the 3D Viewport press **Shift + S**, and then move the mouse cursor over the left option reading Cursor to World Origin or press the **1** key as suggested by the tooltip.
2. Next, we'll set the 3D Cursor as our Transform Pivot Point, via the upper middle menu in the 3D Viewport, appearing as two chain links, and selecting the 3D Cursor option from the drop down. Now, any transforms will use this point as its origin, which means movement happens relative to this point, rotation pivots at this point, and scaling happens from this point, or to this point. This is a difficult concept to convey through words, so the best thing to do is try it out.
3. Clear any selection by clicking away and then select all by pressing **A**, then press **S** and move the mouse cursor toward the 3D cursor. This will shrink the selected geometry down. Left-click to confirm when the cube pedestal base is approximately the same width as the pedestal in the reference image as shown in *figure 4.19*. Don't worry about the cylinder during this operation. Please refer to the following figure:

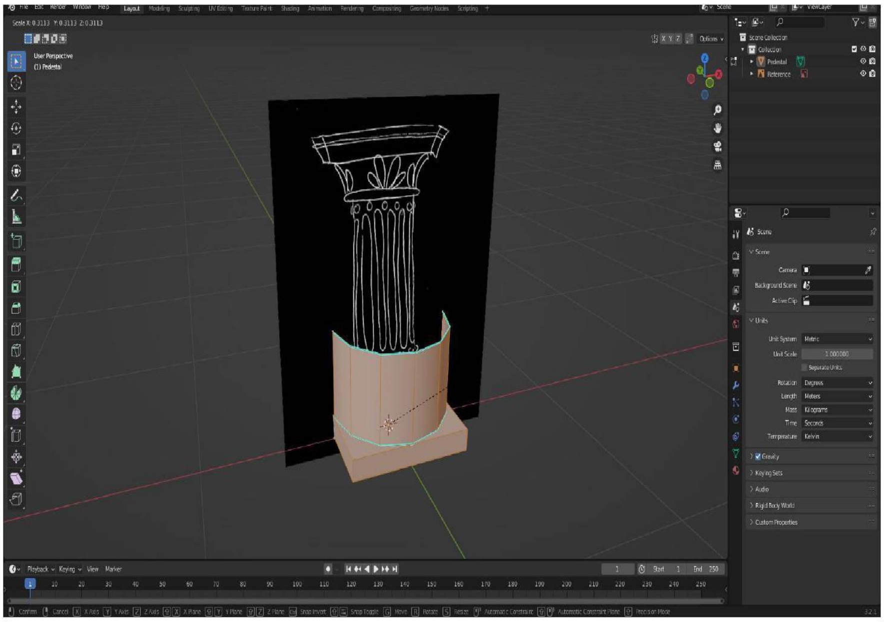


Figure 4.19: Cylinder and cube sub-object primitives scaled using 3D Cursor as transform pivot point

If we refer to the reference again, we will note that the cylinder is currently wider and shorter than intended but scaling it relative to the floor would cause the cylinder to shrink downward and intersect with the cube. We want it to stay snapped to the top of the cube, so we will use the cube's top face as the Transform Pivot Point next. To do this, we will need to perform the same steps as done previously, but with one small difference.

- Ensure that the 3D Cursor is at the center of the cube's top face
- Set our Transform Pivot Point to the 3D Cursor
- Scale all the objects relative to this point

4. To move the 3D Cursor to the center of the cube's top face, first clear any selection, then left-click to select the top face.

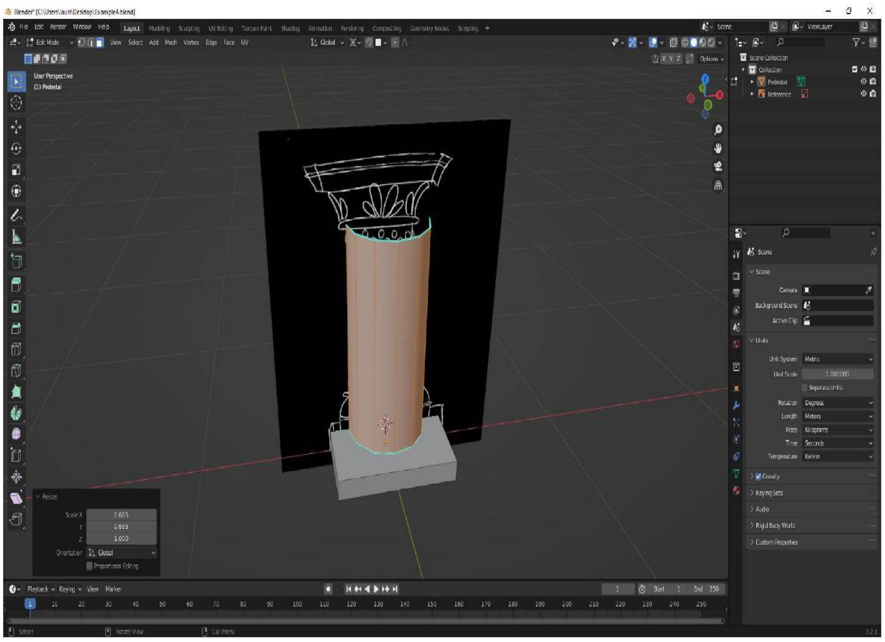
5. Next, press **Shift + S**, and this time choose Cursor to Selected, or press the 2 key. Then clear the selection and select the cylinder by hovering the mouse cursor over the cylinder shape and pressing L for Select Linked.

   Now, when we press S, scaling of the cylinder will occur from the top of the cube (pedestal base).

6. Before adding intrusions and extrusions, we'll scale the cylinder up in the z axis until its top edge is just below the first rounded extrusion in the reference image by pressing S, then Z.

   Next, lets scale the cylinder in the x and y axes to make it thinner, but exclude the z to maintain its height.

7. To do this, first press S, then **Shift + Z**. Scale the cylinder approximately 0.6 in the x and y axes as shown in *Figure 4.20*, but don't worry too much about matching the reference perfectly, since it is lopsided. Hand drawn reference images don't need to be followed precisely but can still provide useful landmarks.



Resize
Scale X 0.665
Y 0.665
Z 1.000
Orientation Global
Proportional Editing

Figure 4.20: Cylinder sub-object primitive scaled up in the z and down in the x and y axes

Be sure to save your progress at regular intervals, especially at points when losing your progress would be devastating. From here, we will begin modeling the remaining large forms of our pedestal using several essential poly modeling tools and techniques. First, let's finish the forms at the top of the pedestal.

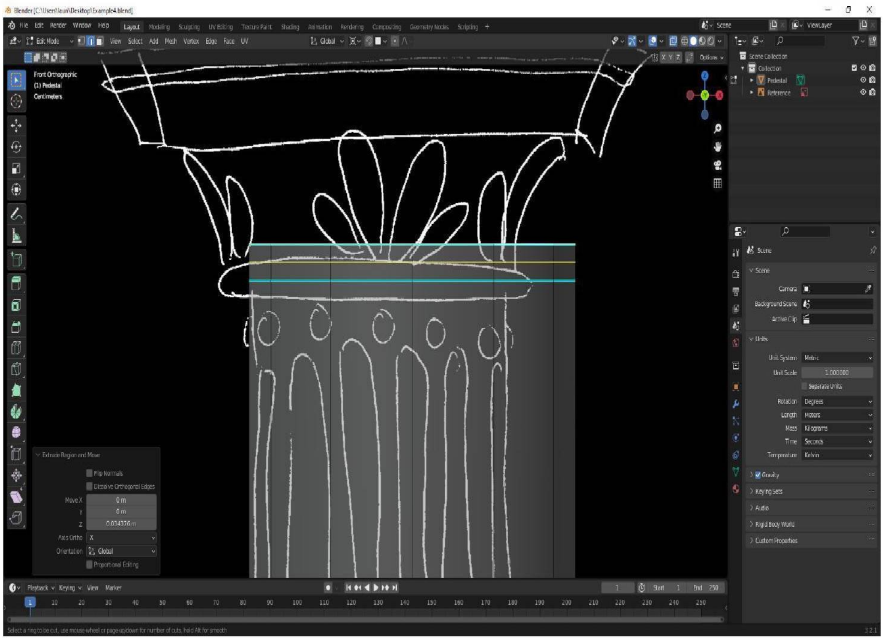
Note: The remaining steps will be shortened, based on the assumption that we are now familiar with techniques we have practiced multiple times previously, such as zooming, orbiting, panning, clearing selections, and making sure the mouse cursor is positioned correctly for contextual commands.

In the reference, at the top of the cylindrical form is a short, rounded extrusion. There are many ways to approach creating any given shape, but here we will begin with the most common.

8. To make a space from which to model the rounded extrusion, lets first select just the top edge of the cylinder by first switching to **Edge select mode** (2 key), then pressing **Alt** + mouse clicking an edge in the loop, and extrude it upward a small amount by pressing **E + Z**.

Note: If we accidentally press E and confirm the extrusion with a left-click without having moved the mouse cursor, the extrusion will have occurred, but with no visible indication. If we're ever unsure whether we confirmed an extrusion or not, it is wise to Undo a few steps or check the Undo History. Otherwise, invisible doubled geometry will remain which can cause unexpected results, such as selections that will inexplicably fail. Mistakes invariably happen, so we will also discuss helpful mesh checks and cleanup in Chapter 11, Using 3D Objects in Production.

9. At this point, it may help to switch back to X-Ray mode from the upper right 3D Viewport menu, and Front Orthographic view, by pressing the number pad period key. With the new loop of faces – referred to as a "face loop" – framed in the view, we'll divide the faces in half horizontally once by inserting a new horizontal edge loop through the middle. Press **Ctrl + R** and hover the mouse cursor over one of the vertical edges in between the top edge loop and the and second edge loop from the top, until a yellow horizontal line appears as shown in *Figure 4.21*. Left-click to confirm:


Blender [C:\Users\lauri\Desktop\Example4.blend]

Front Orthographic
(1) Pedestal
Centimeters

Extrude Region and Move
Flip Normals
Dissolve Orthogonal Edges
Move X: 0 m
Y: 0 m
Z: 0.034376 m
Axis Ortho: X
Orientation: Global
Proportional Editing

Select a ring to be cut, use mouse-wheel or page-up/down for number of cuts, hold Alt for smooth

Figure 4.21: Newly inserted edge loop via Ctrl + R

We have just performed our first loop cut. In the left-hand Tool bar (opened and closed with the T key), mid-way down, there is an icon appearing as a cube sliced vertically by a blue line which also activates the loop cut tool. For now, though we'll leave our tool set to Select Box and continue using hotkeys.

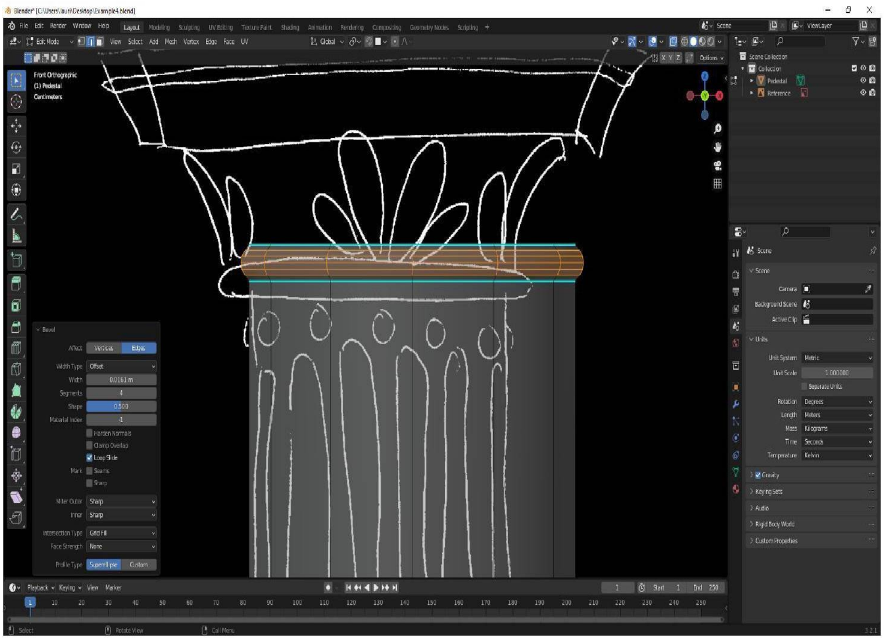
Tip: As with the Select tool and all other Tool bar icons with a small triangle in the lower right corner, if we click and hold this icon, additional options will appear.

With this new edge loop still selected, press **S** to scale it outward a small amount to resemble the reference. If we haven't changed the Transform Pivot Point, this scale will still be originating from the 3D Cursor, which is an undesirable result, so press **Shift** + **Z** to exclude scaling up or down and left-click to confirm.

Now is a good time to set the Transform Pivot Point back to Median Point, or Bounding Box Center. All Transform Pivot Points are calculated from the current selection.

In one more step, we will round the extrusion to better match the reference.

10. With the new edge still selected, press **Ctrl** + **B**, and then move the mouse cursor away from the cylinder to increase the bevel amount. Before confirming, roll the mouse wheel 3 clicks forward, which will increase the number of segments added to 4. This will provide the 6 minimum sides to make our curve appear rounded. Then continue moving the mouse toward or away from the cylinder as needed, until the extrusion is rounded and the segments are evenly spaced as shown in *Figure 4.22*. Left-click to confirm:

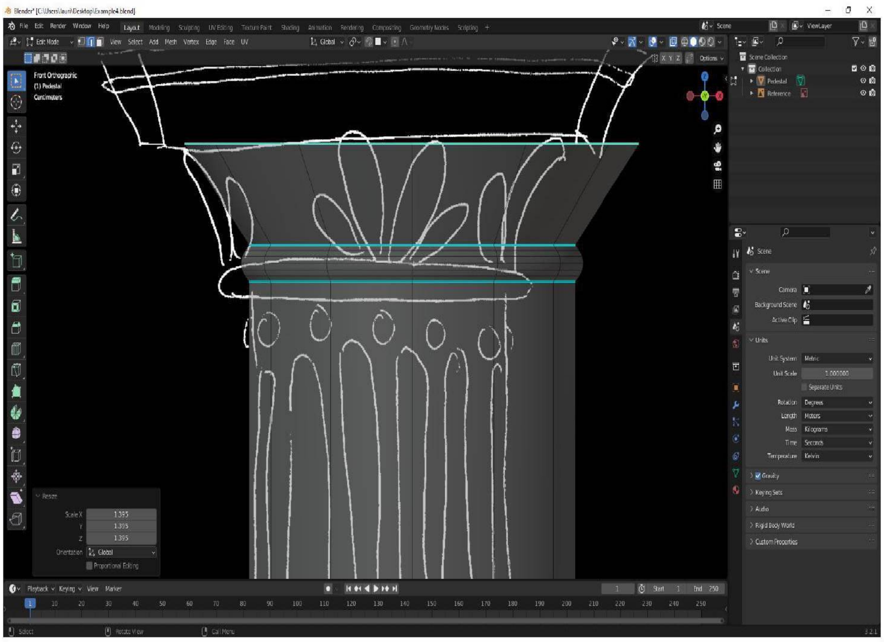


Bevel
Affect: Vertices [✓ Edges]
Width Type: Offset
Width: 0.0161 m
Segments: 4
Shape: 0.500
Material Index: -1
☐ Harden Normals
☐ Clamp Overlap
☑ Loop Slide
Mark: ☐ Seams
☐ Sharp
Miter Outer: Sharp
Inner: Sharp
Intersection Type: Grid Fill
Face Strength: None
Profile Type: [✓ Superellipse] Custom

Figure 4.22: Beveling the inserted, outward scaled edge loop, to create a rounded extrusion

The width and number of segments can then be adjusted to better match the reference if needed. Next, we'll perform the same operation in reverse, to achieve the rounded intrusion underneath the squared pedestal top.

11. Select the top-most edge of the cylinder, extrude it upward in the z axis, confirm the extrusion, and this time, press **S** to scale the new edge outward in the x and y axes as shown in *Figure 4.23*:



Resize

Scale X	1.395
Y	1.395
Z	1.395
Orientation	Global
Proportional Editing	

Figure 4.23: *Extruded edge loop, scaled outward*

12. Once again, insert a new edge loop in the middle of the new faces and confirm. This time, scale the new edge inward, to approximate the inner curve, and bevel again. The same settings that were used before will be reapplied as shown in *Figure 4.24:*

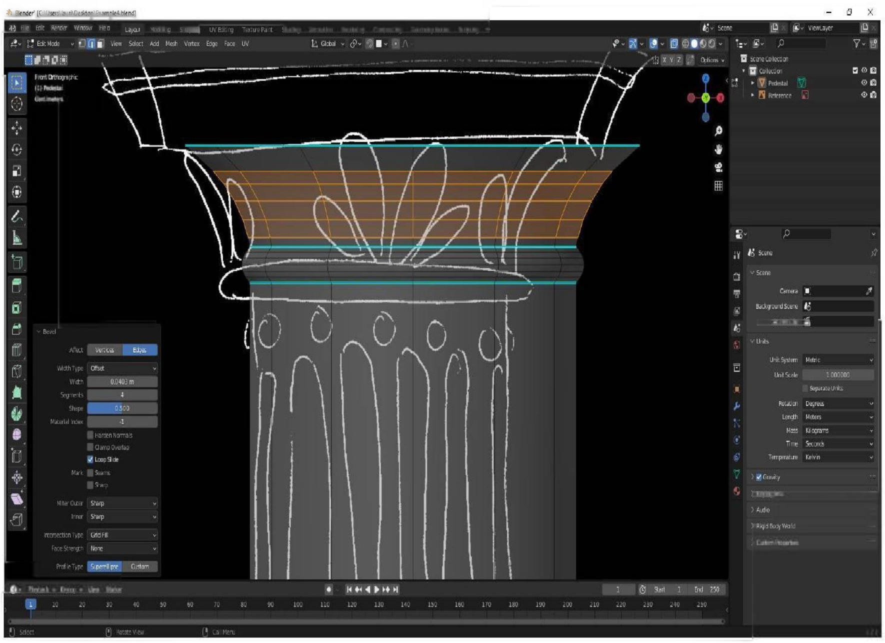

Bevel
Affect: Vertices [✓ Edges]
Width Type: Offset
Width: 0.0403 m
Segments: 4
Shape: 0.500
Material Index: -1
☐ Harden Normals
☐ Clamp Overlap
☑ Loop Slide
Mark: ☐ Seams
☐ Sharp
Miter Outer: Sharp
Inner: Sharp
Intersection Type: Grid Fill
Face Strength: None
Profile Type: [✓ Superellipse] Custom

Figure 4.24: Beveling the inserted, inward scaled edge loop, to create a rounded intrusion

Unfortunately, the top face loop is now taller than the bottom face loop of this curved section of the cylinder.

13. To correct this, from the Bevel contextual menu, change the **Width Type** to **Percent**, and increase the **Width Percent** to something like 70% as shown in *Figure 4.25*:

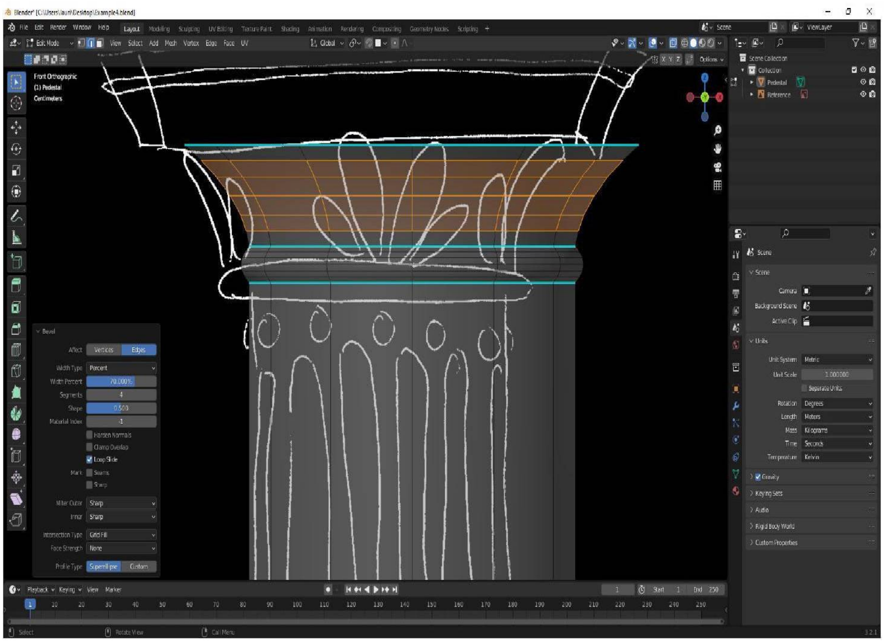

Bevel
Affect: Vertices [✓ Edges]
Width Type: Percent
Width Percent: 70.000%
Segments: 4
Shape: 0.500
Material Index: -1
☐ Harden Normals
☐ Clamp Overlap
☑ Loop Slide
Mark: ☐ Seams
☐ Sharp
Miter Outer: Sharp
Inner: Sharp
Intersection Type: Grid Fill
Face Strength: None
Profile Type: [✓ Superellipse] Custom

Figure 4.25: Adjusting the placement of the new bevel by changing the Width Type

Now that we have some medium level details started, let's be sure to save, and take a more objective look at our progress by disabling **X-Ray mode** and exiting **Edit Mode** for a moment. Orbit the view around the rudimentary pedestal and evaluate.

Before we begin refining the medium level details any further, lets finish the larger forms by capping off the pedestal with the remaining rectangular shape. Since our pedestal's top and bottom would likely be created in the same dimensions in real life, let's use the existing base to our advantage.

14. Back in Edit Mode, in **Face select mode** (3 key) select the top face of the cube shape, press **Shift + D** to create a duplicate face, and move the mouse around. Before left-clicking to confirm, press **Z** to constrain movement in the z axis, and position the new face where the top of the pedestal will be.

    Next, let's adjust the snapping settings from earlier to snap the new face to the top edge of the cylindrical shape.

15. Under the Snap To icon in the upper middle menu of the 3D Viewport, to the right of the icon appearing as a magnet, select **Edge** as the new target. Then press **E**, then **Z** to extrude this face downward (press **Z** again if the blue line showing that transforms are constrained to the z axis doesn't appear), press and hold **Ctrl** to activate snapping, and left-click to confirm when the new face touches the top edge of the cylinder.

    It might help to return to **X-Ray mode** with **Alt + Z**, and Front Orthographic view with the number pad 1 key. By comparing our result to the reference, we can see that the top face should be bigger to create an angle.

16. In Face select mode, box-select just the top face by pressing the **B** key, then left-clicking and dragging a rectangular selection around the top of the top cube. When just the top face is highlighted brighter than the rest of the cube, scale it outward to match the reference.

    The smaller rounded extrusion can be added here in much the same manner as before, but with two additional steps.

17. Insert a new horizontal edge loop with **Ctrl + R**, and this time, when we left-click, let's then move the mouse upward, causing the new loop to slide along the vertical edges. We want to left-click again to confirm when the new loop is in the middle of the upper third of the pedestal cap where the reference extrusion is drawn, as shown in *Figure 4.26*:

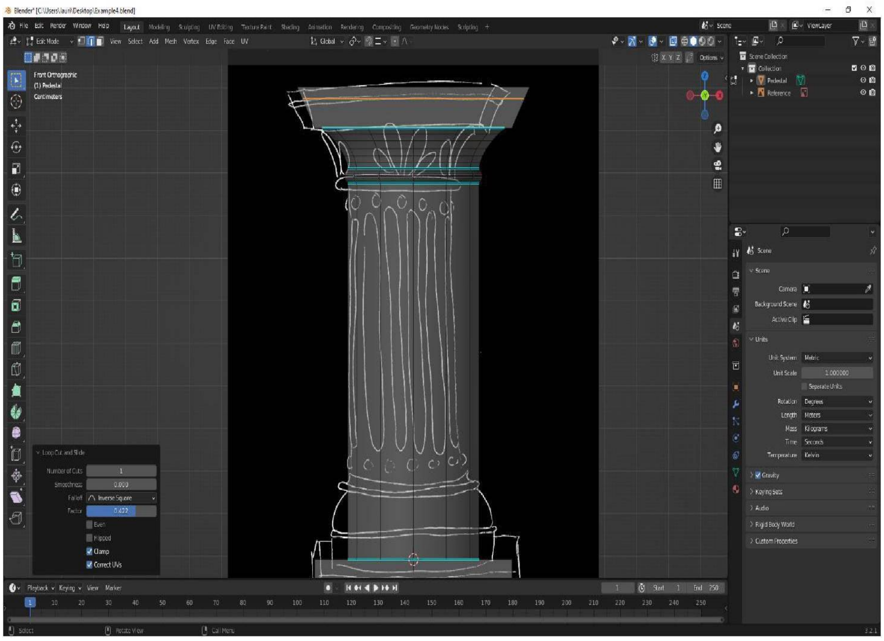

Loop Cut and Slide
Number of Cuts: 1
Smoothness: 0.000
Falloff: Inverse Square
Factor: 0.422
☐ Even
☐ Flipped
☑ Clamp
☑ Correct UVs

Figure 4.26: Adding an edge loop from which to create a second rounded extrusion

In the future, if we accidentally confirm a loop cut position too soon, we can still **Alt + left-click** an edge in the loop, then press **G** twice in succession (**G + G**), to re-activate "edge slide" behavior, but this time let's undo with **Ctrl + Z**, and redo the operation if needed, so that we can use one more setting in the Loop Cut and Slide contextual menu.

18. The top setting in the Loop Cut and Slide contextual menu is the number of cuts. Feel free to try different values to see what happens, but to match the reference we'll want to change this to 2.

    Now, we can follow the same steps we performed earlier, to create a new middle edge loop, scale it out, and then bevel it round. If we zoom out, we can also see that the larger cylindrical form above the base and the rounded extrusions and intrusions just above that are missing, so let's add those now.

19. First, let's add a horizontal loop cut to the main section of the cylindrical shape, and while it is still active, slide it down, to just above the first rounded intrusion.

20. Below that, add another loop cut and slide it up, for the bottom of that intrusion.

21. Below that, add one more loop cut for the bottom of the small extrusion, so that there are two stacked face loops, of approximately the same size.

22. Follow the same steps as done earlier to create a rounded intrusion and

extrusion to match the reference.

For the last two large forms and the small, rounded extrusion between them, we'll use similar approaches.

23. First, we'll add a loop cut to the middle of the bottom section.

24. Next, in Face Select mode, we'll select the bottom-most face loop by **Alt + left**-clicking on a vertical edge within the loop. Then, we'll scale the bottom face loop outward without changing its height by pressing **S**, then **Shift + Z**.

25. Insert a new horizontal edge loop in this bottom-most section, and slide it up, leaving space for the small, rounded extrusion above.

26. Insert another edge loop in the middle, scale it outward, and bevel it round. Repeat the process in the remaining section to create the large dome shape, and we're done.

Updating shading

If we check the pedestal in **Object mode** with X-Ray turned off, we'll see some unwanted shading going on. This is partially because we added brand new geometry which has no shading information applied.

1. So, let's make sure the Pedestal object is selected, then re-apply Shade Smooth, via the upper left Object menu.

   Back in **Edit Mode**, we can see that only the edges that were extruded from sharp edges are still marked sharp, as indicated by the light blue edge marking.

2. To correct this, we need to select every edge that we want to be sharpened and mark them all sharp via **Ctrl + E | Mark Sharp**.

   Fortunately, selecting entire edge loops with **Alt + Click** makes quicker work of this task, but the top and bottom edge loops of the base and cap pieces can't be selected this way, because a cube is essentially two four-sided face loops joined perpendicular to each other, so that the loop selection operation runs into a corner – also called a "pole" – and doesn't know which way to go.

3. For the pedestal base, we can get around this by selecting the whole cube with **Shift L**, since we also want its side edges marked sharp.

   For the cap piece though, we'll need a more targeted approach.

4. While in **Edge select** mode, we can loop-select each corner and mark them sharp one at a time.

5. To sharpen the top and bottom edges of the cap, it will be easier to switch to **Face select mode** via the 3 key, then select the top face, hold Shift to add to the selection, then left-click the bottom face, then still holding **Shift**, select the top and bottom face loops above and below the rounded extrusion by also pressing **Alt**, and clicking on a vertical edge along the face loop.

The needed edge marks will for our current progress will look something like the light blue lines shown in *Figure 4.27*. From here on out, we'll try to remember to add sharp markings as we create new geometry and reapply Shade Smooth as needed:

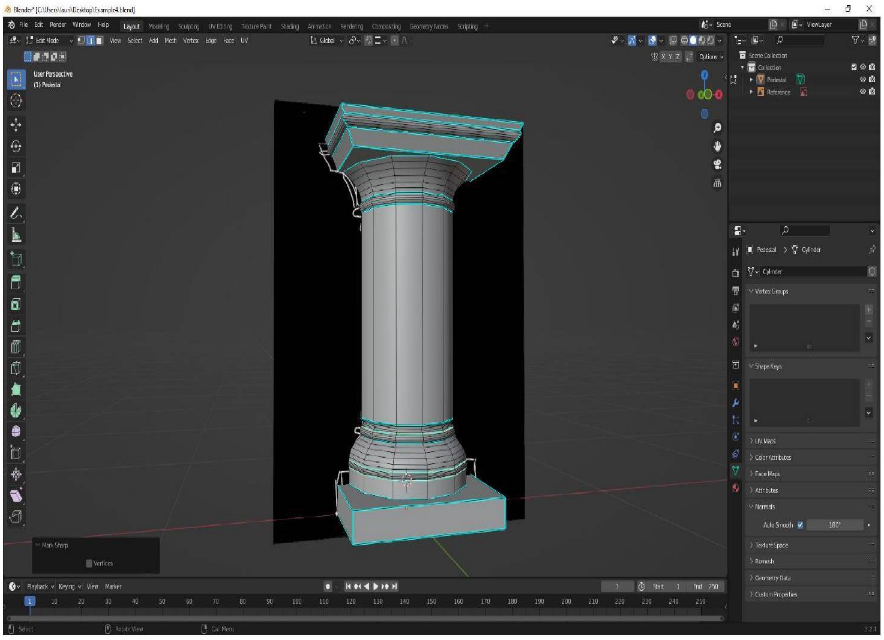


Figure 4.27: *Large forms complete, and all sharp edges marked, with smooth shading reapplied*

Now is a good time to save. With the largest shapes roughed in, this is also a good point to adjust the proportions of anything not quite to our liking.

Tip: One of the easiest ways to adjust entire sections at once is, from X-Ray mode, switching to any Orthographic perspective (number pad key 1 - front, 3 - right, or 7 - top, or the inverse with Ctrl) and Vertex select mode (1), then Box Selecting layers of the mesh by pressing B and dragging a box that encompasses all the vertices to be moved, and then moving the selected vertices up or down. Pressing B to activate Box select, and then pressing and holding the middle mouse button will draw a deselect box, removing vertices from the current selection.

In *Figure 4.28*, we've selected and moved just the lower rows of intrusions and extrusions to make the cylindrical base a bit taller:

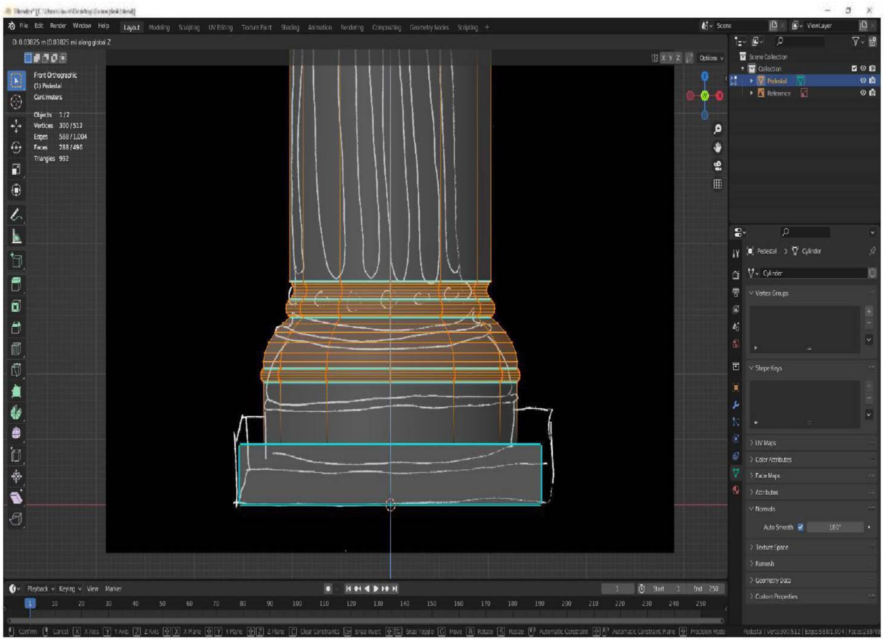


Figure 4.28: Large forms

Combining, reusing, and repeating shapes

The idiom, "don't reinvent the wheel" is particularly good advice for 3D creators. It means don't waste your own effort by creating something new when it already exists. This is the same principle behind starting from primitive objects, but we can also apply it to objects and sub-object components which we have created or edited ourselves.

The final touches we will add to our pedestal before moving on to sculpting high resolution details in *Chapter 6, 3D Sculpting* are fluting along the core column, and a radial pattern of half-spheres above and below the fluting.

1. We'll start by switching to **Object Mode** and – because new objects are added to the scene at the location of the 3D Cursor – ensuring that our 3D Cursor is located at word center, by pressing **Shift + S** and choosing **1**, Cursor to World Origin.

2. Next, we'll add a quad sphere to our scene via **Shift + A** | **Mesh** | **Round Cube**. If you've recently upgraded Blender, you may need to re-enable this option under **Edit** | **Preferences** | **Addons** | search for "**extra**" and enable **Add Mesh Extra Objects**. The Round Cube isn't very spherical yet by default, so in the **Add Round Cube** contextual menu, set the Radius to **1**, and below the **Size**,

increase the **Arc number** to **6**. Our new sphere is too large, but we can scale it down by eye in a moment.

Note: The difference between UV Spheres, Ico Spheres, and Round Cubes is their construction. UV Spheres are essentially cylinders subdivided horizontally, and cast into the shape of a sphere, with the end poles pinched into triangles causing an uneven distribution of mostly quad faces. Ico Spheres are made entirely of triangles. Round Cubes are subdivided cubes cast into the shape of a sphere and are constructed entirely of quads.

3. Next, we'll move our new sphere up in the world so that the center (origin point, appearing as a small yellow/orange dot) is at the right height for the bottom row of half-spheres in the drawing by pressing G, then Z.
4. Next, scale the sphere down, to just a little bigger than the hemi-spheres in the sketch. It may be easiest to do this in two or more steps, scaling close, confirming, zooming in, and scaling again if needed.
5. With the sphere still selected in **Object mode**, under the **Object menu**, **Shade Smooth**, and in the **Object Data Properties** tab of the **Properties** panel under Normals, enable **Auto Smooth** and increase the amount to **180**.
6. In **Edit Mode**, with X-Ray mode still enabled, switch to Right Orthographic view by pressing the number pad **3** key. Switch to **Face select** mode, clear selection, and Box select the right half of the sphere as shown in *Figure 4.29*, then delete the selected faces.

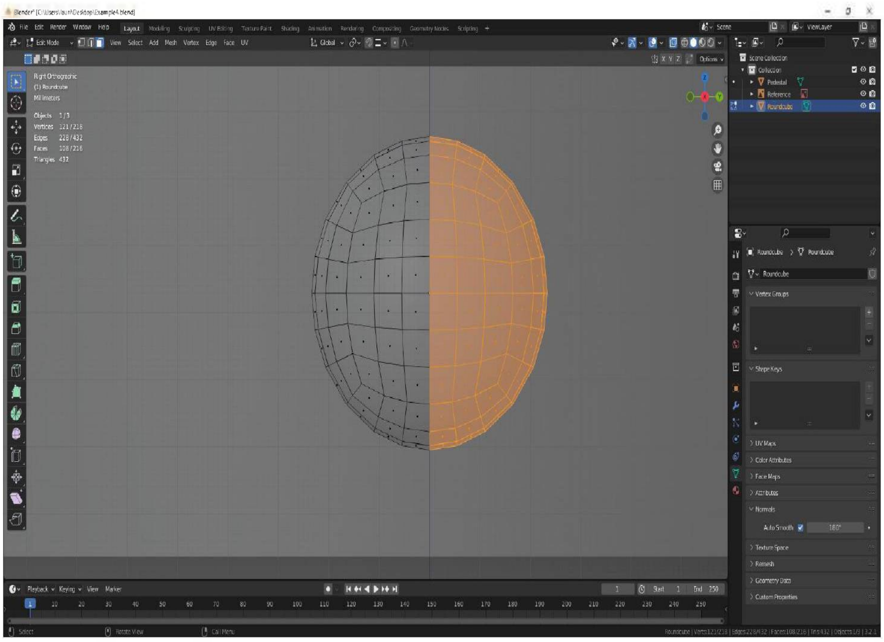


Figure 4.29: Round Cube object, with right half selected in Right Orthographic view mode

If we enable scene statistics in the bottom-most right menu or from Viewport Overlays as shown previously in this chapter, and then select a middle edge loop of this sphere, we will see that the number of segments remaining is 12 in both directions. Since this is only half a sphere, it doesn't require all 12 subdivisions to appear round, so it would be wise to optimize this mesh before repeating it around the pedestal by removing every other edge.

7. To do this, first switch to Front Orthographic view and **Edge Select** mode, then ignoring the boundary loop on the open edge, loop-select every other interior edge loop (including the "corner" edges) by **Alt + clicking** each loop, as shown in *Figure 4.30*. Then, press **Delete** or X, and this time choose **Edge Loops**.

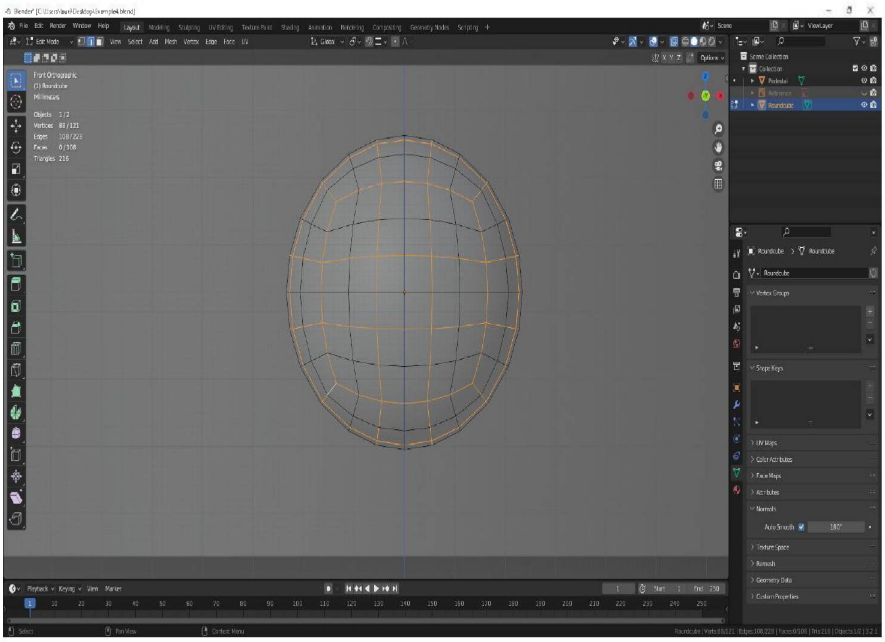


Figure 4.30: *Every other edge selected for removal*

Now is a good time to save. Next, we'll snap the open edge of this hemisphere onto the surface of our pedestal.

8. Back in Right Orthographic **view mode** (number pad **3** key) zoom out enough to see the surface of the pedestal. Clear the selection, and from **Edge select mode**, **Alt + click** to select the open/flat edge loop, then press **L** to select the whole hemisphere at once.

9. With **Snap To** still set to **Edge**, type **G**, then **Y** to constrain movement to the y axis, press and hold **Ctrl**, and move the hemisphere left in the viewport, until it snaps to the surface of the pedestal.

We have successfully created our first "greeble": a detail which helps to break up the uniformity of large surfaces and make a more interesting composition.

If we look closely, there is a small gap on the sides of the hemisphere because we aligned it with the cylinder's front vertical edge rather than a flat surface. One advantage to poly modeling is that we can edit parts of a mesh separate from the rest. An advantage to having begun our pedestal with a 12-sided cylinder is that it can be rotated to have a flat face on the front, back, and sides if needed.

In this case, we will rotate the cylinder 15 degrees, separate from the pedestal's base and cap so that the front and sides are flat, allowing us to close the gap between cylinder and hemisphere.

10. Exit **Edit Mode** from the Roundcube object. With the pedestal object selected in X-Ray mode, lets view the geometry from top-down by pressing number pad key **7**. Enter **Edit Mode** for the Pedestal object and clear any selection.
11. The cylinder shape is visible from the top, and since it is still a separate sub-object, selectable as linked, via the **L** key. With the whole cylinder selected, press R then 15, and this time press the Enter key to confirm.
12. Exit Edit Mode, select the hemisphere (named Roundcube) and enter **Edit Mode** again. In the side view (number pad 3 key), press **G**, then **Y**, then press and hold **Ctrl** while moving the hemisphere back, to snap the hemisphere flush with the front face of the pedestal.

The next step of our process could be to copy and rotate the hemisphere in a radial pattern around the top and bottom of the column, but the fluting pattern happens to match the shape of the sphere inverted, so a smarter move would be to reuse a copy of the hemisphere to make the fluting.

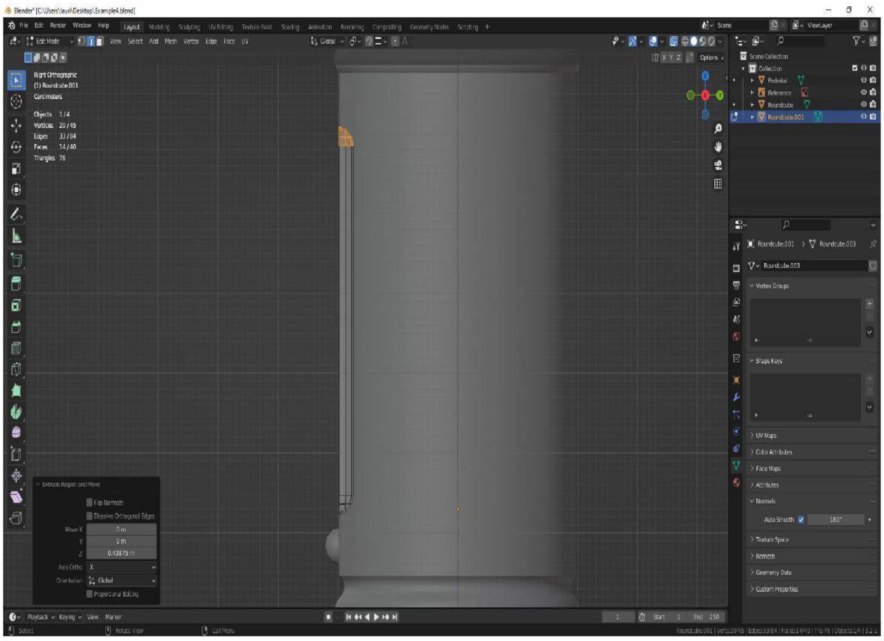
In front Orthographic view, **Object Mode**, in Solid shading view, select the hemisphere and press **Shift + D** to duplicate it, then Z to constrain its movement and place the copy slightly above the original where the fluting starts in the reference sketch.

The fluting appears offset from the hemispheres in our reference, so we'll have to rotate them as we did with the cylinder, but that can wait until we're ready to incorporate the fluting into the cylinder. The first step to creating the fluting is to invert the hemisphere in the y axis.

13. From **Edit Mode** with **X-Ray mode** enabled, select the open edge loop on the back of the hemisphere. Next, press Shift + S, and select Cursor to Selected (2). Link Select the entire hemisphere mesh, change the Transform Pivot Point to 3D Cursor, and click Ctrl + M, then Y, then Enter.

It is a helpful habit to reset the 3D Cursor to World Origin via **Shift + S**, **1**, and switch the Transform Pivot Point back to Median Point or Bounding Box after using the 3D Cursor as a temporary pivot point this way.

14. With the hemisphere inverted, let's enter Face select mode, clear our selection, and then **Box Select (B)** the entire top half of the inverted hemisphere. Zoom out far enough to see the whole core cylinder of the pedestal, and press **E** to extrude. It may be necessary to press **Z** two or three times in a row to constrain movement in the z axis – indicated by a blue vertical line – and move the top of the fluting up, leaving enough space for the planned top row of hemispheres above, as shown in *Figure 4.31*:



Extrude Region and Move
Flip Normals
Dissolve Orthogonal Edges
Move X 0 m
Y 0 m
Z 0.43875 m
Axis Ortho X
Orientation Global
Proportional Editing

***Figure 4.31:** Top half of duplicate hemisphere extruded upward, to form fluting*

15. Orbiting the view, we will see that our extrude operation inadvertently extruded front faces as well as the dome, so in Solid shaded view, let's left-click select these six front faces – four wide and two narrow – and press **Delete** or **X**, and select Faces.

The hemisphere's normals are pointing outward by default, so we'll need to flip them to make their insides outside as intended.

16. Select the whole hemisphere by clearing the selection and pressing **A**, then press **Alt + N** and choose **Flip**. Exit **Edit Mode**.

Now is the time to join the hemispheres to the Pedestal object.

17. From Object Mode, **Ctrl + left-click** the two Roundcube objects, one at a time, then **Ctrl + click** the Pedestal object last to make it Active, then press **Ctrl + J** to join the hemispheres to the pedestal.
18. In Solid shading mode, enter **Edit Mode**. Clear the selection, and then in Face select mode, then **Alt + click** one of the vertical edges of the tall face loop at the center of the pedestal to select the whole face loop.
19. Then, pressing and holding Shift, right-click the face underneath the hemispheres deselecting it and leaving the rest of the pedestal's geometry unselected as shown in *Figure 4.32*:

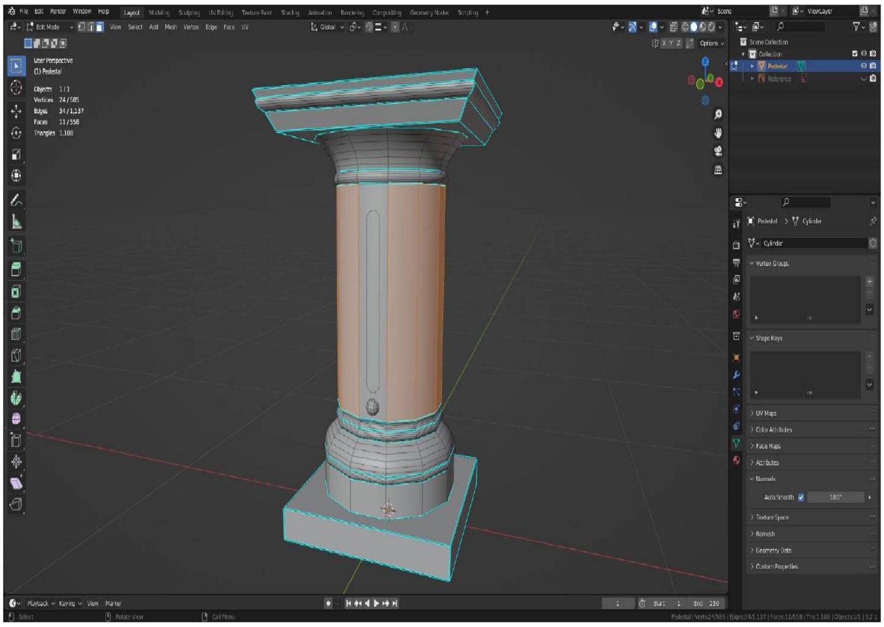


Figure 4.32: *Core face loop of Pedestal selected, except for the center front face*

20. Delete the selected faces; we will replace them in just a moment.
21. Next, select the remaining face underneath the hemispheres, and press **Delete** or **X**, but this time, choose to delete Only Faces, which will leave the bounding edges in place.

22. Finally, we'll mark the open edges of each hemisphere sharp, by first selecting each hemisphere via L, then under the **Select** menu in the upper left menu in the 3D Viewport, choose **Select Loops** | **Select Boundary Loop** as shown in *Figure 4.33*, then **Ctrl + E** | **Mark Sharp**.

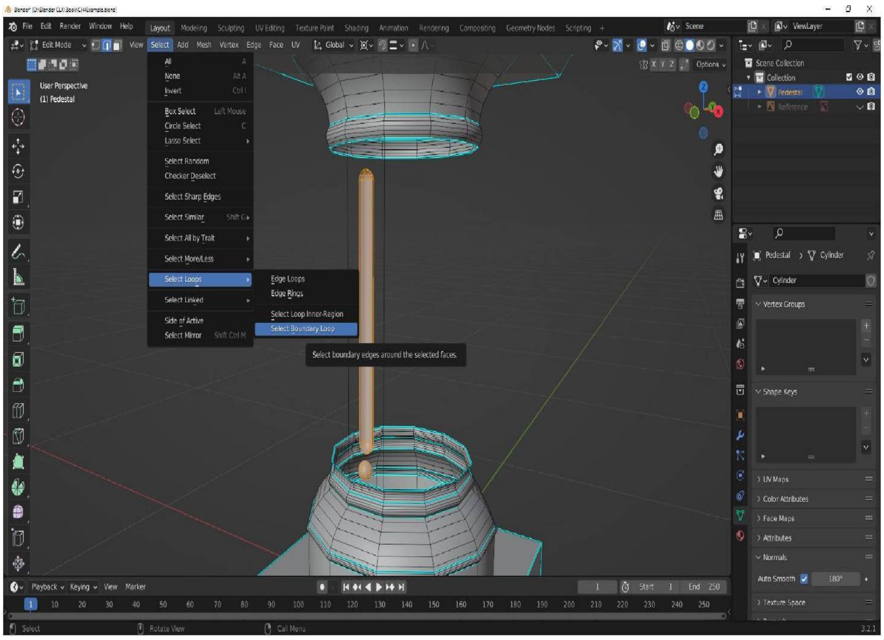


Figure 4.33: Selecting Boundary Loop to mark sharp, via Select | Select Loops | Select Boundary Loop

23. Next, switch to **Edge select** mode, and select two neighboring edges, one from the deleted face of the cylinder, and the other, the closest open edge of the elongated hemisphere. Then press F to fill the space between them with a face as shown in *Figure 4.34*.

24. Repeat the process for the opposite side, selecting the open, elongated edge from the hemisphere and the neighboring edge of the deleted cylinder face, and press F to fill the space between them.

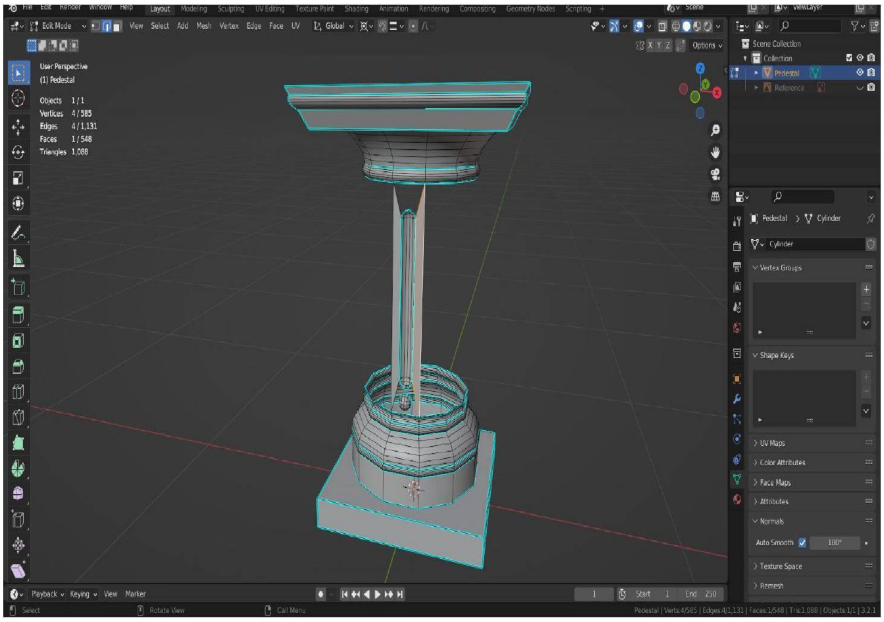


Figure 4.34: New faces bridging the elongated hemisphere and remaining edges of the previously deleted cylinder face

25. For the next step, it will help to zoom in to the remaining space between the top of the elongated hemisphere and the cylinder, then left-click select a single edge at the top of one of the new side faces, then press and hold **Ctrl + left-click** the opposite edge at the top of the other new side face, until the whole edge along the top of the hemisphere is selected, then press **F** to fill the gap as shown in *Figure 4.35*:

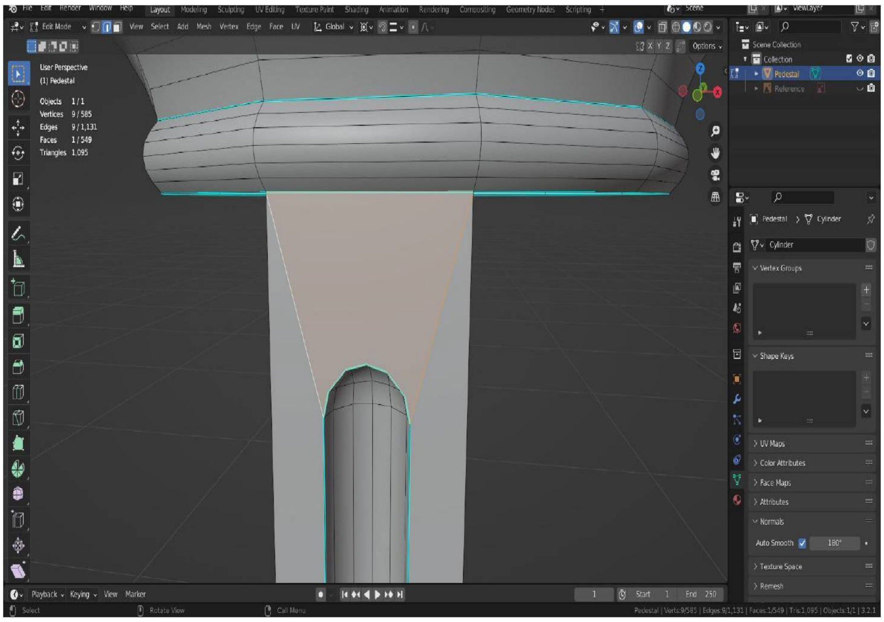


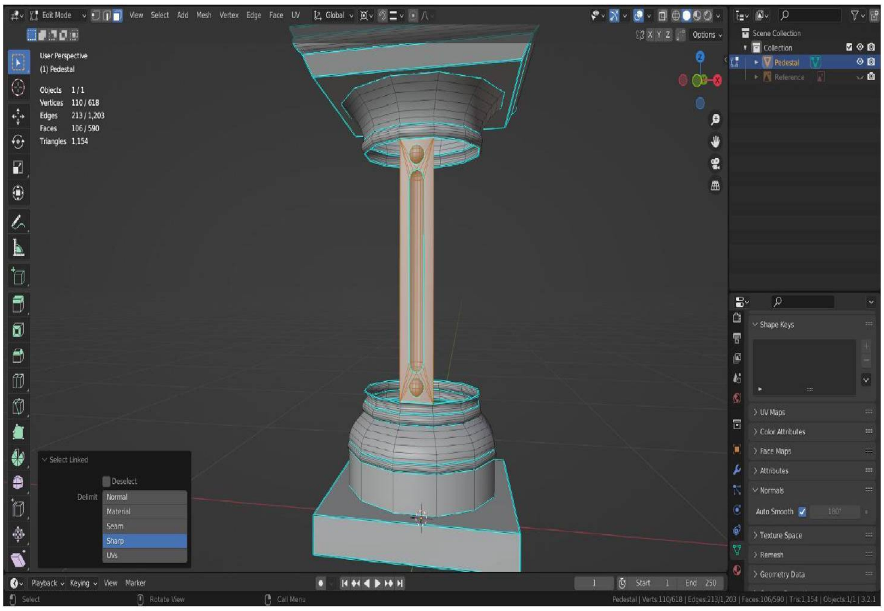
Figure 4.35: Ngon filling the space above the elongated capsule, between two new faces on the sides

We have created a new ngon, which is not a problem, but since we're planning to repeat this newly created fluted face in a radial pattern around the cylinder, it is better to clean it up now than to individually edit twelve sides later.

26. With the new ngon selected, press **Ctrl + T** to triangulate it.
27. the process of filling and triangulating the remaining space at the bottom of the elongated hemisphere in the same way, as shown in *Figure 4.36*.
28. We'll also select and press **Shift + D** to duplicate the small hemisphere detail at the bottom, then use **G + Z** to move it into place above the elongated hemisphere.

    The pattern of hemispheres doesn't quite match the reference yet, but we will offset them in a later step.

29. The final step to prepare this new geometry for repeating around the pedestal's core is to hover the mouse over the unselected hemisphere and press **L** to Select Linked, then change the Delimit method to Sharp, then hover the mouse over the elongated hemisphere, continuing to press **L** over the new surrounding faces until only the fluted face and hemispheres are all selected as shown in *Figure 4.36*:



User Perspective
(1) Pedestal

```
Objects    1 / 1
Vertices   110 / 618
Edges      213 / 1,203
Faces      106 / 590
Triangles  1,154
```

Select Linked
Deselect
Delimit: Normal, Material, Seam, [✓ Sharp], UVs

Figure 4.36: Fluted face and hemispheres selected by Linked, Delimited by Sharp

This is a good time to save. Now that we have a single fluted face and the decorative hemispheres in place, we can duplicate the selection, rotate it 30 degrees on the z axis, and repeat these steps 11 more times, to recreate the pedestal's cylindrical body with detail added.

30. To prepare, we'll verify that the 3D Cursor is set to the world origin via **Shift + S + 1**, then we'll make sure that the Transform Pivot Point is set to 3D Cursor.

31. Next, we'll duplicate the selected geometry by pressing **Shift + D**, and before confirming we'll press **R**, then **Z**, then type **`30`**, and **Enter** to rotate the duplicate geometry 30 degrees in the z axis.

    To repeat a single step, we could use **Shift + R**, but since this process requires two separate steps, we'll need an alternative to quickly repeat them 10 more times in order. Let's make a quick macro, so we'll only need to press one button to perform both steps. First, we should separate the selected geometry, making it a separate object, so that we can record and play rotation around the world origin, rather than the origin of the sub-object components.

32. From **Edit Mode**, with the mouse cursor inside the 3D Viewport and the duplicated geometry still selected, press **P** to open the Separate menu, and choose **Selected**.

33. Exit **Edit Mode**.

34. From the top-most menu in Blender, select the Scripting tab on the far right, to change the Active Workspace.

35. In the Scripting workspace, the 3D Viewport is shrunken, while a new Editor Type: Text Editor fills most of the screen.

36. From the middle of the menu at the top of the Text Editor window, press the **New** button. This creates a new, unsaved text file, with a cursor appearing where text can be entered.

37. In this window, type the code "**`import bpy`**" as shown in the following, then press **Enter** twice, to start a new line:

    ```
    import bpy
    ```

    To record the steps into memory for our macro, let us repeat the preceding process but in Object Mode.

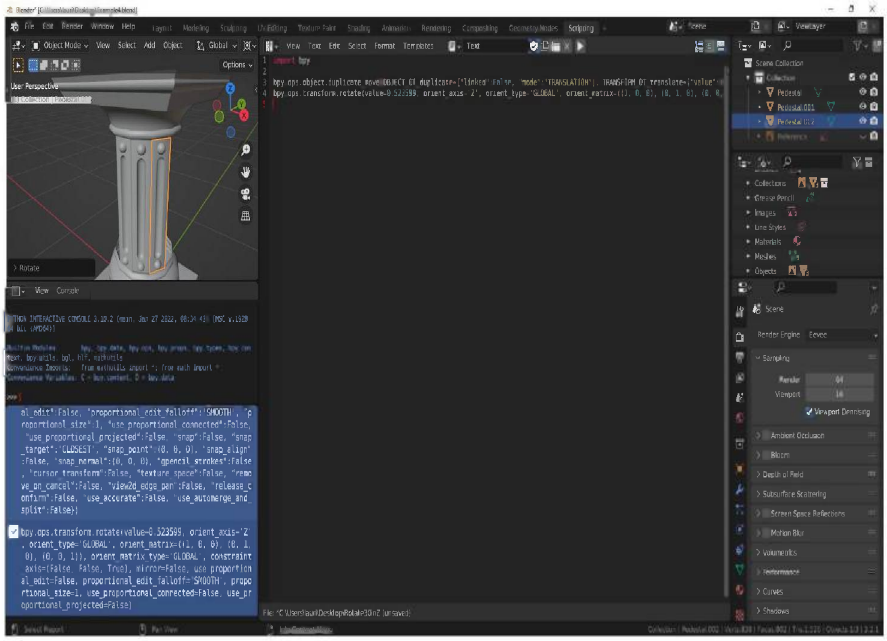
38. First, zoom in to the 3D Viewport to get a better view. Select the new **`Pedestal.001`** object from the Outliner, then with the mouse cursor over the 3D Viewport press **Shift + D** to duplicate, and before confirming press **R**, then **Z**, then type **`30`**, then press **Enter**.

    In the lower left of the Scripting workspace, there is also a Python Console window, showing a recording of the last operations performed in Blender. After following the preceding steps, this window will display two large blocks of code showing all the previous operations in Python script.

39. Scroll up to see both blocks of code, or expand the Python Console window if needed by hovering the mouse cursor just above the text until the double arrow icon appears, then click and drag upward.

    This code may look intimidating, but it is easy to work with. Left-clicking any block will select all the code with it.

40. Press and hold **Shift + left-click** to select both the last, and the previous step in the list as shown in *Figure 4.37*:



```
import bpy

bpy.ops.object.duplicate_move(OBJECT_OT_duplicate={"linked":False, "mode":'TRANSLATION'}, TRANSFORM_OT_translate={"value":
bpy.ops.transform.rotate(value=0.523599, orient_axis='Z', orient_type='GLOBAL', orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0,
```

```
PYTHON INTERACTIVE CONSOLE 3.10.2 (main, Jan 27 2022, 08:34 43) [MSC v.1928
64 bit (AMD64)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.con
text, bpy.utils, bgl, blf, mathutils
Convenience Imports:   from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>>

al_edit":False, "proportional_edit_falloff":'SMOOTH', "p
roportional_size":1, "use_proportional_connected":False,
 "use_proportional_projected":False, "snap":False, "snap
_target":'CLOSEST', "snap_point":(0, 0, 0), "snap_align"
:False, "snap_normal":(0, 0, 0), "gpencil_strokes":False
, "cursor_transform":False, "texture_space":False, "remo
ve_on_cancel":False, "view2d_edge_pan":False, "release_c
onfirm":False, "use_accurate":False, "use_automerge_and_
split":False})

bpy.ops.transform.rotate(value=0.523599, orient_axis='Z'
, orient_type='GLOBAL', orient_matrix=((1, 0, 0), (0, 1,
 0), (0, 0, 1)), orient_matrix_type='GLOBAL', constraint
_axis=(False, False, True), mirror=False, use_proportion
al_edit=False, proportional_edit_falloff='SMOOTH', propo
rtional_size=1, use_proportional_connected=False, use_pr
oportional_projected=False)
```

Figure 4.37: Previous two steps recorded as blocks of code, highlighted, copied, and pasted into a new Text file

41. Right-click the selected blocks of code and choose `Copy`, then paste the two blocks of code into the new Text file to the right, underneath the first snippet of code.

    For your reference, the code to copy and paste is as follows:

```
bpy.ops.object.duplicate_move(OBJECT_OT_
duplicate={"linked":False, "mode":'TRANSLATION'}, TRANSFORM_OT_
translate={"value":(0, 0, 0), "orient_axis_ortho":'X', "orient_
type":'GLOBAL', "orient_matrix":((0, 0, 0), (0, 0, 0), (0, 0,
0)), "orient_matrix_type":'GLOBAL', "constraint_axis":(False,
False, False), "mirror":False, "use_proportional_edit":False,
"proportional_edit_falloff":'SMOOTH', "proportional_size":1,
"use_proportional_connected":False, "use_proportional_
projected":False, "snap":False, "snap_target":'CLOSEST', "snap
```

```
projected":False, "snap":False, "snap_target":'CLOSEST', "snap_
point":(0, 0, 0), "snap_align":False, "snap_normal":(0, 0, 0),
"gpencil_strokes":False, "cursor_transform":False, "texture_
space":False, "remove_on_cancel":False, "view2d_edge_pan":False,
```

```
"release_confirm":False, "use_accurate":False, "use_automerge_and_
split":False})
```

```
bpy.ops.transform.rotate(value=0.523599, orient_axis='Z', orient_
type='GLOBAL', orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)),
orient_matrix_type='GLOBAL', constraint_axis=(False, False,
True), mirror=False, use_proportional_edit=False, proportional_
edit_falloff='SMOOTH', proportional_size=1, use_proportional_
connected=False, use_proportional_projected=False)
```

With the code pasted into the Text file as shown in *Figure 4.37*, all that is left to do to repeat the operation is press the large right-pointing triangle icon at the top middle of the Text Editor viewport, appearing as a Play button.

42. Press **Play** as many times as necessary and orbit the view in the 3D Viewport as needed, until the cylinder is complete.

This macro can be saved to your hard drive via the menu option **Text | Save As** in the upper left of the Text Editor, and reopened later for reuse, but we won't need it again for this exercise.

Merging loose geometry

Now, we will follow the following steps further:

1. Return to the Layout workspace from the top-most workspace tab group.
2. Now we'll join the objects we just created back to the Pedestal object, by first selecting the top-level Pedestal object from the Outliner (expanding the Outliner window downward as needed), then **Shift** + selecting all the new Pedestal objects (appended with .001, .002, etc.)
3. Press **Ctrl** + **J** to join.

   While the new objects are now joined with the old, just as with the cube and cylinder, they are still separate "islands" of geometry. To connect them back to the cylindrical body, we will merge all the vertices that now occupy the same space.

4. In **Edit Mode**, clear any existing selection, then select All by pressing **A**. Press **M** to bring up the **Merge** menu and select **By Distance**.

For this object, the default settings will work just fine, but it is helpful to know that the contextual menu for Merge offers a Merge Distance setting which can be adjusted as needed. After merging by distance, a success message appears in the bottom middle of the Blender application confirming that 44 vertices have been removed, one for every vertex that was overlapping.

Note: Blue confirmations, yellow warnings, and red errors will all appear at the bottom middle of the Blender application during and/or after many operations. In-progress operations can be cancelled by pressing the grey X appearing to the right of a loading bar, while warnings and errors will automatically fade after a short time. Clicking on the text of warnings and errors brings up the Info Editor window, with a log that may help in diagnosing unexpected issues.

It is helpful to note that the default merge distance is set to 0.0001m. This is a very small amount, relative to our 1.2-meter pedestal, which is why it successfully merged only vertices occupying shared space within this small distance from one another. If the Merge Distance were set to 1, most of this 1.2-meter pedestal's vertices would merge. Operations like merge are calculated relative to the object's unit scale, which is another reason to model 3D objects to real world dimensions, ensuring predictable behavior.

Advanced selections

Now that we have the larger forms of our pedestal complete, we'll circle back to a detail that we put off earlier: offsetting the placement of the hemispheres. Because of the simplicity of our object so far, to select just the hemispheres and nothing else, we can work backwards by selecting the cap and base of the pedestal as well as the cylinder and hide them via the **H** key. The remaining hemispheres could then easily be selected via the **A** key.

But what if our object were more complicated? A 3D model of a vehicle might have hundreds of similar shapes in different locations and orientations, such as nuts and bolts; and we might need to perform a single operation on multiple, to give them all a metallic texture, for example. This is a great opportunity to explore more advanced methods of selection. Within the Select menu at the top left of the 3D Viewport, we will find multiple selection tools in addition to those we have already employed, such as Box, Circle, Lasso, Linked, and Boundary Loop. The section we will explore next is Select Similar, which can also be invoked by pressing Shift + G.

1. Every Select Similar option requires an initial selection to compare with, so first, we'll select one of the hemispheres via Select Linked.

   Select Similar also presents different options and produces different behaviors depending on whether **Vertex select**, **Edge select**, or **Face select mode** is currently enabled because some object data is not available to all sub-object components. Materials are not assigned to vertices or edges, for example.

Choose **Face select mode** via the **3** key, then press **Shift + G** and choose the first option from the top, Material.

At this point, our model doesn't have any Materials, so nothing happens; but if it did, the selection would extend to all the faces with the same Material applied as the original selection. Materials are containers for textures that can be applied to the surface of 3D objects to give them color and other surface properties like roughness, or a glow. We will delve further into Materials, Textures, and other surface treatments in *Chapter 7, 3D Surfaces*, but for now it is good to know that one way to select similar objects is by selecting those that have the same surface treatment.

2. From the Select Similar contextual menu that appears in the lower left corner of the 3D Viewport, change the Type from Material to Area, and notice how the selection has changed.

   Area refers to the mathematical formula measuring the total size of the polygon. Notice also, in the **Select Similar** menu, the two available settings for the **Area Type** are **Compare** and **Threshold**. The Compare options are **Equal by default**, **Greater**, or **Less**. When comparing to Equal, only polygons of the identical area will be added to the selection, Greater will select polygons with a larger area, and so forth. Much like the distance setting in the **Merge** menu, the Threshold setting is relative to the current scene's unit scale: meters by default.

   Feel free to switch between any of the available options, to see the resulting selections. If needed, we may cancel the operation, reselect the initial selection, and invoke **Select Similar** again.

3. Eventually through experimentation, we will find that **Select Similar** Type: **Perimeter | Equal** at the default Threshold of 0.01000 provides the closest result to hemispheres only, although the top and bottom of elongated hemispheres are also selected.

   This is to be expected because the fluting was made from the hemispheres, extruded at the center.

4. Fortunately, because we bridged edges of the original cylinder to the elongated hemisphere, while the original hemispheres remain separate islands of geometry, we can use the hotkey combination **Ctrl + number** pad minus sign twice, to make the selection decrease, or retreat from the boundary of only the sub-object components with neighboring geometry.

   Alternately, we could deselect the column by hovering the mouse cursor over it, then pressing **Shift + L** to deselect linked. It will come in handy to know multiple ways to achieve the same result when faced with very complicated selections in the future. Regardless of how we achieve the selection of the hemispheres, we'll wrap up this stage of 3D creation by rotating the selected hemispheres by 15 degrees, so they are offset from the fluting, and then we'll adjust their sale to better match the reference.

5. With only hemispheres selected, and either the 3D Cursor, Median Point, or Bounding Box Center set as the Transform Pivot Point, simply press R, then Z, then 15, then Enter.

Either of these three Transform Pivot Points works, because the selection is equidistant from its midpoint.

You may observe that parts of the hemispheres are now extending beyond the surface of the fluted faces into the interior of the pedestal. This is because earlier, we moved the hemispheres closer to the world center, after rotating the cylindrical shape. For our purposes this is acceptable. So long as unwanted gaps between geometry are closed, intersecting geometry is not a problem in most 3D creation situations. In *Chapter 6, 3D Sculpting*, we'll discuss instances where this may not be the case, such as for 3D printing, where designs must meet a standard called "watertight," meaning that if the object were filled with fluid, none would leak out of gaps, in which case every island of geometry would need to be enclosed.

Checking for, and correcting common issues

With as many steps as may be involved in any 3D modeling process, it isn't difficult to lose track of actions that could cause a cascade of unintended, unwanted effects. Occasionally, as through the course of this exercise, we may need to retrace and redo various steps, which can result in a variety of unexpected behaviors like unwanted shading artifacts, inconsistencies between the scale and orientation of sub-object components, misplaced origins, and the like.

Before moving on to modeling and sculpting smaller decorative details, we'll apply all the transforms we may have performed on the object so far.

1. Press **Tab** to exit **Edit Mode**, then with the Pedestal selected, press **Ctrl + A**, then choose All Transforms.

   Building on concepts introduced in *Chapter 3, General 3D Concepts*, applying transforms is an important way to reconcile any conflicts that might arise from long sequences of actions. One issue that applying transforms can reveal is whether we have inadvertently created inverted faces. With larger forms roughed in, this is also a good time to check that all our object's normals are facing outside.

2. Refer to *Chapter 3, General Concepts* for a reminder of alternate ways to visualize an object's normals, but for this exercise, check via the Viewport Overlays menu in the upper right of the 3D Viewport and select Face

Orientation from near the bottom of the menu.

3. If, after applying all transforms, any normals remain inverted as shown in *Figure 4.38*, re-enter **Edit Mode**, unhide all via **Alt + H**, clear selection, select all via **A**, then press **Alt + N** and choose to Recalculate Outside.

4. Occasionally, these steps do not fix all inverted normals but rather flip new ones the wrong way, in which case selecting just the red, inverted faces then pressing **Alt + N** and choosing **Flip** is necessary.

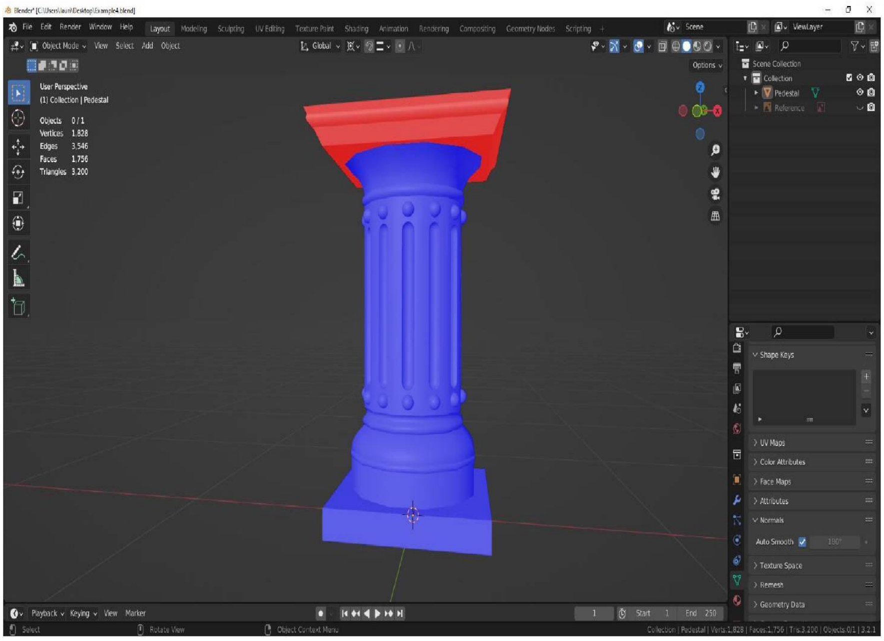


***Figure 4.38:** Pedestal with Face Normals overlay enabled, showing two sections with inverted normals*

Those who have been following along with the reference provided should now have a pedestal very similar to the example shown in *Figure 4.39*. It is plainly visible that the unstructured feel of the hand-drawn reference sketch has been lost in translation. Rest assured though, we will be able to interject and overlay much more interesting, organic detail in the following chapter, even to this thematically hard surface object. Please refer to the following figure:

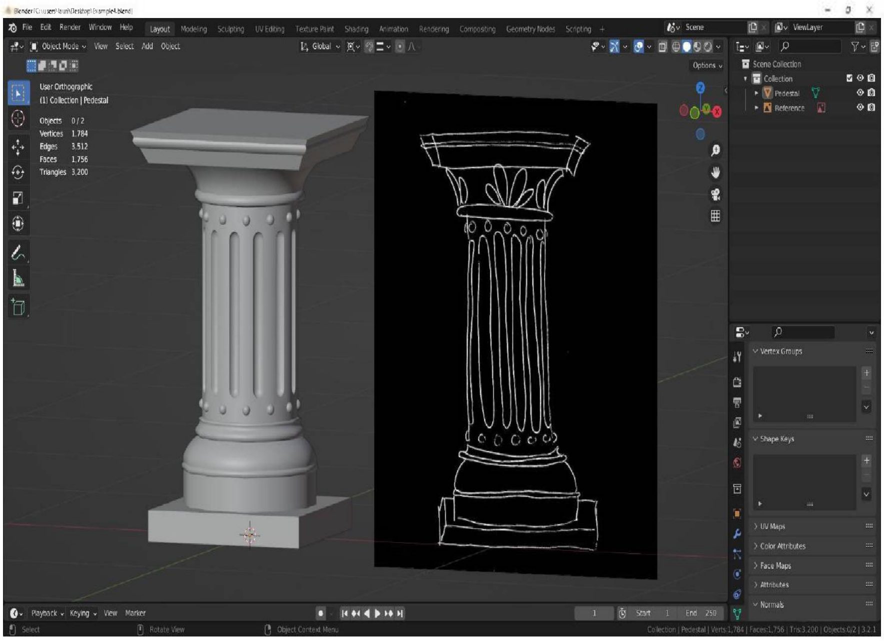


Figure 4.39: Progress so far, on pedestal poly modeled to match the exercise reference sketch, shown in Orthographic perspective

Conclusion

In this chapter, we have learned how to plan and begin implementing a 3D modeling project, as well as how to decide when to start with poly modeling versus sculpting. We have practiced essential techniques for poly modeling a 3D object from our own designs and prepared an original 3D object for further customization through sculpting, surface treatments, animations, and simulations. In the next chapter we will practice sculpting 3D objects directly and add detail for visual interest to our first 3D creation.

Essential shortcuts and hotkeys

In this chapter we used the following additional Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from the 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show/Hide Side Bar	N
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9

Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement	Alt + G
Revert all Rotation	Alt + R
Revert all Scale	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J

Shortcut	Hotkey
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Table 4.1: *Essential Shortcuts and Hotkeys*

Points to Remember

- Planning a 3D project by gathering reference images, even if only a rudimentary sketch or 3D grey-box is a valuable time saver.
- Considering the categories of a 3D project will lead us to the most helpful tools and efficient workflows.
- Paying attention to established budgets for polygon counts is essential for interactive and real time 3D products.

Questions

1. How do we choose when to use polygon modeling over sculpting?
2. What is the difference between the organic and hard surfaces in 3D?
3. What is one advantage of quad topology?
4. What are the hotkeys for hiding and unhiding geometry?

5. When would we choose to apply transforms?

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

CHAPTER 5
Poly Modeling Extras

Introduction

Now that we have learned when to start with poly modeling, practiced basic poly modeling techniques that optimize editability, and learned where to find the essential tools for creating a 3D object from our imaginations, we are better equipped to explore enhancements to our poly modeling process.

In this chapter, we will learn the purpose and practical applications of additional tools in the poly modeling toolbar, as well as explore new primitive objects to begin from. We will discover the power of modeling with modifiers, to reduce redundant operations and maintain the editability of a 3D object's sub-object components. Lastly, we will learn how to work with curves and text, converting them to polygonal models and back again.

Structure

This chapter covers supplemental tools and techniques for poly modeling. Topics to be covered include:

- Additional poly modeling tools
- 3D viewport wrap up

- The power of modifiers
- Working with curves and text
- Exercise: making 3D text
- Additional primitive objects

Objectives

After reading this chapter, we will be familiar with multiple enhancements to our poly modeling workflow that will simplify the 3D creation process, ensure precision, and allow for reverting back to a base mesh when needed.

Additional tools

One of the very best ways to learn what Blender tools can do is to experiment with them. It is highly advisable, therefore, to follow along with this chapter by opening a new instance of the application, adding a few primitive objects, and beginning to test each tool without a specific goal in mind.

From Blender's **Edit Mode**, with the left-hand **Tool bar** menu toggled on (by pressing the T key), we can see several tools that we have not yet explored through previous exercises. In *Chapter 3, General 3D Concepts*, we learned each of the transform tools: **Move**, **Rotate,** and **Scale**, as well as the **Transform multi-tool** enabling moving, rotating, and scaling all at once. In *Chapter 4, Polygonal Modeling*, we skipped **Annotate** and **Measure**, but we've already practiced **Extrude**, **Bevel**, **Loop Cut**, and **Edge Slide**.

Annotate and measure

While they are not technically modeling tools by default*, **Annotate** and **Measure** are still essential to the accurate and simplified creation of 3D objects.

*By enabling the built-in add-on Mesh: Bsurfaces BPL Edition via **Edit** > **Preferences** > **Add-ons** > search "Bsurfaces," drawings made with the Annotate tool can be converted into curve objects which can then be converted into polygonal meshes, as we will explore later in this chapter.

Annotate allows hand drawing of notes and sketches in the 3D viewport. Simply select the Annotate tool and begin drawing using the mouse, or a pen if using a drawing tablet. As with most tools, adjustments to the parameters of the annotate tool can be made via the right-hand **Sidebar** menu, toggled by pressing **N**. Unless converted, annotations are not technically objects but are saved with the **BLEND** file which is convenient for collaboration and planning.

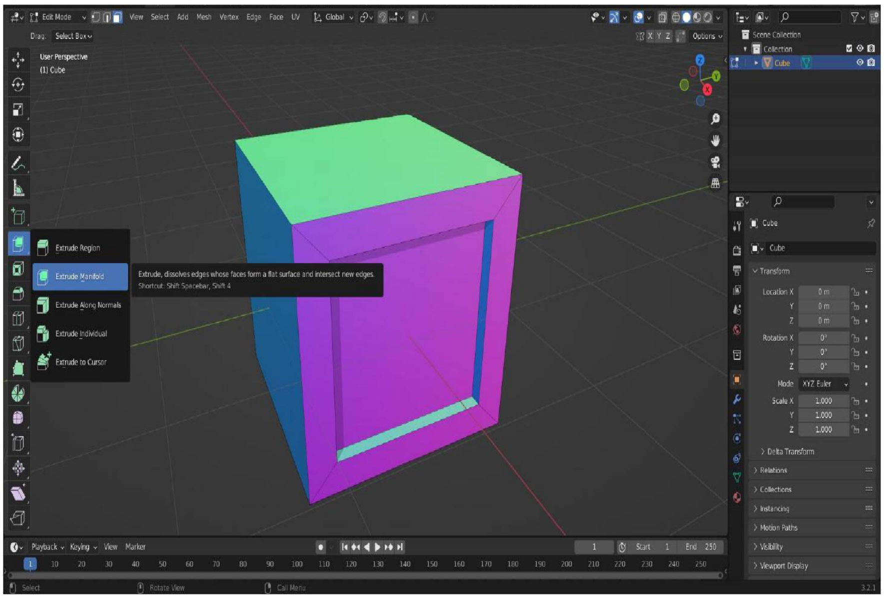
The **Measure** tool is similarly self-descriptive. Pressing the measure tool icon enables the freehand click and drag drawing of a line, which will display a readout of the line's length in the scene's currently selected unit of measure. To ensure accurate measurements, use an orthographic view while placing the measuring tool. Once a measurement line is drawn, the ends of the line can be clicked and dragged to move them, or a bending point may be inserted by clicking along any midpoint of the line and dragging, which will form an angle whose degrees or radians will also be displayed.

Add

In the **Tool bar** menu, below **Annotate** and **Measure**, an icon appearing as a cube with a plus sign to its upper left is the **Add Cube** tool, which is the default setting for adding primitives in a unique way. Underneath the Add Cube option are other familiar primitives, including cones, cylinders, and spheres. In this **Add** mode, a circular grid overlay appears under the mouse cursor when hovering over a surface. The orientation of the grid overlay shows the orientation that the added primitive will be drawn as you click and then drag. Then as you click and drag again, the depth of the added primitive will be drawn.

Extrude

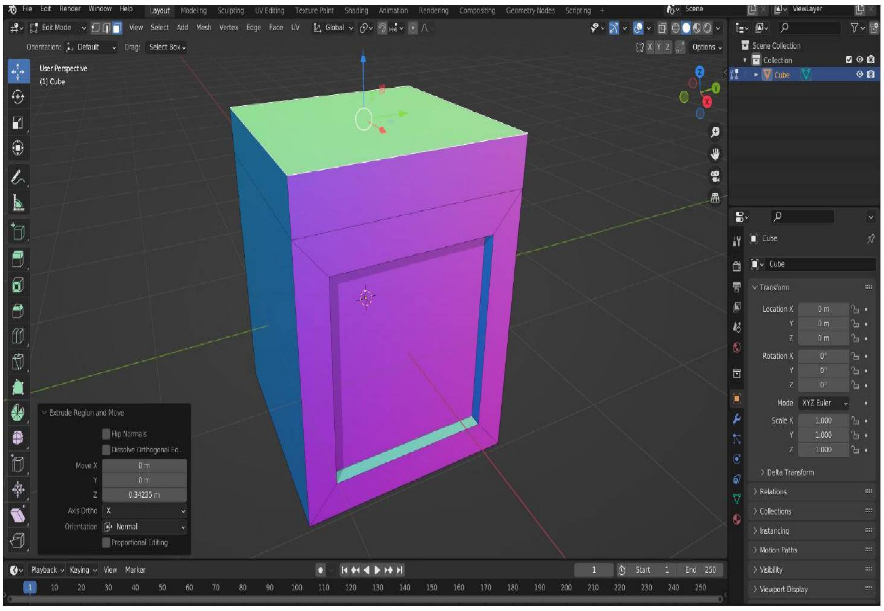
Although we have practiced a basic **Extrude operation**, it is helpful to note that multiple alternative methods of extrusion can be found under the **Extrude Region** icon in the **Tool bar** menu. As with the other tools, and menu items in Blender, hovering the mouse cursor over each Extrude option will display a tooltip describing what the operation entails as shown in *Figure 5.1*:



Extrude, dissolves edges whose faces form a flat surface and intersect new edges.
Shortcut: Shift Spacebar, Shift 4

Figure 5.1: Additional Extrude options with tooltip shown highlighted

In most cases, the functionality accessed through the **Tool bar** menu is identical to the options available through parameters in the contextual menu but serves as a convenient shortcut for common configurations as shown in *Figure 5.2*:



Extrude Region and Move
Flip Normals
Dissolve Orthogonal Ed...
Move X 0 m
Y 0 m
Z 0.34235 m
Axis Ortho X
Orientation Normal
Proportional Editing

Figure 5.2: Contextual parameters accessed using hotkeys matching additional Tool bar options

Inset Faces

The **Inset Faces** tool is very similar to Extrude. In this, the operation performs an extrusion which is constrained to the same depth of the selected face or faces as shown in *Figure 5.3*:

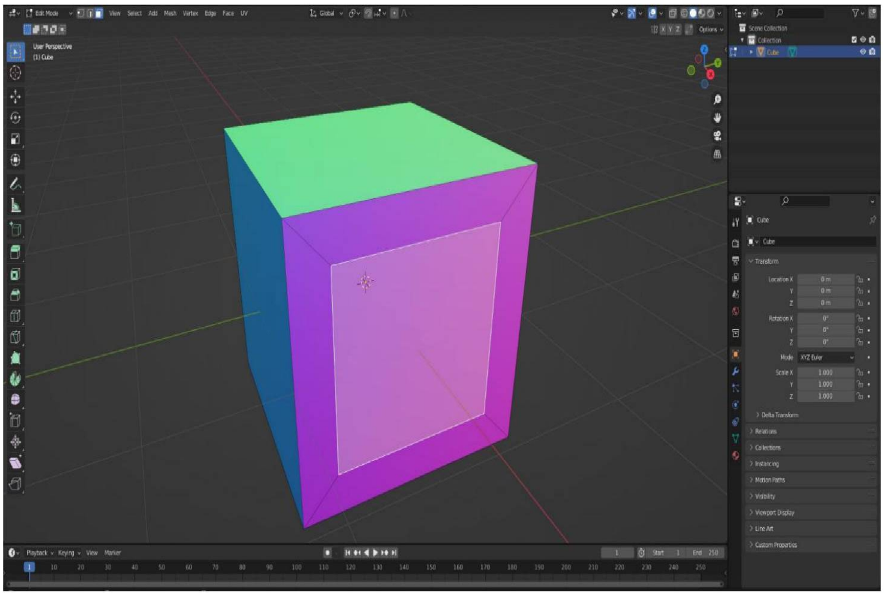


***Figure 5.3**: Single face of a cube, inset*

So, while we might expect the term "inset" to refer to an inward extrusion, in 3D, an inset face remains parallel to the surface of the original until it is extruded inward or outward in a second operation. Inset followed by Extrude can be useful in the creation of many real-world objects such as windows and doors inside frames, concave boxes, picture frames, and electronic screens, as shown in *Figure 5.4*:

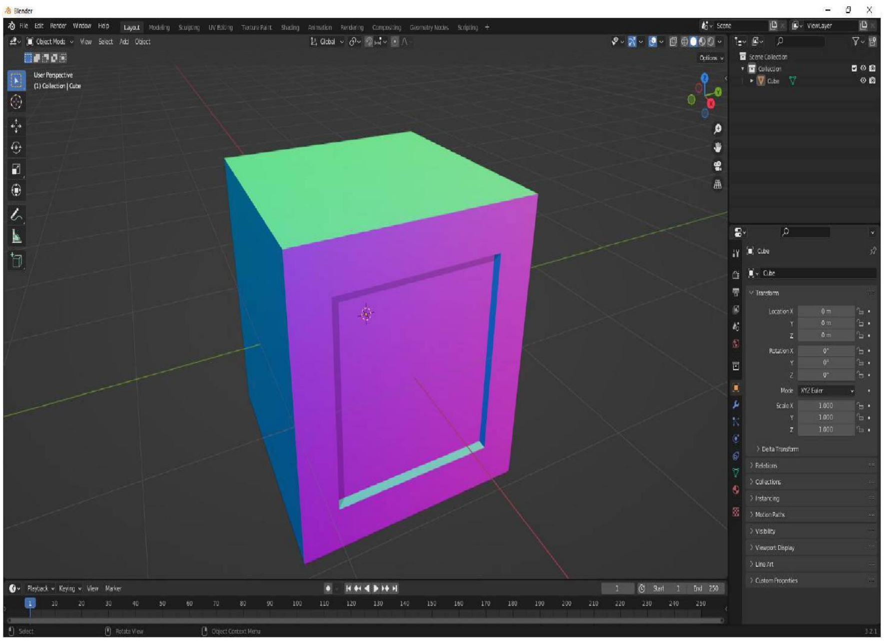


***Figure 5.4**: Single face of a cube, inset, then extruded inward*

Bevel

Accessed via the **Tool bar** menu, **Bevel** performs in the same way that we practiced in *Chapter 4, Polygonal Modeling*, but as with many tools in the **Tool bar** menu, can be activated prior to selecting upon which to apply it, and can be adjusted with a gizmo.

Loop Cut

Like Bevel, **Loop Cut** – when accessed via the **Tool bar** menu – performs in the same way that we practiced in *Chapter 4, Polygonal Modeling*, with one additional mode: **Offset Edge Loop Cut**.

Offset Edge Loop Cut works with an existing edge loop, creating a new edge loop on either side of the selected loop, which can then be slid further from or closer to the original edge via clicking + drag.

Knife and Bisect

Accessed via one icon appearing as a box with a line drawn through it, **Knife and Bisect** are like the Loop Cut tool above it. In it, they cut edges into the faces of a 3D mesh. Unlike Loop Cut, however, Knife and Bisect do not necessarily create contiguous loops of edges that are easily selectable by pressing **Alt + mouse clicking** because they allow for edges to be terminated perpendicular to an edge instead of extending through it and connecting on the other side.

The **Knife tool** enables freehand cutting of a new edge through any face, edge, or vertex, and creates a new vertex wherever the mouse is clicked on an existing face or edge. To begin cutting a mesh with the Knife tool, press the icon in the **Tool bar** menu and then click anywhere on the mesh with the knife cursor. **Ctrl Z** will undo individual clicks, rather than exiting the Knife operation. Green squares will appear wherever new vertices will be placed, and edges will highlight when the mouse cursor is placed over them. A red square will appear after clicking, and a yellow square outlined in red will appear anywhere that an existing vertex will be reused instead of created anew.

Tip: Notice the long string of options that appear in the Status Bar at the very bottom of the application when the Knife tool is in use, as shown in Figure 5.5. These options are hotkeys that can be pressed and held to modify the current

Knife operation as described, such as: pressing and holding Shift to toggle midpoint snap, C to cut through the whole mesh, or A for angle constraint. At the end of a Knife cut, press the Enter key to confirm, or press the Esc key to cancel.

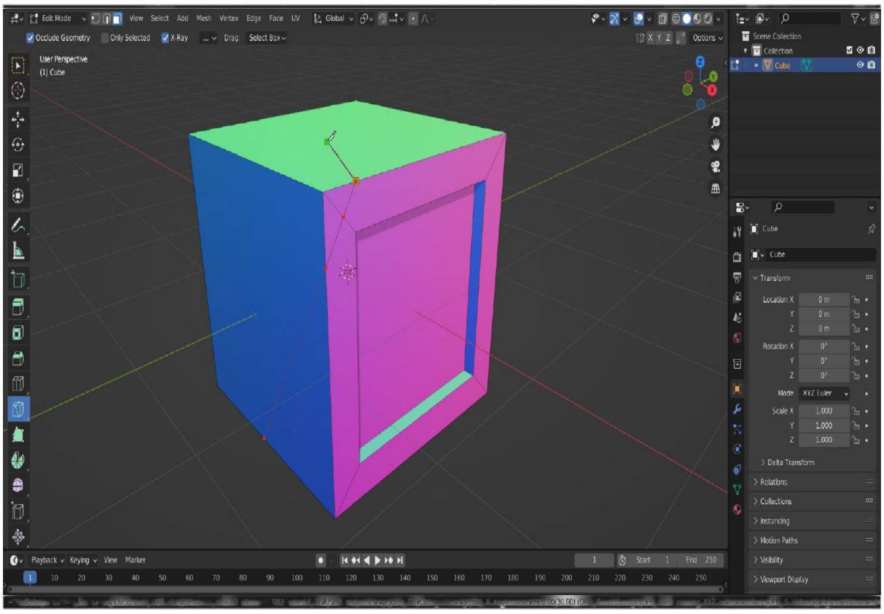
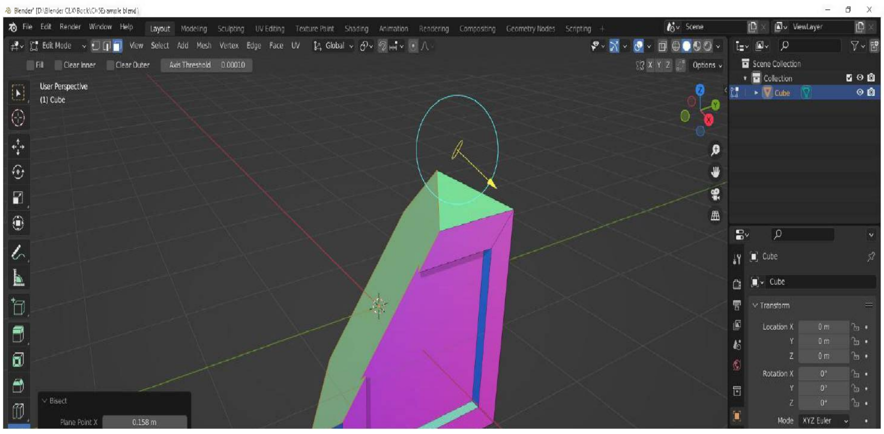


Figure 5.5: Knife cut shown mid-operation

Bisect also cuts geometry, by drawing a single straight line representing a plane extending infinitely in space, perpendicular to the viewport. The **Bisect tool** requires a prior selection to work, to avoid cutting unselected geometry. To use Bisect, select the geometry to cut, then press the icon in the Tool bar menu, then click and drag the mouse to draw the line representing a plane. After confirming a Bisect operation by releasing the mouse button, a gizmo will appear allowing additional adjustments to be made to the position of the bisecting plane, and the contextual menu in the lower left of the 3D Viewport will display additional parameters that can be changed, such as filling the resulting cut with ngons or removing one or the other of the bisected pieces as shown in *Figure 5.6*:


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Fill
Clear Inner
Clear Outer
Axis Threshold 0.00010
User Perspective
(1) Cube
Bisect
Plane Point X 0.158 m

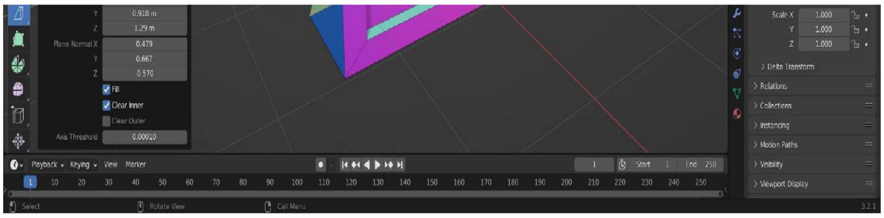

Y 0.918 m
Z 1.29 m
Plane Normal X 0.479
Y 0.667
Z -0.570
[✓ Fill]
[✓ Clear Inner]
Clear Outer
Axis Threshold 0.00010

Figure 5.6: Bisected, filled cube, with contextual parameters and gizmo shown

Tip: To create two separate objects that fit perfectly together via Bisect, mark the created edge Sharp while the Bisect operation is still in effect by pressing Ctrl + E and selecting Mark Sharp. Now, the two halves can be selected via Select Linked delimited by Sharp, and the selected part can be made a separate object by pressing P then choosing Selection. If Fill is enabled in the Bisect contextual menu both objects separated this way will have the openings between them filled, but normals of the new faces will all point in the same direction, so half of them will be inverted and may need to be flipped by selecting and pressing Alt + N and choosing Flip.

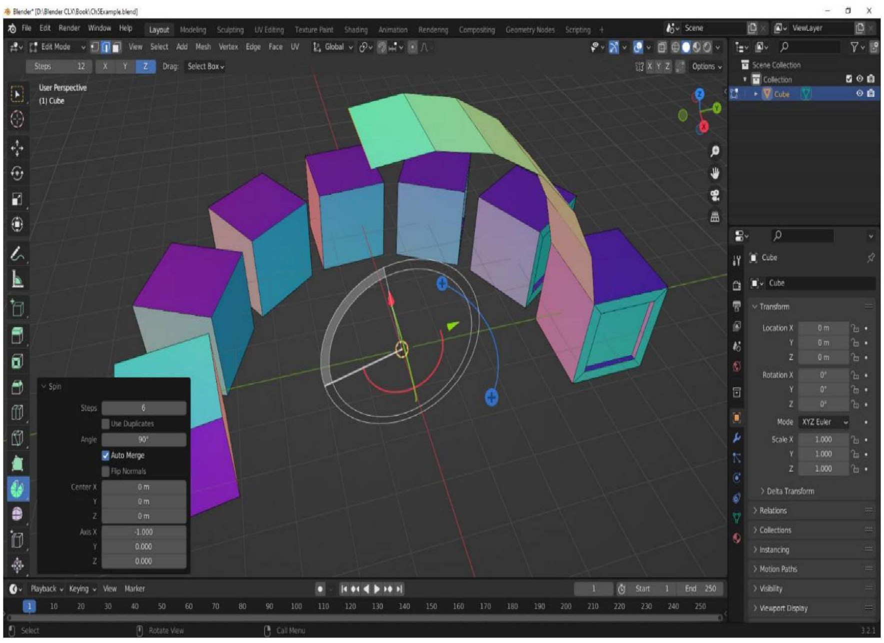
Poly Build

Poly Build, like Add Cube, is an alternate method to achieve similar functionality to other tools like Extrude, but in a more convenient way. Poly Build requires an open vertex or edge – called "non-manifold," meaning it is not filled with a face and is not watertight – and with the click of a button, will extrude a quad (or triangle, if Create Quads is deselected in the upper left of the 3D **Viewport** menu) in the direction of the cursor movement, from any highlighted edge. Or it will extrude an edge from a single highlighted vertex. To create an open edge to work from, press **Shift** with Poly Build active, and left click to delete a face.

Spin and Spin Duplicates

The **Spin tool** and its secondary option- Spin Duplicate works with a selection, and either extrudes, (in the case of Spin), or duplicates the selection, in the case of Spin Duplicates. Both Spin tools enable a gizmo multi-tool, first appearing as a blue arc with plus signs at either end which we can click and drag to control the angle of the spin operation. Once the Spin operation is confirmed, the Spin gizmo displays transform controls, so that the pivot point can be moved or rotated in any direction, while the contextual menu allows for direct adjustment of parameters for more precision.

Tip: If getting oriented to the way gizmos work is daunting, try moving and rotating the gizmo, paying attention to the corresponding axes of world space. Grabbing and rotating the red circle, for example, will rotate the selection around the red (x) axis as shown in Figure 5.7. By default, an operation's number of steps may be set so high as to cause overlap, so also try adjusting the number of steps and angle (amount) via the contextual menu. Feel free to zero out all but one parameter, to better see the effect of each one individually, and use undo as needed.



Spin

Steps	6
☐ Use Duplicates	
Angle	90°
✓ Auto Merge	
☐ Flip Normals	
Center X	0 m
Y	0 m
Z	0 m
Axis X	-1.000
Y	0.000
Z	0.000

Figure 5.7: Cubes duplicated in a circle using Spin Duplicates, one cube with a single edge extruded on the x axis via Spin

Smooth and Randomize

Smooth and Randomize are grouped together in the **Tool bar** menu and perform opposite operations of a similar kind. Smoothing in this context refers – not to the surface shading as introduced in **Chapter 4, Polygonal Modeling** – but rather to softening the angles between the faces of an object so that they become less jagged and more rounded.

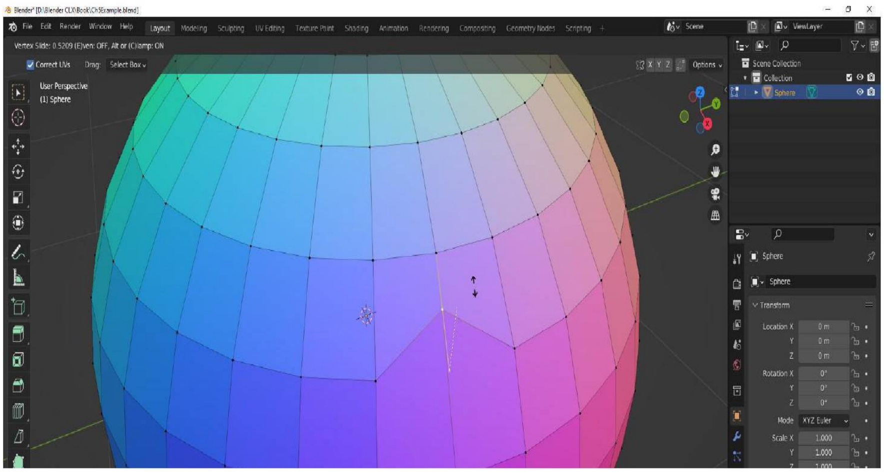
Randomize moves the selected vertices in random directions, which increases the sharpness of angles between faces. Randomize can be very useful for adding natural irregularity to organic shapes.

Edge Slide and Vertex Slide

Edge Slide and Vertex Slide work in **Edge** and **Vertex Edit** Modes respectively. Sliding edges or vertices along an edge is useful for adjusting placement while keeping the moved component constrained to surrounding edges. In *Chapter 4, Polygonal Modeling* we practiced using Edge Slide on edge loops via the hotkey combo **G + G**. Vertex Slide can also be activated with two presses of the **G** key in succession and works the same way.

When either **Slide operation** is in effect, a yellow line will appear to indicate which surrounding edge the movement is currently constrained to, two arrows will appear to show the direction of movement, and a white dotted line will point to the

mouse cursor as shown in *Figure 5.8*. If the selected edge or vertex is surrounded by perpendicular edges, which edge the slide movement is constrained to can be changed mid-operation by moving the mouse in a direction perpendicular to the indicator, left and right or up and down. Please refer to the following figure:


Vertex Slide: 0.5209 (E)ven: OFF, Alt or (C)lamp: ON

Correct UVs [✓] Drag: Select Box

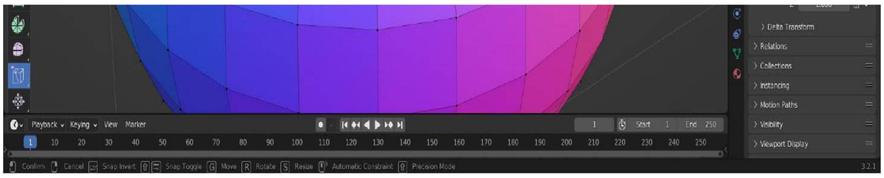


Figure 5.8: Vertex Slide in effect, with indicators showing the edge and direction movement is constrained to

Shrink and Fatten, Push and Pull

The difference between Scale and Shrink and Fatten is easier to see than describe, as shown in *Figure 5.9* While **Scale** is performed relative to the selected transform pivot point, **Shrink** and **Fatten** are relative to the selected sub-object component normals, which can be useful when we need to scale a convoluted selection relative to itself. Shrink and Fatten can also be invoked in **Edit Mode** via the **Alt + S** hotkey combination.

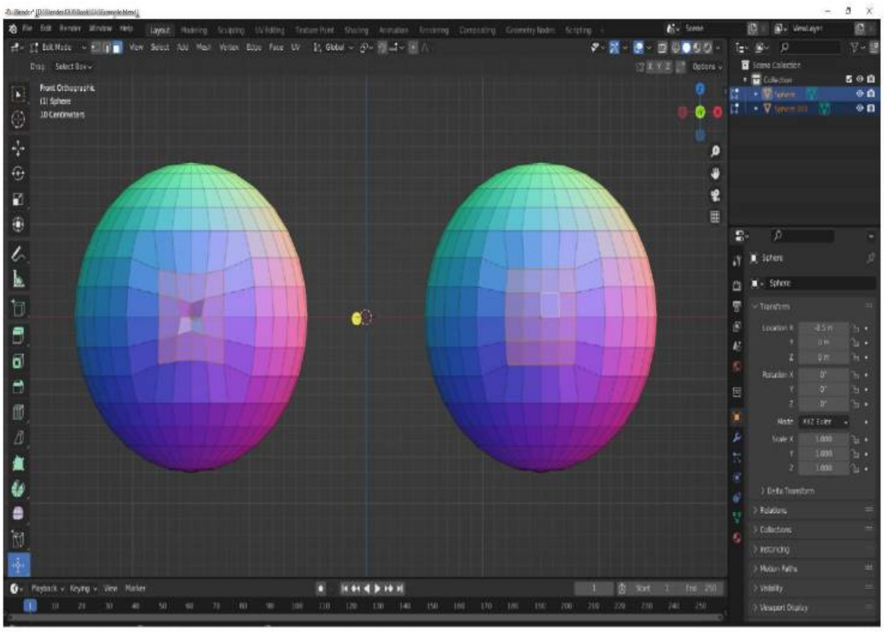


Figure 5.9: The same selection of vertices scaled smaller using Shrink on the left, and Scale on the right

In a similar way, **Push** and **Pull** are corollaries to the transform move, but Push and Pull do work relative to the transform pivot point. Unlike Move, Push and Pull moves the selected sub-object components toward each other or away from each other relative to the pivot point.

Shear and To Sphere

Shear and **To Sphere** are quite different from one another in terms of results, but similar in that they both affect the selection with a shape as shown in *Figure 5.10*.

Shear has the effect of stretching the selection relative to a plane parallel to the current view. The Shear tool in the **Tool bar** menu enables a rectangular gizmo that indicates the plane along which the selection will be stretched and handles from which to push and pull. This plane stays oriented to the view, even as we orbit around the selection.

To Sphere does not enable a gizmo, but rather works via clicking and dragging anywhere in the viewport, in any direction. To Sphere does exactly as described, deforming the selection's original shape towards a spherical shape. **The Factor** setting in the contextual menu that appears in the bottom left after confirmation of To Sphere is a range where 0 represents no deformation, and 1 represents 100%. Please refer to the following figure:

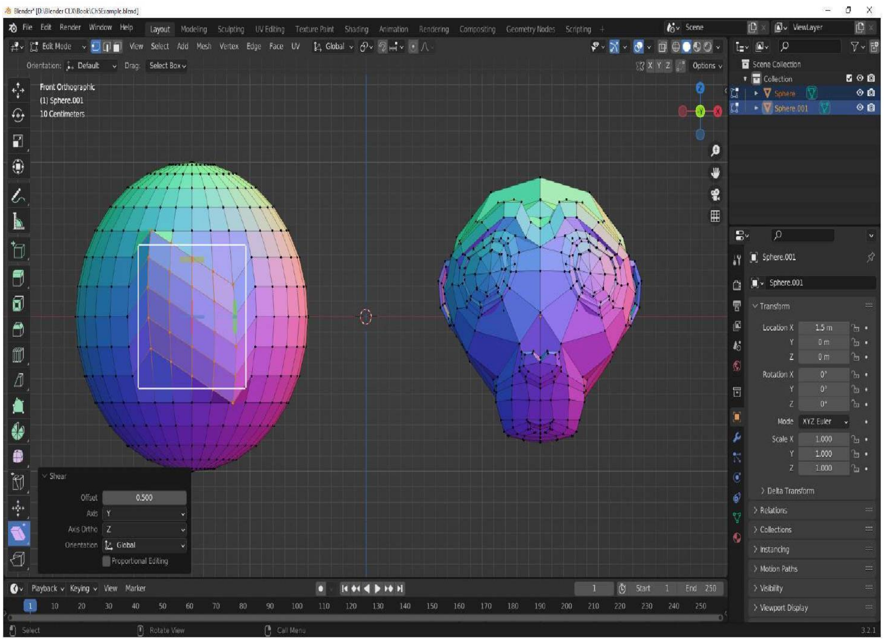


Figure 5.10: UV Sphere with 0.5 Shear applied on left, Suzanne with 0.75 To Sphere applied on right

Rip Region and Rip Edge

The **Rip Region** icon, appearing as a box with one corner peeled away, is a clear representation of this tool's usage. When two or more edges share a vertex, the common vertex is considered "welded" or merged, as we practiced in *Chapter 4, Polygonal Modeling*. Each edge has its own vertex existing in the exact same space, but for computations like selection they are considered as one.

Note: The concept of vertices which share the same space being welded/merged or not has implications in many 3D creation scenarios. Certain 3D file formats such as those commonly used in CAD do not weld vertices or save information about whether vertices are welded or not, which means that every single vertex of a 3D model saved in that format has between double and quadruple the number of vertices of the same model with all the vertices sharing space being welded. This can easily have a negative impact on processing power, especially in interactive applications and on low power devices, which is one reason that choosing the appropriate file format is important.

Sometimes, however, we need to open that edge like a door. The purpose of Rip Region is to break one vertex shared by two edges into two, so that each edge's vertex in that space can be moved and operated on independently, or in the case of

a larger selection, to separate the boundary vertices so the selection can be moved independently.

Rip Edge is a bit more complicated than Rip Region because it simply moves the edge without breaking apart the vertices and creates ngons between the edge's original position and wherever it is ripped to in the process.

3D viewport wrap up

As extensive as it is, Blender's **Tool bar** menu contains just a selection of more commonly used poly modeling tools. In Edit Mode, there are also several operations under each of the 3D **Viewport** headings – Mesh, Vertex, Edge, Face, and UV – that we won't explore here, but will come into play in future chapters. For any functionality that piques your curiosity, it is advisable to open an empty Blender instance and simply begin to experiment, referring to the Blender official documentation for deeper understanding.

What we will do here is cover the remaining everyday functionality for polygonal modeling. From the top middle of the **3D Viewport** in either **Object** or **Edit Mode**, we have used the Transform Pivot Point, and Snap tools, but to their right and left, the Transform Orientation and Proportional Editing options have thus far gone ignored. Far from inconsequential, these poly modeling essentials have operated behind the scenes in all our previous exercises.

Transformation orientation

In *Chapter 2, Installation and Interface*, we learned that there are significant differences between viewport orientation and object transforms. Navigating the viewport is like moving a camera around the scene, while manipulating objects is independent of the view.

Like the Transform Pivot Point, the **Transform Orientation** setting dictates how to transform operations are performed, and what or where they are performed relative to. When we transform along a given axis, for example, the direction of the movement is relative to the selected Transform Orientation. It's a bit like the difference between the directions up, down, left, and right which is relative, or North, South, East and West which are global. Sometimes we want to "give our model directions" in terms of the world, and other times we may want to give those directions relative to a specific landmark.

As the name implies, the default Transform Orientation, Global, refers to the application's world space, which in the previous analogy constitutes North, South, East, West, as well as up (skyward) and down (toward the ground). Every transform we have performed so far has been relative to this global, or world, space. Working in any of the alternate Transform Orientations can be confusing, and only needed for very specific operations, so it is usually best to revert to the default **Global Transform Orientation** when in doubt.

Instead of explaining every Transform Orientation, we'll discuss a few ways in which they are different from Global. The additional options available under the Transform Orientation drop-down menu, appearing by default in the middle top of the **3D Viewport** as a gizmo next to the word Global, are as follows: **Local**, **Normal**, **Gimbal**, **View**, and **Cursor**.

A **Local Transform Orientation** is, as it sounds, relative to the whole selection. If we were to lie flat on our backs, our local z or "up" and "down" axis would run through the top of our head and bottoms of our feet, even though it would be parallel with the ground, or the global or world x and y axes. Local Transform Orientation is handy when the selected object or sub-object component is oriented at an odd angle to the world front, back, side, top or bottom views. For example, if we wanted to raise our hat straight up from the top of our heads while lying on the ground, it might be helpful to switch to Local Transform Orientation for a moment and translate the hat "upward" in our local z axis rather than skyward along the z axis of global space.

Likewise, the **Normal Transform Orientation** option is relative to the selection's normal direction, which is dependent on whether the selection is a vertex, edge, face, or multiple sub-object component. Since whole objects and contain potentially infinite sub-object component normals pointing in infinite directions, the Normal Transform Orientation doesn't work on whole objects, and in **Edit Mode**, operates on the Active selection.

Tip: Refer to Chapter 2, Installation and Interface for a refresher on the difference between Selected and Active and remember: to visualize a sub-object component's normal direction, Normals overlays can be enabled in the Viewport Overlays menu in the upper right of the 3D Viewport.

The **Gimbal Transform Orientation** can be thought of as relative to the selection's combined rotations in all three axes as shown in *Figure 5.11*. This Transform Orientation is more advanced than is useful for introductory 3D creation techniques and won't be referenced further in this text.

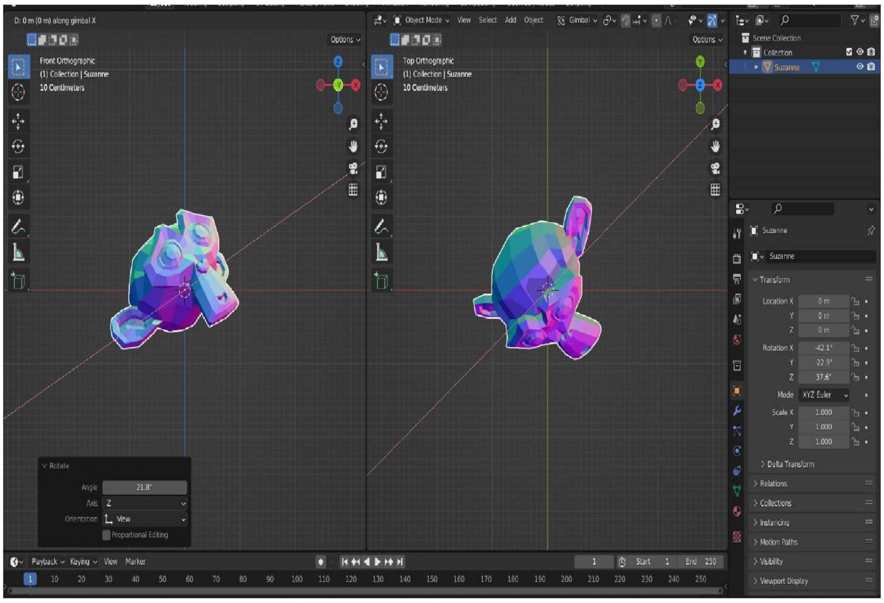


Figure 5.11: Moving along the x axis of a rotated object in Gimbal Transform Orientation mode

The **View Transform Orientation** is relative to our current perspective as displayed on the screen, and changes as we orbit, pan, and zoom, as well as when we change from Orthographic to Perspective views. A common usage for the View Transform Orientation is to make precise transforms at irregular angles by first orienting the view to the selection by using the Shift key to modify the viewpoint shortcuts to local space, and then using the View Transform Orientation to move, rotate, and scale along the view axes as you would along world axes as shown in *Figure 5.12*:

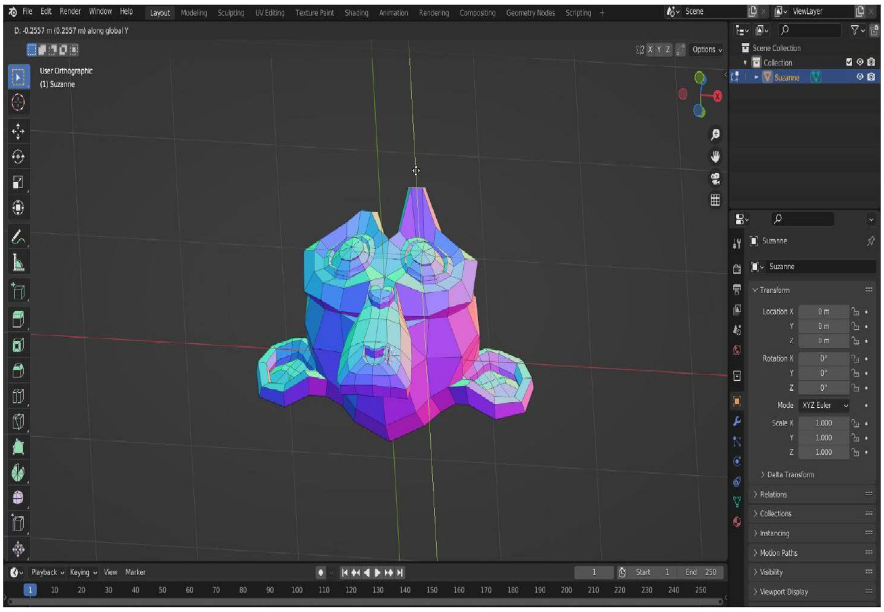


Figure 5.12: Moving a face in View Transform Orientation mode,

with movement constrained to the viewport's y axis

The **Cursor Transform Orientation** operates relative to the 3D cursor's position, the utility of which is akin to its use as a temporary pivot point, since the 3D cursor can be manually placed anywhere, or snapped to other objects and sub-object components.

Once again, these alternate Transform Orientation options have narrow, specific uses, and are most often a temporary setting which we will revert to Global immediately after using.

Proportional editing

Within the contextual menus of many of the operations we have performed, options for Proportional Editing have appeared, although we have not yet activated the option. Proportional Editing can be thought of as creating a ripple effect with any transform, where the selected geometry is the center point which the transform operation radiates out from.

In this way, when Proportional Editing is enabled and a transform is initiated, neighboring geometry will also be transformed, but at a proportionally reduced amount from the effect on the selected geometry. Enabling Proportional Editing mode also enables a circular indicator of the boundary of the transform's effects, which can be made bigger or smaller by scrolling the mouse wheel. The amount that neighboring geometry is affected by the transform is greatest nearest the selection and falls off gradually between the selection and the boundary, depending on the falloff type selected, as shown in *Figure 5.13*.

The falloff type is the shape of this falloff, or the ripples of the effect, and can be changed under the Proportional Editing Falloff drop-down menu icon appearing by default as an upward arcing curve. It may help to imagine that the shape of the falloff displayed in the menu icon is the side profile of the transform effect. Please refer to the following figure:

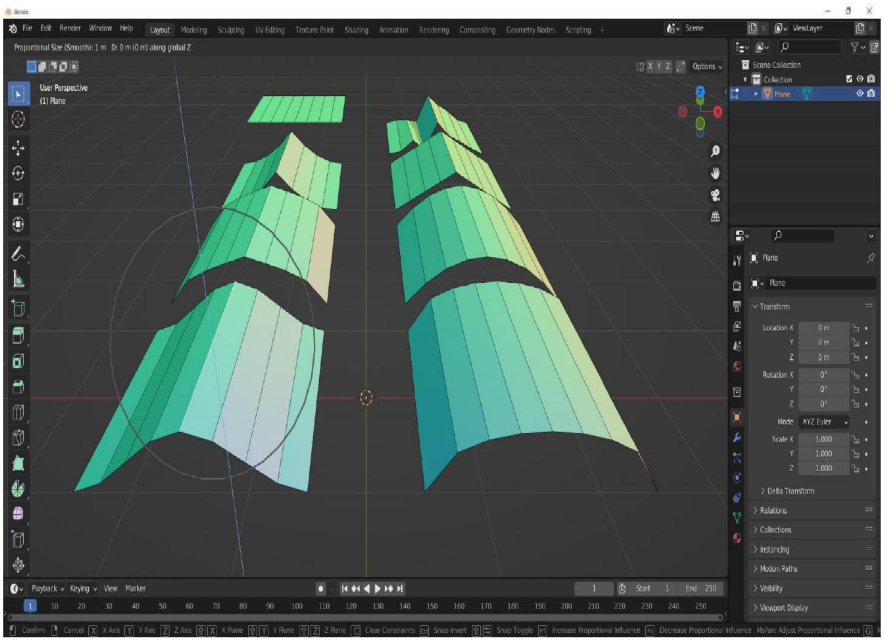


Figure 5.13: Falloff options left to right, front to back: Smooth, Sphere, Root, Inverse Square, Sharp, Linear, Constant, Random

Modifiers

As the name implies, **Modifiers** modify whole 3D objects. The special characteristic of Modifiers is that they do not alter the underlying sub-object geometry directly. Instead, Modifiers represent a sort of container that changes the object in a temporary way until permanently applied or removed. We can think of Modifiers as akin to garments. A girdle or a belt, for example, may squeeze and push the human form underneath to give it a new appearance, but once the garment is removed, the underlying shape will return to its original form.

Applying and removing modifiers

The benefit to using Modifiers is that, unlike transforms, Modifiers do not directly edit the underlying sub-object geometry unless and until they are applied. This maintains the editability of the mesh and allows for infinite adjustment of the modifying effect.

Unlike a garment, once the Modifier's desired effect is achieved, the Modifier's effects can also be made permanent via the Apply command found in the upper right of a given Modifier under the down arrow icon. Until it has been applied, we can always remove Modifiers by pressing the X icon in the upper right of the Modifier or leaving the Modifier's parameters empty. Then, as with the garment analogy, the affected geometry will revert to its underlying state.

Modifiers are accessed in the Modifiers Property tab of the Properties panel, and are grouped under the headings Modify, Generate, Deform, and Physics, although all of them technically "modify" the underlying geometry. As shown in *Figure 5.14*, the list of available Modifiers is lengthy, so rather than outlining each one, we'll explore the commonly used Modifiers Array, Boolean, and Mirror. Once we have a strong grasp of general Modifier usage, the Blender official documentation for the remaining Modifier options will make more sense. Please refer to the following figure:

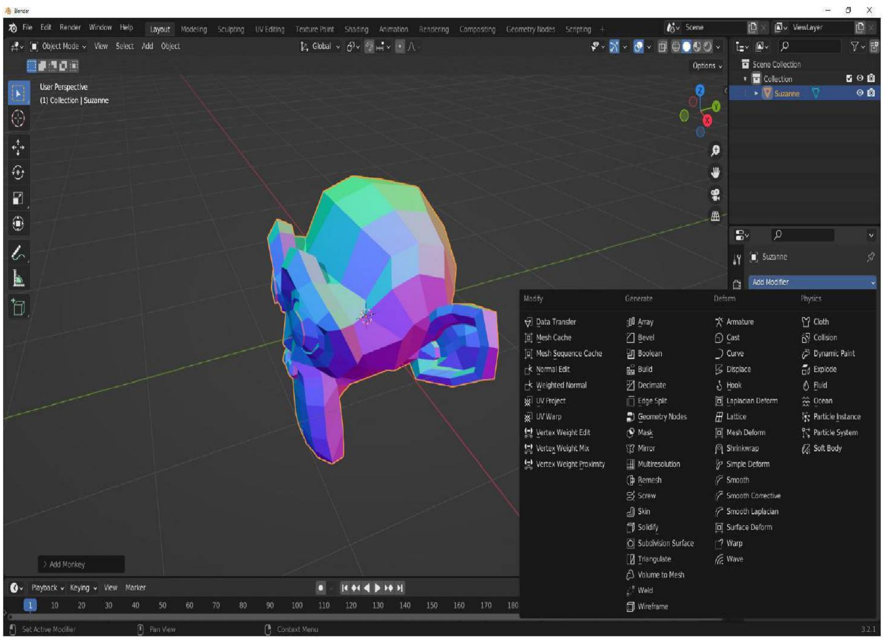


Figure 5.14*: Modifier options grouped under headings in the Modifier Properties tab of the Properties panel*

Modifier artifacts

Before we delve into employing Modifiers in our 3D creations, it is important to understand the consequences of doing so, so that we can plan accordingly. The incredibly flexible nature of Modifiers means that artifacts – or unintended results – do occur. Mirrored geometry, for example, will have inverted normals from the original, which means that its faces are inside out until corrected.

Similarly, **Boolean Modifiers** can create ngons where we might prefer quads. Even though these artifacts can require a few extra steps to resolve, Modifiers can still potentially save many more steps in return. Whether the number of steps required and saved results in a positive or negative should always be considered when planning whether to use Modifiers.

Empty Objects

It is also helpful to introduce the concept of Empty objects, and the role that they can play in the use of Modifiers. Empty objects can be added to a scene for the express

purpose of being used by a Modifier, as a transform origin much like the 3D Cursor, but with a unique name that appears in the Outliner and with transforms that can be edited like a real 3D object.

This separation from the 3D Cursor makes it easier to create multiple for multiple purposes in one scene, to select, deselect, and transform without losing properties like transforms, when other objects need to use it. As such, Empty objects are essentially placeholders, and like a gizmo, provide visual reference points and some transform data, but not much else (hence the name).

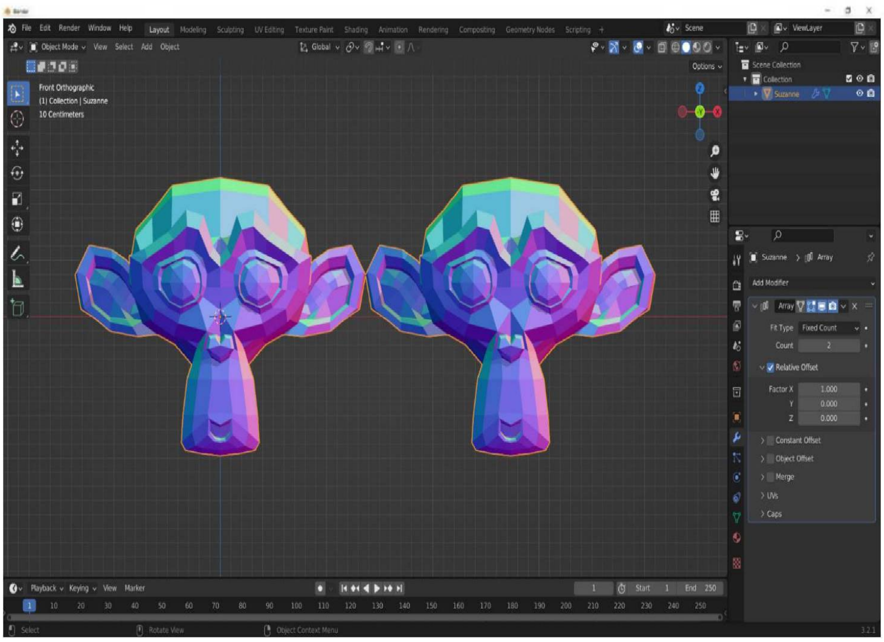
Tip: Actual 3D objects can also be used in place of Empty objects as the transform origin for Modifiers when it makes sense to do so. For example, the axle of a wheel might be a logical object to use as the origin for a radial Array of spokes.

Available from the **Add menu** in **Object Mode** only, Empty objects come in a variety of shapes and sizes, from plain axes to arrows and circles. In *Chapter 4, Polygonal Modeling*, we encountered one type of Empty object in the form of a reference image that we imported into the scene. In this case, the Empty object was not actually empty, as it behaved as a container for the image's visual properties. However, unlike a 3D object with an image applied to its surface, the reference image Empty is not itself a 3D object and is not intended for export or use in other 3D applications.

Note: Empty usually aren't included in exports to common 3D file formats, which means that their use is typically limited to the application they are created in. Occasionally, 3D objects imported into Blender from external programs may contain Empty objects as containers for multiple other objects which involves a new concept called "parenting" that we will explore in more detail in Chapter 8, 3D Animation.

Array modifier

Simply put, the **Array Modifier** duplicates the object and allows us to choose an offset amount and a shared origin for the duplicates. As shown in *Figure 5.15* the default Array creates a second copy of the object with an Array Modifier, the number of which can be increased via the Count parameter, and by default offsets the duplicate by 1 unit relative to the original. This means that even if the original object's scale changes, the duplicate will remain the size of the original object away from the original.



Fit Type: Fixed Count
Count: 2
Relative Offset ✓
Factor X: 1.000
Y: 0.000
Z: 0.000

Figure 5.15: *An Array Modifier on the left Suzanne, creates a second duplicate Suzanne on the right by default*

Beyond simply duplicating objects a specified distance and direction apart, Arrays are helpful for creating editable repeating patterns in rows, columns, and 2D or 3D grids, such as multi-paned windows as shown in *Figure 5.16*. Arrays can also be used in conjunction with other Modifiers such as Curves, to determine where the duplicated objects are placed along a path, such as railroad ties on a train track. With Object Offset enabled in the Array Modifier, and a second object such as an Empty assigned, a radial array can even be created by rotating the second object.

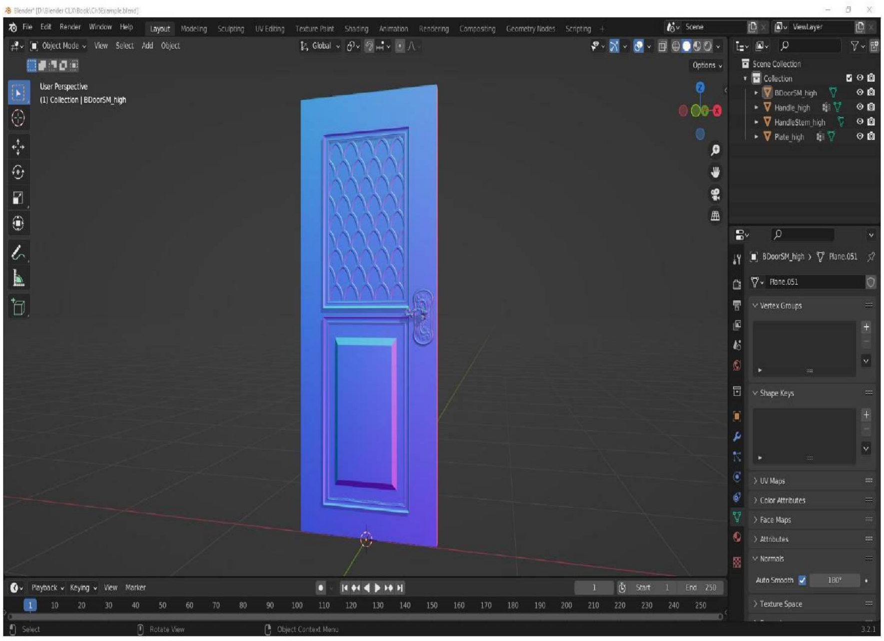


Figure 5.16: Leaded glass windowpanes arranged with an Array Modifier

Boolean Modifier

Whereas the Array Modifier is purely additive, **Boolean Modifiers** can be additive or subtractive. As such, the Boolean Modifier requires two separate objects to work: the object to be modified with the Modifier, and a separate object to cut away from or to be merged with the first.

Note: Unlike the Array and many other Modifiers, the Boolean Modifier requires watertight or "manifold" objects to work, which means that vertices, edges, and planes cannot be cut with the Boolean Modifier. To cut one non-manifold object with the edges of another, Knife Project might be preferable.

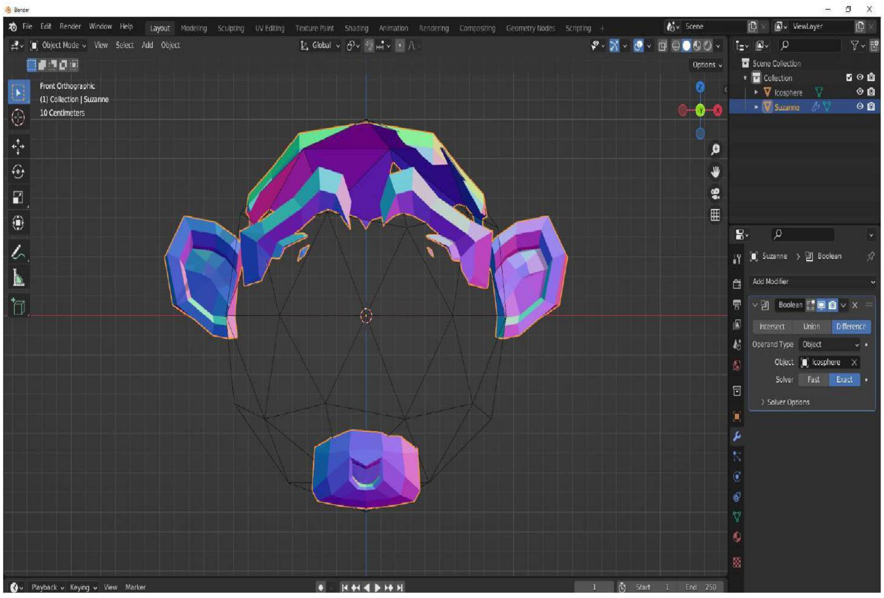
To use Knife Project, position the two objects in Object Mode, select the object to be cut and enter Edit Mode, then select the geometry to be cut. Next, press Ctrl and select the cutting object, then choose Knife Project under the Mesh menu in the top of the 3D Viewport.

By default, the Boolean Modifier is set to the Difference type. The Difference Boolean is a subtractive operation, which means that the object selected as the **Operand Object** is the object that will cut away from the modified object.

Tip: Selecting the Operand Object for any Modifier is as simple as dragging and dropping the object's name from the Outliner into the Object parameter, clicking within the open parameter, and selecting the object's name from the drop-down panel that opens, or selecting the eyedropper tool, then clicking on the cutting object in the 3D Viewport or Outliner.

Like leaving an imprint in sand, being able to cut the shape of one object away from another object, can save a significant amount of time. Instead of having to model or sculpt a shape once and then again in the inverse, we can simply use one shape to create its own impression in another.

In a Difference Boolean operation, the cutter object remains intact as a separate object and can be hidden at any time as shown in *Figure 5.17*. After the Boolean Modifier has been applied via the **Apply** option under the down arrow icon in the upper right of the Boolean Modifier's panel, the cutter object can be deleted or hidden, although it may be useful to retain the cutting object when we want two separate objects that fit together seamlessly.



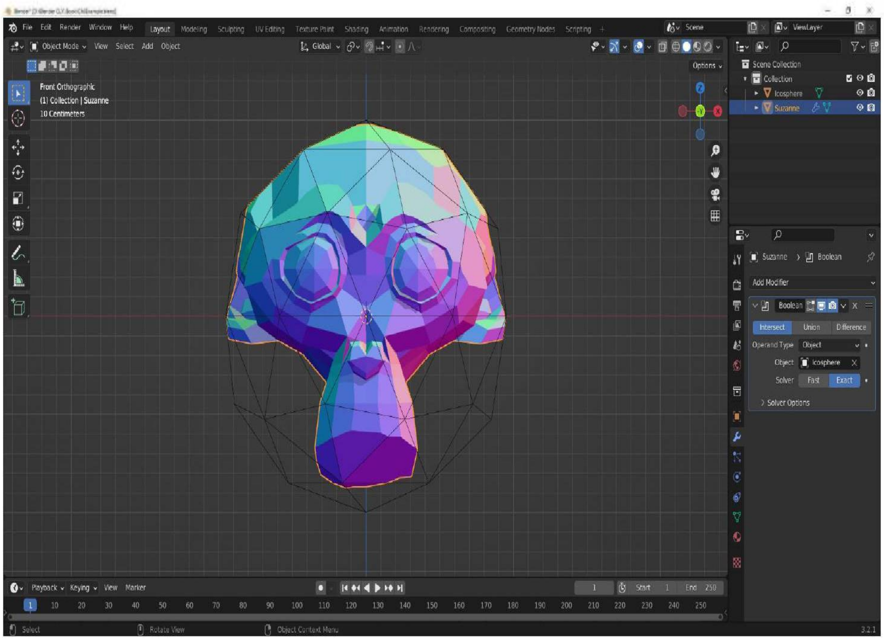
Boolean
Intersect Union Difference
Operand Type: Object
Object: Icosphere
Solver: Fast Exact
Solver Options

Figure 5.17: Suzanne with Difference Boolean and Icosphere (rendered as wireframe) Operand Object

A Union Boolean, as it sounds, joins a second object to the first modified object. In a Union Boolean operation, the joining object also remains intact as a separate object and can be hidden at any time. After the Boolean Modifier using it has been applied, the joining object can be deleted or hidden.

Intersect is essentially the inverse of Difference, where the modified object is cut away from the **Object Operand**, leaving behind a new shape where the two shapes intersected as shown in *Figure 5.18*. As with Difference and Union, the Object

Operand in an Intersect Boolean remains a separate object and can be hidden at any time or deleted after the Modifier is applied.



Intersect | Union | Difference
Operand Type: Object
Object: Icosphere
Solver: Fast | Exact

Figure 5.18: Suzanne with Intersect Boolean and Icosphere (rendered as wireframe) Operand Object

Mirror

The **Mirror Modifier** might well be the most used Modifier of all. The Mirror Modifier creates a symmetrical duplicate of the geometry to which it is applied, so any 3D object with symmetry in any dimension is faster to model with a Mirror Modifier in effect. In practice, this means that we can focus our efforts on modeling only one half of a symmetrical object, and the Mirror Modifier will do the rest.

Used in conjunction with the Array Modifier, Mirror Modifiers make quick work of everything from modeling character base meshes to buildings, vehicles, furniture, and everything in between. Even when the goal is an asymmetrical model, we can and everything in between. Even when the goal is an asymmetrical model, we can often avoid significant repetition by starting out with a Mirror Modifier, applying it, and adding asymmetrical detail as a second step.

The typical workflow for using a Mirror Modifier is to enter **Edit Mode** and toggle X-Ray, select one half of the geometry or the other (using the Knife tool to split the model down the middle if needed), and delete half of the model along one axis. But Mirrored modeling can even be begun with the Mirror Modifier on a Single Vert object.

Once the Mirror Modifier is in effect, it will mirror geometry from one side to the other based on the world axis indicated in the Axis parameter, or an object if chosen in the Mirror Object field. Enabling the **Clipping** setting will prevent new geometry from being moved across the axis, and Merge will weld any vertices along the Mirror's midline.

Additional modifiers

In future chapters, we will discuss additional modifiers relevant to the current lesson, such as the Multiresolution Modifier, which we will use in the next chapter on *3D Sculpting*; *Cloth, Collision, Fluid,* and *Particle System Modifiers*, which we will use in *Chapter 9, Effects and Simulations*; and finally *Decimate and Remesh Modifiers*, which we will discuss in the final chapter on using 3d objects in production.

Curves and Text

It may seem illogical to group Curves and Text together at first, but in Blender, 3D text is made of a special type of 3D object called a Curve. Curves can be thought of as a 3D corollary to Vectors in 2D art – such as those created in Adobe Illustrator – in that they are composed of line segments drawn between control points, with adjustable thickness and sometimes fills. In fact, Blender conveniently allows the import and editing of vector art in the **SVG** file format, as well as exporting of Curve objects as SVG for editing in external vector art editing programs. Unlike 2D art which relies on shading and color to give the impression of depth however, Curve objects in Blender can be made truly three dimensional.

The line segments of a curve are sometimes referred to as a path, and the shape of a path is determined by the length of its segments and the direction they enter or exit any given control point. Like Mesh objects, Curve objects can originate from a single point, but for most purposes 3D objects need two or more. For every two points along a 3D curve, a mathematical formula dictates whether any segment of line coming into or going out of each control point is straight or curved. As shown in *Figure 5.19*, the formula for this blending, or "interpolation," between control points may make the path sharp and flat, or smooth and curved, or a mixture of both. Fortunately, we don't have to learn or apply these formulas, we just pick a curve type based on its properties.

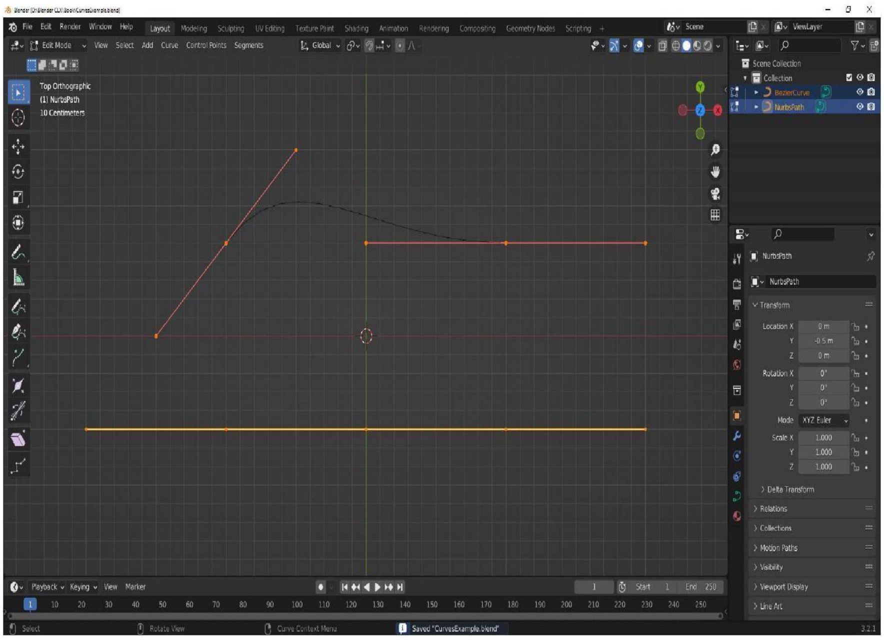


Figure 5.19: Two Curve objects with different interpolation: Bezier type on top, Path type on bottom

From the Curve section of the Add menu, we can choose Bezier, Circle (which is a closed Bezier Curve), Nurbs Curve, Nurbs Circle (which is a closed Nurbs Curve), or Path. Each of these Curve object types can be switched to any of the other types from Edit mode. Many online sources claim that Nurbs objects in Blender are more akin to CAD than modeling tools, and aren't well implemented, so we'll focus on the more common Curve object types: Bezier and Path.

In 3D creation, the opposite of Curve objects is the Mesh objects we have worked with so far, which are made up of straight lines and flat planes between points and can only give the appearance of curvature through shading tricks and subdivision as we first learned about in *Chapter 3, General 3D Concepts*. Like Vector to Raster in graphics, Curve objects are often used as a preliminary stage for Mesh objects while the shape is still in flux, because the interpolation between the vertices of a Mesh is essentially "baked in" the way that Raster graphics are pixelated.

When we convert a Curve object to a Mesh to make it more usable in more applications, control points are replaced with vertices along the curve with straight edges replacing the path segments in between. The number of vertices that will be "baked" along a Curve object's path when converted into Mesh is the curve's "resolution," which is a parameter which can be increased or decreased infinitely while the object is still in a Curve form.

To begin working with Curves and understand the available options, we will perform the following exercise: making 3D text.

Exercise 5.1

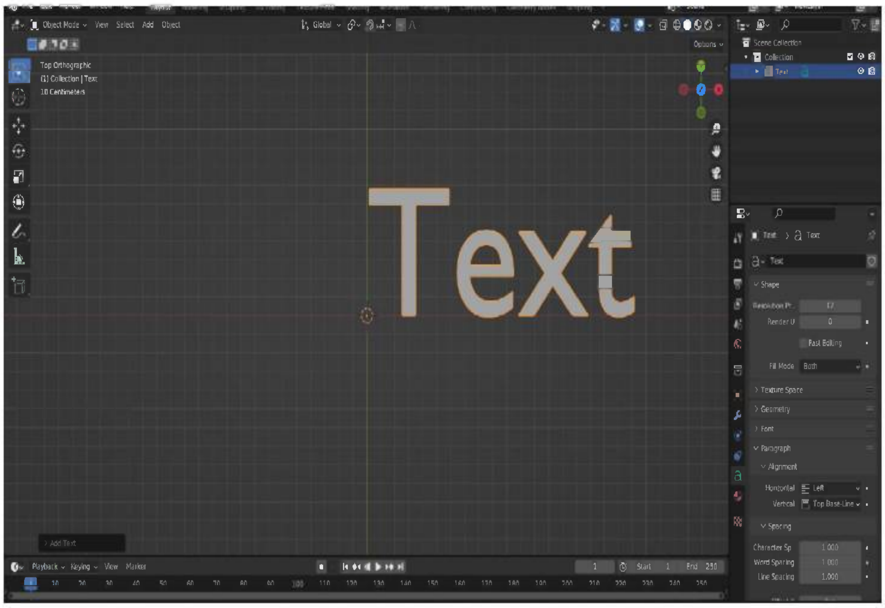
Text is in its own category in Blender, as a special kind of Curve object with relevant options including fonts and common text editing parameters such as paragraph alignment, character spacing, and so forth.

1. For this simple exercise, we will begin by adding a Text object into the scene by pressing **Shift + A** with our mouse cursor inside the 3D Viewport, or via the Add menu at the top of the 3D Viewport, then selecting Text.

   Text objects in Blender start out flat in the z axis, so to see and edit our text, we either need to rotate the object or our view.

2. For this exercise, let's orient our view to Top by pressing the Number pad **7** key, or by clicking the Z icon in the 3D Viewport's upper right gizmo.

3. From here, we can open the special Object Data Properties panel for Text, from the Properties panel tab appearing as a lowercase letter a, as shown in *Figure 5.20*:



Shape
Resolution Pr... 12
Render U 0
Fast Editing
Fill Mode Both
Texture Space
Geometry
Font
Paragraph
Alignment
Horizontal Left
Vertical Top Base-Line
Spacing
Character Sp 1.000
Word Spacing 1.000
Line Spacing 1.000

Figure 5.20: *A new, editable Text object*

4. To change the words displayed from Text to whatever we want to say, we press **Tab** to enter **Edit Mode**, then **Backspace** to erase the word Text, and simply type.

   Pressing **Enter** will move the cursor to the next line and pressing **Ctrl A** will select all, in a similar way as many programs for editing text.

5. Since our word (or phrase, sentence, or paragraph) is currently off center, let's also change the Paragraph Alignment to Center via the Object Data Properties panel, in both the Horizontal and Vertical dimensions.

Properties panel, in both the Horizontal and Vertical dimensions.

6. Press **Tab** again or choose **Object Mode** from the upper left of the 3D Viewport, to continue editing the Text object.
7. From preceding the Paragraph section of the Text's **Object Data Properties** panel, let us also choose a new Font for the Regular font type, by pressing the icon appearing as a folder to the right of the heading Regular.
8. From the Blender **File View** that pops up, navigate to your system's Fonts folder, and choose any TTF type font you have installed.

Note: OTF or Open Type Fonts can be used in Blender but must first be converted to TTF via an external program. Many font converters can be found by searching online for "OTF to TTF converter."

9. Next, let us rotate the object 90 degrees along the x axis by pressing R, X, 90 then Enter, and rotate the view by clicking the middle mouse button and moving the mouse until we can see the Text object at an angle.
10. Back in the **Object Data Properties** panel, open the Geometry section, and in the Extrude parameter, click, and drag inside the parameter field to "scrub" the thickness of the text to about 0.1m (we can also click and type 0.1 inside the parameter field).

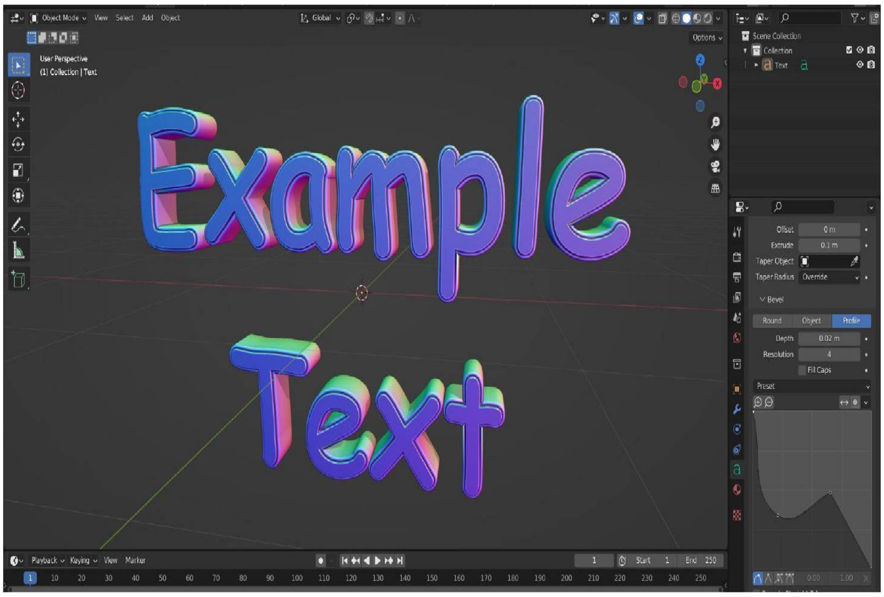
    Offset would expand each letter from the center, so we want to leave that parameter set to 0 for now, but feel free to experiment to see the results and revert.
11. Next, let's change the bevel Depth under the Bevel section in the Object Data Properties panel, from 0 to 0.02m, and switch the Bevel type to Profile.

Note: Through experimentation with various parameters such as Bevel Depth, we may have discovered that some operations can cause unwanted twisting and overlap of geometry in corners and at extremes. In many contextual menus, an option called "Clamp" may be enabled, to stop the creation of such overlap, while other times it may be necessary to simply enter parameters cautiously.

Enabling clamp also has the effect of preventing deep bevels because it stops the bevel operation when it would otherwise cause overlap. This is expected behavior, since such operations can produce "impossible" geometry otherwise, when two vertices are scaled past one another at corners, for example, but the limitation can require some advanced planning to work around

Clicking the Profile Bevel type in the Text's **Object Data Properties** panel opens a new section appearing as a square divided diagonally by a straight line. This diagonal line represents the side profile of the shape that is currently affecting the Text object's edges, in a similar way that the Falloff Type of Proportional Editing mode controls the falloff's side profile as discussed earlier in this chapter.

12. To change the Bevel Profile from flat to something more interesting like a routed edge, click along the diagonal line to add one or more points to the Profile curve, and then click and move each point, or choose from one of the available Presets in the drop-down, observing the resulting effect on the Text object as shown in *Figure 5.21*:



Offset 0 m
Extrude 0.1 m
Taper Object
Taper Radius Override
Bevel
Round | Object | Profile
Depth 0.02 m
Resolution 4
Fill Caps
Preset

Figure 5.21: Example 3D Text object extruded 0.1m in the Comic Sans font with a custom Bevel Profile

Tip: Many of the figures throughout this book – such as Figure 5.21 – show 3D objects in a colorful Viewport Shading mode called a MatCap. MatCaps are preset surface shading modes that fake lighting, shading, and color, which is very useful for visualization. MatCaps are not exported with 3D objects, do not alter textures or materials, and do not appear in renders (2D images) made with 3D cameras.

To enable different MatCaps, click the down arrow appearing to the right of the Viewport Shading options in the upper right of the 3D Viewport, switch Lighting from Studio to MatCap, and click on one of the colorful spheres there. To add the faked shadows and highlights as shown in Figure 5.21, enable the options for Shadow and Cavity below the panel of spheres.

To render exactly what is seen in the 3D Viewport including gizmos, grid lines and MatCaps, in the upper left of the 3D Viewport, choose View > View Render Image. A new Blender Render window will open with an image that can be saved from the Image menu in the upper left of the window.

Handle types

Handle types

Control points along a curve's path are the corollary to vertices along an edge in a Mesh object, but as discussed in the previous section of this chapter, unlike Mesh

objects, Curve objects are not restricted to having straight lines between these points until they are converted to Mesh. Also, unlike the edges and vertices of Meshes, the control points and path segments of Curve objects act more like tent poles than actual geometry, forming the underlying structure over which an infinite curve is draped like fabric depending on the Curve object type.

At the bottom of the Bevel Profile panel we opened in the previous step, there appear four Handle Type options: **Auto**, **Vector**, **Free**, and **Aligned**. Whether it is a Profile curve like the one we just edited for the Bevel, or a Curve object, or an Animation Curve which we will explore in *Chapter 8, 3D Animation*, the Handle Type of any curve changes the blending, or "interpolation" between the incoming and outgoing path segments from a given control point along the curve.

By default, newly added points along the curve in the **Bevel Profile** panel will be set to Auto, which causes incoming and outgoing path segments to curve based on the angle and distance of the point from neighboring points on either side. The Vector Handle Type behaves in a similar way, in that the interpolation is automatic rather than manually adjustable, except the Vector type sets the incoming and outgoing segments to a straight line, creating a flat or sharp angle.

Free Handle and **Aligned Handle** Types both allow manual adjustment of this interpolation through clicking and dragging two handles that extend beyond the control point, and both allow the blending effect to be freely increased or decreased. Aligned type handles remain parallel to each other like a teeter-totter and remain the same length when scaled via clicking and dragging. On the other hand, Free Handles can be "broken" and rotated 360 degrees in different directions around the control point (which may cause unwanted buckling or a loop in the path). Points set to the Free Handle type can also be given a sharp angle on one side of the point like Vector, and a curve on the other like Auto.

Note: Throughout the chapter we have referred to Curve objects with a capital C to distinguish between the common language term "curve" and the 3D object Curve. At times we have also referred to various curves as paths, but within Blender's interface and in other 3D applications that utilize Curve objects, a curve may be alternately referred to as a Spline or even a Vector. Each of these terms has its own English language and mathematical definitions, but for the purpose of 3D creation it is helpful to consider them one and the same.

Converting curves to mesh and mesh to curves

As we wrap up this exercise, it is helpful to note that while the Text object in Blender is a special kind of Curve, it must first be converted to a true Curve for its control is a special kind of Curve, it must first be converted to a true Curve for its control points and path segments to be edited like one. Converting a Text (or Mesh) object to Curves is as simple as selecting the object in **Object Mode**, then in the menu at the top of the 3D Viewport choosing **Object** > **Convert** > and Curve or choosing Mesh

to convert Text and Curves to Mesh. When we convert a Text object into a Curve or a Mesh, the text editing options go away, and we can only edit the individual letters as if they were any other 3D shape.

Tip: Converting non-curve objects to Mesh – even if they are already Mesh – via the Object > Convert > Mesh command is also a handy way to Apply all the Modifiers on a selection, even if a group of items is selected. Because Modifiers can only be applied on the Active selection via the Modifier panel, using Convert > Mesh can be a significant time-saver.

Additional primitive objects

In *Chapter 3, General 3D Concepts* we learned about an additional primitive object, the Single Vert. Under **Edit** > **Preferences** > **Add-ons** as shown in *Figure 5.22*, there are several additional primitive objects under the Add Mesh and Add Curve headings that can also be enabled, including Archimesh and BoltFactory, Discombobulator, Extra (Curve) Objects, IvyGen and Sapling Tree Gen.

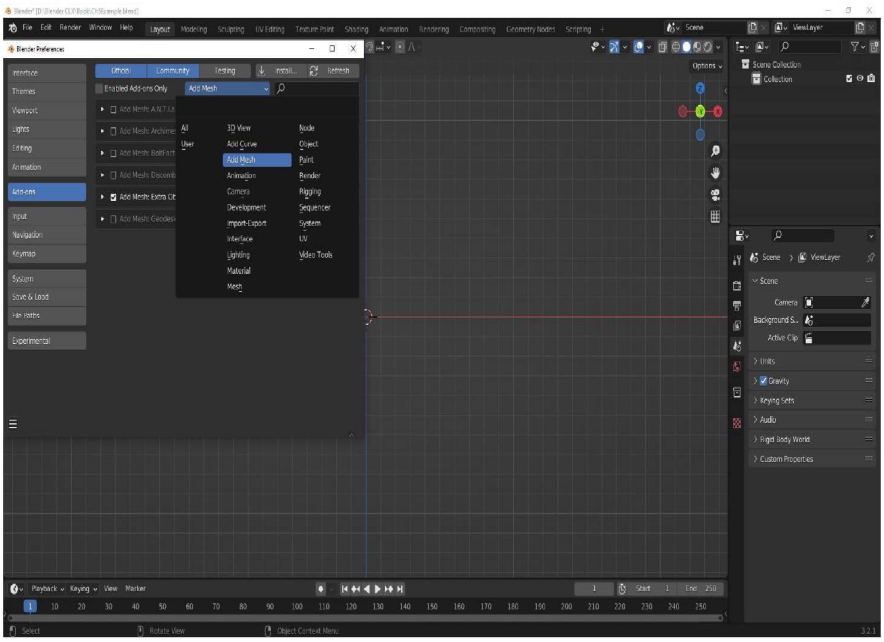


Figure 5.22: Additional primitive objects available in the Edit > Preferences menu

Each of these options is versatile enough that a whole chapter could be devoted to their use, so rather than outlining each one, we will simply note that when we come across a need for a common shape, chances are very high that there is either a default Blender add-on to handle it, or someone somewhere on the web is providing a free or paid plug-in we can download or buy. And if not, we might be on to a marketable idea that we could develop ourselves!

Conclusion

In this chapter, we have learned about every additional tool in the **Blender Tool bar** menu and rounded out our understanding of the 3D Viewport with a deep dive into Transform Orientations and Proportional Editing. We've also gained an introduction to the power of using Modifiers as removable containers that affect 3D objects in a variety of ways. We learned the unique properties of 3D objects called Curves, which can be converted to Mesh and back again, and practiced using a special type of Curve object, Text, which enables the creation of editable 3D text.

As always, we are reminded that the official Blender Manual is the ultimate reference for in-depth explanations of Blender's capabilities. Having had an introduction to these tools will serve us well as we begin expanding our use Blender into creation of 3D objects of greater complexity and customization.

Essential shortcuts and hotkeys

In this chapter, we used the following additional Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete

Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G

Shortcut	Hotkey
Shrink and Fatten (in Edit Mode)	Alt + S
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show / Hide Side Bar	N
Show / Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D

Fill Selected (in Edit Mode)	F

Table 5.1: Essential Shortcuts and Hotkeys

Shortcut	Hotkey
Lock Number Pad	Num Lock
Local View	Num /
N/A	Num *

Shortcut	Hotkey
Zoom Out	Num -
Top View	Num 7
Local Top View	Shift + Num 7
Bottom View	Ctrl + Num 7
Rotate Up	Num 8
Reverse View	Num 9
Zoom In	Num +
Rotate Left	Num 4
Perspective / Orthographic	Num 5
Rotate Right	Num 6
Front View	Num 1
Local Front View	Shift + Num 1
Rear View	Ctrl + Num 1
Rotate Down	Num 2
Right Side View	Num 3
Local Right Side View	Shift + Num 3
Left Side View	Ctrl + Num 3
Camera View	Num 0
Frame Selected	Num .
Confirm	Num Enter

Table 5.2: Viewport Navigation - Shortcuts and Hotkeys

Points to remember

- Blender's Tool bar menu gives us quick access to commonly used poly modeling tools in common configurations.
- The default Global Transform Orientation is all purpose, while the alternate

- The default Global Transform Orientation is all purpose, while the alternate Transform Orientations are situational.
- Modifiers are a powerful way to make temporary changes to a mesh which can later be made permanent through the Apply command.
- Curve objects are the 3D corollary to 2D vectors.
- The official Blender Manual is the ultimate reference for in-depth explanations of Blender's capabilities.

Questions

1. How is adding a primitive through the Add menu different from adding it through the Tool bar menu?
2. Which operations have we used where Proportional Editing was an optional parameter?
3. Where could we have used Modifiers in the exercise from *Chapter 4, Polygonal Modeling*?
4. Which of the built-in primitive objects might speed up our workflows that we haven't explored?

Join our book's Discord space

CHAPTER 6
3D Sculpting

Introduction

Now that we have learned and practiced polygonal modeling to create 3D base meshes, we are better equipped to begin adding high resolution details to our creations by sculpting 3D objects as if from clay. First, we will consider the appropriate approach depending on the desired result. Whether the aim is a low poly mesh that is performant for games and interactive applications, or a high-resolution object for print or render, 3D sculpting can be used to add key finishing touches.

Structure

This chapter covers the use of Blender's sculpting tools and multi-resolution workflows to create high poly 3D objects for high-detail applications such as 3D printing, or as we will practice here, for giving a low poly model the appearance of higher detail. Topics to be covered include:

- When to sculpt 3D models, and how to plan
- Low to high versus high to low workflows
- Layering detail and controlling mesh resolution
- Blender sculpting tools overview

- Exercise: Sculpting a leaf detail for the pedestal

Objectives

After reading this chapter, we will understand how and when to add high resolution details to our 3D objects and how to plan for revising the topology of high resolution meshes for a low poly goal. We will also be prepared for transferring high resolution detail onto low poly meshes through surface treatments we will discover in *Chapter 7, 3D Surfaces*, while having gained hands-on experience with sculpting by creating a decorative leaf detail for the pedestal object from previous exercises.

Planning to sculpt

Now that we understand how to create a base mesh upon which to sculpt using poly modeling techniques, planning to sculpt is simply a matter of picking the workflow that will get us to our desired result in the most accurate and efficient way. The first consideration to plan for is the desired topology of the resulting mesh. Questions to ask during the planning phase include:

- Does the result need to be optimal for real time rendering, games, animation, or interactivity?
- Does the result require freedom to create without limits, as if working with a raw lump of clay, or does the desired result have precise requirements?
- Does the whole mesh need to be intricately detailed or just part?
- Will I need to revise the overall shape as I go along, or will I just add detail to a finalized design?

Low poly goal

In *Chapter 4, Polygonal Modeling*, we explored the concept of logically increasing complexity, from large, simple forms, to small, complex detail in the section on Low to High, Large to Small. In this chapter, we are shifting to a similar but separate question of whether to start sculpting from high-resolution or low-resolution base meshes, or some combination of the two. The answer depends entirely upon our desired result for any given project.

When the desired result will be animated or rendered in real time with interactivity – such as in a game or on the web – we will need the result to be low poly. To achieve a low poly result, whether we start from a high-resolution or low-resolution base mesh – or even sculpt only part of the mesh as we will do in *Exercise 6.1* – is mostly a matter of personal preference. In this case, there are two equally valid approaches.

One approach, **low to high**, is to begin by poly modeling and finalizing the desired overall form in low poly, and either sculpting higher-resolution details directly onto a subdivided duplicate of the low poly version or using a Multiresolution Modifier to allow sculpting on the low poly version itself. Both workflows involve proceeding with the intention of transferring high-resolution detail onto the low-resolution base mesh later as a surface treatment.

Note: Subdividing any 3D object through any means – including manually or with the Multiresolution Modifier – has the potential multiply the number of polygons on screen so rapidly that a computer cannot handle displaying them. This can cause a software crash, or in the worst case, even a hardware crash. Aside from potential data loss, it is usually possible to recover from such an accident by restarting, but it is wise to avoid the risk altogether, and proceed slowly whenever increasing a parameter.

The opposite approach, **high to low**, involves free sculpting any high-resolution mesh – such as a heavily subdivided sphere or cube – with the intention of later making a low-resolution duplicate that conforms to the surface through a process called "**retopology**."

This low-resolution duplicate can be made either through a time-consuming manual process of drawing new vertices, edges, and faces all over the high-resolution mesh, or by using an algorithm like *Remesh* or *Decimate*. These algorithms use different methods to automatically reduce the number of vertices, edges, and faces of a mesh, with the tradeoff of less predictable, less intentional edge flow than can be produced manually. This randomness can negatively impact later operations such as animation or adding surface treatments through texture and materials.

One of the advantages of starting a low poly goal by creating a very high-resolution, or highly "subdivided" mesh, is the creative flexibility this approach affords. Like beginning with a lump of clay, we can allow our imaginations to run free with a high-resolution mesh, pushing and pulling wherever we are inspired to do so in the moment. On the other hand, 3D objects sculpted entirely from the imagination can be quite challenging to retopologize.

The main disadvantage to starting from a high-poly base mesh is the time and effort required to bring the result back to a more readily usable, low poly state when needed. However, since the pre-planning needed to create a low poly mesh that allows for later creative touches is also effort intensive – just on the front end – the two approaches are nearly identical and mostly up to personal preference.

High poly goal

When the desired result does not need to be animated or rendered in real time – such as with sculptures intended for 3D printing, or fantastical creations only meant for 2D render – our creations can both begin and remain high poly throughout the process. In this case, starting from a high-resolution mesh may be inconsequential, as long as we are mindful of how many polygons we add onscreen at one time, to prevent significant slow-down of our sculpting process.

This scenario is a great use case for the Dynotopo sculpting mode which appears in the Sculpt Tool bar **Brush Settings**. Dynotopo (dynamic topology) allows us to only add more resolution where we add brush strokes, leaving the rest of the mesh low poly enough to keep our computers running smoothly. In this paradigm, we will still want to start by sculpting larger, simpler forms and work our way down to smaller, more complex details, but the base mesh can be more subdivided, and the direction of our creative impulses less restricted.

Destructive or non-destructive

In previous chapters, we have briefly discussed a second consideration for sculpting, which is whether to use a destructive or non-destructive workflow. Destructive essentially means permanent or irreversible beyond the **Undo History** (which, as a reminder, is not saved in between closing and reopening the application). In *Chapter 5, Poly Modeling Extras*, we learned that while Modifiers remain unapplied, they are non-destructive because they can be removed at any time and the underlying mesh will resume its original shape.

Note: The impermanence of Modifiers also means that changes to a mesh that are achieved through them – such as sculpting done with the Multiresolution Modifier – are permanently lost when the Modifier is removed. To preserve the effect and not lose the underlying mesh, make a duplicate by pressing Shift + D, then remove the Modifier from one copy, and apply the Modifier to the other.

To determine the best approach, it is helpful to identify a personal level of comfort with potentially losing hours of work, versus accumulating an unmanageable history of mistakes, and proceed with that in mind. A popular compromise is to frequently create and hide duplicates at various stages, then delete the copies from time to time.

What to save and when to save, in 3D as with many digital art forms, is a complex dance. It may help to reflect on the fact that in analog creation, there is often no **Ctrl + Z**. We cannot unbreak an egg or uncut timber. Embracing the need to start over, when it arises, can be quite helpful. Some people believe that the second time will always be better.

Resolution and subdivision

All 3D sculpting operations are either limited by existing topology in how far the brush can move the mesh, or we must subdivide the mesh – either manually or algorithmically – to expand its limits. If all we want to do is move existing vertices algorithmically – to expand its limits. If all we want to do is move existing vertices

around, we might as well use Proportional Editing in **Edit Mode**, but if we want to carve into and layer onto new geometry, the topology of our mesh will be affected.

To manually create the needed resolution to employ the full power of sculpting tools, to stretch and carve further, we can enter **Edit Mode**, select the geometry we plan to sculpt, and under the **Edge menu**, choose **Subdivide** and increase the Number of Cuts parameter in the contextual menu. Alternately, we can add a Subdivision Surface Modifier to our object, increase the Levels, and then apply the Modifier.

Adding then applying a Subdivision Surface Modifier is often preferable to manual subdivision because it also applies smoothing – a kind of melting algorithm – which helps rounded surfaces retain their rounded appearance as shown in *Figure 6.1*. Manually subdivided flat surfaces will remain flat, so conversely, the Subdivision Surface Modifier can have the unwanted effect of rounding surfaces we want to remain flat or sharp. Please refer to the following figure:

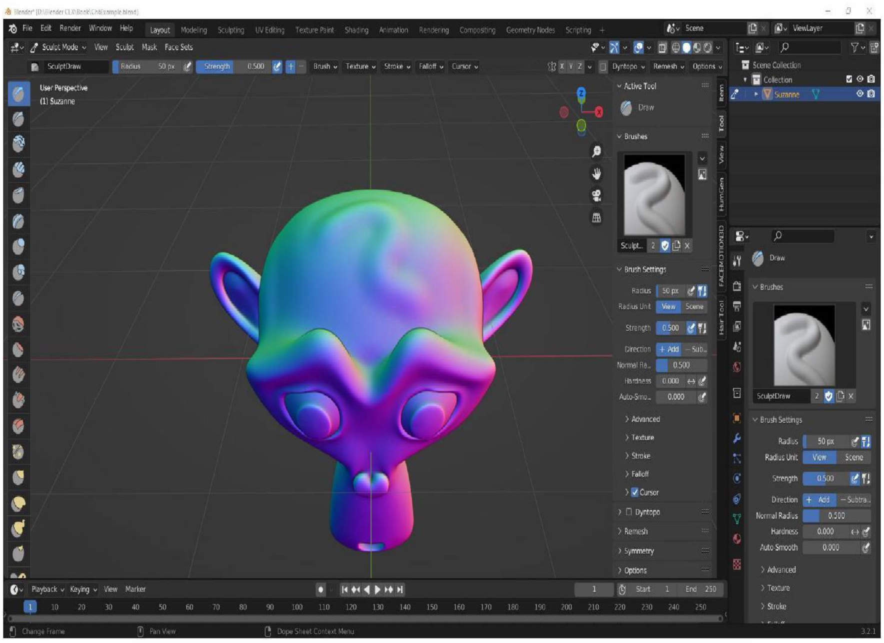


Figure 6.1: Subdivision Surface Modifier applied before sculpting, subdividing the whole mesh

Note: The object's shading settings will also impact the appearance of smoothness, as we learned in Chapter 4, Polygonal Modeling. When beginning to sculpt a new mesh, it is often necessary to set the object's shading to Shade Smooth and enable Auto Smooth in the Normals section of the object's Object Data Properties tab of the Properties panel to achieve the intended appearance. Smooth shading will not physically smooth the surface in the way that a Subdivision Surface Modifier does.

While some Modifier effects can be constrained to specific vertex groups, another significant drawback to adding and applying a **Subdivision Surface Modifier** is that it subdivides the whole mesh, including areas that do not need it. This can cumulatively slow down our computer's performance while sculpting, which is a problem even if our desired result is a high poly mesh.

As mentioned earlier, Blender's sculpting tools can also create high resolution only where we need it, using the Dynotopo (dynamic topology) sculpting mode, but this method comes with one significant drawback too: enabling Dynotopo triangulates the whole mesh, potentially destroying carefully planned topology as shown in *Figure 6.2*. All these options require careful weighing of the respective pros and cons.

Tip: In Blender, there is an algorithm that can attempt to convert a triangulated mesh back to quad topology, especially if it was originally modeled that way. However, because algorithms lack human intelligence, results may vary. To attempt to convert a triangulated sub-object selection into quads, in Edit Mode, press Alt + J.

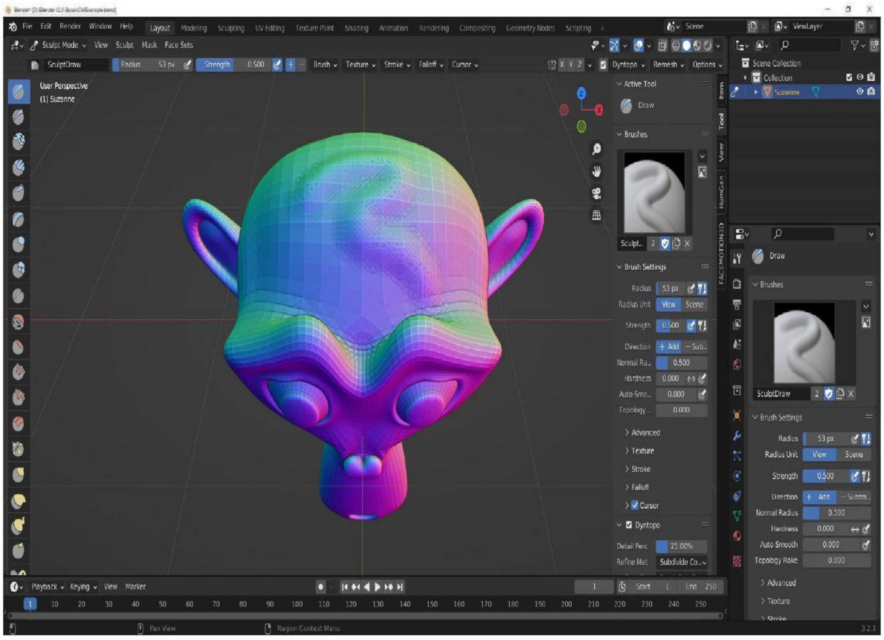


Figure 6.2: Dynotopo sculpting mode enabled, subdividing the mesh only where more resolution is wanted

Starting to sculpt

With a 3D object selected, from the upper left-hand corner of the **3D Viewport**, beneath the **Object Mode** and **Edit Mode** options, is the next mode we will explore:

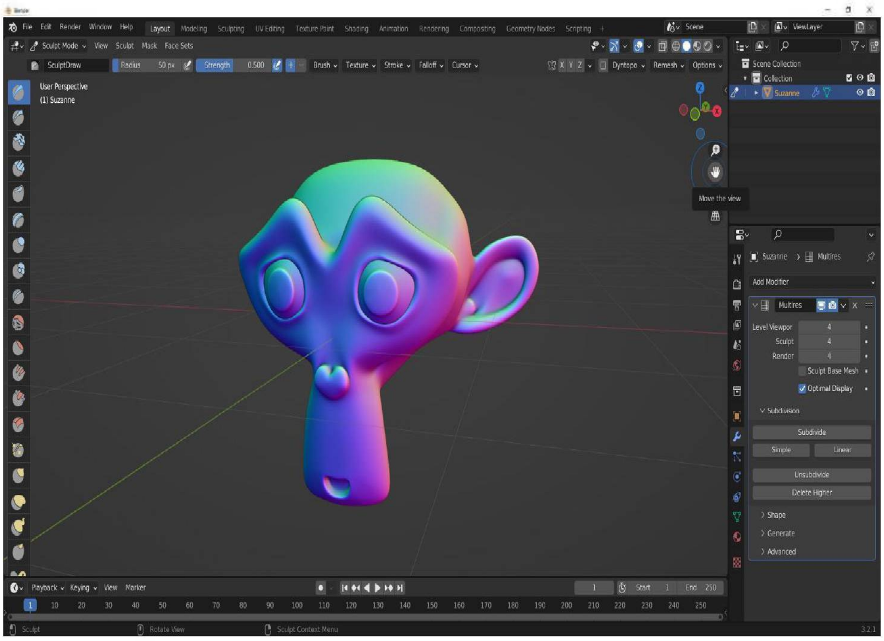
Sculpt Mode. From **Sculpt Mode**, we can see that the **Tool bar** menu has expanded, so much so that a scroll bar appears to the right of the tools.

Note: The Tool bar menu is collapsed by default, to save space, but if we hover our mouse over the space just to the right of any tool until the double ended arrow icon appears, we can click and drag the toolbar to make it wider, revealing text labels for all the tools.

Brush categories

In Blender, the physical action of sculpting is performed like digital painting, in that we select a tool to apply to the object's surface through movements of the mouse akin to brush strokes. In 3D, just like real life, painting and sculpting tools come in all shapes and sizes, with narrow pointed tips or wide fan bristles. Even sponges, spatulas, or the sharp end of a brush can be used to create certain effects.

As with many features of Blender, to gain a better understanding of each tool's capabilities, it is highly advisable to create throwaway projects just for the purpose of experimentation. To test out various Sculpt Mode brushes, a very helpful experiment is to add a Monkey (Suzanne) object to the scene, then a Multiresolution Modifier to that, increasing the Subdivision to 4, setting the Object to Shade Smooth, and testing out various brushes in this setup as shown in *Figure 6.3*.


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Sculpt Mode View Sculpt Mask Face Sets
SculptDraw Radius 50 px Strength 0.500 Brush Texture Stroke Falloff Cursor
Dyntopo Remesh Options
User Perspective
(1) Suzanne
Move the view
Scene ViewLayer
Scene Collection
Collection
Suzanne
Suzanne > Multires
Add Modifier
Multires
Subdivision
Subdivide
Simple
Linear
Unsubdivide
Delete Higher
Shape
Generate
Advanced
Playback Keying View Marker
Start 1 End 250
Sculpt Rotate View Sculpt Context Menu

Level Viewport 4
Sculpt 4
Render 4
Sculpt Base Mesh
Optimal Display ✓

Figure 6.3: Multiresolution Modifier added, with Levels of Subdivision increased to 4

As with painting, however, it isn't necessary to preview the properties of every possible option if we know what to try that may produce the creative effect we have in our minds. Like the **Modifiers** panel, the available options under Blender's Sculpt **Tool bar** menu are too numerous to outline in a single book. Instead, we will look at the major categories of Blender's Sculpt tools, to get a better idea of the purpose and use of each.

The first two major categories of brushes in Blender's Sculpt Mode Tool bar menu can be considered additive and subtractive, indicated by blue and red icons respectively. Below the blue and red icons, the brushes with yellow icons can be considered as movement brushes. Below these two categories is a mixture of mode-specific tools like masking, and below that, more familiar functionality such as the basic transform tools. As usual, accessing the tool tip by hovering the mouse cursor over the associated icon is an important way to become familiar with each option, as well as learning its Shortcut.

Brush settings

Whether we move the mouse back and forth across the surface in a rapid and haphazard manner or slowly and precisely also affects whether the effect we achieve is more organic and layered or chiseled and mechanical. The settings we choose affect how far and how much the effect is spread. Frequently increasing and decreasing various settings such as the brush size and strength while sculpting is very common practice.

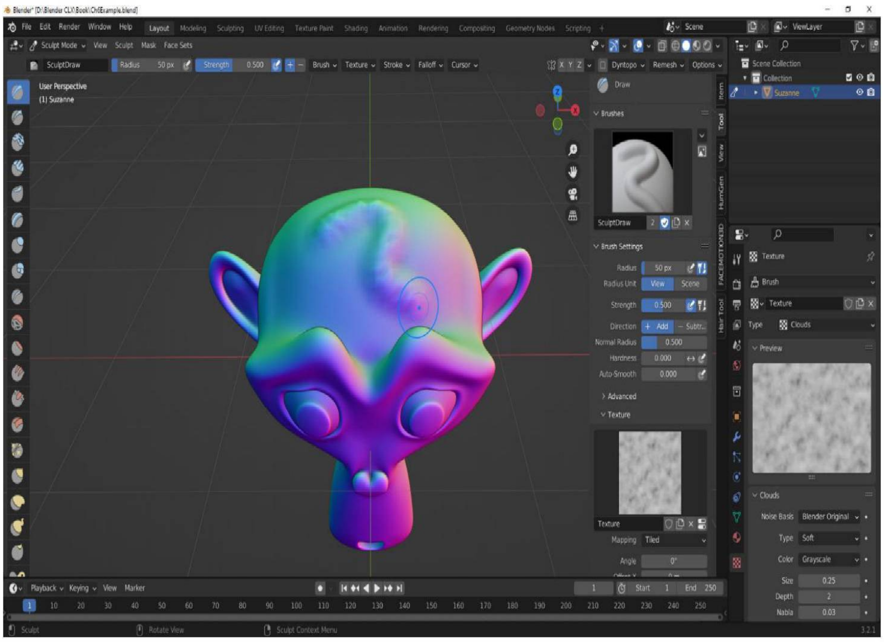
In Sculpt Mode with a brush selected, in either the **Sidebar** menu (toggled by pressing the **N** key) under the **Tool tab**, or from the **Properties** panel under the **Active Tool** tab as shown in *Figure 6.4*, each brush presents a variety of settings which will affect its operation:

Note: Brush Settings are specific to the currently selected Sculpt Tool, and changes to the settings will persist for each brush, even after saving the file. This means that when we reselect a brush after using another, the settings we previously set will persist. New settings, when desired, must be chosen for each brush upon first selecting.

Commonly adjusted settings also appear across the top of the **3D Viewport** in **Sculpt Mode**, and include:

- **Radius**, the overall scale of the brush
- **Strength**, the intensity of the stroke
- **Brush**, the hardness of the brush edge and auto smoothing of the stroke
- **Texture**, a greyscale image that can be added to adjust the current tool's intensity in a pattern as shown in *Figure 6.4*
- **Stroke**, the spacing of the stroke and jitter (randomization)
- **Falloff**, the side profile of the brush effect

- **Cursor**, the appearance of the mouse cursor
- **Symmetry**, whether to mirror strokes across the mesh and in which axes


Blender* [D:\Blender CLX\Book\Ch6Example.blend]
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Scene
ViewLayer
Sculpt Mode View Sculpt Mask Face Sets
User Perspective
(1) Suzanne
Dyntopo Remesh Options
Scene Collection
Collection
Suzanne
Playback Keying View Marker
Sculpt
Rotate View
Sculpt Context Menu
3.2.1

SculptDraw Radius 50 px Strength 0.500 Brush Texture Stroke Falloff Cursor

Draw

Brushes: SculptDraw 2

Brush Settings
Radius 50 px
Radius Unit View / Scene
Strength 0.500
Direction + Add / − Subtr...
Normal Radius 0.500
Hardness 0.000
Auto-Smooth 0.000

Texture: Texture
Mapping Tiled
Angle 0°

Texture: Brush, Texture
Type Clouds

Clouds
Noise Basis Blender Original
Type Soft
Color Grayscale
Size 0.25
Depth 2
Nabla 0.03

Start 1 End 250

Figure 6.4: Cloud Texture added to Draw brush creating a textured stroke

Tip: A helpful behavior of Sculpt Mode Tools is that additive and subtractive brushes can be made to produce the inverse effect by pressing and holding the Ctrl key while "painting." Most brushes also switch to the Smooth tool when used with the Shift key pressed.

Exercise 6.1

In this exercise, we will revisit the pedestal we began poly modeling in *Chapter 4, Polygonal Modeling*, to add the remaining detail from the reference sketch: a sculpted leaf detail.

1. To begin, open the saved pedestal project or follow along with a new cylinder.

   Referring to the pedestal reference sketch from *Chapter 4, Polygonal Modeling* as shown in *Figure 6.5*, we can see one remaining detail: a sort of leaf shape repeated under the cap, intentionally left open to interpretation:

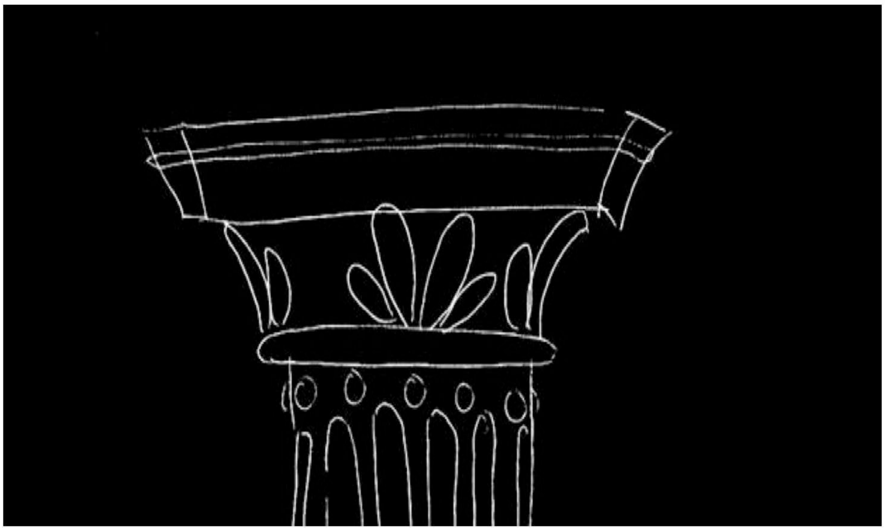


***Figure 6.5:** Leaf shaped detail from hand drawn sketch of pedestal concept*

   In the exercise where we created the base for this detail, we scaled a section at the top of the cylinder to be larger than the body and added a bevel to make it curve inward. The underlying topology is clean and meant to be low poly enough to make the pedestal useful for games and other interactive applications, so we won't be subdividing the whole object to add this detail.

2. With the Pedestal object selected, enter **Edit Mode**, and using Circle Select by pressing the **C** key and right-clicking and dragging, paint a selection of just the front-most faces in this curved top section, as shown in *Figure 6.6* middle mouse-click if needed to deselect anything outside this section, then right-click to confirm.

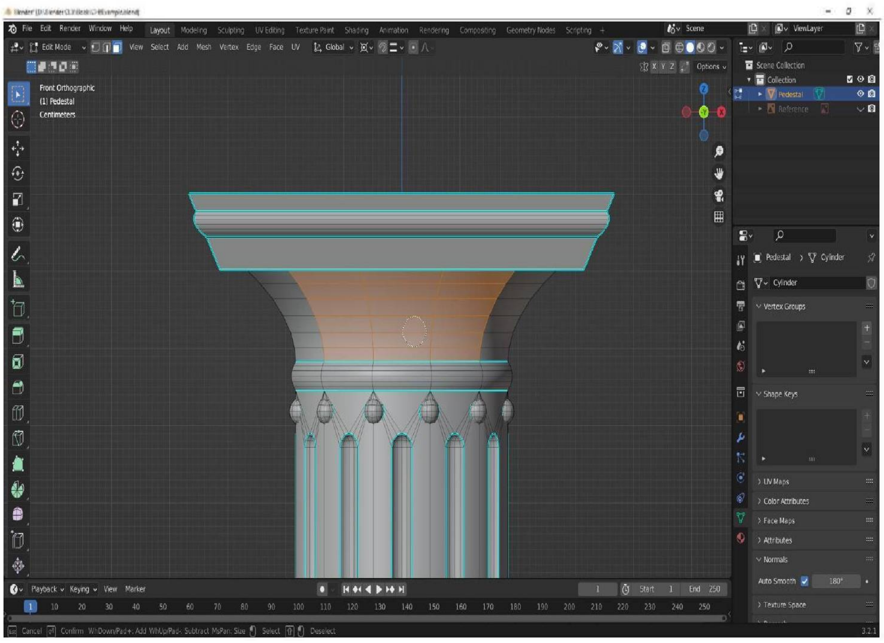


Figure 6.6: Circle Select mode enabled to paint a selection of the geometry we intend to sculpt

3. With this section selected, let us duplicate the geometry by pressing **Shift D**, and right-click to confirm.
4. Without deselecting the duplicated geometry, press **P** and choose Selection, to separate the selected geometry as a new object.
5. Exit **Edit Mode** and select only the newly duplicated geometry now named Pedestal.001.

   This is the start of what will be the base mesh for our sculpting work.
6. In the **Properties Panel**, **Modifier Properties tab**, add a new Subdivision Surface Modifier to Pedestal.001 and increase the Levels Viewport setting to 4.
7. Press the **Shift + H** keys to hide all but the new geometry, and once again enter **Edit Mode**.
8. Next, with Circle Select again, paint a selection over just the middle column of faces.

9. Under Mesh in the upper-left menu of the **3D Viewport**, select **Split** > **Selection** as shown in *Figure 6.7*:

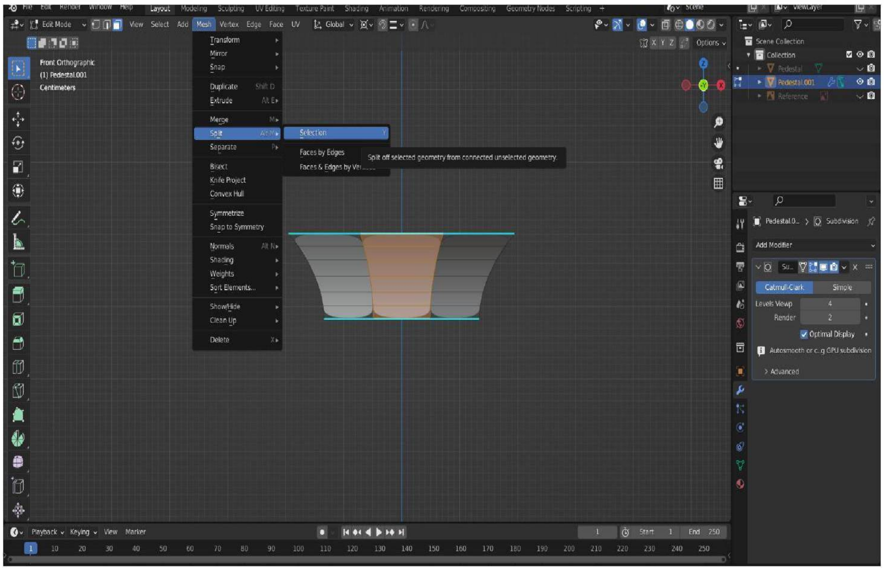


Figure 6.7: Middle column of faces selected

Next, we will give the three separate base meshes some thickness.

10. In **Face Select mode**, clear the selection, and select all.

11. Press E to extrude, and move the mouse cursor downward, with no additional keypresses until the new leaf shapes are approximately as thick as each of the horizontal subdivisions are tall, as shown in *Figure 6.8*, then left-click to confirm.

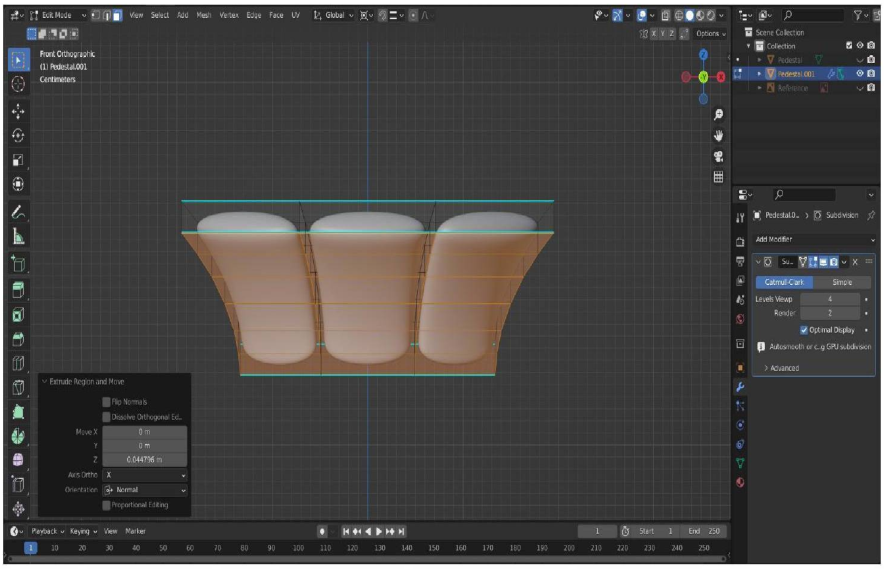
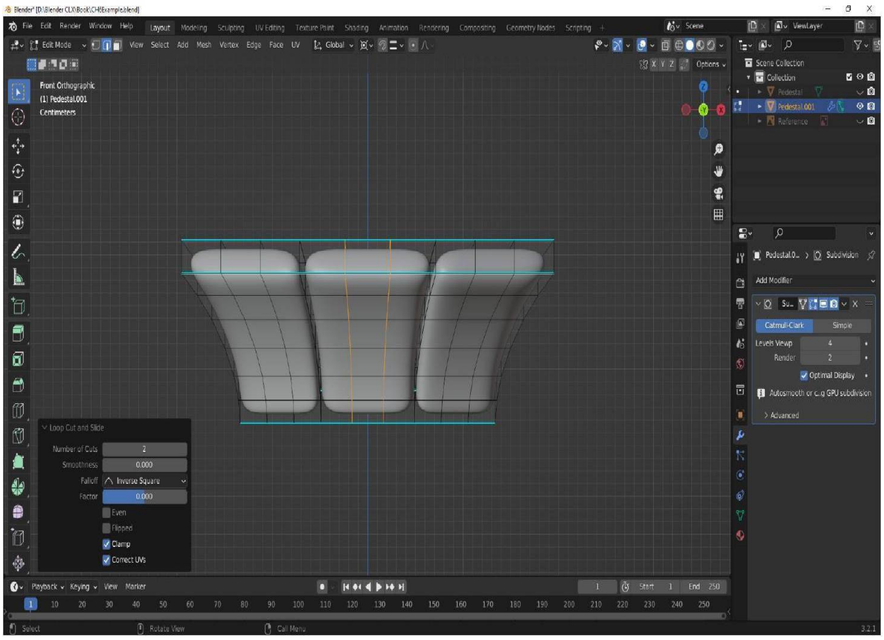


Figure 6.8: Extruded leaf shape base meshes

12. Let us also add some vertical edge loops, so that the new topology of these bases for our leaf detail will be more evenly distributed, by pressing **Ctrl + R**. Hover the mouse cursor over a horizontal edge and roll the mouse wheel up one notch to add two vertical edge loops to the middle of each of the three shapes as shown in *Figure 6.9*, and left-click, then right-click to confirm:


Loop Cut and Slide

Number of Cuts	2
Smoothness	0.000
Falloff	Inverse Square
Factor	0.000
Even	☐
Flipped	☐
Clamp	✓
Correct UVs	✓

Figure 6.9: *Base meshes for three leaf details with edge loops added for even distribution of resolution*

Next, let us elongate the leaf shapes.

13. First, we will ensure that the Transform Pivot Point is set to Bounding Box Center, in the upper middle of the **3D Viewport**.

14. Then in **Face Select mode**, **Alt + click** along one of the middle horizontal edges of each of the three shapes to select the middle face loop of each.

15. Press the **S** key to begin scaling, then press the **Z** key and move the mouse cursor upward to stretch the leaf meshes into an oblong shape along the z axis as shown in *Figure 6.10*:

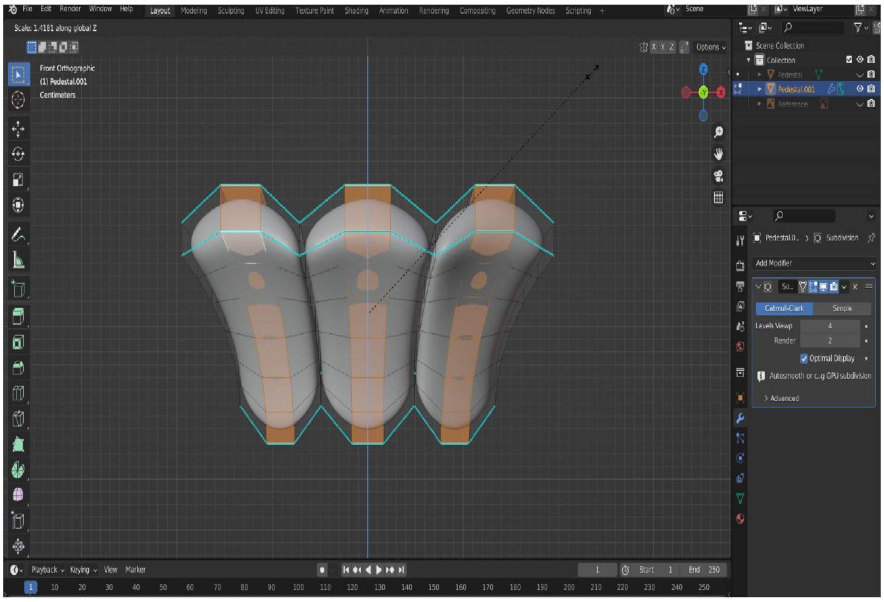


Figure 6.10: *Scaling the leaf detail base meshes along the z axis*

Finally, before we begin sculpting, we need to apply the Subdivision Surface Modifier. To apply any Modifier, we must first exit **Edit Mode**.

16. After pressing **Apply** in the Modifier's upper right-hand drop-down menu, if we enter **Edit Mode** again, we can see what 4 levels of subdivision looks like as shown in *Figure 6.11*:

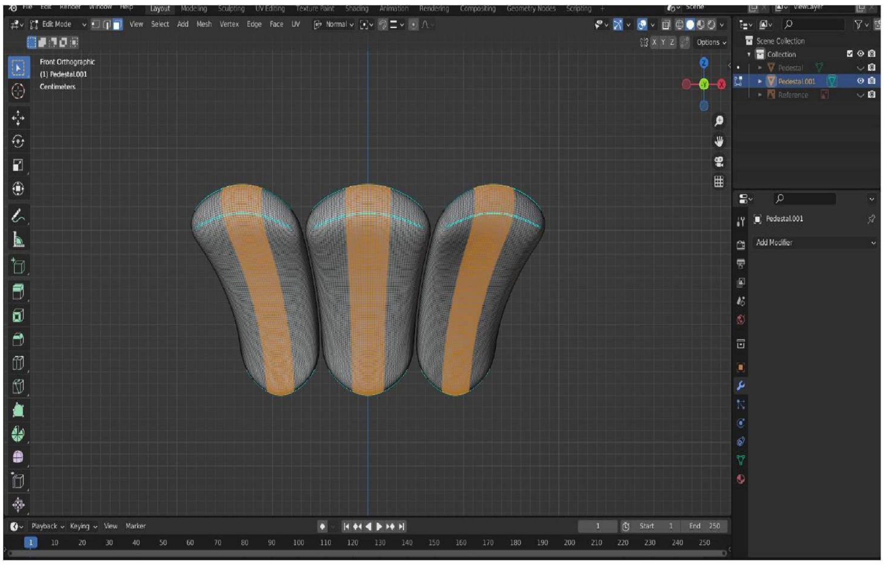
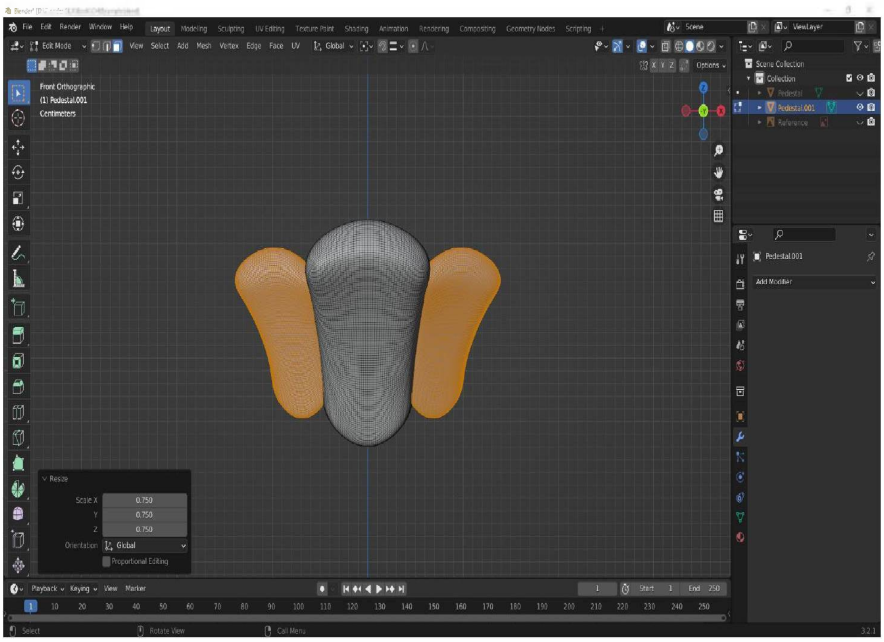


Figure 6.11: *Mesh with applied Subdivision Surface Modifier set to 4 Levels, as seen in Edit Mode*

This view reminds us that the Split operation left sharpened edges around the top and bottom of each shape.

17. To correct this, clear the selection, select all, then press **Ctrl +E,** and from the **Edge menu** that appears, select **Clear Sharp**.

18. As one last touch of preparation, let us select just the left and right shapes and scale them smaller until they are nestled against the middle shape, as shown in *Figure 6.12*:


Front Orthographic
(1) Pedestal.001
Centimeters

Resize
Scale X 0.750
Y 0.750
Z 0.750
Orientation Global
Proportional Editing

Figure 6.12: Resized left and right leaf base mesh shapes

Note: The previous sequence of steps, scaling the vertical middles of the three shapes while they were still the same size, ensured that the overall scale of the shapes remained proportionally similar. If we had first scaled the side shapes smaller than the middle, and then scaled the vertical height of all three, the smaller shapes would not be equally as rounded as the middle shape.

Now, we are ready to begin sculpting.

19. First let us switch to Sculpt Mode.

    Next, we will enable symmetry, to save effort sculpting on each side, and make the results more precise. Using symmetry is a personal preference depending on the desired result. If a more organic result is desired it is fine to skip this step, or even experiment with enabling and disabling it off and on as we work.

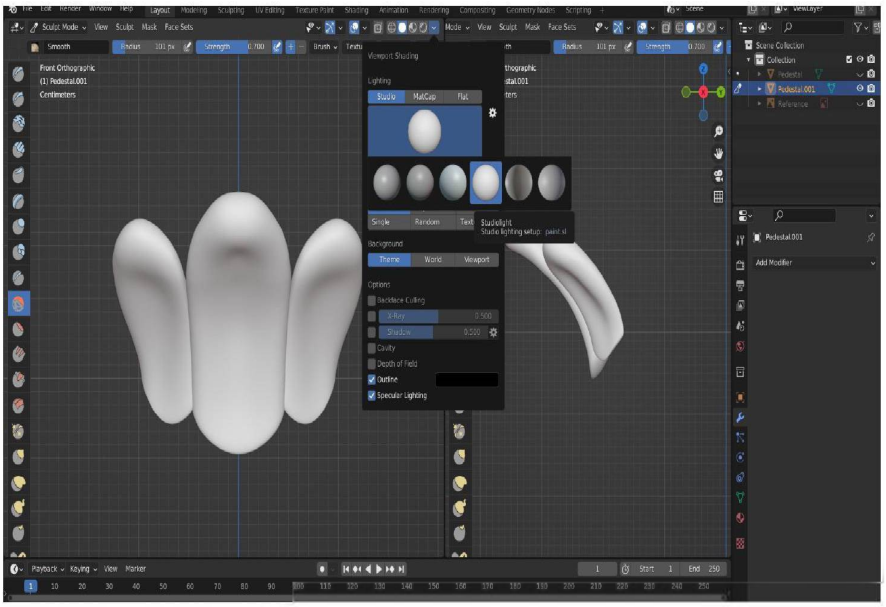
20. To enable symmetry along the x or front axis, in the upper right of the **3D Viewport** in Sculpt Mode, click the X button next to the icon appearing as a butterfly half drawn with dotted lines.

Tip: Splitting the view as we first practiced in Chapter 2, Installation and Interface is very helpful to the process of sculpting.

From here on out, it is recommended to test out each brush, and try different **Radius**, **Strength**, **Texture**, **Stroke** and **Falloff** settings as we go. To sculpt one area of the mesh without affecting another, we can enter **Edit Mode** to hide selections and reenter **Sculpt Mode**. Whenever a brush stroke produces unexpected results, it may help to remember the underlying topology, or even view it in **Edit Mode** for clarification. We can undo individual strokes via **Ctrl + Z** if we do not like them, keeping in mind that just-changed brush settings will be undone as well.

21. Zoom in and zoom out and rotate the view to get a better look as needed and notice how the brush size stays relative to the view.

Tip: The default Viewport Shading mode, Studio Lighting Default, fakes angled shadows across the mesh which can deceptively cause sculpted shapes to appear asymmetrical even when they are not. For sculpting, it may be helpful to switch to a Viewport Shading Lighting mode that fakes straight-on lighting instead. Accessed under the upper right-most drop-down menu in the 3D Viewport by clicking the shaded sphere, multiple other settings such as the one shown in Figure 6.13 can be chosen to give a more realistic impression of brush stroke effects.



[illegible]

Figure 6.13: Alternate Viewport Shading mode without angular shadows, Studio lighting setup: paint.sl

The ultimate design of these shapes is entirely up to us, so the following steps that were used to achieve the result shown in *Figure 6.14* should be viewed as mere suggestions. Some alternate design options are feathers, palm fronds, beads, or even abstract shapes.

Tip: Remember that Shortcuts to these and other tools can be found by hovering the mouse cursor over their icon. If we find ourselves repeatedly reaching for the brush, that is a good sign that memorizing the Hotkey combination will save us time.

22. Using the **Elastic Deform brush** with a yellow icon in the **Movement** category, we can click and drag parts of the mesh to reshape the leaf overall. Snake Hook in the same section has a similar but more intense effect.

23. Using Inflate with a blue icon and its inverse Deflate, activated by pressing the **Ctrl** key with the Inflate brush active, enables the curling of the top of the leaf shapes and carves a concave area underneath. The Blob brush achieves a similar effect with a bit more control, while Clay adds a bit of natural roughness.

24. Tapping quickly with the Smooth brush from the section of red icons rounds out the corners of the former rectangle shape and can repair artifacts created when surfaces accidentally twist, overlap or invert.

25. Scrape, with a red icon can be used to flatten an area. Scraping from two angles can form a corner, while Multi-Plane Scrape achieves this effect in two dimensions at once.

26. Using the Crease brush with a blue icon set to a small Radius at the highest Strength, with the Stroke Spacing setting at the lowest, 1% and Stabilize Stroke enabled, we can cut in veins or separate smaller leaf shapes like acanthus, or a palm frond. Using Crease in the inverse will pinch or sharpen an edge.

27. We can bring the sides of a Crease cut closer together with a Pinch brush from the yellow icon section, with a larger radius than the Crease.

28. The Rotate brush with a yellow icon can be used from the side to create a curl.

When we are happy with our **Sculpt Mode** results or at any time in the process of sculpting, we can unhide the Pedestal object and further adjust the scale and position of the leaf detail relative to the Pedestal until the result is to our liking. To finish the exercise, we will use a radial Array to duplicate the leaf detail in a circular pattern.

29. First, we will Apply Transforms on the leaf detail mesh, to clear out any movement away from the object's origin that we may have inadvertently caused while sculpting.

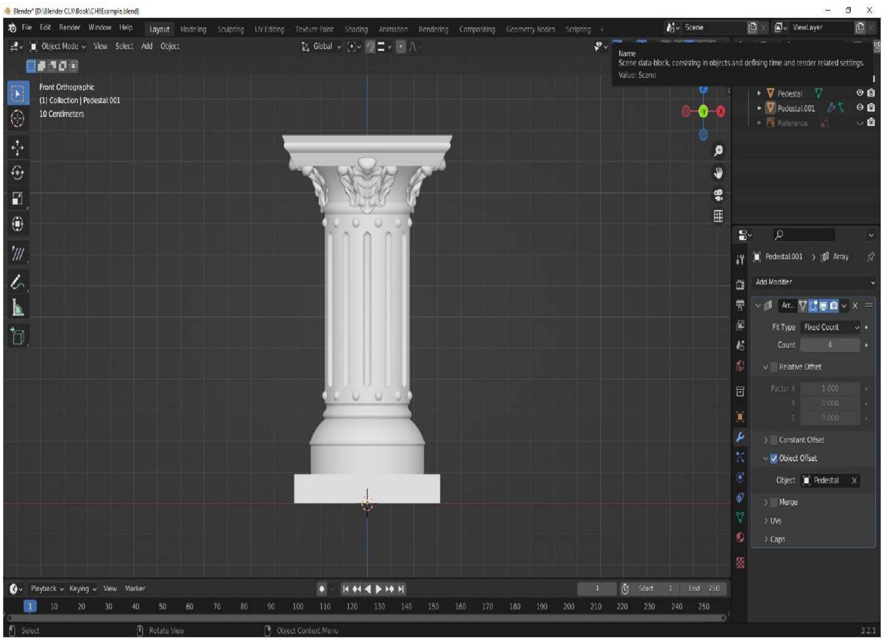
30. Next, let us add an Array Modifier to the leaf detail mesh, and set the Count to 4.

31. Disable Relative Offset and enable Object Offset, then expand its options.

32. Using the **eyedropper tool** in the Object field of the Object Offset or scrolling through the names of objects listed there, select the original Pedestal object.

33. With the leaf detail mesh selected, in the **3D Viewport** press the **R** key, and 90 to rotate the duplicate meshes around the Pedestal object 90 degrees.

The resulting pedestal should look a bit more like the original reference sketch, as shown in *Figure 4.8*. Now is a great time to save our project.


Front Orthographic
(1) Collection | Pedestal.001
10 Centimeters
Pedestal
Pedestal.001
Reference
Name
Scene data-block, consisting in objects and defining time and render related settings.
Value: Scene

Pedestal.001 › Array
Add Modifier
Fit Type: Fixed Count
Count: 4
☐ Relative Offset
Factor X: 1.000
Y: 0.000
Z: 0.000
☐ Constant Offset
☑ Object Offset
Object: Pedestal
☐ Merge
UVs
Caps

Figure 6.14: Finished sculpted detail repeated around the pedestal using an Array Modifier

Conclusion

In this chapter we have learned how to plan for adding sculpted details to any kind of 3D object, whether the desired result is high or low poly, as well as how to navigate Blender's Sculpt Tool menu of brushes. We have learned how to add the necessary subdivisions to our mesh, and choose which method best suits our desired result. We have created a high-resolution detail which we can later transfer onto the mesh using techniques we will explore next, in Chapter 7 - 3D Surfaces.

Essential shortcuts and hotkeys

In the previous chapters we used the following Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show/Hide Side Bar	N
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A

Adjust Last Operation	F9

Shortcut	Hotkey
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Convert Triangles to Quads (in Edit Mode)	Alt + J
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Table 6.1: *Essential Shortcuts and Hotkeys*

Points to remember

- Working with more polygons than are necessary will slow down any modeling operation, whether the intended result is high or low poly
- When aiming to achieve a low poly result, the choice of whether to work from low to high or high to low is largely personal preference, but there are benefits and drawbacks to each
- The Subdivision Surface Modifier can be applied for a high to low resolution workflow, while an unapplied Multiresolution Modifier is better suited for low to high.
- The official Blender documentation is the ultimate source for more

- The official Blender documentation is the ultimate source for more information about the capabilities and purpose of each brush, but exploration and experimentation are great teachers

Questions

1. What are some use cases for a high poly result?
2. What are the factors that determine whether a low poly goal is appropriate?
3. Where can we find Shortcuts for frequently used Sculpt Tool brushes?

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

CHAPTER 7
3D Surfaces

Introduction

With a firm understanding of low poly and high poly modeling techniques and when to use each, we can now begin preparing our 3D creations for surface treatments to give them color, life, and realism, or even a stylized appearance. We will also explore how to fake higher resolution details on lower resolution objects via these surface treatments, keeping the objects optimized while looking their best for everything from rendering to real time and interactive applications.

Structure

This chapter explains the anatomy of 3D surfaces and the myriad approaches to giving 3D objects a desired appearance, whether that be stylized or realistic. We will learn how colors, images, and effects are applied to the surface of 3D models and how the appearance of each can be edited. In this chapter, we will also learn how 3D surface treatments are mapped to a 3D object, followed by three practical exercises to prepare, then create and set up images to make the surface of the pedestal object from previous chapters more lifelike in an optimal way. Topics to be covered include:

- **Components of a 3D surface**: Textures, Materials, and Shaders
- **Texturing**: Maps, baking, painting, and editing
- **UV unwrapping**, mapping the surface of a 3D object

- **Exercise**: preparing a low poly mesh for baked detail
- **Exercise**: UV unwrapping the pedestal model
- **Exercise**: baking pedestal details to Texture Maps

Objectives

After reading this chapter, we will understand the three main components of 3D surfaces and how to select the appropriate treatments for our goal. We'll also understand how to unwrap UVs – a map to the surface of a 3D object – so that they lie flat, ready for treatments to be applied. With this knowledge, we'll be ready to make our 3D objects look their best, whether for exporting to other 3D software, for interactivity and animations such as those we will explore in *Chapter 8, 3D Animation*, or for rendering, which we will address in *Chapter 10, Images and Video*.

Surfaces

In several previous sections, we have discussed the appearance of 3D objects and touched on the most basic ways that computers display 3D objects on screens, such as through Viewport Shading modes as first discussed in *Chapter 4, Polygonal Modeling* and the MatCaps we encountered in *Chapter 5, Poly Modeling Extras*. We found that some such surface treatments are primarily practical for making work with 3D models easier and more precise.

The surface treatments that we will explore next are those which are meant to be rendered for viewing by an end user or exported with the 3D object for import into other applications. The main components of 3D surfaces meant for such decorative, interactive, and commercial purposes are as follows:

- Textures
- Materials
- Shaders

Textures are typically image files such as PNG, JPG, TIFF, PSD, and – less commonly

Textures are typically image files such as PNG, JPG, TIFF, PSD, and – less commonly due to their extremely large file size – BMP. Sometimes though, Textures are virtual – such as single colors, gradients, or procedurally generated patterns like those we will create in the third exercise later in this chapter – and aren't separate files until they are "baked" or saved externally.

Materials can be thought of as containers for Textures, operating in a similar way to the layers panel common in many image editing applications. Materials can display multiple Textures on the same faces of the same model, with each image driving different properties of the final surface treatment. For example, one Material might have one Texture for color, another for reflectivity, another for transparency, and soon.

Note: Each 3D object can have an unlimited number of Materials applied, although for the sake of performance, it is best to limit the number of Materials on any given 3D object to as few as possible to ensure that each object's surface resolution matches the resolution of other 3D objects in the scene, according to their relative scale. A 3D building, for example, will likely need more materials to display the same pixel resolution as a smaller prop in the same scene, such as a crate. This pixel resolution relative to the object's scale within a scene is called "texel density" and is more important in interactive scenes containing multiple 3D objects than in a single product render.

Finally, Shaders are the code that directs how the Material displays its Textures. In Blender, Shaders are used, and their parameters are edited through a convenient interface called nodes. The default Shader in Blender is the Principled BSDF Shader, which allows for combining over 20 inputs from ordinary Textures like Base Color and Roughness, to specialized Textures such as Subsurface Scattering, Refraction, and even Transmission.

The number of effects that can be achieved with Shaders is so limitless that entire careers are made creating Shader effects from everyday surfaces like dirt, wood, and metal, to animated oceans, fire, and electricity, to the fantastical, iridescent holograms, and black holes in space. Blender provides multiple Shaders for multiple purposes, and multiple Shader effects can be combined through special Mix Shader nodes. Materials can utilize multiple Shader effects blended together this way, but all Shaders must feed into one Material Output for their effect to be displayed on a given 3D object.

Textures

2D images, colors, or patterns applied to the surface of 3D objects known as Textures can also produce a wide range of surface effects all on their own. Some 3D surface treatments employ a single image, a single color, or just one pattern that is generated in Blender, but most surface treatments combine one or more images to generate more complex and ever more plausible surface looks. Individual Textures may cover more complex and ever more plausible surface looks. Individual textures may cover the entire surface of a 3D object, while others may be designed to repeat seamlessly in any direction, providing an effect called "tiling," like the wood pattern shown in *Figure 7.4*.

Tip: Since many 3D applications require image pixel dimensions to be a "power of 2" for important computations such as compression, it is a good habit to use or make textures for 3D objects square (1:1) or rectangular (1:2 and 2:1) in one of the following pixel dimensions.

The most common power of 2 image sizes are 32 pixels by 32 pixels, 128 x 128, 256 x 256, 512 x 512, 1024 x 1024, 2048 x 2048, 4096 x 4096, or 8192 x 8192. Most computers can't handle many textures above 8k at once, and the difference is difficult to detect on even the largest monitor and television sizes, so it is not recommended to use larger images, especially for real time rendering.

Texture maps

Images meant to be used together in a single Material are referred to as Texture Maps. A set of Texture Maps created for one Material are "mapped" to the same faces of the same 3D object, but each Texture uses different greyscale and/or color values to produce different effects on the textured object, such as greyscale driving roughness or smoothness, reflectivity, and transparency, or color for glow, translucence, and faked height detail.

Materials rarely require all the possible Texture Map types, and some use no Texture Maps at all. The most common type of image Texture Maps are RGB color maps, alternately referred to as Base Color, Diffuse, and Albedo. The reason a variety of names exist for essentially the same map is that early attempts to make 3D surfaces look realistic involved adding fake lighting information like shadows and highlights into one color map, either painted digitally or as captured in photographic Textures.

In many early 3D applications, color maps with fake lighting incorporated this way were called Diffuse maps. These early Textures worked well enough in the days before higher resolution screens and elaborate virtual cameras were developed, because viewpoints were often limited and details were pixelated, such that implausible lighting was less obvious. It is no longer common for Diffuse maps to include shadows and highlights, but the term Diffuse is still in use in many 3D applications, and nowadays refers mainly to color maps.

In recent years, computers and gaming consoles have been made capable of far more complicated computations which can be used for simulating 3D lighting that looks ever more lifelike, so a new approach called **Physically Based Rendering** – or PBR for short – has become the norm, and a new type of Texture that contains only color without shadow or highlight called **Albedo** has largely replaced Diffuse maps.

Aside from Color, PBR Texture Map sets also often include Roughness, Metallic, Normal, and somewhat less often, Alpha. Roughness, Metallic, and Alpha maps are greyscale. Fully black areas of Roughness maps are shiny, whereas white areas are fully rough. Fully black Metallic maps are non-reflective, while fully white areas are

fully metallic. Fully black Alpha maps are completely transparent, whereas fully white areas are fully opaque. Pairing black Roughness maps with white Metallic maps produces a mirror finish, which paired with dark grey to black Alpha maps, can begin to approximate the properties of glass.

Note: Like Diffuse Textures, there are other non-PBR Texture Maps still in use, such as Glossy and Specular, which are like Roughness and Metallic in their effect but use inverted greyscale values.

Normal maps are the next most common Texture Map type in the PBR workflow and have replaced both Diffuse maps and an older greyscale Texture type that was used to fake variations in height and directionality on 3D objects called **Bump Maps**. Normal maps are RGB Texture Maps, which means they employ a broader spectrum of color to map the normal directions of the faked details they display, making them much more convincing. The distinctive pastel pink, purple, and teal hues of the Normal map correspond to the red, blue, and green indicators of axes in 3D space, relative to the normal directions of the source object's surface.

Unlike outdated Diffuse maps and their successor Bump maps, Normal maps are more difficult to produce by hand because of that broader spectrum of color, and are more often generated, either with image editing software specifically for creating Textures for 3D, or through a process called Texture "baking," which we'll explore more in the following section.

Some other common Texture Maps that are used more situationally include Ambient Occlusion, which fakes how ambient light is occluded inside crevices and between two surfaces that are very close in proximity; Emission (also called Emit, and Emissive) which creates a glowing effect; and Transmission (Transmissive) or Subsurface Scattering, which control where light can pass through translucent objects. A handful of other common PBR effects such as Refraction and Clear Coat are even more situational but just as important to the plausibility of a 3D object's surface treatment where they are called for, such as with 3D glass or automotive paint.

Baked textures

Texture baking involves setting up two 3D objects with closely matching topology occupying the same space, as we will do in the first exercise later in this chapter. Triggering a bake then generates an image of the source object's surface, remapped to the surface of a target mesh. This process is often used to give a lower resolution object a very convincing look of higher resolution detail, but multiple types of Texture Maps can be baked from one object to another.

Baking Textures is often used convert a number of very complex Materials with multiple image inputs into a single Material using only one set of Texture Maps,

which is much more optimal for interactive applications such as games and 3D viewers on the web. A common workflow is to prepare a source object with different Materials assigned to each of multiple groups of faces – such as a wood Material assigned to faces of a 3D chair's legs and a fabric Material assigned to faces of the seat – and then baking all of those Materials into one, for use on a copy of the mesh that is meant for export.

Note: Even greater optimization of 3D objects meant for real time rendering such as with video game assets can be achieved through an advanced process called "packing" Textures. While we won't practice it here, it is helpful to know that packing Textures involves baking and then saving multiple Texture Maps such as Color, Roughness, and Metallic into the R, B, G, and A channels of a single image, using advanced image editing software like Adobe's Photoshop or Substance Designer.

Texturing maps

We will call the last category of Texture we will discuss here "Texturing Maps" because they are typically used in the process of designing complex surface treatments such as the multiple Material workflow described earlier, which are meant to be baked into a single Texture Map set. For this reason, Texturing Maps are not often used directly in the final output. Texturing Maps include Height, Cavity, Position, and Thickness. Some plug-ins for Blender as well as third party software such as Adobe's Substance Painter can be used to generate these maps based on unique properties of the source mesh like distances between surfaces, position in local and world space, or extreme angles producing cavities and edges.

Texturing Maps are most often used in the Texture creation process to blend two Textures or Materials to produce lifelike effects such as weathering, by essentially acting as a mask between them. For example, Position maps are often used to blend a new, clean looking Material with a duplicate Material that looks faded and peeling, so that it only appears on the upper sides of a mesh to mimic the effects of exposure to sun and rain.

Texture painting and editing

In addition to generating Textures through baking or setting up procedural patterns – both of which we will practice later in this chapter's exercises – there are two other primary methods for creating Textures for 3D objects: painting and editing images. Texture painting, as it sounds, is a manual process using tools much like the sculpting tools used in *Chapter 6, 3D Sculpting*, which appear in a specialized **Tool bar** menu accessed via the **Texture Paint Workspace** tab located at the top of the Blender

application window, or by switching to Texture Paint mode in the upper left of the 3D Viewport.

To begin freehand painting a 3D object, we can assign a new blank image to one of the Material's Image Texture slots such as **Base Color**, via Blender's **Texture Paint Active** Tool tab at the top of the **Properties pane** as shown in *Figure 7.1*. This setup allows us to begin painting directly on the surface of a selected 3D object in the **3D Viewport**, or onto the image itself via the **Image Editor** on the left-hand side. Note that a valid UV Map – which some primitives have by default, and we will learn to create anew in the following section on UVs – is necessary for painted textures to align as intended to the 3D object's surface. Please refer to the following figure:

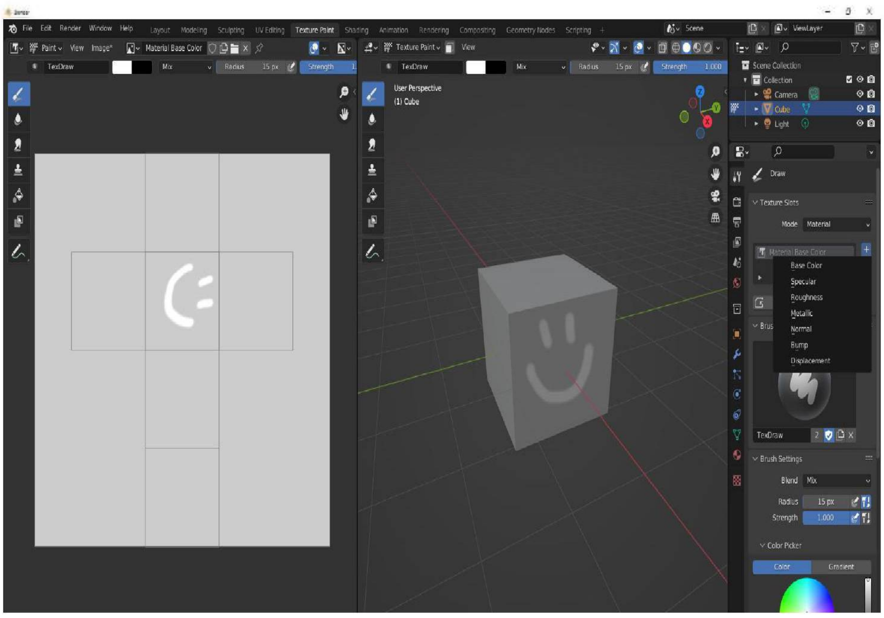


Figure 7.1: Texture Painting in the Base Color Texture Map

Blender's Texture painting tools also allow for the editing of existing images through the same interface, when images are added into the object's Material inputs in the same way, via the **Material Properties Tab**, or even in the **Shader Editor** tab as we will discover later in this chapter. Multiple common image editing tools exist within Blender's Texture painting tool set, including **Draw**, **Blur**, **Smudge**, **Clone** and **Fill**, as well as the ability to add an Alpha mask for creating custom Brush effects. As with painting a blank image, editing existing images on the surface of a 3D model requires mapping where we intend for such Textures to be displayed.

UVs

To prepare the surface of a 3D object to display image Textures or any other effect we may want mapped to specific faces, we must first create a flat proxy of the

object's surface called a UV Map, through a process very much like unwrapping a gift or preparing fabric for sewing a garment, called "UV unwrapping." Like the cuts, creases, and printed side of wrapping paper that once enveloped a package, UV Maps retain information about a 3D object's surface, with vertices, edges, and faces that correspond to the object's own vertices, edges, and faces, as well as which directions their normals face.

In Blender, UV Maps are stored as object data, viewable in a selected 3D object's **Object Data Properties** tab in the **Properties panel**. UV Maps are visualized and edited through the **UV Editing Workspace** tab at the top of the Blender application, or by converting any **Editor Type panel** to a UV Editor through the menu in the upper left corner. UVs appear as a see-through 2D copy of the 3D object's surface as shown on the left in *Figure 7.2*.

It may be useful to think of an object's UVs as being akin to a custom-fitted garment. If we were to cut apart this garment in such a way as to allow each piece to lie flat, we will have created UV "islands" corresponding to specific parts of the object, separated by UV "seams" wherever the cuts were made. 3D surface treatments – like fabric or a pattern printed on a garment – are displayed on the surface of the unwrapped 3D object according to the placement of the pattern on the flattened garment, as it would be if the unwrapped garment were reconstructed on the surface of the model.

UV space

UV Maps and the operations we can perform on them are typically constrained

UV Maps and the operations we can perform on them are typically constrained within a defined section of otherwise infinite 2D space in the UV Editor viewport called "UV space." UV space begins in the bottom left at the coordinates (0,0) and end at (1,1) in the upper right. This space is defined by (x, y) coordinates rather than pixel dimensions because a 3D object's UVs expand or shrink to fit any image applied to the surface of the object for which UVs control the placement, while the object itself remains unchanged.

Note: Advanced UV and Texturing techniques utilize second, third or multiple sets of UV coordinates – as in the UDIM workflow – for expanding the amount of texture resolution that can be applied to multiple surfaces of a 3D object. Multiple UV Maps may also be created for any given 3D object, for layering extra effects such as light maps. In this chapter, though, we will limit our focus to the single (0,0) to (1,1) UV space.

Unwrapping and editing UVs

UV unwrapping is the operation of creating a UV Map for a selected 3D object. Some objects – such as many primitives – have a Generate UVs parameter which can be

enabled to allow them to be UV unwrapped by default upon creation as shown in *Figure 7.2*. In a few cases, the actual process of UV unwrapping can be as simple as pressing a button. With a 3D object or some part of the object selected in **Edit Mode**, we can press the U key, and select the first option Unwrap to perform an automatic UV unwrap.

Tip: It is helpful to note that not all 3D objects are UV unwrapped by default, and it is possible for only part of a 3D object to be UV unwrapped, such as when combining primitives with custom poly modeling. If we come across a mesh whose Textures don't display as expected, it may be helpful to check whether all the object's UVs are properly unwrapped or not.

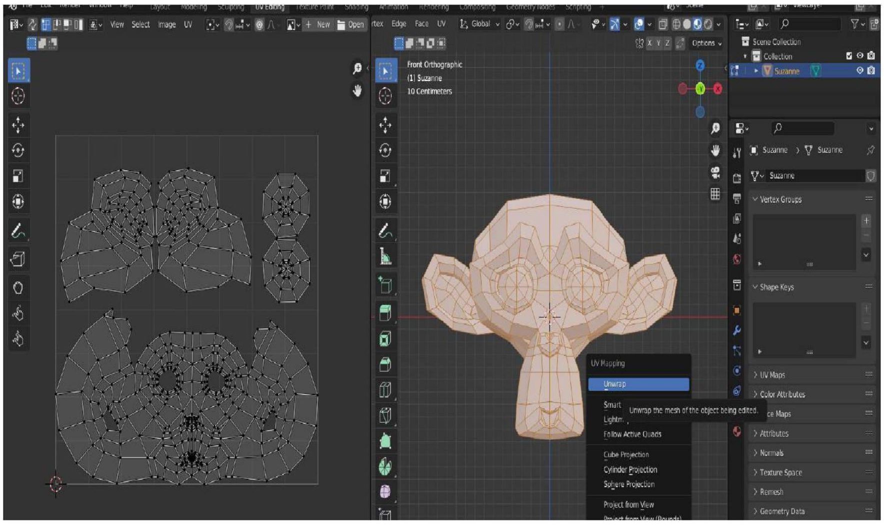
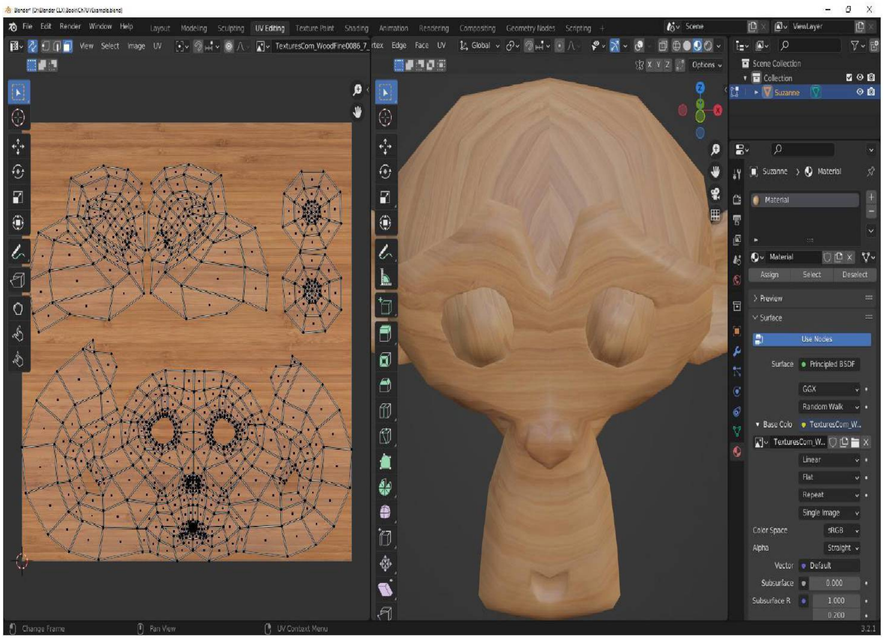


Figure 7.2: Suzanne primitive object on the right, with default UV Unwrap on the left in the UV Editing Workspace

In many cases, another of the available options in the Unwrap menu may be better suited for a particular purpose – such as one of the Projection methods for Cube or Cylinder shapes – followed by manual adjustment of the result. 3D objects that are more convoluted than the average primitive may also require separating or stretching overlapping UVs so that the corresponding mesh can display separate areas of Texture on separate faces. Inadvertently overlapped or twisted UVs can produce a variety of unwanted visual artifacts such as Texture bleed.

As with the vertices, edges, and faces of a 3D mesh, vertices, edges and faces of an object's UVs can be edited through selection, moving, rotating, and scaling for the optimal display of surface treatments. In most cases, some thoughtful planning of the desired size and placement of connected clusters of UV faces – called "islands" – and where they are separated – called "seams" – produces more desirable results than automatic unwrapping, or no UV Mapping at all.

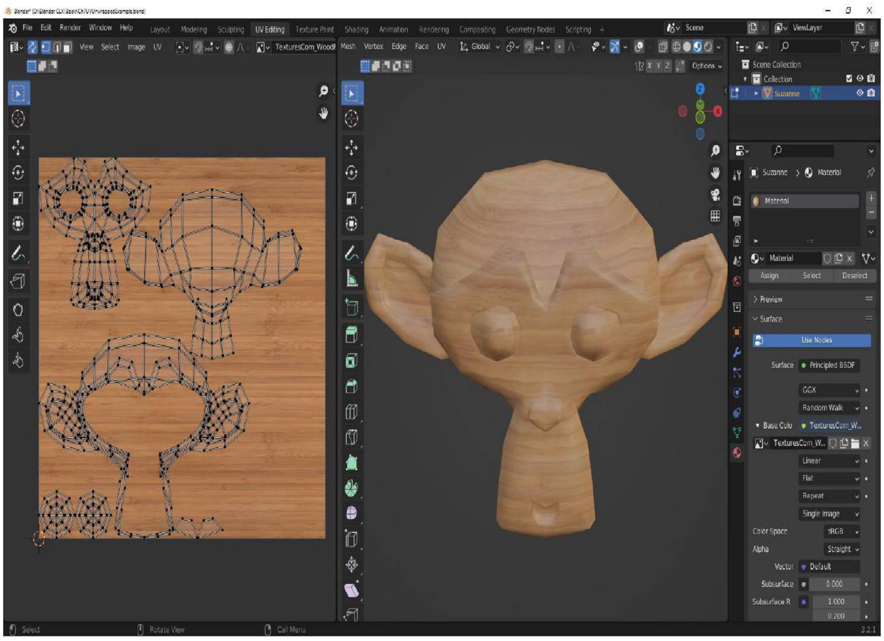
Choosing the preferred placement, position, rotation, and scale of UVs is much like aligning the pattern on a garment or gift wrapping for aesthetic purposes. In some cases, the aim is for parallel lines to appear contiguous, for example, but how we should orient a given 3D object's UVs depends on our intended surface treatment. As we discovered earlier in this chapter, we can create an unlimited number of customized surface treatments for 3D models. For example, if we wish our 3D object to appear as though it is made from wood, we will likely want the direction of the wood grain to be plausible as shown in *Figure 7.3* instead of random as shown in *Figure 7.4*:



Surface: Principled BSDF
GGX
Random Walk
Base Colo: TexturesCom_W...
Linear
Flat
Repeat
Single Image
Color Space: sRGB
Alpha: Straight
Vector: Default
Subsurface: 0.000
Subsurface R: 1.000

Figure 7.3: Suzanne with default UV unwrap and tiling wood texture showing implausible seams and grain direction

In real life, wood is cut with the grain running parallel to longer sides of a wood piece for structural integrity, and the grain of contiguous pieces of wood will run in one direction, which is one reason we may prefer to orient UVs in one direction using a **Projection unwrap** method, as shown in *Figure 7.4*:



Surface: Principled BSDF
GGX
Random Walk
Base Colo: TexturesCom_W...
TexturesCom_W...
Linear
Flat
Repeat
Single Image
Color Space: sRGB
Alpha: Straight
Vector: Default
Subsurface: 0.000
Subsurface R: 1.000
0.200

Figure 7.4: Suzanne primitive object with a custom, front-projected UV unwrap and plausible wood grain

UV seams

When the basic Unwrap method produces undesirable results, it may help to add markings to the mesh where you want Seams to fall. Just like seams in our garment analogy are where separate pieces of fabric are cut for joining, UV Seams indicate where UV islands meet. Similar to marking edges Sharp, marking edges as a Seam can be done through the **Edge menu** at the top of the **3D Viewport**, or by pressing **Ctrl + E**. The inverse of **Mark Seam**, **Clear Seam**, will remove seam markings, but does not automatically undo UV unwrapping, except when a special mode called **Live Unwrap** is enabled.

UV packing

The position and rotation of a 3D object's UVs dictate the position and rotation of any image-based surface treatment that is displayed on the 3D model. Likewise, the size of the UVs relative to the (0,0 to 1,1) UV space dictates the resolution of any surface treatment displayed on the corresponding faces of the mesh. The larger the UVs, the smaller the detail, and in the case of image-based surface treatments, the less pixelated the model's surface will appear because UVs can expand and shrink to fit any image. The distance between UV islands, called "padding," also affects how densely the texture resolution appears on the 3D object's surface.

Together, all these considerations of a 3D object's UV layout are called the UV's "packing." UV packing is a process usually undertaken after UV seams are marked,

UVs are unwrapped, and the desired orientation and scale of UV islands are determined. Packing UVs is often a manual process akin to assembling a puzzle, but Blender does provide an automated UV Packing algorithm – which can employ automatic rotation of UV islands or not – activated via the upper menu of the **UV Editor viewport** under **UV | Pack Islands**. Because Blender's built-in UV packing algorithm ignores holes inside which additional UVs could be packed, UV editing is a task that can be greatly assisted by multiple free and paid add-ons that can be acquired online.

Note: Whether 3D objects created solely for 3D printing need surface treatments or not depends on the real-world materials to be used in the printing process. Most 3D printers for home use only print in one color per print. While full color 3D printing is possible – usually through specialty businesses – it is achieved at a much higher cost than single color printing because the equipment and materials are expensive, and the processes are complicated. Preparing 3D models for full color 3D printing is made more difficult by the high poly count needed for smooth rounded surfaces but is otherwise identical to the process we will learn in this chapter.

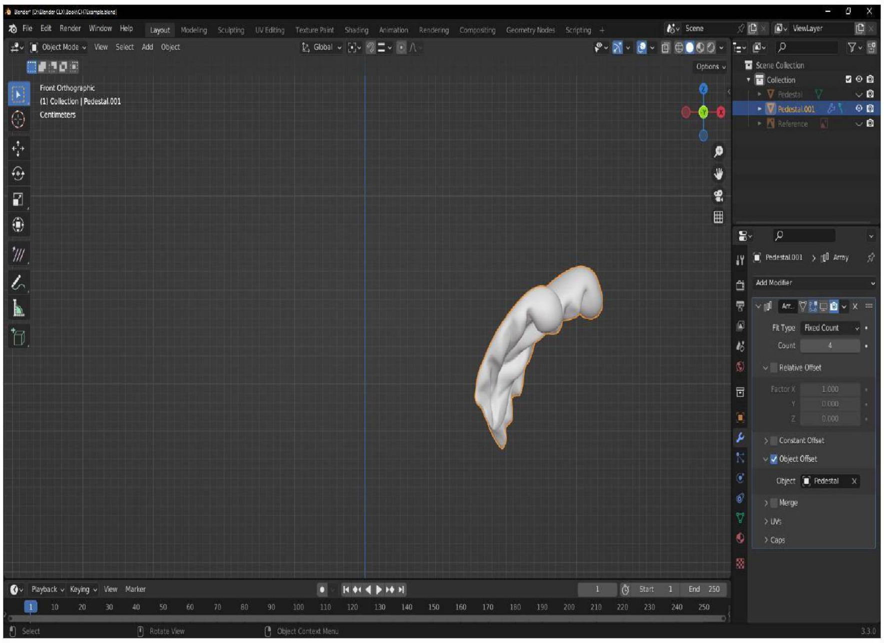
Exercise 7.1

In this exercise, we'll first make a flat, low poly surface on which to project the high-resolution leaf detail we sculpted in *Chapter 6, 3D Sculpting*, and then we will apply it to a texture in a process called "baking" to a special type of texture called a normal map.

1. First, we'll reopen our Pedestal project.
2. Next, we'll select the high poly leaf mesh named `Pedestal.001`, and press **Shift + H** to hide all but the leaf detail from view.

   We only need to model the front facing portions of the mesh, and since we can use the radial Array again, we only need to create a low poly surface for this operation once.
3. To make it even easier to navigate around the leaf detail in the 3D viewport, in the **Array Modifier** accessed via the **Modifier Properties tab** appearing as a wrench icon in the **Properties panel**, click the icon appearing as a computer monitor to disable **Realtime display**, so that all the copies of the leaf detail are temporarily hidden from view.

4. Orient the view to look at the leaf detail from one side, as shown in *Figure 7.5*.



Fit Type: Fixed Count
Count: 4
Relative Offset
Factor X: 1.000
Y: 0.000
Z: 0.000
Constant Offset
Object Offset ✓
Object: Pedestal
Merge
UVs
Caps

Figure 7.5: Sculpted leaf detail with Array copies hidden, viewed from the side

Next, we'll place the 3D cursor where we plan to draw new geometry.

5. Tab into **Edit Mode** and select any vertex at the bottom side of a leaf detail near the **view pane**, then press **Shift + S** and press the **2** key or choose Cursor to Selected.

6. Exit **Edit Mode**.

7. From **Object Mode** via the **Add menu**, with **Add Mesh Extras** enabled in the **Preferences** panel, **Add** | **Mesh** | **Single Vert** | **Add Single Vert**.

   Remember that **Edit Mode** will be automatically enabled.

8. From the **Snapping menu** in the upper middle of the **3D Viewport**, enable continuous **Snapping Mode** by clicking the icon appearing as a magnet, then in the drop down, choose **Face Project**, and disable **Include Active**.

9. Toggle **X-Ray mode** via **Alt Z**.

10. Press **E** to extrude, then place the new vertex along the bottom of the same leaf detail.

    Exactly where to place this and each subsequent vertex will depend on the shape of your mesh but refer to *Figure 7.6* for an idea of the topology to aim for.

11. Select each vertex one at a time and extrude them upward, snapping so that when the space between them is filled with a face it will be approximately square, rather than rectangular

    If we extrude an entire edge, it may be necessary to deselect the new edge and adjust the position of the vertex that didn't snap to the surface.

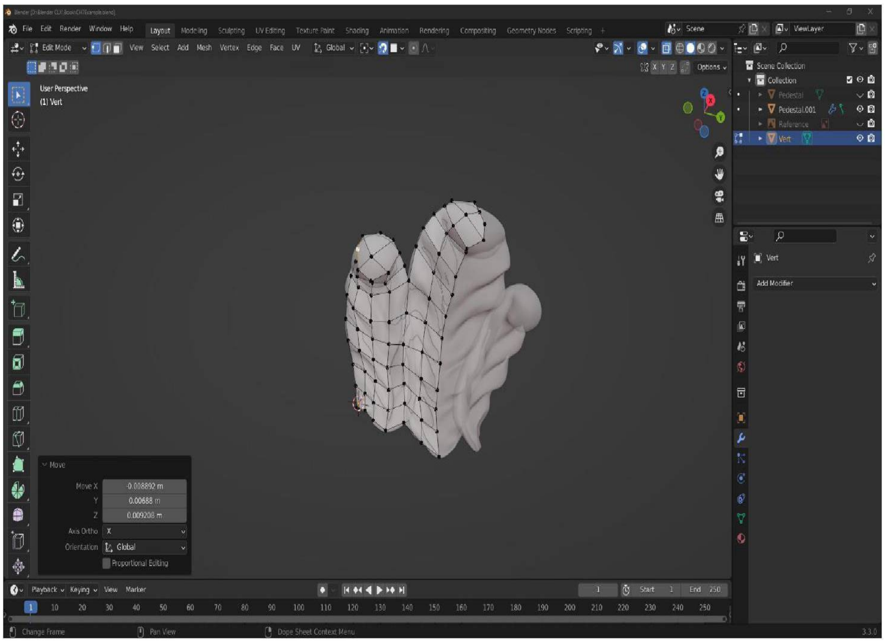
12. Select the four vertices and press **F** to fill them with a face.

13. Insert edge loops and use any of the previous tools and techniques introduced in *Chapter 4, Polygonal Modeling* as needed.

14. To merge any two vertices, press **M** and choose **At First to merge** the second and subsequently selected vertices to the first, or **At Last to merge** the previously selected vertices to the last.

15. Attempt to keep the new mesh on top of the detail mesh, select each new vertex created, and press **G** to snap the new vertices as well.

16. Continue this way, extruding and snapping vertices into square shapes along the surfaces and then filling them, rotating the view as needed, until the result covers just the front of one half of the leaf detail as shown in *Figure 7.6:*



Move

Move X -0.008892 m
Y 0.00688 m
Z 0.009208 m
Axis Ortho X
Orientation Global
Proportional Editing

Figure 7.6: Sculpted leaf detail half covered in new, lower resolution geometry

It is acceptable if some of the new mesh is inside the detail mesh, but it is preferable for the vertices to sit on top of peaks rather than inside valleys, wherever possible. It's also fine to create triangles if needed, because this

mesh will neither be animated, nor do we need to UV unwrap it in a very specific way.

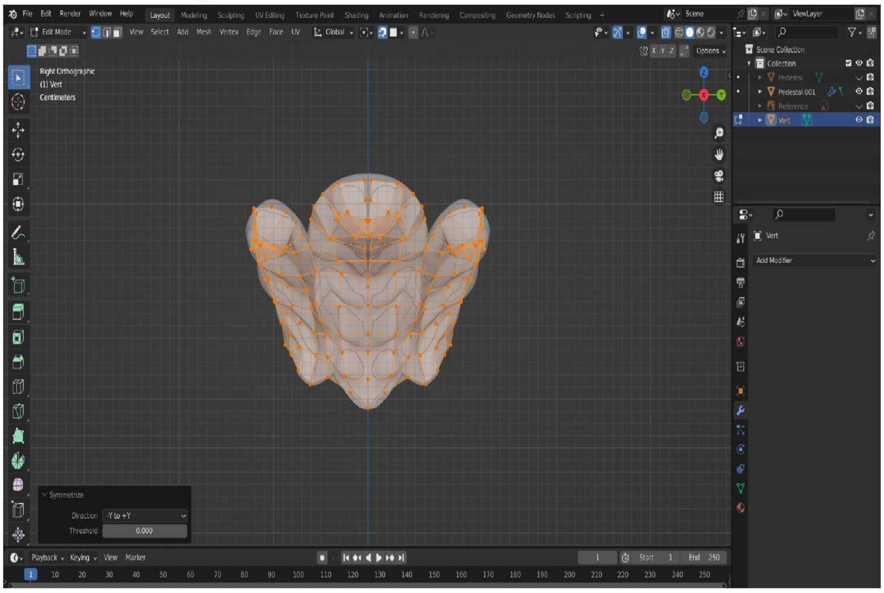
This is a good time to save. Now, that we have half the mesh covered, we'll use a new approach to mirroring the geometry from side to side (or in this case, front to back).

17. Start by exiting **Edit Mode**.
18. Then reset the 3D Cursor to the world origin, by pressing **Shift + S** then **1**.
19. Next, from the **Object menu** at the top of the **3D Viewport**, select **Set Origin** | **Origin to 3D Cursor**.

    In effect, we have just Applied the Transforms to the new mesh (currently named Vert) so that any further operations will be based on the correct origin.
20. Next, reenter **Edit Mode**, and select the whole mesh.
21. Then, under the **Mesh menu** at the top of the **3D Viewport**, select **Symmetrize**.

    Don't be alarmed when some or all the mesh disappears. This is because the default axis is incorrect for this operation on this mesh.
22. From the contextual menu that appears for the Symmetrize operation in the bottom left of the **3D Viewport**, select **-Y to +Y**.

    Assuming that our leaf detail was created through the same steps as we performed in *Chapter 6, 3D Sculpting*, the resulting mesh will be mirrored across as shown in *Figure 7.7*. If our results vary, trying one of the other Direction options should work:

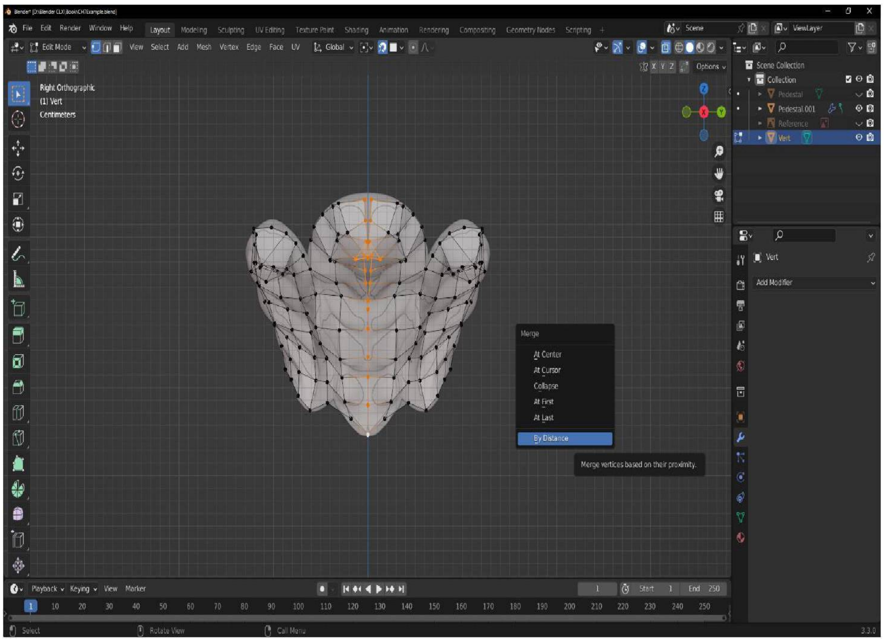

Right Orthographic
(1) Vert
Centimeters

Symmetrize
Direction: -Y to +Y
Threshold: 0.000

Figure 7.7: New low poly mesh mirrored across the X axis, viewed from Right Orthographic perspective

As shown in *Figure 7.7*, our newly mirrored mesh may have some gaps. Let's close those now.

23. Clear the selection, then box select just the middle vertices as shown in *Figure 7.8*.



Merge vertices based on their proximity.

Figure 7.8: Middle vertices selected in X-Ray mode

24. Press **M** to activate the **Merge menu** and select **By Distance** as shown in *Figure 7.8*.

25. In the **Merge by Distance contextual menu** that appears in the lower left of the **3D Viewport**, increase the **Merge Distance** by pressing the right arrow icon or by left clicking in the parameter field and dragging the mouse to the right, adding the **Shift** key to slow down the scrubbing movement, until just the middle vertices have merged.

26. If any of the vertices merge with their vertical neighbors, reduce the Merge Distance amount, and merge them individually as described in the following step.

27 Clear the selection, then select any pairs of vertices that weren't merged by Distance, then press **M** again, but this time choose **At Center**.

We can now toggle **X-Ray mode** off, hide the **`Pedestal.001`** object, and set the new low poly leaf mesh to Shade Smooth under **Object** in the upper left menu of the **3D Viewport**. The resulting mesh will look something like *Figure 7.9*.

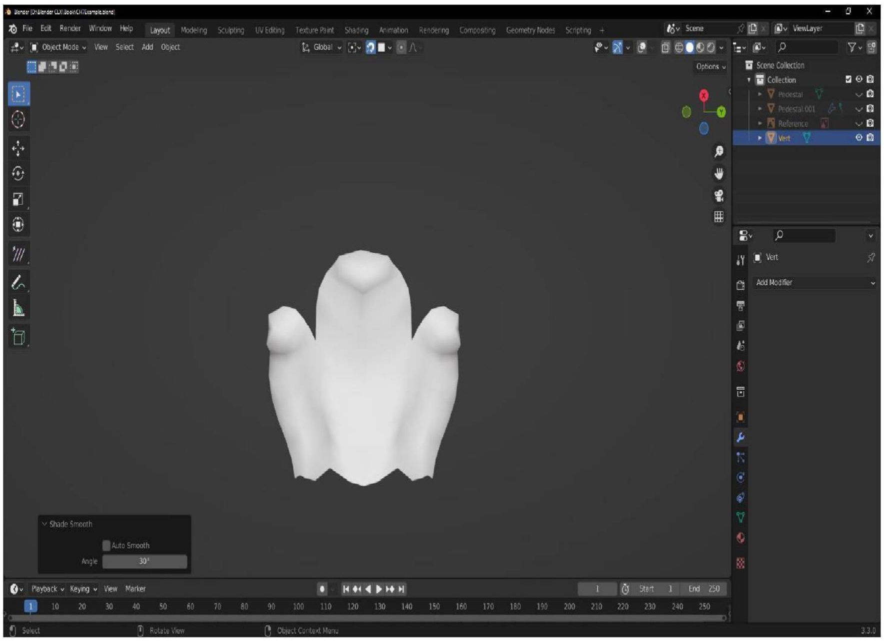


Figure 7.9: Low poly leaf mesh matching the overall shape of the high poly detail mesh

Before we apply the radial Array from the `Pedestal.001` object to our new low poly leaf detail, we'll take one more step to prepare it for baking high-resolution details from the high poly mesh onto a normal map texture, because baking normal maps works best when the target mesh is slightly above the source mesh.

28. With the `Pedestal.001` made visible again, and with the Vert object selected, in the **Modifier Properties tab** of the **Properties panel**, add a **Shrinkwrap Modifier** to the Vert object.

29. At the top of the **Shrinkwrap Modifier**, click the icon appearing as a triangle with circles at each corner to enable the **On Cage display mode**.

    This will enable us to see the Modifier's Effects in **Edit Mode**.

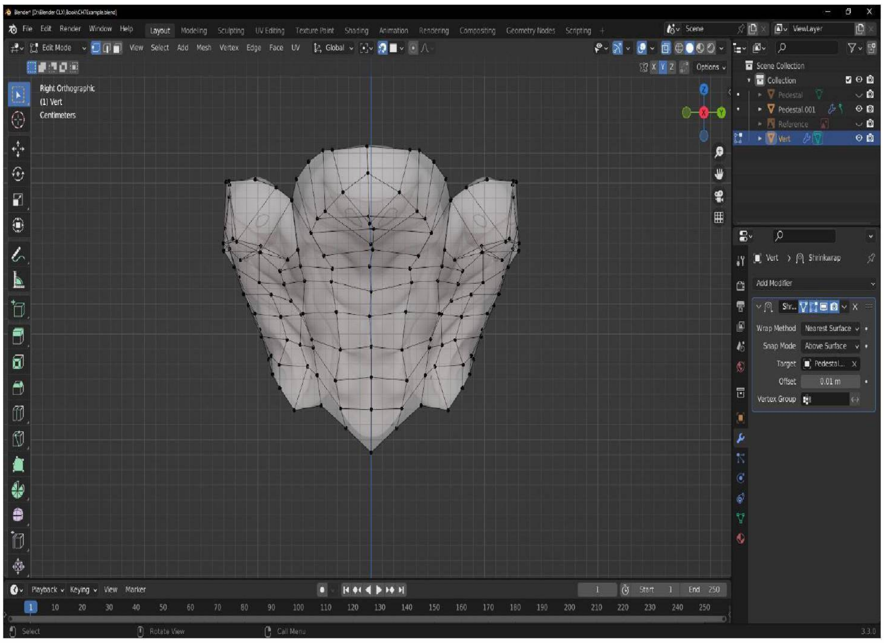
30. In the Shrinkwrap Modifier's parameters, choose the **Snap Mode** above Surface.

31. Then set the Target to **Pedestal.001**.

32. Next, increase the Offset amount gradually, until most of the low poly mesh is just outside the surface of the high poly mesh.

33. If a few vertices stubbornly remain under the surface, enter **Edit Mode**, enable mirroring Transforms across the y axis by clicking the **Y** button next to the butterfly icon in the upper right of the **3D Viewport**, and manually adjust

their placement until the result is like *Figure 7.10*, with barely any of the low poly geometry under the surface of the high poly mesh.



Vert › Shrinkwrap

Wrap Method: Nearest Surface
Snap Mode: Above Surface
Target: Pedestal...
Offset: 0.01 m
Vertex Group

Figure 7.10: Finished low poly leaf mesh

Note: If our leaf detail has sharp direction changes that need Seams marked, it would be better to mark them now than to manually place seams on multiple copies of the same mesh later. See Exercise 7.2 for the steps to mark Seams.

Now we can proceed to repeat the leaf detail around the Pedestal object. Because the `Pedestal.001` object retains the Array from our previous exercise and we've applied the transforms on this low poly duplicate, we can just copy that Modifier over.

34. Exit **Edit Mode**, and first let's apply the Shrinkwrap Modifier, by pressing **Apply**, under the down arrow icon at the top of the **Modifier**.

35. With the Vert object still selected, Ctrl + click the **Pedestal.001 object** in the Outliner panel.

36. With the mouse cursor inside the 3D Viewport, press **Ctrl** + **L** to bring up the **Link/Transfer Data menu**, and choose **Copy Modifiers**.

    At first, nothing will happen because we disabled the **Realtime display** in the Modifier earlier.

37. Click the icon at the top of the **Array Modifier** appearing as a computer monitor to re-enable **Realtime display**, and then with the mouse cursor in the **3D Viewport** again, press **Ctrl** + **L** and choose **Copy Modifiers** again.

38. Next, we'll select only the Vert object, and as in *Exercise 6.1*, press **R, Z, 90**, to rotate the object, distributing the array around the Pedestal Object Offset in a radial pattern.

    The final step to prepare the low poly leaf mesh for baking is to Apply the Array Modifiers on the Vert and **`Pedestal.001`** objects, and then join each of them to a copy of the Pedestal mesh.

39 First, let's unhide and select the original Pedestal mesh, then press **Shift** + D to duplicate it.

40. Then unhide and select the **Pedestal.001 object** (which is the sculpted leaf detail) and Apply its **Array Modifier** by pressing **Apply**, under the down arrow icon at the top of the **Modifier**.

41. Select the **Pedestal.002 object** (which is the duplicate of the original), then Ctrl + click the **Pedestal.001 object**, and press **Ctrl + J** to join the duplicate Pedestal to the sculpted leaf detail.

42. Hide the **Pedestal.001 object**.

43. Next, we'll select the Vert object and join it to the original Pedestal mesh by making the Pedestal object visible, selecting the Vert object, then Ctrl + clicking the Pedestal object, and pressing **Ctrl + J**.

The desired result is that we have two full pedestals occupying the same space: one original Pedestal object with the low poly leaf objects joined to it as shown in *Figure 7.11* and one hidden **`Pedestal.001`** object with the high poly sculpted leaf details joined to it. Please refer to the following figure:

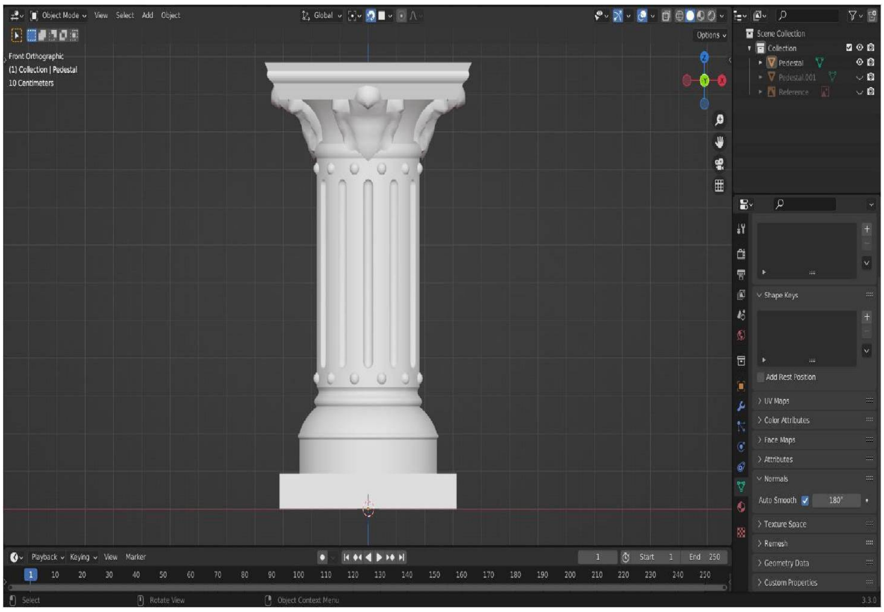


Figure 7.11: New low poly leaf meshes joined to the original Pedestal mesh

Exercise 7.2

The next step in the process of baking sculpted high-resolution detail onto a low poly duplicate mesh is to prepare the mesh for texture. While shaders, materials, and textures can be applied to any object regardless of whether it is UV unwrapped or not and regardless of whether any UVs are thoughtfully placed or not, if we want control over where and how well textures are displayed on the mesh, we need to unwrap its UVs.

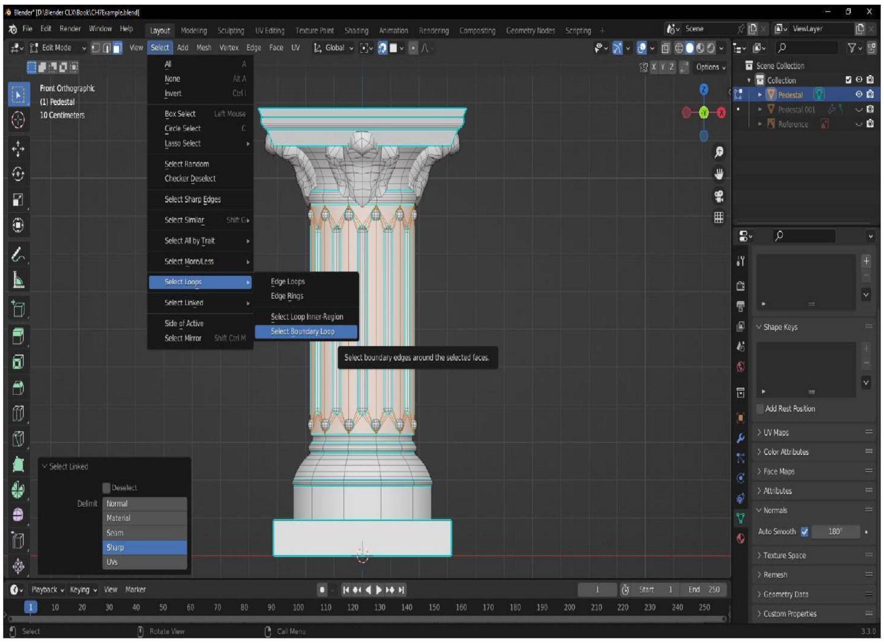
1. The first step in this process is to select the Pedestal mesh and enter **Edit Mode**.

2. The next step is to determine where seams should be placed.

   In our case, many of the Sharp markings are excellent choices for seam placement.

3. In the **Face Select mode**, select the body of the main column via **Select Linked**, and change the **Delimit method** to **Sharp** as shown in *Figure 7.12* so that the fluting is deselected.

4. Next, under the Select **menu**, choose **Select Loops** | **Select Boundary Loops**, also shown in *Figure 7.12*:



Select boundary edges around the selected faces.

Select Linked
Deselect
Delimit: Normal, Material, Seam, Sharp, UVs

Figure 7.12: *Body of the Pedestal's main column selected by Delimit method Sharp*

Note: Edge Select operations automatically enable Edge Select mode.

With the **Sharp markings** enabled, it is difficult to see the selection, so it may be helpful to hide only **Sharp markings** under the **Viewport Overlays menu** in the upper right of the **3D Viewport** as shown in *Figure 7.13*:

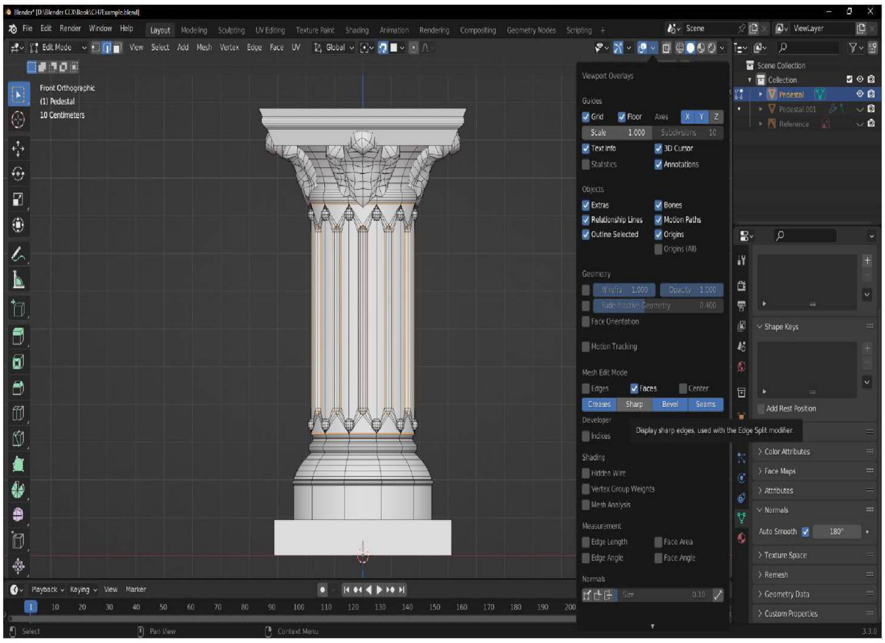


***Figure 7.13:** Boundary loops of the body of the Pedestal's main column selected and Sharp markings hidden*

5. Mark these edges as Seams, which will turn them red, by pressing the E key to access the Edge menu and choosing Mark Seam.

   Next, because cylindrical shapes must be split top to bottom in order to lie flat, we need to choose a single seam that runs the entire length of each cylindrical shape to mark as a Seam. In most cases it is best to choose the least visible areas for seams. In our case, since this object could be viewed from any angle and there are no seams directly in the center of the back or sides, let's create a new seam that will at least be partially hidden by a leaf detail.

6. Press **Ctrl** and Number pad key **1**, to rotate the view to the "back" of the Pedestal object.

7. Select Linked the body of the cylinder, and press the **Shift H** to hide all but the selected column.

8. In the **Vertex Select Mode**, with the whole cylinder visible as shown in *Figure 7.14*, press the **K** key to activate the **Knife tool**, then press and hold **Shift** to activate Midpoint Snap, click the top-most edge of the column, then with **Shift** still depressed, click the bottom-most edge and press **Enter** to confirm,

so that a new edge is cut along the body of the column as shown in *Figure 7.14*:

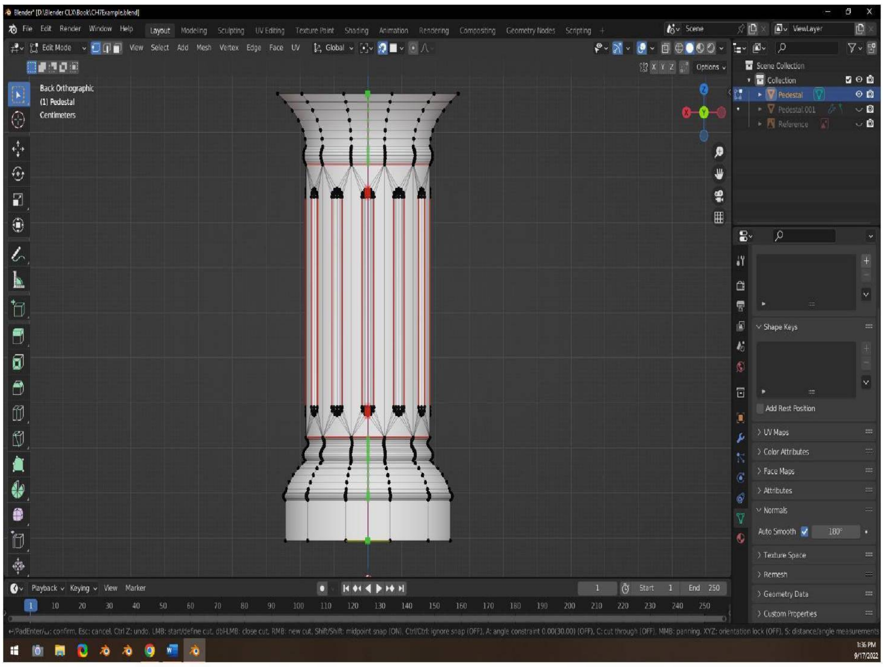


***Figure 7.14:** Boundary loops of the body of the Pedestal's main column selected and Sharp markings hidden*

9. Mark this new edge as a Seam.

10. Next, switch back to **Edge Select mode**, then Alt + click to loop select a horizontal edge anywhere that the silhouette of the column sharply changes direction as shown in *Figure 7.15*, then press and hold **Shift** to select the remaining horizontal edge loops with similar direction changes.

Tip: Open edges will automatically become seams, and do not need to be marked.

11. Mark these edges as Seams.

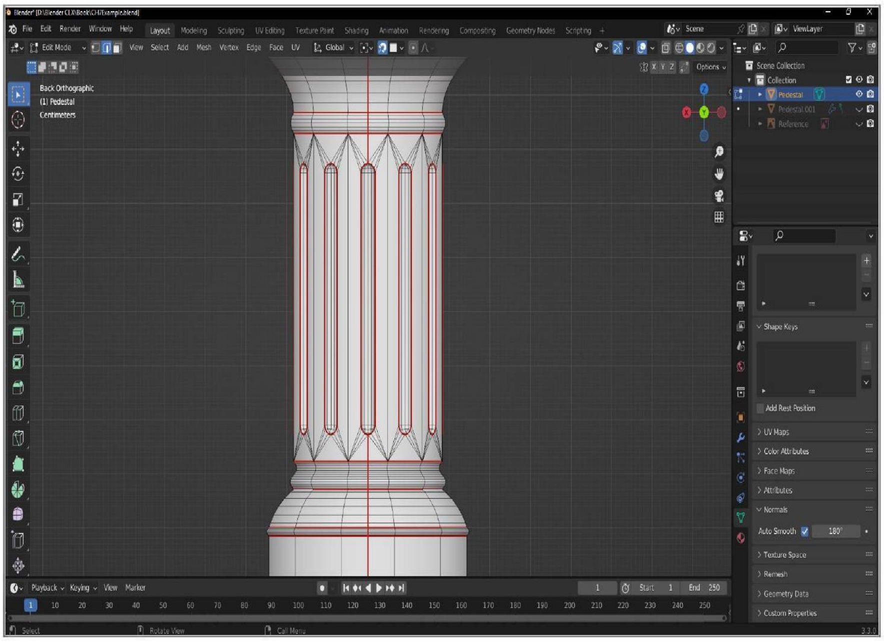


Figure 7.15: Horizontal edge loops marked as Seams at points where the silhouette sharply changes direction

12. Unhide all by pressing **Alt + H**.

13. Next, Select Linked the cap and base shapes, and press **Shift + H** to hide the rest of the mesh.

    It is neither necessary nor optimal to separate every face by a seam, so we do not need to mark every Sharp edge as a Seam. The desired approach is to only place seams where necessary so that the resulting UV islands can lie flat without overlap, twisting, or buckling, while keeping as many neighboring UV faces together as possible.

    The base and cap of our Pedestal object are essentially cube shapes, so we will follow a common unwrap approach for cubes, which is akin to cutting out a paper pattern to make a box, Leaving one "flap" open as a lid.

14. Beginning with the simpler base shape, in **Edge Select mode**, first select the bottom back and two bottom side edges of the bottom face and mark these as Seams as shown in *Figure 7.16*.

15. Next, select and mark all the vertical edges as Seams.

16. Lastly, select the top back and two top side edges of the top face of the cap, and mark these as Seams as shown in *Figure 7.16* as well:

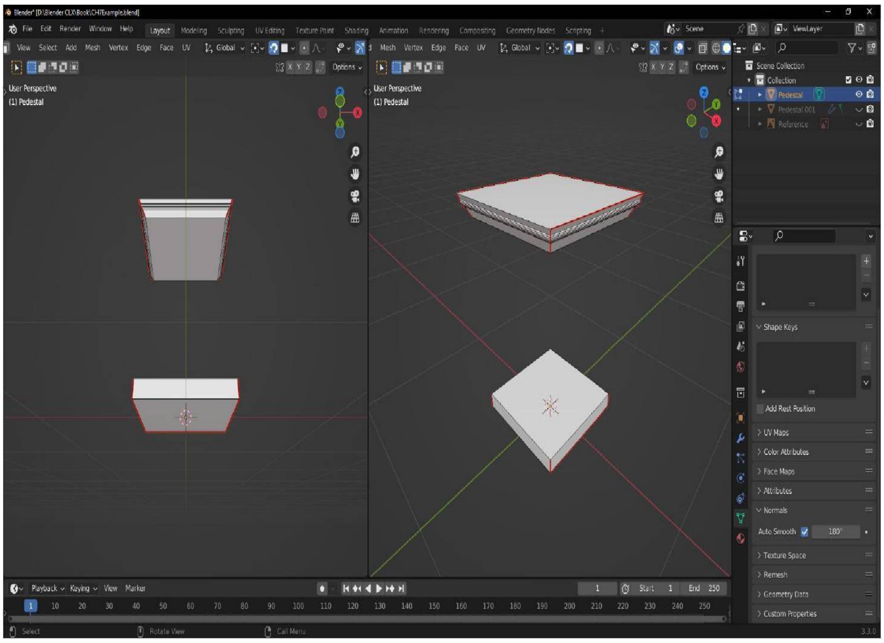


Figure 7.16: *Base and cap with Seam markings in the typical "t" shape or Box unwrap configuration*

Now our pedestal is ready to unwrap.

17. From the top-most menu in the Blender application, select the **UV Editing Workspace tab**.

18. Unhide all, select all, and zoom in to view the Pedestal object in full.

The left-hand panel is the **UV Editor Type**, and the right-hand panel is the **3D Viewport**.

Note: Shading and Overlay settings are unique to each Workspace, so it may be necessary to enable and disable desired settings when switching between Workspaces. Most of these Workspace settings are saved with the file, with the exception of the Rendered Viewport Shading mode, which is reverted to Solid between saving and reopening BLEND files, to prevent unnecessary slowdowns upon file opening.

If we've been following along with the previous exercises, the UV Editor view will look very similar to *Figure 7.17*. As noted previously in this chapter, not every object is UV unwrapped by default, so not all the UVs have been created yet.

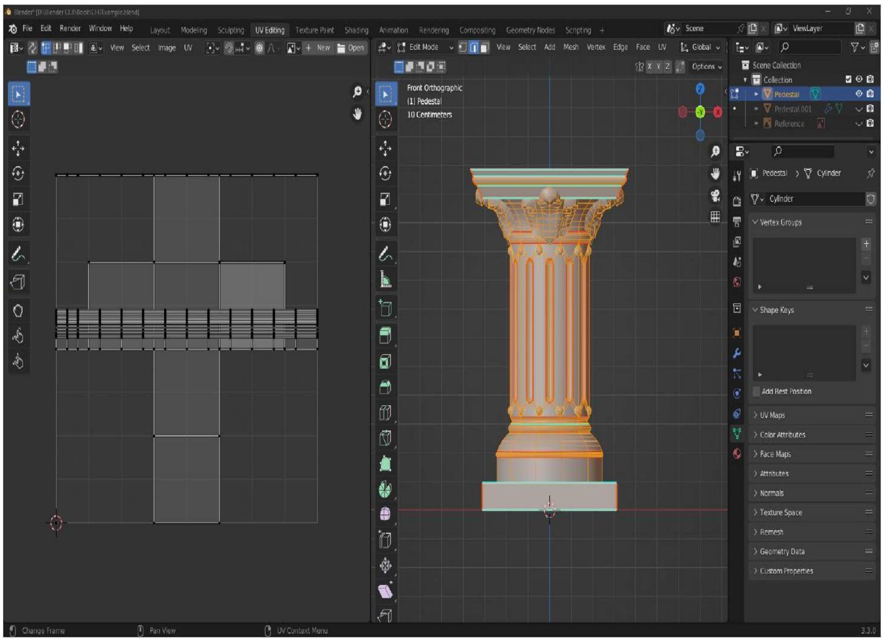


Figure 7.17: Pedestal mesh with seams marked, not yet fully UV unwrapped, shown in the UV Editing Workspace

19. With the mouse cursor over the **3D Viewport**, press the **U** key and choose **Unwrap**.

    If all goes well, the resulting UV Map will appear like *Figure 7.18*:

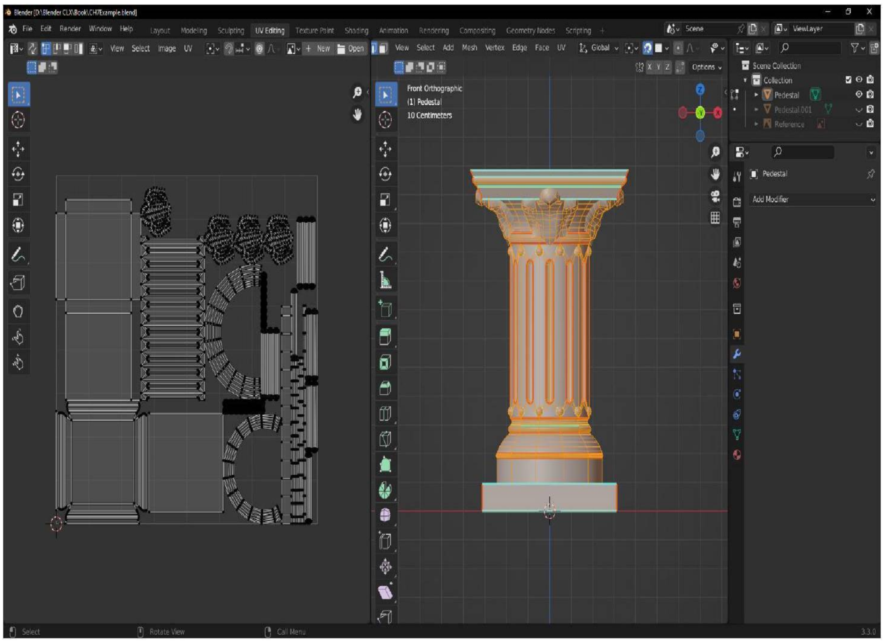


Figure 7.18: Pedestal on right with unwrapped UVs on left, shown in the UV Editing Workspace

Note: Our Pedestal example is particularly convenient to show an uncomplicated automatic UV unwrap, however it should be noted that many 3D shapes are more challenging to unwrap, and better results may be obtainable by employing one of the alternate Unwrap methods listed in the Unwrap menu accessed via the U key.

Convoluted 3D shapes may also require adjustment of the Unwrap operation's contextual parameters and may be aided by use of a plug-in designed to make specific UV unwrapping tasks easier. We will not be covering advanced UV unwrapping here, but as with many of Blender's advanced capabilities, we should not hesitate to seek out additional information on specialized workflows from the vast Blender user community as well as from official Blender documentation.

Exercise 7.3

For the final exercise of this chapter, we will set up our high poly duplicate mesh with a tiling marble material, assign a blank texture to the low poly original to receive the baked detail from the high poly mesh, perform the texture baking, and then save our texture maps.

First let's assign a blank texture to our low poly original Pedestal mesh with the newly unwrapped UVs.

1. Switch to the default **Layout Workspace**.
2. With the original Pedestal object selected, open the **Material Properties tab** of the **Properties panel** appearing as a dark and light checkered sphere, and in the panel press the **+ New button** to assign a new Material.
3. Click the name **Material** and give it a new name like Pedestal, or Pedestal Material if it prevents confusion.

   With the new Pedestal Material selected, the Material's properties will appear, and the Surface section will be expanded by default.
4. To add a new blank texture to our Pedestal, click the yellow dot to the right of the words Base Color which will open the texture slot menu in the Material's Shader, and choose **Image Texture** as shown in *Figure 7.19*.

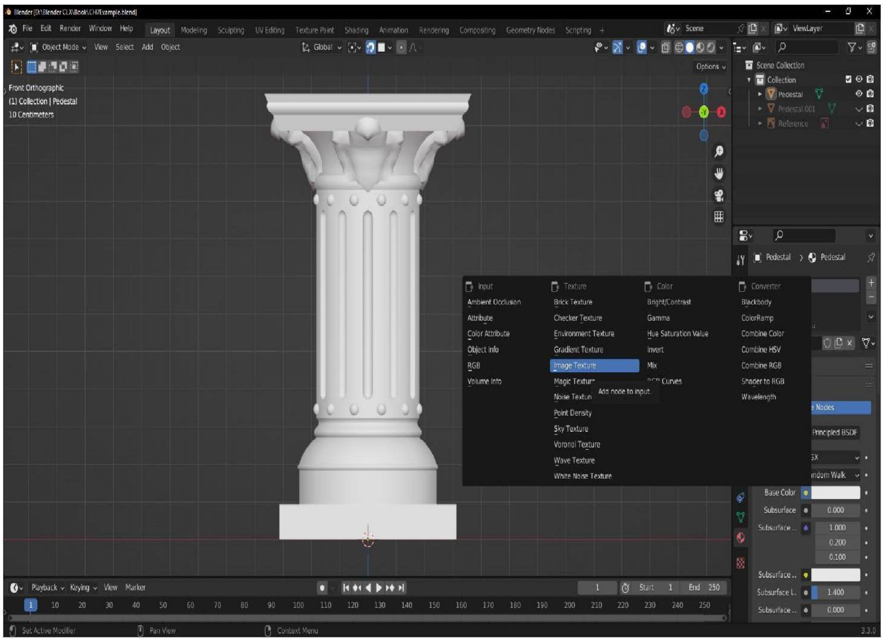


Figure 7.19: Adding a new Image Texture to the new Pedestal Material in the Shader's Base Color slot

Now the Base Color slot has a **+ New** button, as well as an **Open** button, where we could select a preexisting texture map.

5. Press the **+ New** button, and in the **New Image** menu that appears, type a new name into the **Name** parameter, such as Pedestal Base Color.

6. Uncheck the **Alpha** checkbox, as Alpha channels make the texture map larger to allow transparency, but the **Pedestal** object will not need transparency.

7. Leave the other settings the same and press **OK**.

8. Switch the **Viewport Shading mode** to **Material Preview**, from the icon appearing the same as the **Material Properties tab** icon, a sphere with a light and dark checker pattern in the upper right of the **3D Viewport**.

   If all goes well, the low poly Pedestal object will appear black in the viewport, the color of the texture map we just assigned to the Shader's Base Color slot in the Pedestal Material. This is a good time to save.

9. Next, let's hide the low poly original Pedestal object and unhide the high poly `Pedestal.001` copy.

10. Let's also add a new Material to the high poly copy, and name it Marble or something descriptive of the texture we plan to apply to it.

    Now, we could add an image texture node to the Base Color slot of the Shader as before, and either bake or paint a new image texture, or even use the **Open button** to add a premade texture, but instead let's make a simple procedural

marble to bake onto our low poly Pedestal's blank Base Color image. The advantage of procedural textures is that they wrap seamlessly around any object, which will make our pedestal look as if it was carved from a single piece of marble.

11. Switch to the **Shading Workspace** tab from the top of the Blender application window, zoom in to the high poly pedestal in the **3D Viewport**, and zoom out a bit in the **Shader Editor** below that, to get a better view of the Shader nodes.

    We'll see two nodes here, the Principled BSDF Shader that we've been working with in the Material Properties tab of the Properties panel, and the Material Output node, which displays the Shader output on the surface of the 3D object.

12. To begin, we'll add a **Noise Texture node**, by either pressing **Shift + A** with the mouse cursor inside the **Shader Editor**, clicking the **Search box**, and typing **`Noise Texture`** into the Search box that appears, or by conducting the same search under the Add menu at the top of the **Shader Editor window**.

13. Click the name **Noise Texture** that appears in the search, and a **Noise Texture node** will attach to the cursor, then click anywhere within the **Shader Editor** to place the new node, preferably to the left of the Principled **BSDF node**.

14. Repeat the search procedure and this time locate and place a **Mapping node** to the left of the **Noise Texture node**.

Tip: Most Editor Type panels can be navigated with the same Hotkeys we have already used in the 3D Viewport, such as A to select all, and Number pad period key to frame up the selection in the view, so any time we get lost in the Shader Editor viewport, it can be helpful to select all, then reframe the selection this way.

15. Repeat the search again and locate then place a **Texture Coordinate node** to the left of the **Mapping node** as shown in *Figure 7.20*:

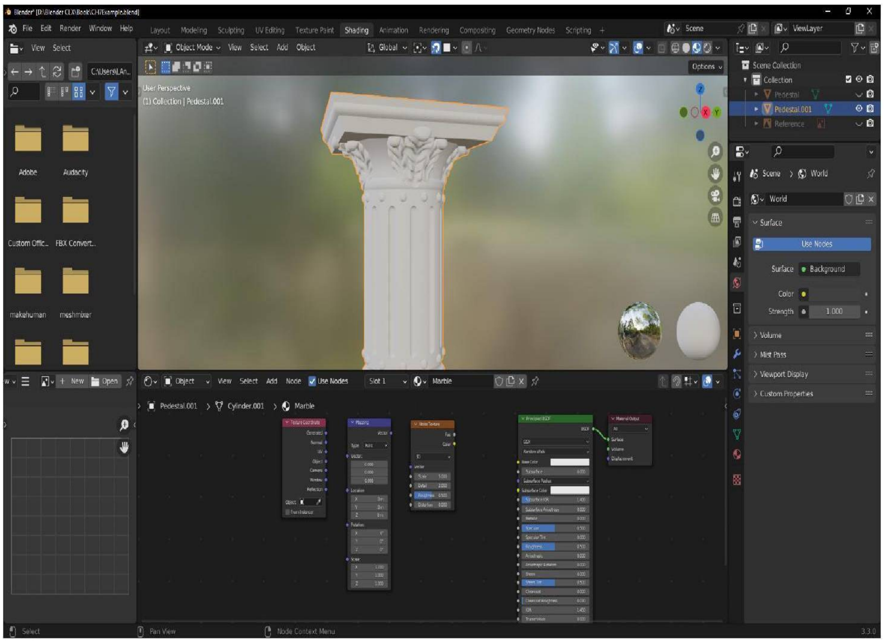


Figure 7.20: New unconnected nodes added to the Shader Editor panel in the Shading Workspace

16. Connect the new nodes in sequence by clicking the small circle to the right of the word **Object** in the **Texture Coordinates node** and dragging the wire to the right until it clicks into to the small circle to the left of the word Vector on the left side of the **Mapping node**.

17. Then connect the **Mapping node** to the **Noise Texture node** by clicking the small circle to the right of the word Vector on the right side of the **Mapping node** and dragging the wire to the right until it clicks into the small circle to the left of the word Vector on the left side of the **Noise Texture node**.

18. Next connect the **Noise texture Fac** output to the **Principled BSDF Base Color** input by clicking and dragging in the same manner.

19. In the **Noise Texture node**, scrub the Detail parameter as far to the right as it can go by default, which is **15**.

20. Move the **Texture Coordinate node** to the left of the **Mapping node**, far enough for another node to fit between them.

21. Select the **Noise Texture node**, and press **Shift** + **D** to duplicate it, and before confirming, drag the duplicate **Noise Texture node** over the wire between the **Texture Coordinate** and **Mapping nodes**, then click, and the wire will automatically reconnect the input and the output.

The Material will update on the mesh each time a change is made to nodes feeding into the **Material Output**, and by now should look a bit like marble. If we'd like the pattern to be larger or smaller, we can adjust the Scale parameters of either **Noise Texture**.

22. To allow adjustment of the colors of the **Noise Texture**, we can search again for a **ColorRamp** (omit the space in the search term) **node** and drag it to the wire between the **Noise Texture node** and the **Principled BSDF node** to insert it.

    The **ColorRamp node** is a gradient with "stops" appearing as small squares with triangles on top that correspond to the greyscale values of any input node. By default, the ColorRamp has one black stop at zero which maps to the darkest color of the **input node**, and one white stop at 1 by default which maps to the lightest color of the input node, so nothing much changes at first.

    To change the color of a stop in the **ColorRamp node**, click the small colored square with a triangle on top of it, and with the stop selected, click the large block of color below the stop number and Pos parameters to open the color picker and choose a new color from there. To change the position of the stops, click and drag them. Additional stops can be added and subtracted via the plus and minus signs, and the whole ramp can be inverted by choosing Flip under the drop-down arrow above the gradient.

23. Adjust the gradient as desired.

24. Next, let's add a small amount of the **Noise Texture** into the Material as height variations, by first searching and adding a **Bump node**, then feeding the ColorRamp into the **Bump node**'s Height input, then feeding the **Bump node**'s Normal output into the **Principled BSDF node**'s Normal input.

25. The resulting effect is too strong, so next we'll search and add a Clamp node in between the **ColorRamp** and the **Bump node**, leave the Min parameter at 0.000 and set the Max parameter to **0.200**.

26. For a final touch, we'll also feed the Clamp node's Result output into the Roughness channel of the **Principled BSDF node**, to produce variations in the shininess of the surface that match the variations in height.

The resulting effect should be like the Material as shown in *Figure 7.21*:

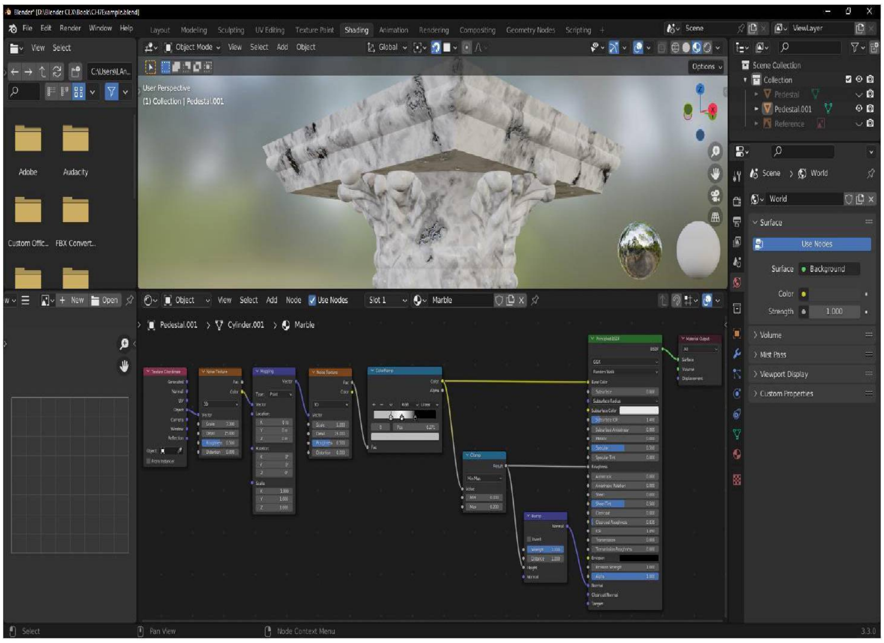


***Figure 7.21:** Fully connected nodes creating a procedural marble effect*

27. Next, let's switch back to the default **Layout Workspace** and begin to set up our bake.

28. Unhide the original low poly Pedestal object.

29. In the **Render Properties tab** of the **Properties panel**, under the icon appearing as the rear view of a digital camera screen, first change the Render Engine under the drop-down menu from **Eevee** to **Cycles**.

30. Next, scroll down inside the **Render Properties panel** to the **Bake section** near the very bottom, and expand the section.

31. Under the **Bake Type** drop down, select **Diffuse**.

    As mentioned earlier, **Diffuse** – along with Albedo or simply Color – is another name for Base Color, a texture map without lighting or height information added in.

32. Under **Influence**, deselect **Direct** and **Indirect**, as our scene has no lighting that we want to bake into our textures.

33. Enable the checkbox **Selected to Active** and expand the **Selected to Active section**.

    Here, we can adjust the **Extrusion** and **Max Ray Distance settings** which control how far away the source mesh can be from the target mesh before the texture

is ignored, which helps with more convoluted meshes, but may not be necessary for our pedestal. A typical workflow involves testing the default settings and adjusting as needed from there, which is how we will proceed.

34. Leave the remaining settings as they are by default.

35. Remembering again that Selected is darker orange and Active is brighter yellow by default, first select the high poly `Pedestal.001` object and then **Ctrl + Click** the low poly Pedestal object to make it Active.

36. Now, we can press the Bake button that appears at the top of the Bake section of the Render Properties panel and wait.

Note: Since version 2.8 or so of Blender, a warning about "Circular Dependency" will often appear at the beginning of a bake operation in the bottom-most Status Bar of the application, but this can safely be ignored.

Baking is a very computationally intensive operation and may take several minutes to complete. Once the bake starts, a loading bar will appear in the **Status Bar**, and then a completion message will briefly appear to alert us that the baked texture has been saved internally but must be saved externally or be packed into the `BLEND` file or it will be lost.

37. Before we save the baked image, let's first hide the high poly `Pedestal.001` mesh, to verify that there are no unexpected artifacts in the texture, which now appears in the Base Color slot of the original low poly Pedestal's Shader. Unhide Pedestal.001 after confirming the bake was successful.

    If unexpected artifacts appear in the new Base Color, such as a pixelated effect, see step 44 of this exercise for instructions on increasing the Extrusion amount to correct the issue, then re-bake before proceeding.

38. To save our new Marble Base Color image, let's switch to the Texture Paint Workspace from the top-most menu of the Blender application, and in the left-hand panel, we should see the **Image Editor** window displaying our newly baked texture as shown in *Figure 7.22*.

39. From the top menu of the **Image Editor panel**, select **Image**, then **Save As**, and save this image externally to our hard drive as `Pedestal_BaseColor` in the default PNG format. Select **RGB** rather than **RGBA** in the **Color options** of the **export settings**, as the A is for Alpha, and our image does not contain transparency.

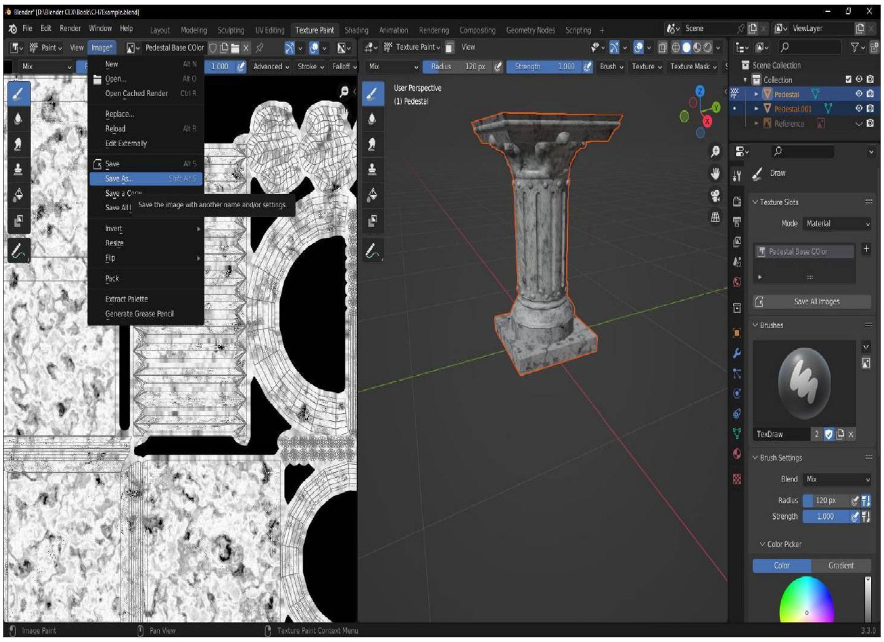


Figure 7.22: Saving the baked Base Color image externally via the Texture Paint Workspace

40. Next, let's switch back to the **Layout Workspace**, and back in the Bake section of the **Render Properties tab** of the **Properties panel**, change the **Bake Type** in the drop-down to **Normal**.

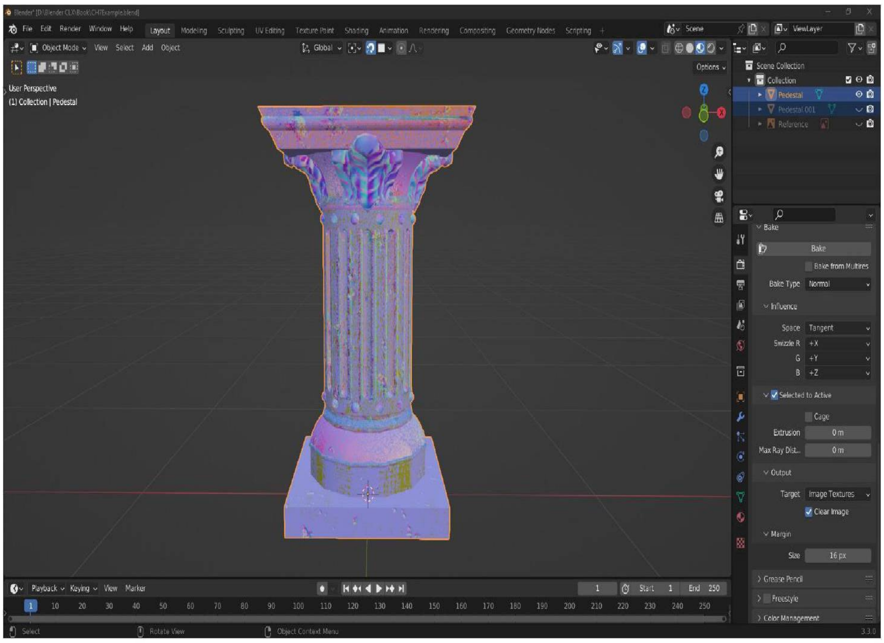
41. Once again, select the high poly `Pedestal.001` first, then **Ctrl** + click to make the low poly Pedestal object Active.

    We are going to bake our Normal map over the Base Color map in the next step, but that's alright because we just saved that image externally.

42. Press the **Bake** button again, and once again wait.

43. When the Normal map bake is complete, hide the **Pedestal.001 object** again to verify that the whole map has been baked. The Normal map will appear

as shades of purple like the MatCap used in earlier figures, but there should not be any bright green artifacts, as shown in *Figure 7.23*:



Bake

Bake from Multires

Bake Type: Normal

Influence

Space: Tangent

Swizzle R: +X

G: +Y

B: +Z

✓ Selected to Active

Cage

Extrusion: 0 m

Max Ray Dist..: 0 m

Output

Target: Image Textures

✓ Clear Image

Margin

Size: 16 px

Grease Pencil

Freestyle

Color Management

Figure 7.23: Baked Normal map with unwanted green shaded artifacts appearing

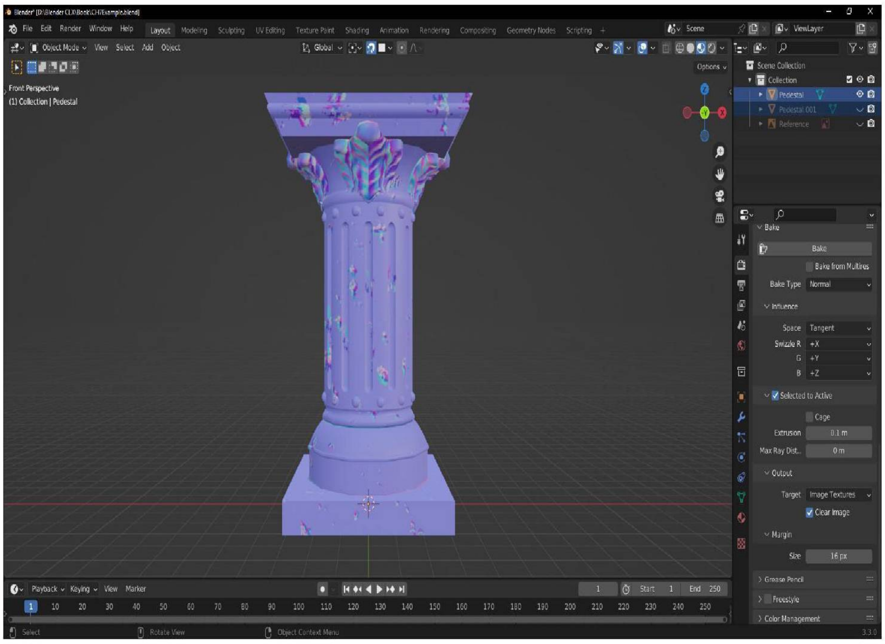
Green shading in baked Normal maps is usually the result of inverted normals on one or both source or target mesh but may also result if the distance between the source and target meshes is too small. In this case, since parts of our source and target meshes were identical, the problem is that there is no space between them.

44. To correct the green artifacts appearing from no distance between the source and target mesh, increase the `Extrusion` parameter a small amount – by 0.1m in this case – under the **Selected to Active section** in the **Bake section** of the **Render Properties tab** of the **Properties panel**.

    If our Normal map has green artifacts from inverted normals, we should refer to the methods of checking normal directions that we learned in *Chapter 3, General Concepts* and *Chapter 4, Polygonal Modeling*.

45. Once the problem is found and corrected, rerun the bake operation.

The desired Normal map should appear as shades of teal, blue, pink, and purple, like *Figure 7.24*:



Bake

Bake from Multires

Bake Type: Normal

Influence

Space: Tangent

Swizzle R: +X

G: +Y

B: +Z

☑ Selected to Active

Cage

Extrusion: 0.1 m

Max Ray Dist..: 0 m

Output

Target: Image Textures

☑ Clear Image

Margin

Size: 16 px

Figure 7.24: Correctly baked Normal map

46. Return to the **Texture Paint Workspace** and save the newly baked Normal map as **`Pedestal_Normal`** in PNG format.

47. Once again, we'll return to the **Layout Workspace**, and in the **Bake section** of the **Render Properties tab** of the **Properties panel**, we'll change the **Bake Type** to **Roughness**, select the high poly object first, then Ctrl + click to set the low poly object as **Active**, press the **Bake** button, wait, and save the resulting map as **`Pedestal_Roughness`**.

48. Finally, to give an added touch of depth to our low poly pedestal, we will switch the Bake Type to Ambient Occlusion, run the bake again, and save the resulting map as **`Pedestal_AO`**.

Note: Because marble is shiny but does not produce a mirror reflection, a default value of zero can be set for the Metallic value when we go to assemble these texture maps into the Pedestal object's Material. For any 3D software such as some game engines which may require an image map to drive the Material's Metallic value, a very small (32 pixels by 32 pixels, for example) solid black image can be used.

To assign our newly baked texture maps to the Material of our low poly Pedestal object, let's first enable a handy built-in add-on, which will automatically wire up the nodes.

49. Via **Edit | Preferences | Add-ons**, type the word "**node**" into the search box, and enable the Add-on Node: **Node Wrangler**.
50. Then, with the high poly **Pedestal.001 object** hidden, and the low poly Pedestal selected, switch to the Shading Workspace again.
51. The image we added that was originally blank is still wired into the Base Color input of the **Principled BSDF Shader**, so select the whole node and delete it by pressing the **Delete** or **X** key.
52. Next, select the **Principled BSDF Shader** by left clicking, then all at once, press **Ctrl + Shift + T** which will open **Blender's File View** where we can browse to the location where we saved our baked textures.
53. In the **File View**, click the first Pedestal texture map, then **Shift +** click to select all four, then press **Enter**, or the button in the bottom right. This will automatically assign all the baked textures to our Pedestal, except the Ambient Occlusion map which will be added to the **Shader Editor** but not connected.

    To connect the Ambient Occlusion map to the Pedestal's Material, we need to blend it with the Base Color.
54. In the **Shader Editor panel**, search and add a **MixRGB node** between the **texture node group** and the **Principled BSDF node**.
55. Then connect the Color outputs of the Ambient Occlusion and Base Color texture nodes to the **MixRGB node**'s Color1 and Color2 inputs, change from Mix type to Multiply in the node's drop-down menu, increase the Fac parameter to 1, and replace the Base Color input in the **Principled BSDF node** with the **MixRGB node**'s Color output as shown in *Figure 7.25*:

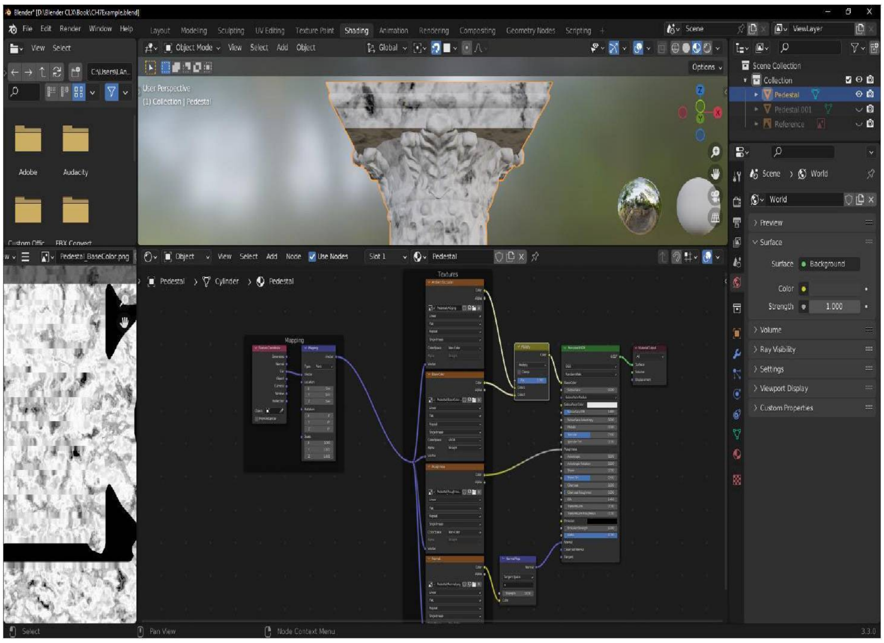


Figure 7.25: Low poly Pedestal object with baked textures assigned to the Material's Principled BSDF Shader

That's it! We have now completed modeling, detailing, and texturing a 3D object that is suitable for games and interactive applications. Many of the stylistic choices we have made along the way are optional and can be recreated in alternate ways to achieve different effects now that we know the results that each of the steps in our process can be expected to produce. Please refer to the following figure:

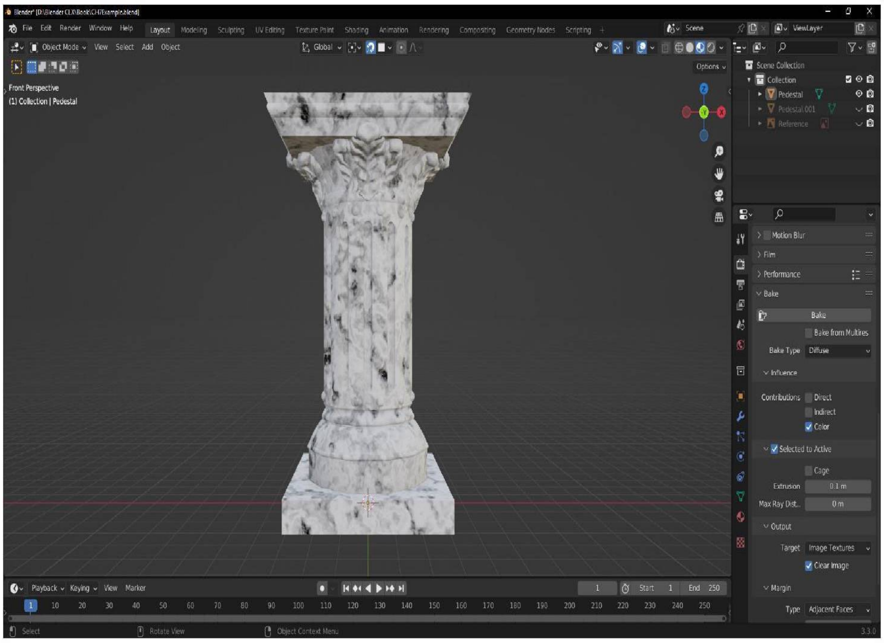


Figure 7.26: Completed low poly 3D Pedestal object

Conclusion

In this chapter, we have learned the anatomy of 3D surfaces and how a variety of surface treatments are constructed. We also learned the necessity of UV Maps and how to get started UV unwrapping our objects in preparation for a variety of surface treatments. Finally, we obtained hands-on experience preparing our own poly modeled pedestal for Texture, set up a procedural marble material, and baked a Normal map to transfer the sculpted detail from the previous exercise onto a low poly model that is ready for real time and interactive applications.

Essential shortcuts and hotkeys

In this chapter we used the following shortcuts and hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R

Snap Scale	Ctrl + S
Show/Hide Side Bar	N

Shortcut	Hotkey
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Table 7.1: *Essential Shortcuts and Hotkeys*

Points to remember

- Some 3D surface treatments are primarily for practical purposes and don't transfer into other applications, while others are meant for viewing by end users.
- An unlimited variety of surface treatments can be created by combining Textures in Materials and modifying the properties of their Shaders.

- 3D objects must be UV unwrapped for certain treatments to map as intended to their surfaces.

- Baking surface treatments into Textures makes the intended appearance of our 3D objects portable into other applications.

Questions

1. What are some of the properties of real-life materials that can be produced with 3D Materials?
2. What are commonly used Texture sizes and why are they common?
3. When are UV Maps needed, and when are they not necessary?
4. What is the purpose of marking UV Seams?
5. How do Normal maps fake height information?

CHAPTER 8
3D Animation

Introduction

Now that we have a firm understanding of how to navigate Blender, create 3D objects, and edit their appearance, we are ready to make our 3D objects move.

Structure

This chapter covers getting started with animating our 3D creations. Topics to be covered include:

- The anatomy of an animation: Timelines, Keyframes, and animatable properties
- Exercise: recording Keyframes to move the default Cube
- Common animation types: mesh, bone, and physics animation

- Exercise: practicing bone and mesh animation
- Editing animations
- Introduction to advanced 3D animation concepts

Objectives

After reading this chapter, we will understand how to get started moving our 3D creations, as well as which other animatable properties exist.

Anatomy of animations

The base unit of any animation is the **Timeline**. Timelines are linear markings of the time in which an animation plays within the sequence of frames that are displayed on a computer screen. Each segment of a **Timeline** corresponds to a single frame in a complete animation. Frames in digital animation are the corollary to still frames in a physical reel of film, which is a sequence of images strung together and played so fast that the human eye is fooled into believing the evolving motion displayed on each frame is fluid.

Every 3D object in Blender has its own **Timeline**, which can contain an infinite number of frames where various properties such as the object's location, rotation, and scale at that point in time can be recorded in a type of object data called Keyframes. Then, when the animation is played, Blender automatically calculates – or "interpolates" – the movement or change that should occur between any two recorded **Keyframes**. Recording a 3D object's Transforms, then moving, rotating, or scaling it, then recording its **Transforms** again, is the basis of 3D Animation.

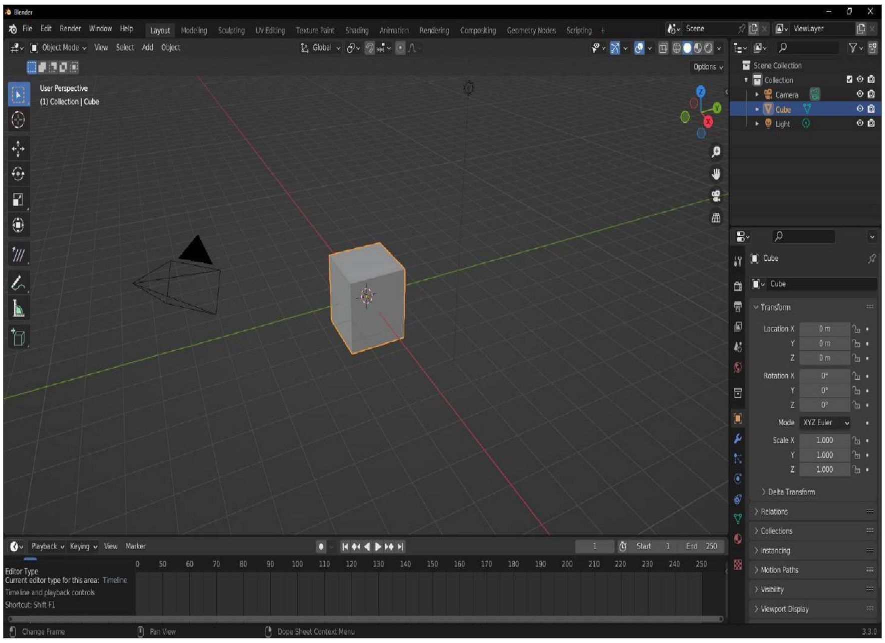
Note: Since the possible number of frames is infinite in 3D, the first step to animating 3D objects is to establish a total number of frames, which can later be adjusted as needed. In Blender, the default number of frames for a new animation is set to 250, as seen in the box to the right of the word End in the upper Timeline menu on the far right of the Timeline viewport. The total number of frames in an animation can be changed by typing a new number into the End parameter field.

Animatable properties

The properties of a 3D object which can be animated vary widely between different 3D applications, with some core similarities. In Blender, the basic **Transforms** of a 3D object, such as its location, rotation, and scale, can be animated as expected, but almost any property that can be changed in Blender can also be recorded as **Keyframes** on the **Timeline** for later playback. Some examples of animatable properties besides basic **Transforms** include a **Material or Shader's parameters**, such as the settings we adjusted to customize the surface of the Pedestal object in *Chapter 7, 3D Surfaces*, as well as the growth of particles and the progression of physics simulations like cloth and water, which we will explore in *Chapter 9, Effects and Simulations*.

Exercise 8.1

By default, an **Editor Type** that we have not yet explored in Blender called the **Timeline** exists directly below the familiar **3D Viewport**, as shown in *Figure 8.1*. By clicking and dragging the boundary between the top and bottom **Viewport panes**, we can expand the **Timeline**, which is where the **Keyframes** for any selected 3D object are recorded. Please refer to the following figure:



Editor Type
Current editor type for this area: Timeline
Timeline and playback controls
Shortcut: Shift F1

Figure 8.1: The Timeline Editor Type at the bottom of the Blender application, expanded vertically

As with most 3D operations, working with the **Timeline** requires a current selection. In this short exercise, we will record a simple animation of the default **Cube** object. For this, we can open a new instance of Blender.

1. First, you must ensure that the object we wish to animate – in this case, the **Cube** – is selected.

2. Next to limit which **Transforms** any new **Keyframes** will record, we will choose an **Active Keying Set** by opening the drop-down menu under **Keying** in the upper menu in the **Timeline**.

In our case, we want to choose the **Location, Rotation & Scale** as the **Active Keying Set**, as shown in *Figure 8.2*:

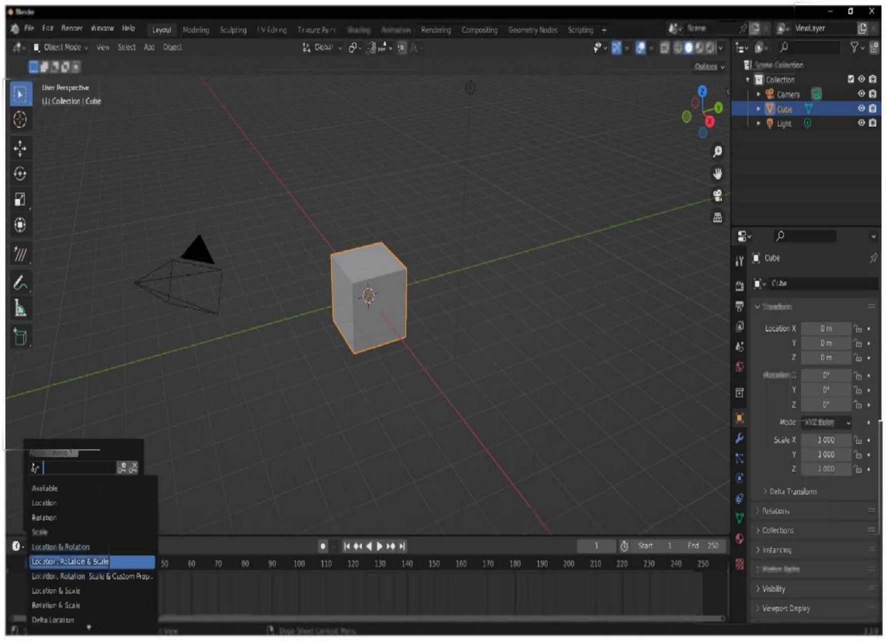
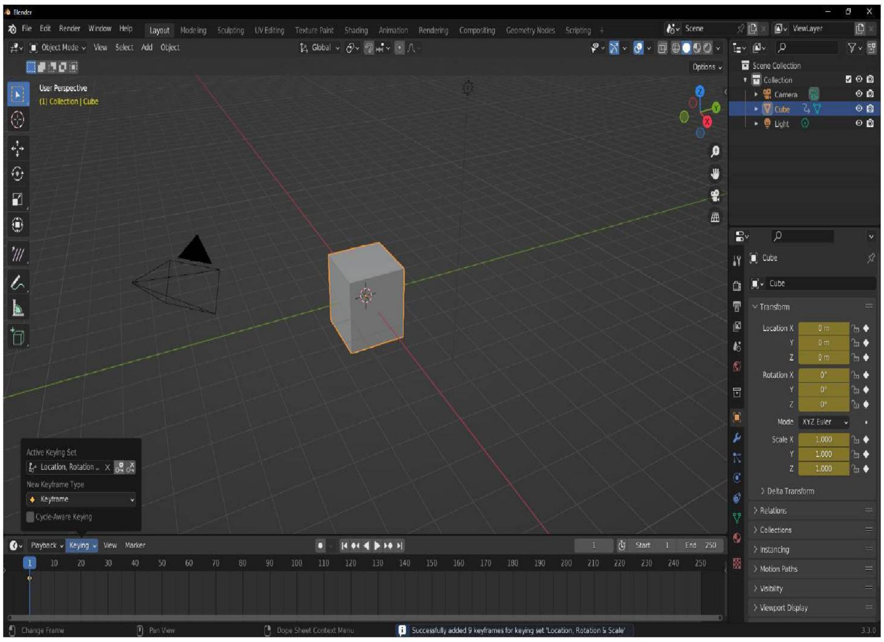


***Figure 8.2:** Setting the Active Keying Set to record Location, Rotation & Scale*

3. Then, we will record the current **location, rotation, and scale** of the **Cube** object by opening the **Keying** drop-down menu again and pressing the icon appearing as a key with a small plus sign, as shown in *Figure 8.3*:



Successfully added 9 keyframes for keying set 'Location, Rotation & Scale'

***Figure 8.3:** Setting a starting keyframe for the animation we are about to record*

Note: The Transform parameters displayed in the Object Properties panel of the Properties panel have become highlighted to indicate the existence of a Keyframe recording their state at that point in the Timeline. The same highlight can be observed in the Side bar menu's Item tab, enabled by pressing N with the mouse cursor inside the 3D Viewport. Keyframes can also be recorded in both panels by right-clicking and choosing Insert Keyframe for the highlighted parameter.

4. Once we have a **Keyframe** recorded on the first frame of the animation, we can click and drag the play head – appearing as a rounded square icon with a thin line extending below it around the number 1 on the **Timeline** – to the right until it highlights the frame number 100 as shown in *Figure 8.4*:

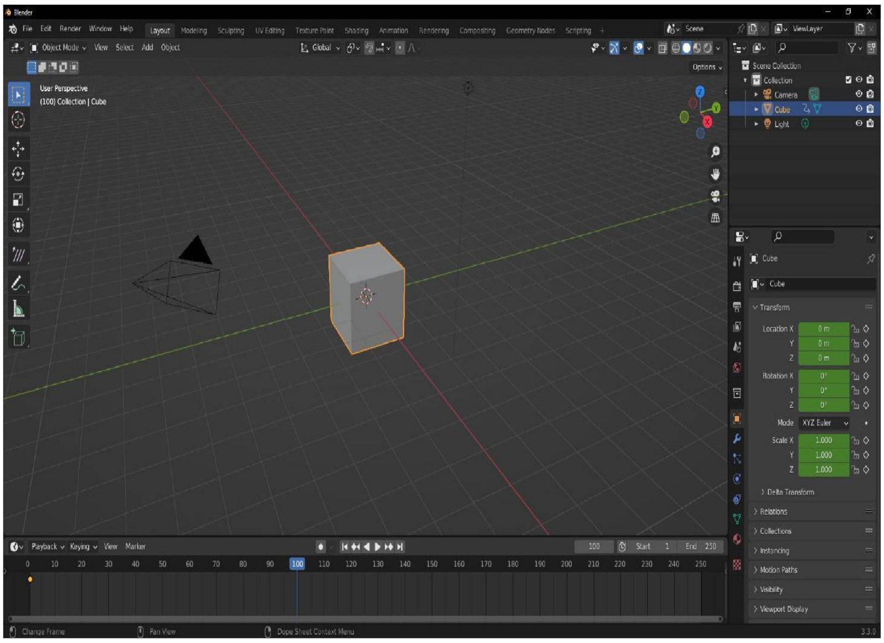
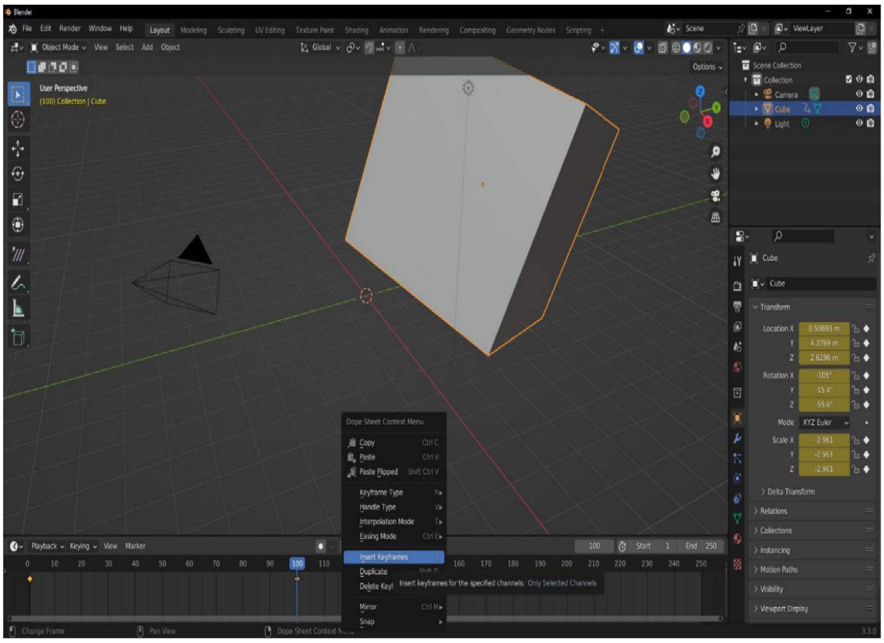


Figure 8.4: Animation play head moved to frame 100 in the Timeline

5. At this frame, let us **move, rotate, and scale** the **Cube** object by pressing **G**, moving the mouse and left-clicking to confirm, pressing **R** and confirming, and then **S** and confirming.

6. Right-click anywhere along the **Timeline** and choose **Insert Keyframes** as shown in *Figure 8.5*, and a new **Keyframe** will appear in the **Timeline** where the play head is currently set, recording the **Transforms** we indicated with the **Active Keying Set** we chose earlier.



Insert keyframes for the specified channels: Only Selected Channels

Figure 8.5: Inserting a Keyframe which records the chosen Active Keying Set in the Timeline

7. We can now move the play head back to frame 1, either by clicking and dragging it, or by pressing the "**jump to end point**" button appearing as a left-facing arrow against a vertical line in the middle menu above the **Timeline**, and then press the right-pointing triangle to activate the **Play mode**, and see our animation run.

   After the play head passes all the **Keyframes** we set, nothing will happen until the play head reaches the end of the **Timeline**, at which point it will loop back to frame 1.

Note: Whether an animation loops seamlessly or not depends on how the Start and End Keyframes are set up, as well as whether the Curves of the animation – which we will explore later in this chapter –coming to the Start Keyframe and going out of the End Keyframe are set to a Linear interpolation or not.

8. Press the **Pause button** appearing as two vertical lines in the middle menu above the **Timeline** to stop playback.

9. With the mouse cursor inside the Timeline Viewport, clear the selection by clicking away from any **Keyframe**, then left-click or Box select the first **Keyframe**, and press **Ctrl + C** to copy that **Keyframe**.

10. Reset the play head to frame 1 again, and press **Play**.

Now, the animation we recorded runs forward once, then backward before repeating again.

Animation types

Now that we have experienced the first steps of animating in Blender, let us explore the most common methods for making 3D objects move. There are two primary ways to animate 3D objects. One method is to record **Transforms** directly on the object, or onto the sub-object components of the mesh itself, called "mesh" animation. Another method involves recording the Transforms of a proxy object which serves as a virtual skeleton to the associated – or "rigged – 3D object called "bone" animation. The choice of which approach to employ largely depends on the target software where the animation will be played.

For animations to be played and rendered exclusively in Blender, either option works equally well, but many external applications cannot read or play mesh animations. Bone animation is often preferred for mechanical and character animations because manipulating a bone that controls predetermined groups of vertices a predetermined amount dramatically simplifies the **Transform operations**.

Bone animation allows precise control over how much each Transform influences each sub-object component of the mesh through a process called "weight painting," as well as allowing for different relationships to control movement called "constraints." Mesh animation, on the other hand, allows for quicker "on the fly" animation, with the trade-off of less precise and more time-consuming control.

Bone Animation

The process of bone animation begins with adding the skeletal structure – alternately referred to as an "armature" or "rig" – into the scene, followed by assigning a 3D object to the Armature object and setting up which vertices of the mesh that each bone of the armature will control. To add an **Armature** to the 3D scene, we can use the **Add menu** and select **Armature**, which by default will add a single bone. Through Blender's **Edit** | **Preferences** | **Add-ons dialogue**, we can also search for `rig` to find and enable Rigging: Rigify, which gives us access to additional **Armatures** that are conveniently premade for humanoids and other common creatures.

How to assign individual vertices to be controlled by individual bones and how to move entire 3D objects that are rigged for animation this way is a complex process that is covered in detail by many free and paid tutorials found online.

Tip: To locate tutorials that are best suited for our animation goals, it is helpful to search using terms that include the animation type, such as "Blender character animation tutorial" or "Blender mechanical animation tutorial" and then filter the search results by tutorials that have been updated within the past year.

Because of the more precise control afforded through the application of vertex weights to each bone in a bone animation rig, bone animation is the preferred method for animating 3D characters. Like the real bones of a humanoid or creature, the bones of an armature can affect more than one area of a character's body, and multiple bones work in chorus with one another. When we rotate our hand in space, for example, the skin around our wrist also moves, most prominently close to our hand, and less prominently closer to the elbow, with a gradual falloff of bone "influence" that varies from one body part to another.

Bone animation and mesh animation can also be combined to great effect, to more accurately mimic the behavior of muscles contracting and elongating underneath skin as a skeleton moves.

Mesh animation

Mesh animation is different from bone animation in that it is recorded onto the 3D object itself, rather than in the **Timeline**, so that it can later be **Keyframed** via the **Timeline** or in another 3D application entirely. Mesh animations can be used to "morph" one object into another, so long as the two shapes share the same exact vertex count and order. Since mesh animations work by recording the **position, rotation, and scale** of individual vertices of a mesh rather than the positions of a proxy object, they are ideal for animating small parts of a 3D object such as eyelids blinking.

In Bender, mesh animations are recorded to an object data type called **Shape Keys**, which are then usable by other 3D software such as game engines, usually referred to as Blend Shapes. To set up **Shape Keys** in Blender, we must first finalize the design of the 3D object and then add a special **Shape Key** called the Basis, which can be used to return the 3D object to its original form. Then, we can add an infinite number of new **Shape Keys**, which each contain data about the alternate shapes that we want the mesh to be able to morph into. After adding a **Basis Shape Key**, many of Blender's Modifiers can be applied as **Shape Keys**, making it possible to transition the mesh from its original state to the Modified state and back again via sliders on each **Shape Key**, which can then be recorded as **Keyframes**.

Physics animation

Another type of animation we will learn about in *Chapter 9, Effects and Simulations* is a form of simulation. Physics animations have the advantage of producing more

plausibly natural results with less human effort, by using simulated forces like gravity, wind, and water.

Exercise 8.2

Before moving on, let's combine our object's animation from *Exercise 8.1* with two more animation types to see how they can work together.

1. First, we'll navigate to the **Object Data Properties tab** of the **Properties panel**, and in the **Shape Keys section**, press the plus sign to the right of the empty area which will add a **Basis key** as shown in *Figure 8.6*.

   This has recorded the **Transforms** of all the sub object components of the selected Cube mesh, as they are now. Please refer to the following figure:

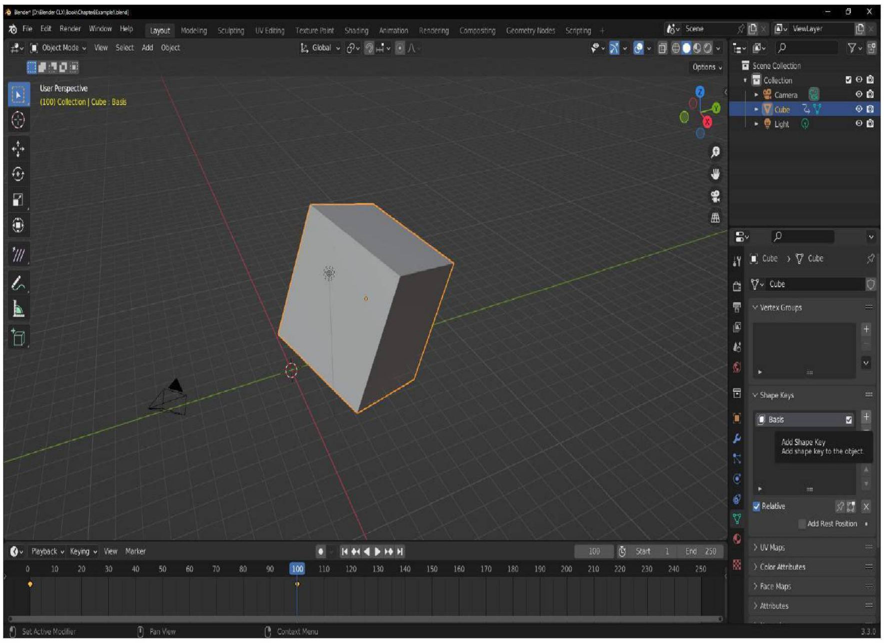


Figure 8.6: Adding a Basis Shape Key via the Object Data Properties tab of the Properties panel

2. Then, press the plus sign again, to create an empty Shape Key named Key 1, that we can record a change to.

3. Next, we'll enter **Edit Mode**.

4. Select any of the vertices, and change its position, as shown in *Figure 8.7*:

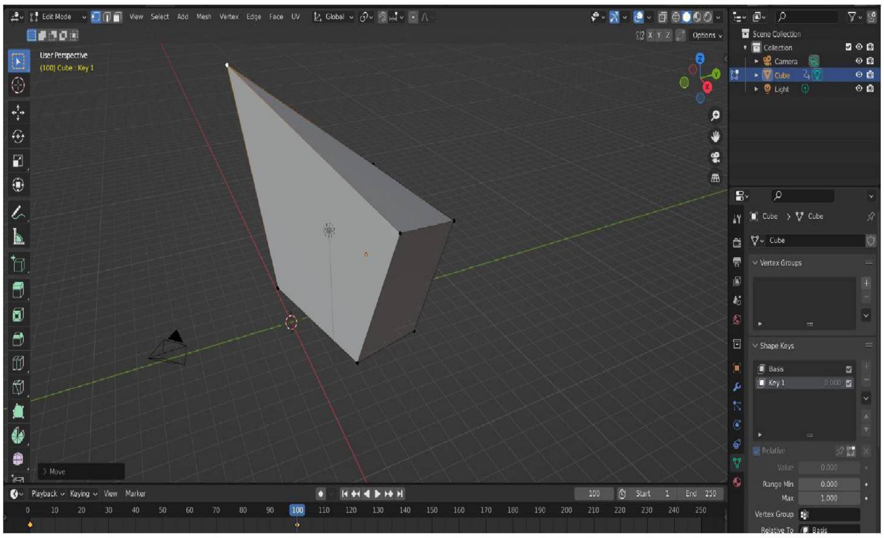


Figure 8.7: Making a change to the mesh that will be recorded to the Shape Key 1

5. Exit **Edit Mode**, and the change that we just made in **Edit Mode** will appear undone.

   This is the desired effect. To make use of this **Shape Key**, we will add **Keyframes** to the **Timeline** recording a change in its **Value**.

6. Set the play head in the **Timeline** to 1.

7. In the Shape Keys section of the **Object Data Properties tab**, with Key 1 selected, move the mouse cursor over the **Value** range below the **Shape Keys**, right-click, and choose **Insert Keyframe** as shown in *Figure 8.8*:

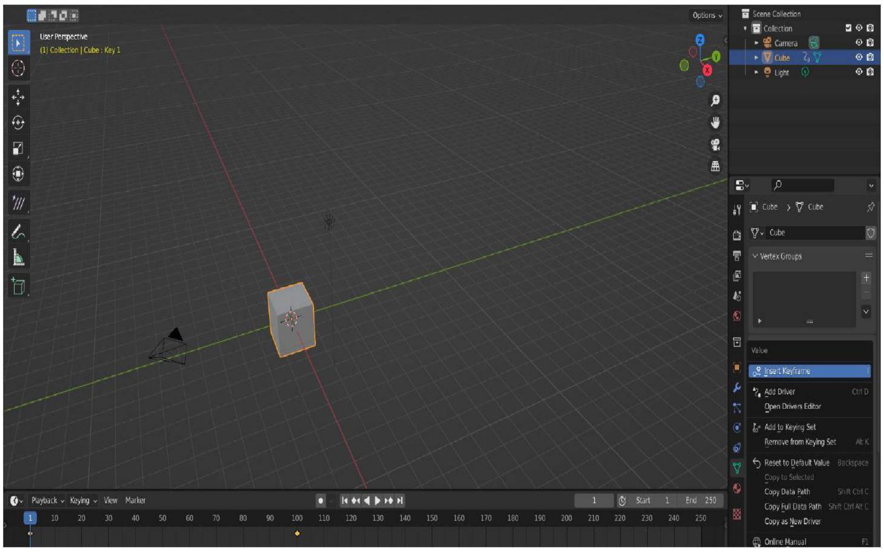


Figure 8.8: Inserting a Keyframe for the selected Shape Key Value 0.000

8. Next, move the play head in the **Timeline** back to frame 100.
9. With the **Key 1 Shape Key** still selected, increase the Value from 0.000 to 1 by typing or scrubbing inside the **Value parameter** field.
10. Right-click in the area over the **Value parameter** again, and select **Insert Keyframe** again.

The **Value field** will become highlighted, and the vertex position change we recorded earlier will be reflected in the **3D Viewport** as shown in *Figure 8.9*:

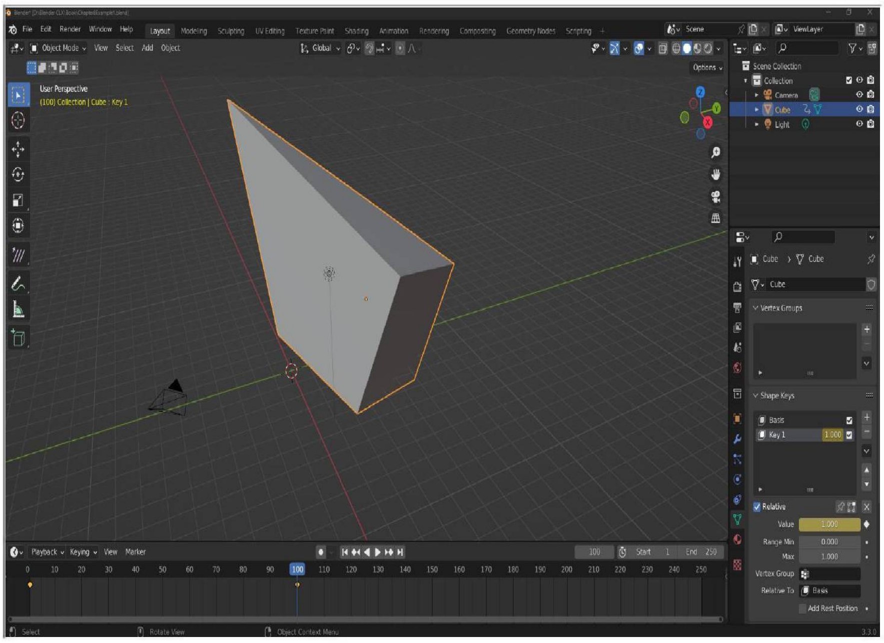


Figure 8.9: Inserting a Keyframe for the selected Shape Key Value 1.000

11. Reset the play head to frame 1 again, and press **Play**.

Now, we have two mesh animations recorded on our cube. Next, we'll add a single bone animation.

12. Set the play head back to 1.

13. In **Object Mode**, bring up the **Add menu** and add an **Armature** | **Single Bone** as shown in *Figure 8.10*:

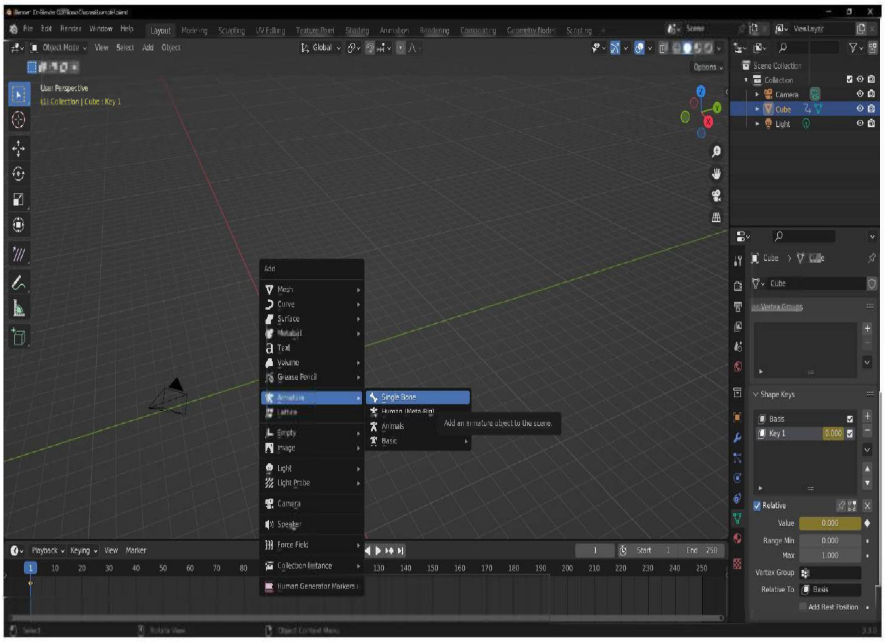


Figure 8.10: Adding a single bone Armature to the scene

The newly created bone is obscured by the **Cube**, so we'll change the **Armature's Viewport Display** to **In Front**, making it visible from any viewpoint.

Note: Armatures do not appear in images or videos such as those we will explore in Chapter 10, Images and Video that are rendered from the 3D Camera, regardless of their Viewport Display mode

14. In the **Object Data Properties** tab of the **Properties panel** for the **Armature** object, expand the **Viewport Display section** and check the box for **In Front** as shown in *Figure 8.11*:

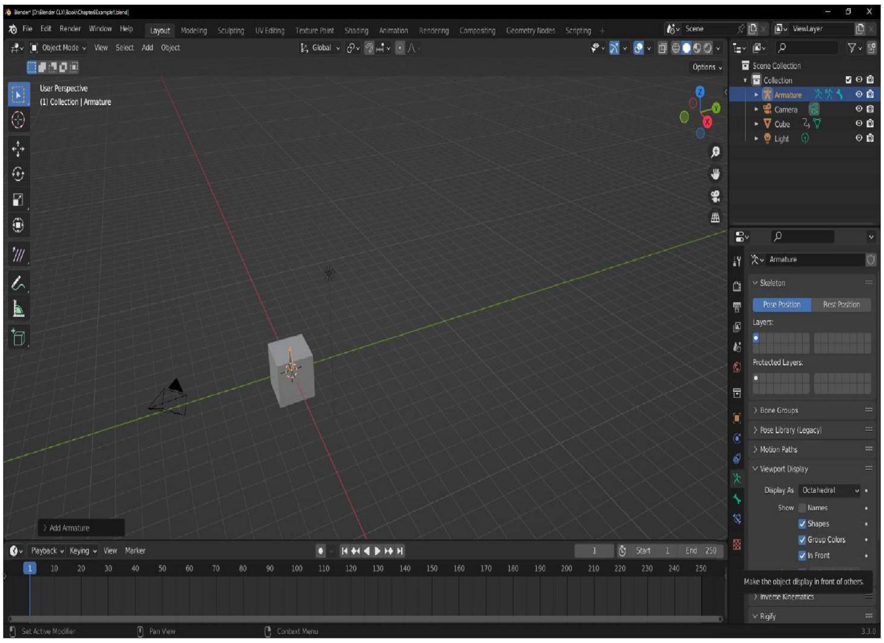


***Figure 8.11:** Enabling the In Front Viewport Display option on the Armature object*

15. Deselect the Armature object, then in the Outliner panel, select the Cube object, then **Ctrl** + click the Armature object to set it as the Active selection.

16. Press **Ctrl** + **P** and select **Armature Deform | With Empty Groups**, to parent the Cube to the Armature.

To assign a group of vertices to be controlled by the Armature, we'll need to create a Vertex Group.

17. With the Cube selected, enter **Edit Mode**.

18. Select any two of the vertices that were not part of the Shape Key animation.

19. In the Object Properties tab of the Properties panel, at the bottom of the Vertex Groups section above Shape Keys, click the Assign button, to assign the selected vertices to the Bone group that was added in a previous step.

20. Exit **Edit Mode**.

21. With the play head at frame 1, select the Armature object.

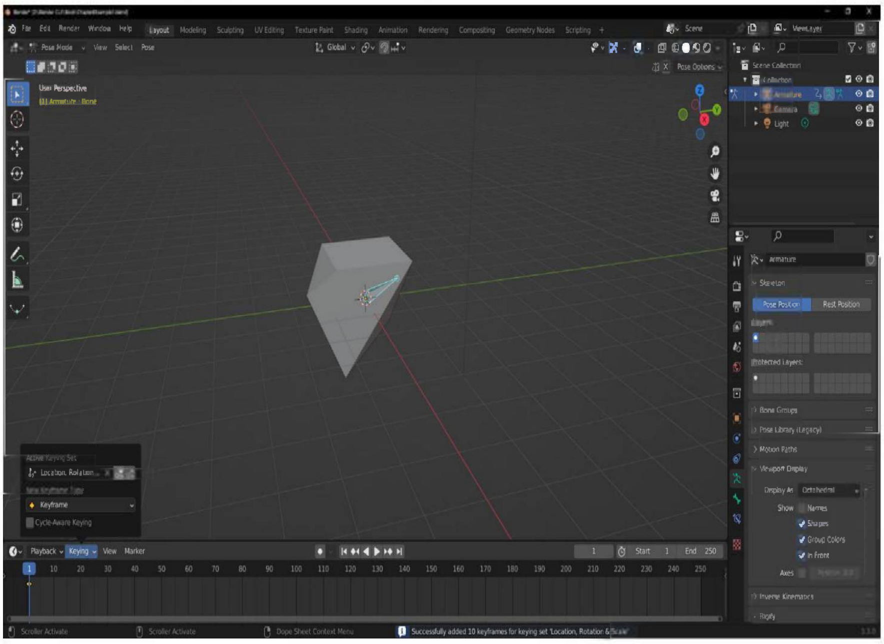
22. From the upper left drop down menu, under Object Mode, select Pose Mode.

    Pose Mode is a special mode only available to Armature objects, for bone animation.

23. Select the bone and rotate it a few degrees in any direction.

The Armature object has its own Timeline, separate from the Cube object's Timeline, so we could select a different Keying Set if we needed to.

24. In the Timeline Viewport menu, under Keying, press the icon appearing as a key with a plus sign inside a circle again, to insert a new Keyframe for the Active Keying Set as shown in *Figure 8.12*:


Active Keying Set
Location, Rotation...
Keyframe
Cycle-Aware Keying

Successfully added 10 keyframes for keying set 'Location, Rotation & Scale'

Figure 8.12: Inserting a Keyframe recording the Armature bone's pose

25. Just as before, let's move the play head to frame 100, and press **Alt + R** to clear the rotation of the bone.

26. Next, let us scale the bone an arbitrary amount.

27. With the mouse cursor inside the **Timeline viewport**, right-click and choose **Insert Keyframe.**

Now, when we move the play head back to frame 1 and press **Play**, all three animations play at once: the whole object's mesh animation, the sub object component's mesh animation, and the **Armature's bone animation**, controlling a group of the Cube's vertices.

28. Exit Pose Mode by pressing **Ctrl + Tab**.

Editing animations

Now that we have practiced the basics of recording 3D animations, we are ready to begin editing animations. In Blender, as with many applications that allow for 3D animation, there are multiple facets of the animation itself. First, there is the

front face of the total animation on a single 3D object, the **Timeline**. Behind that, **Curves**, just like those we worked with in *Chapter 5, Poly Modeling Extras*, control the interpolation of the changes between **Keyframes**.

We can access tools to edit recorded animations by changing any **Editor Type**, such as the **Timeline**, into a **Graph Editor** as shown in *Figure 8.13*. Then we can see that – just as the handle type, position, scale, and rotation of a 3D Curve object's control points change the shape and direction of segments coming into and going out of them – each Keyframe's Handle Type can be set to affect the direction and speed of the changes occurring between them via the **Key menu** in the **Graph Editor** as shown in *Figure 8.13*:

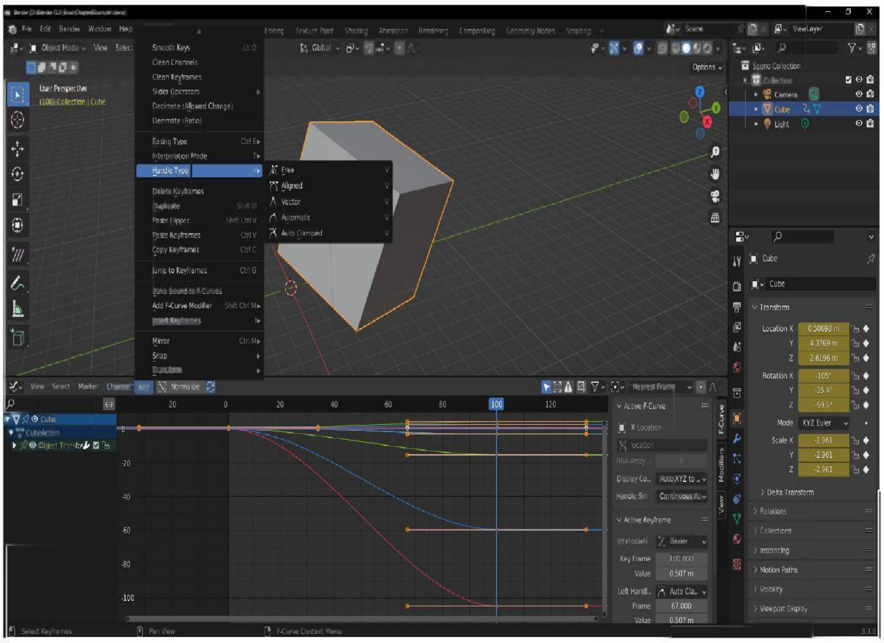


Figure 8.13: Accessing the Key menu from the Graph Editor

Also shown in *Figure 8.13*, in addition to changing the animation curve's Handle Type, we can also choose a variety of *Interpolation Modes and Easing Types*. It is advisable to experiment with these options in an animation that is meant for discarding, like the one we created in *Exercise 8.1*, to become familiar with the effects of each. The **Graph Editor viewport** works in much the same way as any curve editing panel, with the same selection and Transform operations as we have learned to employ in the 3D and other **Viewports**.

Advanced animation concepts

While advanced animation concepts are beyond the scope of this book, it is still helpful to know where to find the advanced tools and what they are capable of. From the **Animation Workspace** tab, selected at the top of the Blender application, we can see another facet of 3D animations, the **Dope Sheet viewport** at the bottom, as well as – by default – a view through the **Active Camera** in the scene in the upper left as shown in *Figure 8.14*:

The **Dope Sheet** looks much like the **Timeline**, however, each **Keyframe** is broken up here into its constituent parts. For any one **Keyframe**, there is a **Summary**, then the selected **Object**, and **Actions** which contain **Object Transforms**. If we expand the **Object Transforms** as shown in *Figure 8.14*, we can see all the **Keying Sets** that we activated before recording – **Location, Rotation, and Scale**, in this case – broken into the separate **X**, **Y**, and **Z axes**.

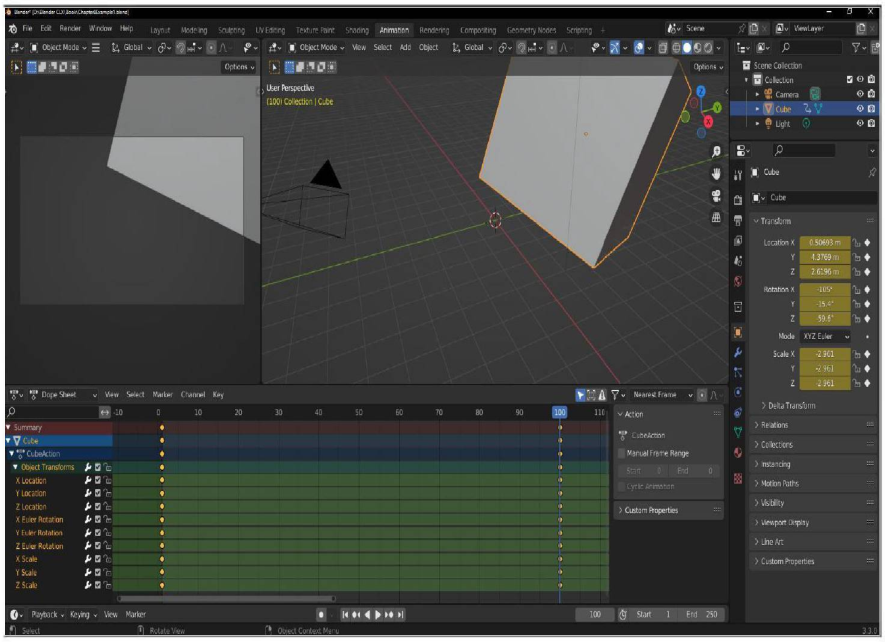
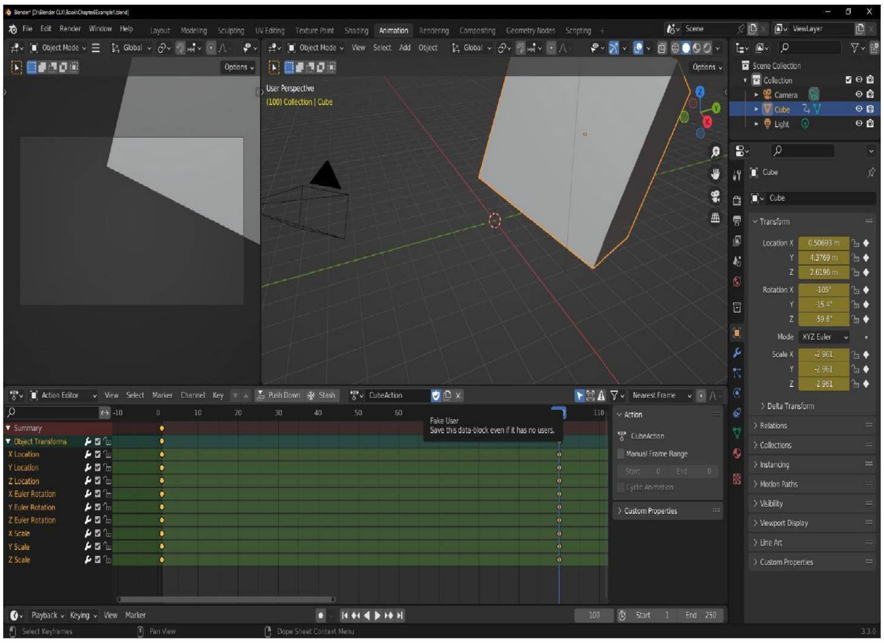


Figure 8.14: The Animation Workspace showing Dope Sheet with the selected Object's Transforms expanded

From the **Dope Sheet viewport**, we can also access the **Action Editor** by opening the drop-down menu just to the right of the **Editor Type** drop down and choosing **Action Editor** as shown in *Figure 8.15*. As we saw in the **Dope Sheet view**, **Actions** are higher in the hierarchy than individual **Transforms**, and serve as a sort of parent or portable container for the **Keyframes** of an animation.



Fake User
Save this data-block even if it has no users.

Figure 8.15: The Action Editor enabled from the Dope Sheet Editor Type

Actions can also be named and saved with a Fake User by clicking to highlight the icon appearing as a shield to the right of the Action's name in the top middle of the Action Editor viewport so that they will be saved for reuse in the **BLEND** file even if they are replaced by another Action on the 3D object they were recorded onto. In this way, we can record and save multiple separate animations, and then switch which animation we wish to use depending on our desired result.

Note: As mentioned earlier, not all animation types can be exported or played back in other programs besides Blender. When planning to animate 3D objects for playback in another software, it is important to investigate which animation types are supported by the target software before choosing an animation type.

In the upper **Dope Sheet viewport** menu to the left of the **Action name**, the buttons **Push Down** and **Stash** appear. When pressed, these buttons will appear to delete or hide the animation; however, they move the animation to another format called NLA strips, which are sometimes required for animations to play in other software besides Blender. Animations Pushed Down or Stashed to NLA strips can be found and edited in the **Nonlinear Animation Editor Type**, which can be switched to from the **Editor Type** drop down menu of any viewport as shown in *Figure 8.16*:

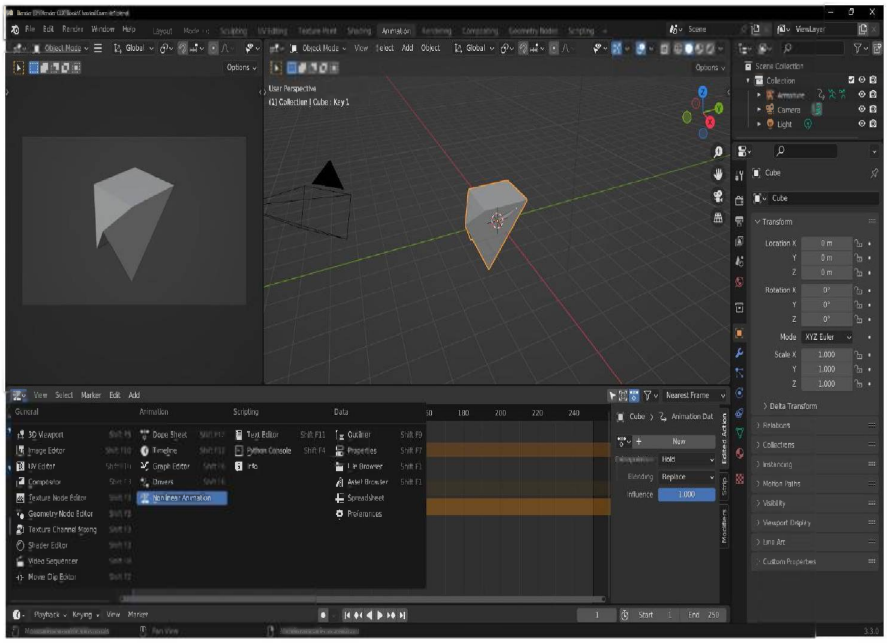


Figure 8.16: The Cube's animation Pushed Down to an NLA strip, shown in the Nonlinear Animation viewport

Finally, from the **Dope Sheet viewport**, we can also switch to the **Shape Key Editor**, as shown in *Figure 8.17*:

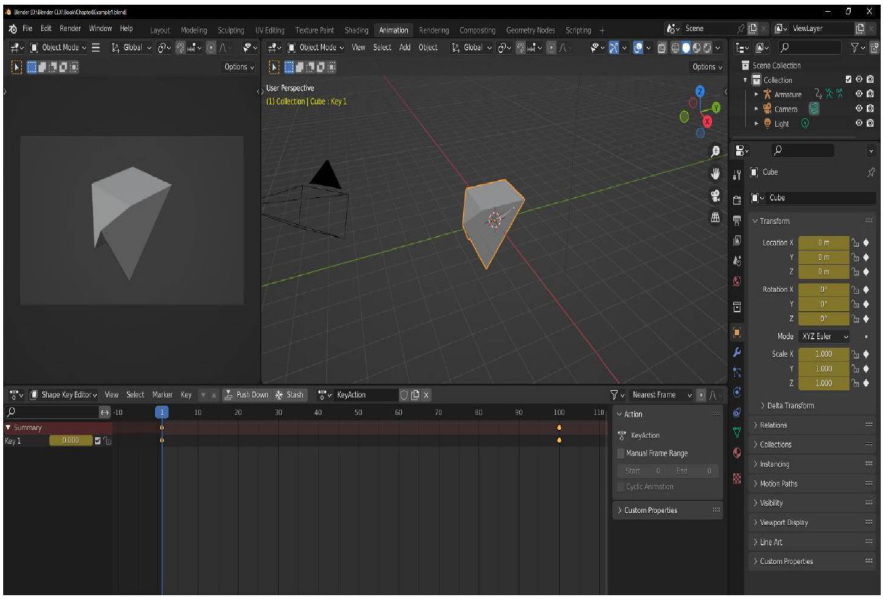


Figure 8.17: The Shape Key Editor mode of the Dope Sheet in the Animation Workspace tab

Conclusion

In this chapter, we have learned how to animate our 3D objects in multiple ways, as well as which animation types are appropriate for specific purposes. We also discovered that many more properties could be animated than just mesh Transforms, and hinted at the subject of the following chapter, *Chapter 9, Effects and Simulations*, where we will learn about creating special effects and moving more than just mesh.

Essential shortcuts and hotkeys

In this chapter, we used the following additional shortcuts and hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S

Show / Hide Side Bar	N
Show / Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab

Shortcut	Hotkey
Toggle Armature Edit Mode and Pose Mode	Ctrl + Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Table 8.1: *Essential Shortcuts and Hotkeys*

Points to remember

- Each 3D object in Blender has its own Timeline and can have multiple animation types applied.
- Keyframes record the information we choose via the Active Keying Set.

- Bone animation uses a proxy object called an Armature or Rig, to move vertex groups we assign to each bone.
- Animations can be edited in much the same way that 3D objects can, using the same selection methods and Transforms.

Questions

1. How can we edit the length of an animation?
2. How are animation curves and Curve objects similar?
3. Where can you find and edit each axis of a Keyframe independently?

CHAPTER 9
Effects and Simulations

Introduction

Effects and Simulations are used in 3D to create everything from fantasy, like magic and sci-fi, to realistic water, fire, smoke, cloth, and even realistic fur and hair.

Structure

This chapter gets readers started exploring special effects. Topics to be covered include:

- Types of effects and simulations
- Particles and hair
- [illegible]

- Exercise: working with particle hair
- The anatomy of physics: behaviors and forces
- Exercise: modeling a 3D tent with cloth physics
- Atmospheric effects
- Exercise: a simple volumetric cloud

Objectives

In this chapter, we will begin to understand the range of special effects and simulations and the fundamentals of how they are produced with Blender. The subject of effects and simulations is vast, but having a firm understanding of this starting point will help us determine which direction to go in should we want to create effects and simulations in our 3D work.

Types of effects and simulations

The overall category of special effects – also known as VFX – refers to a wide range of 3D content that is not quite as solid-appearing as static 3D meshes, although they often affect or use 3D meshes. The effects of gravity and wind, for example, may be applied to a 3D mesh to animate it in a realistic way. In Blender, effects are created through one of a few overlapping and interrelated approaches. Almost all effects and simulations work in conjunction with the Timeline that we explored in *Chapter 8, 3D Animation*.

One very common category of special effects in 3D is Particle effects. Particle effects are composed of emitters and render components operating within a specific set of frames in a Timeline. A fire effect, for example, might be comprised of an empty object like those we worked with in *Chapter 5, Poly Modeling Extras*, from which hundreds or thousands of small 3D objects are emitted over time in a loop, upon which is rendered a glowing, color-changing texture, which together appear to the human eye as fire. Smoke, fog, clouds, rain, electricity, heat distortion, fireworks, confetti, bubbles, and even fantasy effects like magic and sparkles can all be generated with Particle effects.

Note: Not every atmospheric effect that is used in production, like smoke and fire, is a Particle effect. By their very nature of employing many polygons and textures to achieve a level of realism – often with transparency which multiplies the effort required of the computer to render – Particle effects are not always the ideal way to produce such effects. Entire careers are made of inventing less computationally intensive ways of producing the same visual appearance of a Particle effect in a more efficient way, such as saving a flattened view of each frame of a Particle effect to a single image, called a sprite sheet animation.

Physics is another very common category of special effects in 3D, which are often used in conjunction with Particle effects. Adjusting the force of gravity applied to a smoke effect, for example, can make the difference between a cloud in the sky or fog on the ground. Physics effects usually require the preparation of each component to dictate how they will react to one another. For example, one group of 3D objects may be marked as Rigid Bodies, and others may be set up with Colliders so that

Rigid Bodies will not intersect with them but rather react in a desired way, such as bouncing, bending, or shattering.

A third common category of special effects might be described as atmospheric effects. As mentioned earlier, some atmospheric effects, such as fog, are made from Particles, but there are other ways to achieve similar effects that might be realistic and performant, depending on the desired result. One alternative to creating fog effects, for instance, is Volumetrics. Volumetrics rely on a defined space or "volume" within which the effect is rendered. Volumetrics are especially appropriate for creating the appearance of contained gasses and liquids, such as bodies of water, as well as shafts of light.

Whether to use Particles, Physics, Volumetrics, keyframed animations, sculpting, modeling, or other methods to create a specific visual effect is a complex question that is always dependent on the desired result. Every 3D application comes with limitations on which types of effects it is able to render, so the capabilities of the target application are the logical first consideration to investigate.

Blender can produce and render all these effects internally, so if the desired result is an image or video created with Blender, then all of them can be considered. On the other hand, third-party software such as a game engine or 3D viewer will dictate whether and how well these special effects can be transferred outside of Blender.

Particles and hair

At first, it may seem strange to group Particle effects with hair, but if we look back at the components of a Particle effect we just learned, we can see that the concept of a Particle emitter is the corollary to a human or creature's skin from which hair and fur grow. In this respect, hair Particles can also be thought of as a logical way to create effects where lots of small pieces make up a whole, such as with leaves, fronds, and spines which emerge from the ground like grasses or on the surface of plants like bushes, trees, and cactus. The same concepts can also be applied to fantasy effects such as feathers, spikes, and scales growing from monstrous or alien landscapes, foliage, or creatures.

Exercise 9.1

In this exercise, we will become familiar with Blender's **Particle effect systems** by creating and styling Particle hair on a simple Suzanne head primitive model.

Note: As of the release of Blender version 3.3, an additional workflow for creating and editing hair has been added to Blender, using Curves in conjunction with the new Geometry Nodes system. The new method includes some additional tools that may provide preferable results for creating and editing hair and fur for characters but may not be as useful for populating the surface of a model with other 3D objects.

1. To begin, let us open a new instance of Blender and delete the default objects.
2. Next, we will add a **Mesh** | **Monkey**.

   We will use the **Suzanne** monkey head as a stand-in for any humanoid head model.
3. Let us smooth out the appearance of our **Suzanne** head by selecting **Shade Smooth** from the **Object menu** at the top of the **3D Viewport**.
4. Let us also add a **Subdivision Surface Modifier** to **Suzanne** via the **Modifiers tab** in the **Properties panel**, then apply the modifier by choosing **Apply** from the drop-down menu to the right of the Modifier's name plate, as shown in *Figure 9.1*:

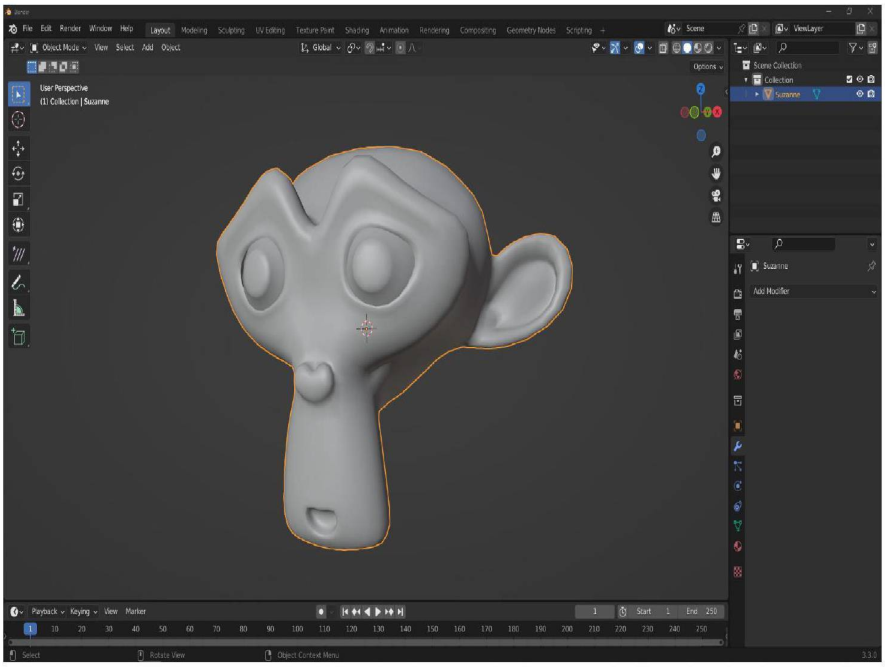


Figure 9.1: *Suzanne monkey head with Subdivision Surface Modifier and Smooth Shading applied*

5. Right below the **Modifiers tab** in the **Properties panel**, open the **Particle Properties tab**.

6. In the upper right of the **Particle Properties tab**, with the **Suzanne** model selected, press the plus sign icon, to add a new **Particle System** to **Suzanne**.
7. Next, from just below the top section where the new **Particle System** is shown under the **Particle Settings** data block, select **Hair** instead of **Emitter,** as shown in *Figure 9.2*.

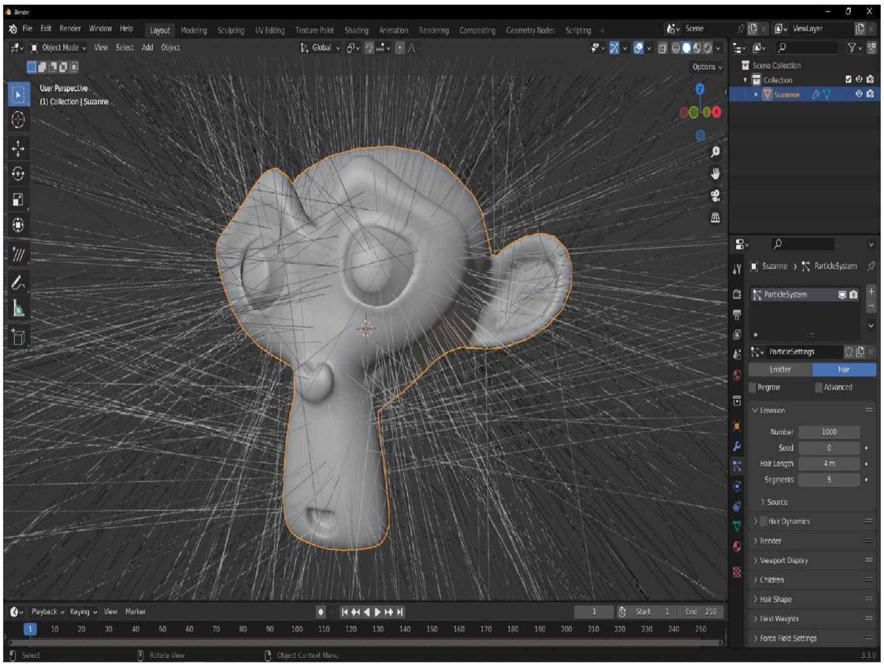


Figure 9.2: *Suzanne with Hair particle system assigned*

To constrain the hairs to a certain part of the model, we will create a new vertex group and assign the vertices we want to have hair to this group, in much the way we assigned vertices to a group to be controlled by bone animation in *Exercise 8.2, Chapter 8, 3D Animation.*

8. With **Suzanne** selected, enter **Edit Mode**.
9. Press the Number pad **3** key with the mouse cursor inside the **3D Viewport** to switch to the **Right Orthographic view**.
10. Press **Alt + Z** to switch to **X-Ray mode**.
11. Press Ctrl + Right mouse click and drag to lasso select the vertices of Suzanne's scalp where there should be hair.
12. Press the **C** key to enable the Circle select brush, then press and hold the **Alt** key to deselect any areas that were lasso-selected accidentally.

Now, we have a selection of the scalp where we want hairs to appear. In some cases, it may be preferable to duplicate the selected area and separate

the duplicate as a new mesh, creating a "wig cap," which can then be applied to the heads of other characters. In this case, it is fine to keep the hair with the mesh.

13. In the **Object Data Properties tab** of the **Properties panel**, under the **Vertex Groups** section, press the plus sign on the right-hand side to add a new, empty **Vertex Group**.

14. Below the **Vertex Groups** list, click **Assign** to assign the selected vertices to the new **Vertex Group** named **Group,** as shown in *Figure 9.3*:

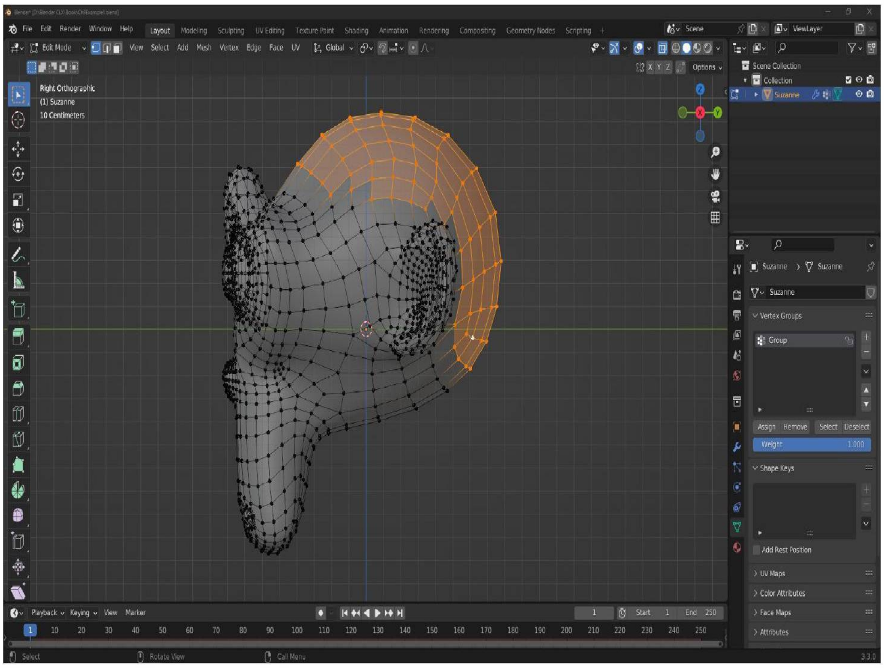


Figure 9.3: Selected vertices added to a new Vertex Group in Object Data Properties tab of the Properties panel

15. Exit **Edit Mode** and disable **X-Ray mode**.

16. Back in the **Particle Properties tab** of the **Properties panel**, scroll to the bottom and expand the **Vertex Groups panel**.

17. Click inside the **parameter field** next to the word **Density** and select the newly made **Group**.

    This will constrain the hair Particles to only emit from the area assigned to the **Group**. Now the hair appears sparse and thin and is sticking out in all directions. Once we start to style the hairs, the Particle settings will be locked in and no longer editable, so next we will edit the hair properties before we move on to practice styling it.

In the top **Emission section** of the **Particle Properties tab**, the default is set to 1000. 1000 is a lot of hairs but not enough to give a realistic look of human hair or fur. While we could increase the overall Emission amount, there are times when the desired result is for individual clumps or "locks" of hair to follow these few "parent" hairs instead.

18. Scroll back up to the middle of the **Particle Properties tab** and expand the **Children section**.

19. Enable the **Simple option**.

    By default, the hair will immediately appear thicker with **Simple** or **Interpolated** hair Children enabled, but it is important to understand that we are still only seeing 10 hairs per 100 that will show when the scene is rendered, as indicated by the **Display Amount**. In many cases, 10 child hairs per parent hair are plenty, so the **Render Amount** can be decreased to 10 as well. In either case, Particle systems can very easily become too intensive for a computer to render, so it is usually best to leave the **Display Amount low** while working and only increase it to the same number as the **Render Amount** temporarily as needed, to verify the results.

20. Also, from the **Children section** of the **Particle Parameters panel**, we can expand the **Clumping section**, and increase the **Clump amount**, which will look more like spikes, as shown in *Figure 9.4*:

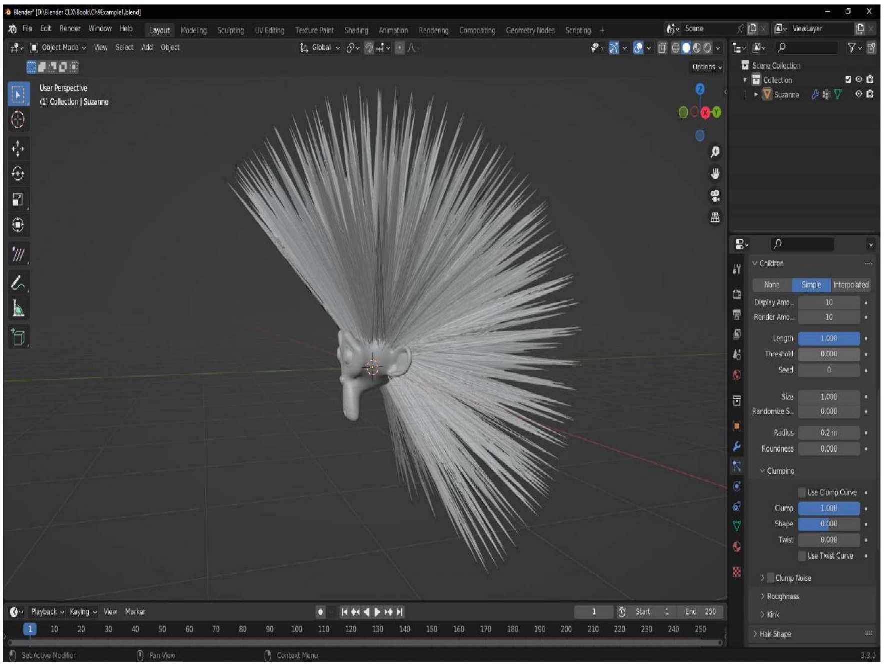

Children: None [✓ Simple] Interpolated
Display Amo... 10
Render Amo... 10
Length 1.000
Threshold 0.000
Seed 0
Size 1.000
Randomize S... 0.000
Radius 0.2 m
Roundness 0.000
Clumping
Use Clump Curve
Clump 1.000
Shape 0.000
Twist 0.000
Use Twist Curve

Figure 9.4: *10 Hair Particle Children per parent hair with Clump amount set to 1.*

Next, we will create a curl shape from a **Curve** object, to modify the shape of each **Clump** of hair.

21. From the far upper-right of the **Outliner panel**, press the **New Collection** icon appearing as a box with a plus sign in the upper right to create a new **Collection** in the **BLEND** file.
22. Select the new **Collection**, named **Collection 2** in the **Outliner**.
23. In the **3D Viewport**, from the **Add menu**, add a new **Curve | Bezier**.
24. Hide the **Suzanne object** from the view and frame the view around the **Bezier Curve object.**
25. With the **Bezier Curve object** selected, enter **Edit Mode**.
26. Delete all the **Bezier Curve control points** (vertices).
27. Press the Number pad **7** key to switch to the **Top Orthographic view**.
28. From the **Tool bar menu**, select the **Draw tool** appearing as a pencil drawing a curve with control points (not to be confused with the **Annotate tool**, one tool above the **Draw tool**.)
29. Hand-draw a spiral from the center outward as shown in *Figure 9.5*.
30. Starting from the center again, draw another spiral shape outward, overlapping with the previous as needed. Please refer to the following figure:

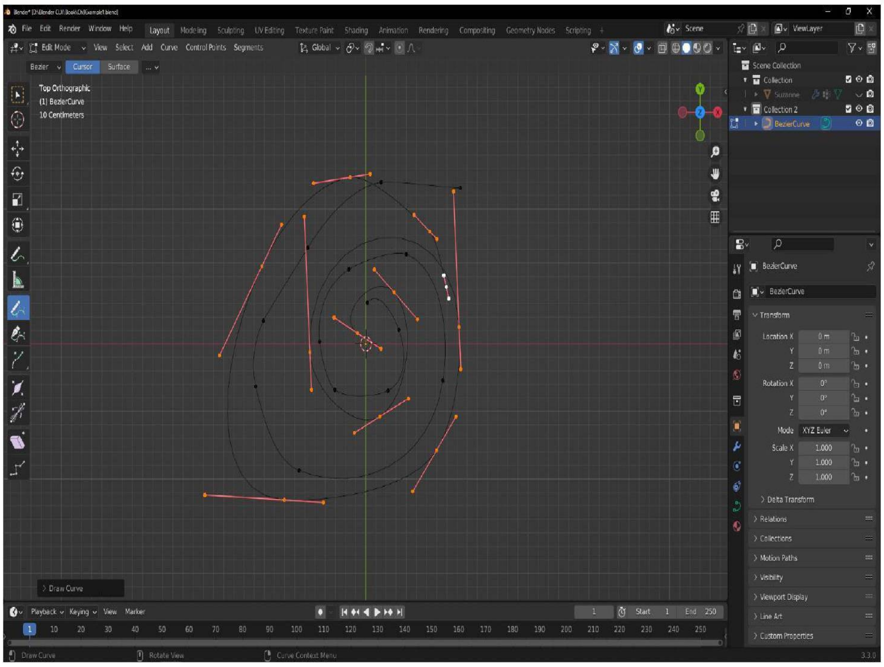


Figure 9.5: Spiral Bezier Curves, hand-drawn in Edit Mode

To keep this exercise simple, we will stop at two spirals, but for added variety, we could draw as many spirals as we want in this way, starting from the center, then applying the following steps to every spiral we create.

31. Reselect the **Select Box tool** from the top of the **Tool bar** menu.
32. Click away from any object in the **3D Viewport** or press the **A** key twice to clear the selection.
33. From the top-most menu bar in the **3D Viewport**, under **Select**, choose **(De) select Last**, to select the outer-most end points of the spirals.
34. From the middle of the top menu bar in the **3D Viewport**, enable **Proportional Editing**, appearing as concentric circles, then, via the dropdown menu to the right of the **Proportional Editing** icon, select **Connected Only** as shown in *Figure 9.6*:

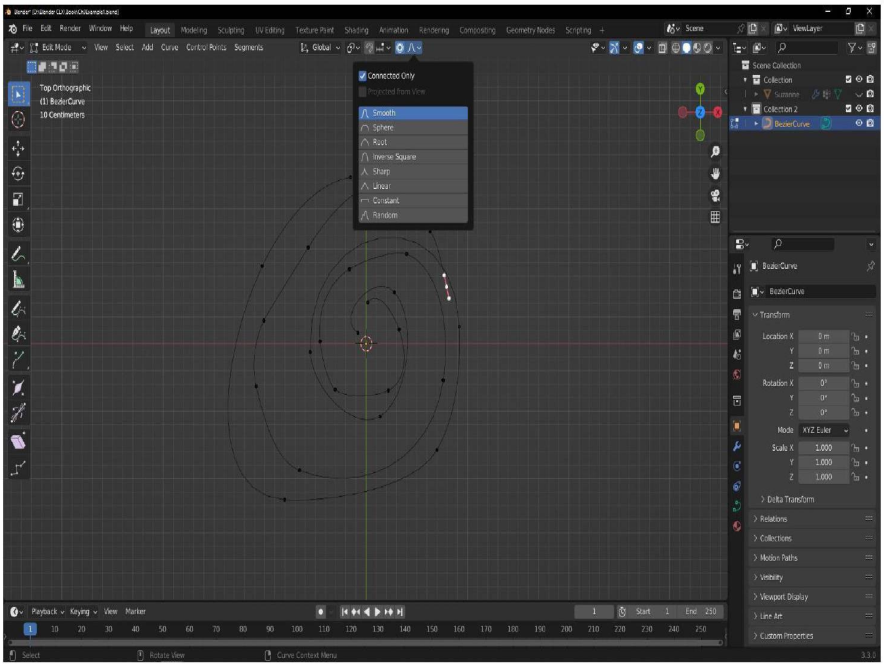
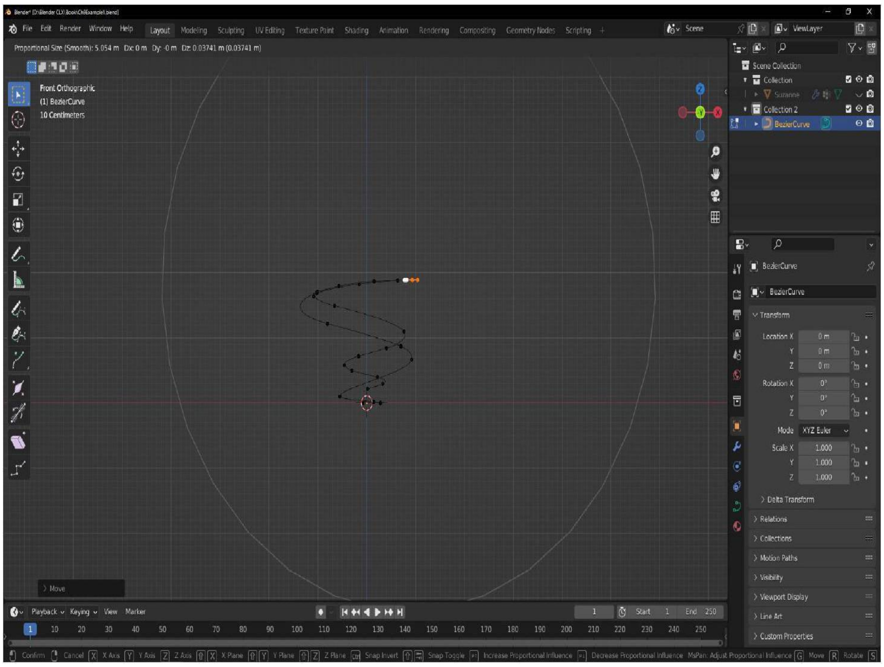


Figure 9.6: Proportional Editing with Connected Only enabled

35. Press the Number pad **1** key, to switch to **Front Orthographic view**.
36. Press **G** to activate the **Move tool**, then scroll the mouse wheel back to increase the **Proportional falloff radius** until all but the ends of the spiral shapes are

affected, then stretch the spirals upward in the z axis until they look like curls as shown in *Figure 9.7*:


Proportional Size (Smooth): 5.054 m Dx: 0 m Dy: -0 m Dz: 0.03741 m (0.03741 m)

Front Orthographic
(1) BezierCurve
10 Centimeters

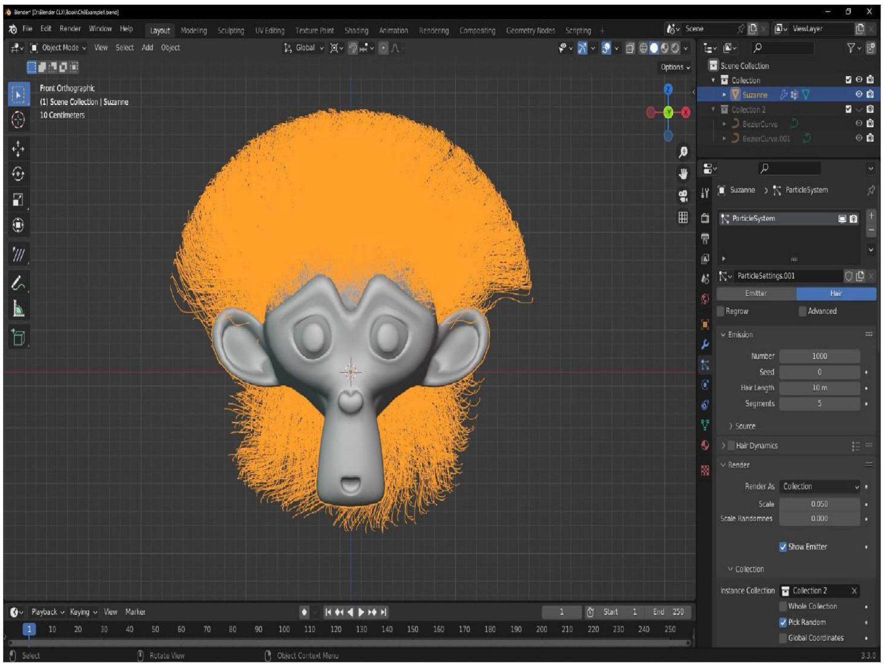
BezierCurve
Transform
Location X 0 m
Y 0 m
Z 0 m
Rotation X 0°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Figure 9.7: Spiral shaped Curve object stretched in the z axis to resemble a curl

37. Next, we will make separate **Curve** objects of each of the spirals by first Linked selecting one spiral at a time, then pressing the **P** key and choosing **Separate**.
38. When each spiral is a separate **Curve** object, exit **Edit Mode**.
39. Select all **Bezier Curves**, and from the **Front Orthographic view**, press **R** then 90 to rotate the Bezier Curves 90 degrees on the Y axis.
40. Next, press the **S** key then X, to constrain scale to the y and z axes, then scale the Curves down, to be thinner than they are long.

Now, fwe can use these Curve objects to control the shape of parent hairs in the **Particle system**.

41. Hide the **Collection 2** from view and make the Suzanne head visible again.
42. Select the **Suzanne model**, and in the middle of the **Particle Properties tab** of the **Properties panel**, expand the **Render** section.
43. Under the **Render** section, **Render As** drop-down menu, choose **Collection**.
44. Below the **Collection** drop-down, in the **Instance Collection field**, select **Collection 2**, and enable **Pick Random**, **Object Rotation**, and leave **Object Scale** enabled as shown in *Figure 9.8*:



[ ] Regrow [ ] Advanced

Emission

Number	1000
Seed	0
Hair Length	10 m
Segments	5

Source

Hair Dynamics

Render

Render As	Collection
Scale	0.050
Scale Randomnes	0.000

[✓] Show Emitter

Collection

Instance Collection: Collection 2

[ ] Whole Collection
[✓] Pick Random
[ ] Global Coordinates

Figure 9.8: *Suzanne model with curl shaped Bezier Curve hairs randomly selected from Collection 2*

From the **Emission**, **Render**, and **Children sections** of the **Particle Properties panel** we can further adjust the **number**, **length**, and **randomness** of the hair. We can also further edit the **Curves** at this point. Once we are happy with the number, clumping, and overall shape of each hair, we can move on to styling the hair with tools much like grooming human hair or the fur of a pet, but once we begin styling, the previous settings will be locked in. This is a good time to save our file.

45. From the drop-down menu in the upper left of the **3D Viewport**, switch from **Object Mode** to **Particle Edit mode**.

    Individual hairs will appear straight while in **Particle Edit Mode**, but will resume the curled shape in **Object Mode**, so it is helpful to style the whole head of hair accordingly. **Particle Edit mode** contains a new **Tool bar** menu full of tools specific to editing hair **Particles**, including **Comb**, **Smooth**, **Add**, **Length**, **Puff**, **Cut** and **Weight**. The behavior of each of these tools is described by their names, but more information about each is always available in the online Blender manual.

46. As with sculpting in 3D, it is easier and more instructive to test each of the tools rather than being directed to use them in a certain way, so the last step of this exercise is to experiment and have fun.

    Entire careers are made from learning to style 3D hair, just as in real life. As with 3D modeling, following reference images is also incredibly helpful

As with 3D modeling, following reference images is also incredibly helpful.

Once the overall style is as desired, the final step is to exit the **Edit Mode** and move on to rendering, which we will begin to explore in *Chapter 10, Images and Video.*

Physics behaviors and forces

Like particles, special effects and simulations using Physics is a vast subject. The component that all Physics effects have in common is the simulation of real-world physics such as mass, gravity, and friction. Real-world physics – like the behavior of light we discussed in *Chapter 3, General 3D Concepts* – is so complex and powerful that consumer-grade computers are not capable of calculating them to exact realism. For this reason, 3D effects use multiple tricks to mimic real-world physics in a convincing way, but not perfectly. The result is that sometimes, 3D Physics does not behave as expected.

Players of video games sometimes experience this fact first-hand when moving two 3D objects together, and one or both objects fly away unexpectedly with explosive force. This is a simple problem that is explained by the fact that games are rendered – like film – as frames, and that 3D objects do not have real thickness. Therefore, in 3D the physically impossible is possible. When one 3D object is moving toward another, unrealistic intersections occasionally occur in between rendered frames, which is detected as a collision on the next frame when BAM! All the calculations of where each object should be getting sorted out in an instant. That is of course, an oversimplification, but then again, so is 3D Physics.

Modeling with physics

Exercise 9.3

There are many potential ways to use Physics in Blender to create natural objects and arrangements of objects. Giving a cluster of objects Gravity and a surface resistant to intersections called Collisions, and then running a Physics simulation by pressing Play in the **Timeline** can produce a much more natural pile of objects, for example. Combining Wind with Smoke or Container and Fluid simulations can create visuals that are not easy to model. In this exercise, we will use the Physics behaviors of Cloth and force of Gravity to help model a realistically draped 3D cloth tent.

1. The first step is to open a new instance of Blender and delete any default objects.

   Next, we will model a simple rope and poles upon which to drape a cloth.

2. Add a **Mesh | Cylinder** and set the number of Vertices to 12 in the contextual menu that appears in the lower left of the 3D Viewport.

3. Set the **Radius** to 0.0125m (approximately half an inch) thick and leave the depth at 2m.

4. Change the **Cap Fill Type** to **Nothing**.

5. Set the **Location** in the z axis to 1m.

6. Set the **Rotation** in the x axis to 90 degrees.

This is the basis for our tent's central rope.

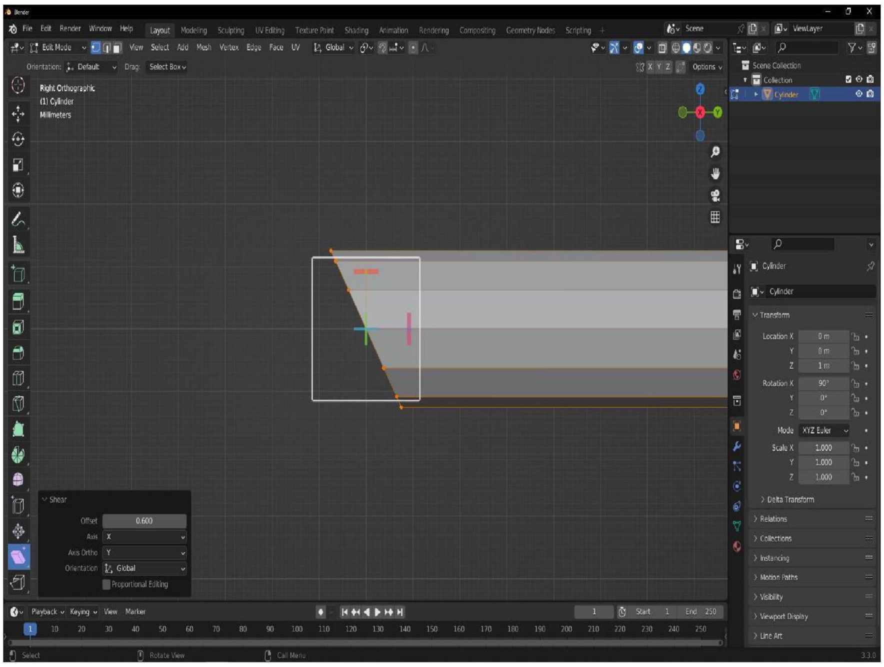
7. Enter **Edit Mode**.

8. Select the ring of vertices at one end, frame the selection in the view by pressing the Number pad period key, and switch to **Right Orthographic view** by pressing the Number pad **3** key.

9. From the **Tool bar** menu, near the very bottom, select the **Shear tool** appearing as a slanted cube.

10. With the mouse cursor, click the top handle on the **Shear gizmo**, just inside the highlighted box, and drag it to the left.

11. In the contextual menu, set the **Offset amount** to around 0.6 as shown in *Figure 9.9*:



Right Orthographic
(1) Cylinder
Millimeters

Shear
Offset 0.600
Axis X
Axis Ortho Y
Orientation Global
Proportional Editing

Transform
Location X 0 m
Y 0 m
Z 1 m
Rotation X 90°
Y 0°
Z 0°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Figure 9.9: Shear applied to the open end of a Cylinder primitive

The **Shear** operation will allow us to extrude faces from this edge at an angle without the new cylindrical shape appearing squashed in its local z axis.

12. Zoom out a bit, and extrude the selected edge downward, close to the green grid line marking the y axis or "ground" level.

13. Use the right-hand handle of the **Shear tool** to turn the open edge parallel to the ground.

14. Press the **N** key to open the **Side bar menu**, and in the Item tab, switch the **Transform Median** from **Local** to **Global**, then type **0** into the **Z** field, to snap the rope to the ground as shown in *Figure 9.10*:

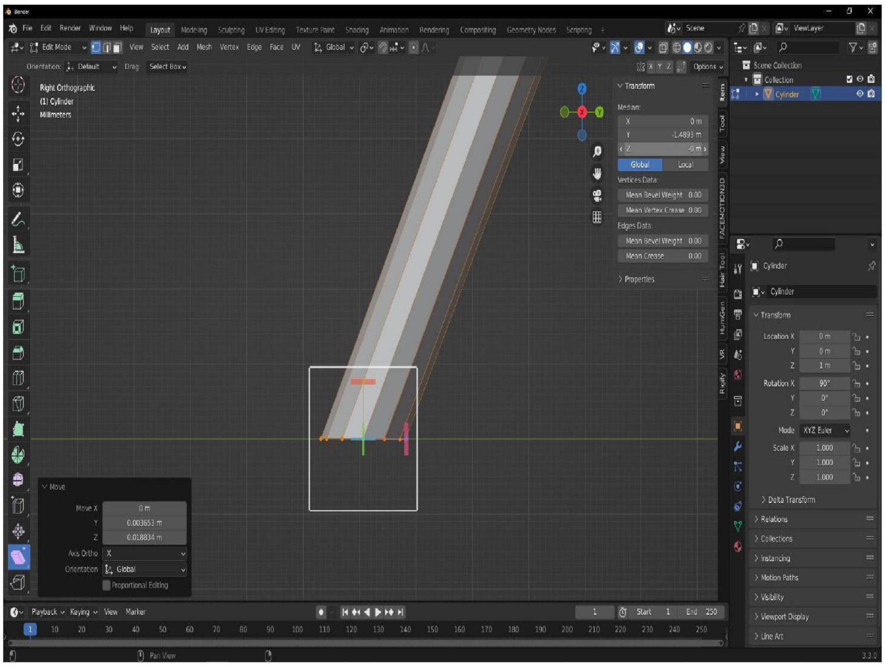


Figure 9.10: Open end of rope sheared parallel to the ground and snapped to world 0 in the z axis

If we zoom out now, we can see the beginnings of a central rope from which we can suspend a cloth tent.

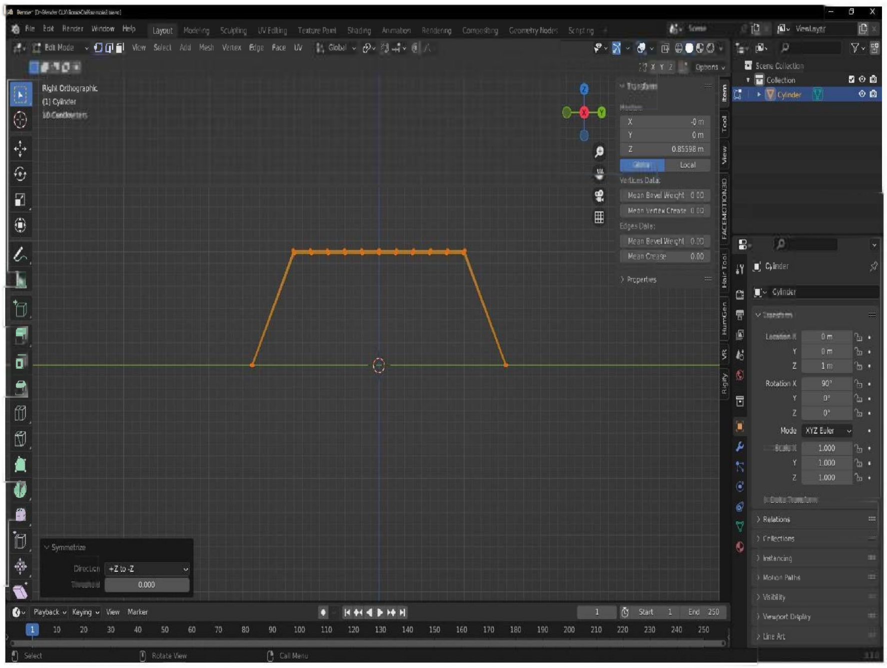
15. Press **Ctrl + R** keys, hover the mouse cursor over the top vertical portion of the rope, left-click to start adding an edge loop, right-click to confirm, and in the contextual menu for **Loop Cut** and **Slide**, enter **9** for the **Number of Cuts parameter**.

    This will give our rope 6 points of curvature on each half when we use a Physics simulation to bend it. Next, we will mirror the angle we created across the z axis, so that the front and back of the 3D rope are symmetrical.

16. Once again, reselect the **Select Box tool** from the top of the **Tool bar** menu.

17. Still in **Right Orthographic perspective**, select all vertices, and from the **Mesh** menu, select **Symmetrize**.

18. From the **Symmetrize contextual menu**, switch the Direction to **+ Z** to **– Z**, as shown in *Figure 9.11*:



Symmetrize
Direction +Z to -Z
Threshold 0.000

Figure 9.11: *Rope object Symmetrized across the z axis*

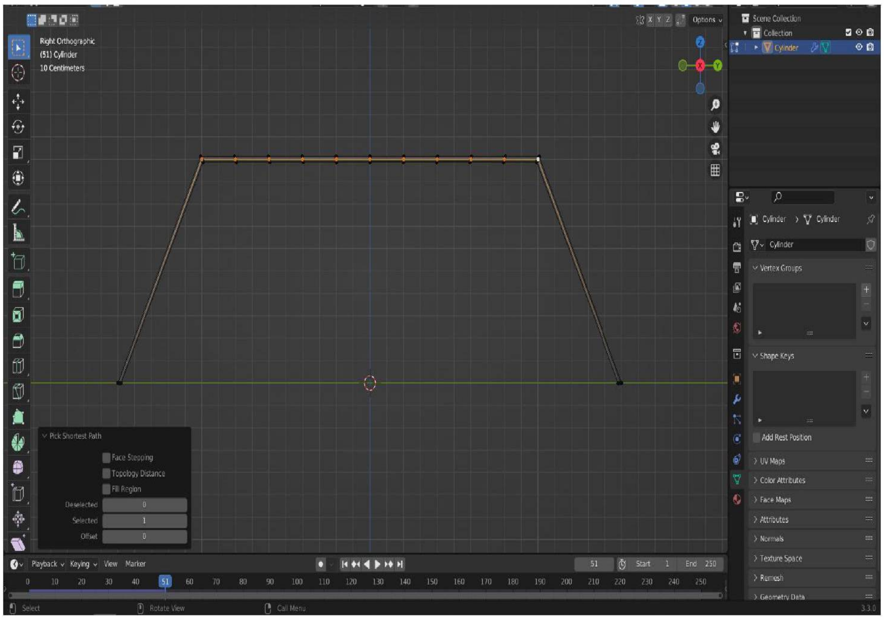
19. This is a good time to save the file.

    To save effort and avoid mistakes, we will create the fabric from the rope by selecting the top-most edge of vertices.

20. From the side view, we can left-click to select the middle vertex at one end of the horizontal section of the rope.

21. Then, we will hover the mouse cursor over the middle vertex at the other end of the horizontal section of rope, press and hold **Ctrl**, then left-click to

select all the vertices along the shortest path between the first selection and the last as shown in *Figure 9.12*:



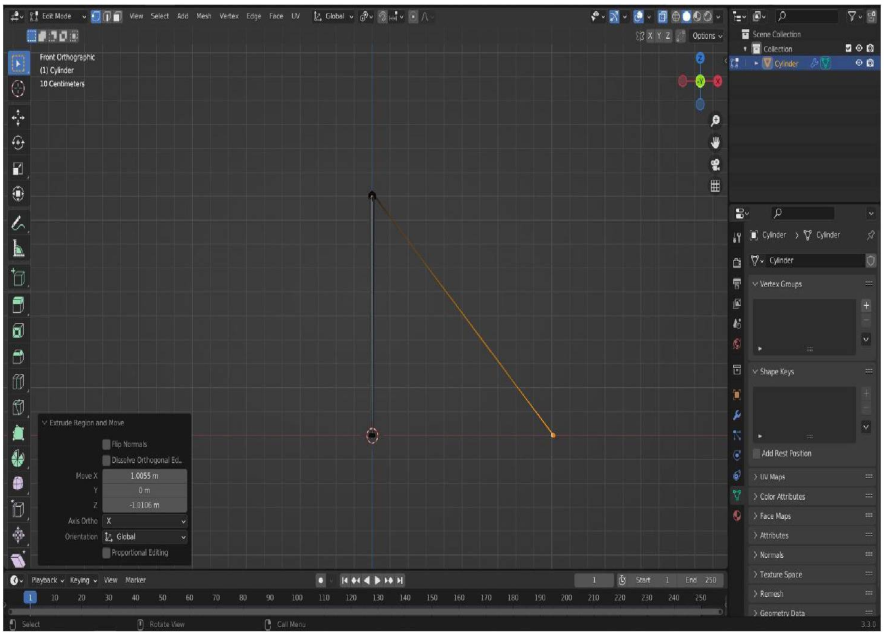
Pick Shortest Path

- Face Stepping
- Topology Distance
- Fill Region

Deselected 0
Selected 1
Offset 0

Figure 9.12: Clicking near a vertex, then using Ctrl + click to select vertices in between the first and last

22. Press **1** on the Number pad to switch to **Front Orthographic view**, then press **E** to extrude the selected edge, and move it down in the viewport to approximately where the bottom edge of a tent would meet the ground as shown in *Figure 9.13*:



Extrude Region and Move

- Flip Normals
- Dissolve Orthogonal Ed...

Move X 1.0055 m
Y 0 m
Z -1.0106 m
Axis Ortho X
Orientation Global
- Proportional Editing

Figure 9.13: Tent fabric edge extruded from duplicate top edge of rope

If we orbit our view to the side, we will see that parts of the new fabric mesh are facing the wrong way, because of the earlier **Symmetrize operation**.

23. It is less visible on the rope part of the object, but the same issue exists here too, so while we are in **Edit Mode**, let us select all, **Alt + N** and select **Recalculate Outside**.
24. Next, press **Ctrl + R** and select a vertical edge along the fabric piece to insert a horizontal edge loop, then roll the scroll wheel forward four clicks to increase the **Number of Cuts** to **5**, making the fabric subdivided in even quads.
25. Select all, and under the **Mesh** menu choose **Symmetrize** again, and in the contextual menu switch the Direction to **+X** to **-X** as shown in *Figure 9.14*:

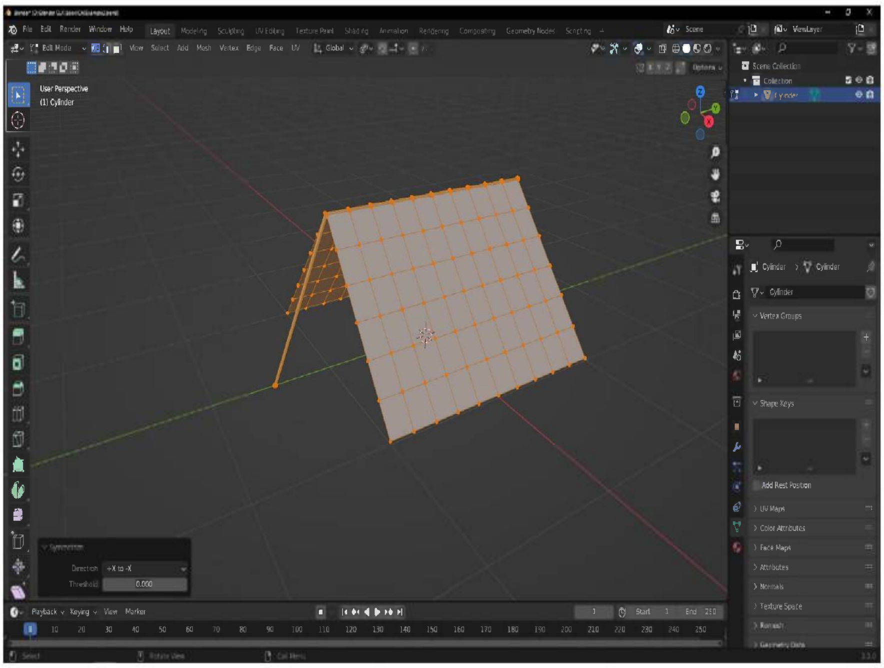

Symmetrize
Direction: +X to -X
Threshold: 0.000

Figure 9.14: *Even quad subdivisions inserted via Ctrl + R and Symmetrized*

26. Repeat the operation to recalculate normals: select all, **Alt + N** and select Recalculate Outside.
27. Next, we will add the vertices of the mesh we want to stay stationary to a Vertex Group by first switching to **X-Ray mode** via **Alt + Z**.
28. Then, clear the selection, and box or circle select just the bottom vertices of the rope.
29. Next, press and hold the **Ctrl** key, then press the Number pad plus key to increase the selection.

30. Lastly, press and hold **Shift**, then select the four corner vertices of the fabric on the ground to add them to the selection.

31. Add a new empty **Vertex Group** via the top section of the **Object Data Properties tab** of the **Properties panel**, and then press the **Assign button**, to assign the selected vertices to the **Group** as shown in *Figure 9.15:*

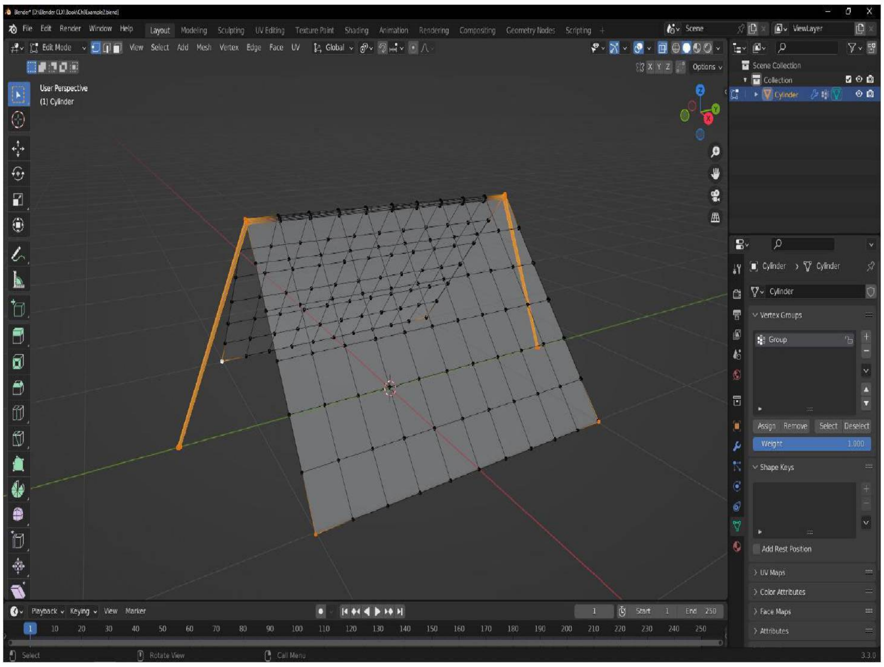


Figure 9.15: Rope ends, and four corners of the fabric section assigned to the Vertex Group

Tip: To verify that only the corner and rope vertices are assigned, deselect all, then press the Select button underneath the Vertex Groups section. Assigned vertices will be re-selected.

Now that our rope and fabric are complete, we can move on to setting up **Physics** to deform them into a natural shape. We can go on to model stakes and poles to make the effect more believable, but since physics in 3D are merely a simulation, we do not need them in place to create the effect.

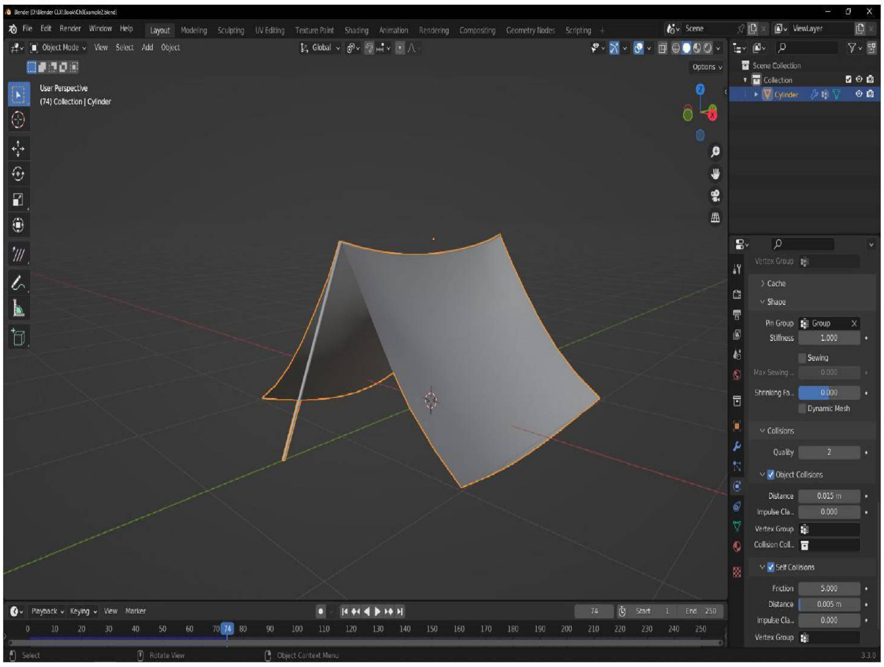
32. Exit **Edit Mode** to enable **Physics**.

33. Open the **Physics Properties tab** of the **Properties panel** appearing as a circle being orbited by a smaller circle and press the **Cloth button** to give the whole tent cloth-like behaviors.

34. Under the **Physical Properties section**, increase the Vertex Mass to around 2kg.

35. Scroll down in the **Cloth panel** of the **Physics Properties tab** and expand the **Shape section** near the bottom.

36. Inside the **Pin Group**, select the **Group** we set up earlier.
37. Below the **Shape section**, expand the **Collisions section**, and enable **Self Collisions**.
38. Under **Self Collisions**, reduce the **Distance** to around 0.005 m.
39. Disable **X-Ray mode**, and with the mouse inside the **3D Viewport**, right-click and choose **Shade Smooth**, then from the contextual menu enable **Auto Smooth** and increase the angle to 180 degrees.
40. Press the **Play** button in the **Timeline**, and after the fabric and rope sag downward as shown in *Figure 9.16*, press the **Pause** button.



Vertex Group
Cache
Shape
Pin Group: Group
Stiffness: 1.000
Sewing [ ]
Max Sewing ...: 0.000
Shrinking Fa...: 0.000
Dynamic Mesh [ ]
Collisions
Quality: 2
Object Collisions [✓]
Distance: 0.015 m
Impulse Cla...: 0.000
Vertex Group
Collision Coll...
Self Collisions [✓]
Friction: 5.000
Distance: 0.005 m
Impulse Cla...: 0.000
Vertex Group

***Figure 9.16:** Tent sagging from pinned Group using Cloth physics simulation*

41. To make this change permanent, we will navigate to the **Modifier Properties tab**, where we will find a **Cloth Modifier** has been added to the **Cylinder object**, then we will **Apply the Modifier.**
42. Set the **Timeline** back to frame 1.

Even without stakes and poles, our tent has a realistic drape. Adding a few more poly modeled meshes makes the effect more believable, as shown in *Figure 9.17*:

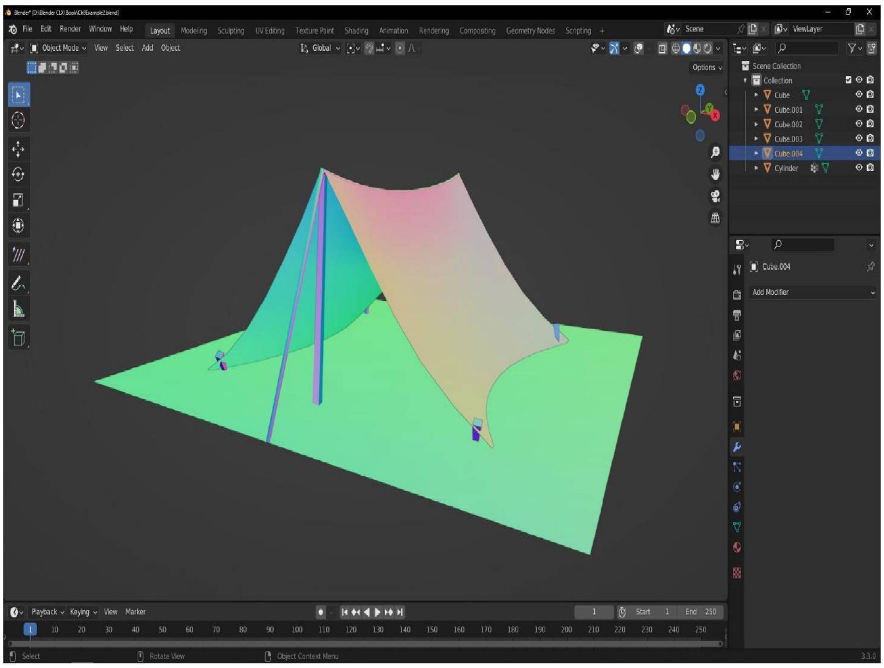


***Figure 9.17:** Physics modeled tent with stakes and pole meshes added*

Atmospheric effects

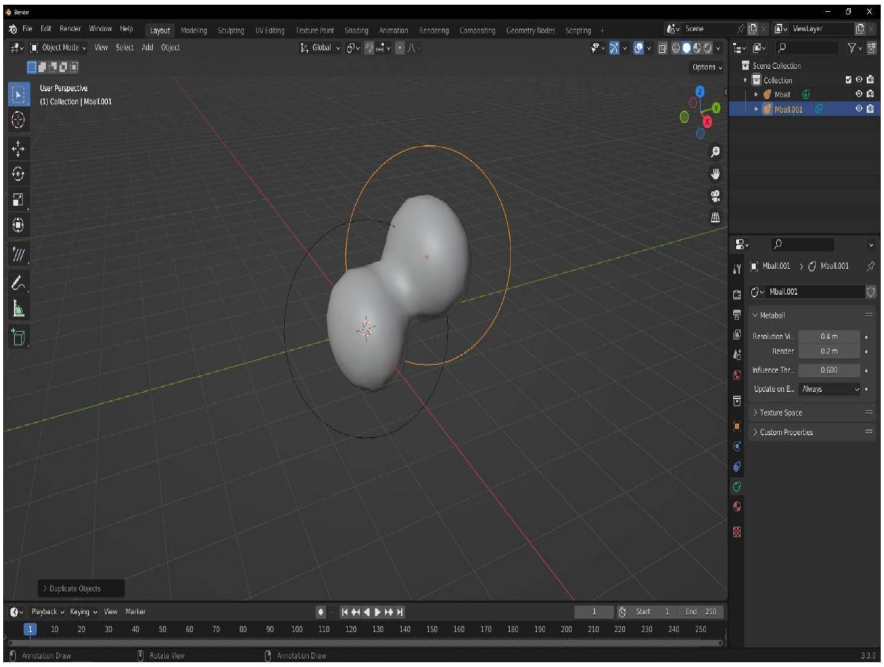
What we are referring to as atmospheric effects in this chapter encompasses multiple approaches to affecting the overall environmental atmosphere of a scene. In this regard, atmospheric effects refer to lights, shadows, reflections, skies, beams of light, fog, dust, clouds, and other volumetrics. While there are equally numerous methods for creating such effects, it is helpful to be aware that Blender can produce all of these, and to know the above terms to use in searches for relevant tutorials and tips online. As the final exercise in this chapter, we will learn how to quickly generate a volumetric cloud out of any 3D object in Blender.

Exercise 9.4

1. The first step is to open a new instance of Blender.

   The next step is to model any 3D object at all, which we might wish to replace with a volumetric cloud of the same shape. In this case, we will create a rough cloud shape from a blobby object that is perfect for cloud forms due to the way it naturally merges with neighboring shapes: the **Metaball**. Metaballs are a special kind of 3D mesh not commonly found outside of Blender which – as with **Curves and Particles** – can be converted to a regular 3D mesh for use in other software as a final step.

2. To begin, delete the default scene assets.
3. Via the **Add menu**, add a **Metaball** | **Ellipsoid**.
4. Working in **Object Mode**, duplicate the selected **Mball** object by pressing **Shift** + D, then move the duplicate and notice how the two are connected like lumps of clay until pulled too far apart as shown in *Figure 9.18*:



Mball.001 › Mball.001

Metaball
Resolution Vi... 0.4 m
Render 0.2 m
Influence Thr... 0.600
Update on E... Always

Figure 9.18: Two Metaballs connected at a short distance

5. Place the duplicate **Metaball** just out of reach of the original, and then scale it up a bit, just until it merges again with the first.
6. Left-click on the ring around any individual **Metaball** to select it separately from the group.

7. We will continue this way, duplicating, moving, and scaling each **Metaball** in one or multiple dimensions until we have created a lumpy blob with a cloud-like appearance as shown in *Figure 9.19*:

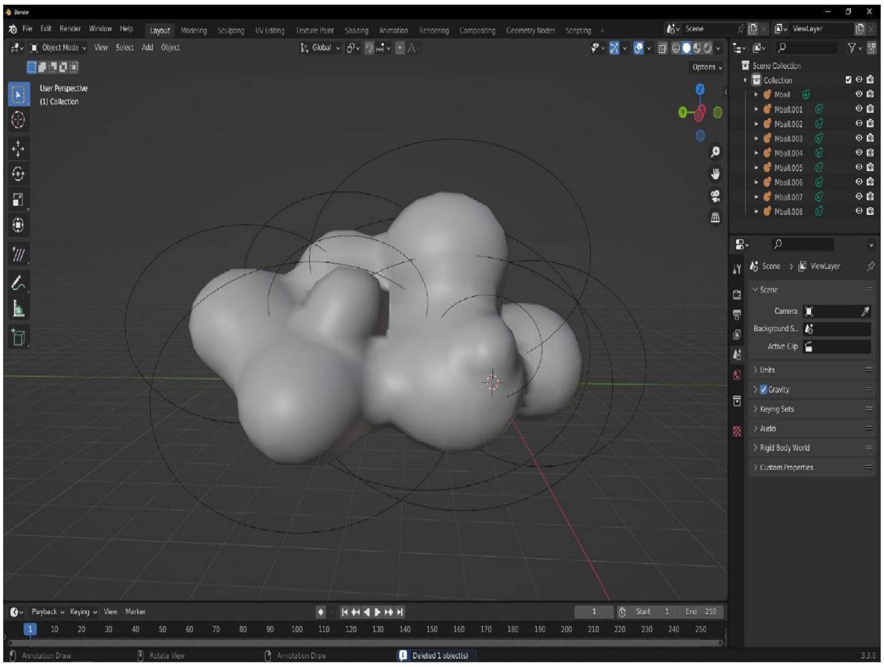


Figure 9.19: Metaballs arranged to resemble a cloud shape

8. The next steps are remarkably simple: first select all the **Metaballs** and via the **Object menu** at the top of the **3D Viewport**, select **Convert** | **Mesh**.

9. Next, add a **Volume** | **Empty object** into the scene.

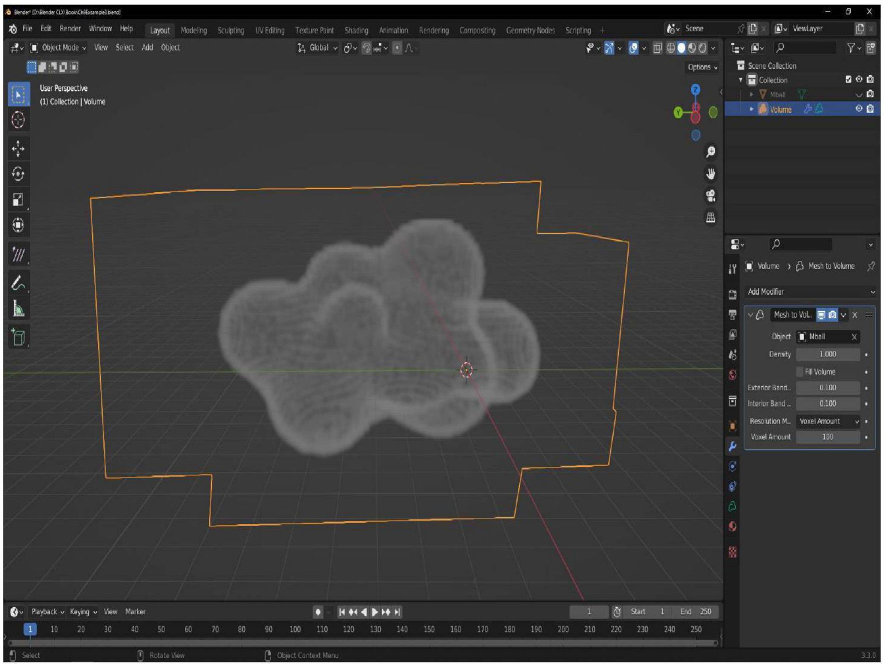
10. Then, with the **Volume** selected, add a **Modifier Mesh** to **Volume**, and from the **Object field**, choose the merged **Mball** object.

The volumetric effect will instantly appear around the **Mball** object.

11. Hide the **Mball** object from view.

12. To reduce the appearance of jagged edges, in the **Voxel Amount**, increase the amount from 32 to something like 100, being cautious not to increase the amount so much as to overburden the computer.

13. Disable **Fill Volume**, to limit the effect to the boundaries of the mesh as shown in *Figure 9.20*:



Object: Mball
Density: 1.000
Fill Volume
Exterior Band...: 0.100
Interior Band ...: 0.100
Resolution M...: Voxel Amount
Voxel Amount: 100

***Figure 9.20:** Empty Volume with Mesh to Volume Modifier applied*

The resulting effect is interesting, especially when viewed from different angles, but it will not win us much work as-is. The next step is to learn how to set up a 3D scene with a virtual camera and objects like this cloud to create visually appealing renders, which is what we will do in *Chapter 10, Images and Video.*

Conclusion

Effects and Simulations are powerful ways to add life and magic to our 3D scenes. We have barely scratched the surface of the power of Effects and Simulations, but now we understand where to begin in Blender when we have more elaborate visions for our 3D creations than simple static meshes.

Essential shortcuts and hotkeys

In this chapter, we used the following Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show/Hide Side Bar	N
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab

Toggle Armature Edit Mode and Pose Mode	Ctrl + Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9

Shortcut	Hotkey
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Table 9.1: *Essential Shortcuts and Hotkeys*

Points to Remember

- The terms Special Effects (SFX), and Visual Effects (VFX) describe a broad range of 3D creations beyond simple 3D objects but often incorporate 3D objects in their creation.
- Like some special objects and some kinds of animations, not every effect that can be created in Blender can be exported and used in another application, but many can, or can at least be converted to another format for export. Look to the capabilities of the target software for information about whether and how well effects from Blender can be transported into them.

- Many effects like volumetrics look best when combined with other effects like lighting. Think of compositions as a whole.
- As with sculpting and many poly modeling techniques, increasing effect parameters too high or too fast can cause the software or computer to crash. Increasing parameters incrementally and slowly is safest.

Questions

1. Why are hairs part of Blender's Particle system?
2. What is a typical use for Physics in modeling?
3. What are some categories of Special Effects?

CHAPTER 10
Images and Video

Introduction

Up until this chapter, we have created multiple 3D objects and effects, but we have not yet begun preparing these 3D creations for others to view or interact with. The focus of these last two chapters will be real-world use cases for 3D creations. In this chapter, we will focus on how to render 2D images and videos of 3D creations for everything from portfolios and graphics to marketing and even game art.

Structure

This chapter covers showcasing 3D objects, animations, and effects through images and video. Topics to be covered include:

- Staging renders
- Render engines and how to choose one
- Rendering styles
- 3D cameras and lights
- Exercise: setting up a 3D camera
- 3D Light and shadow
- Exercise: rendering the Pedestal object
- Image and video editing overview

Objectives

After reading this chapter, we will understand how to prepare and render our 3D creations as sharable images and videos for a wide variety of real-world uses, as well as where to begin editing our images and video.

Staging renders

It is helpful to think about setting up 3D creations to be rendered as 2D images or videos as dressing the stage for a theatrical production. In the same way that viewers find exaggerated animations more believable, as we discovered in *Chapter 8, 3D Animation*, even when realism is the goal, using a bit of fakery can be a surprising key to believability. Using the tricks of theater like cutouts and false fronts, flat backdrops, smoke, mirrors, and colored lights to give the illusion of effects like depth and atmosphere also maximizes the efficiency and versatility of our workflows.

Render engines

In 3D, the term "rendering" refers to creating a 2D image or video of a 3D scene. 3D software creates these 2D images and videos through complex code called a Render Engine. Some Render Engines are universal to multiple applications, and some are standalone, while other Render Engines are built-in or exclusive to a specific application.

In Blender, a few Render Engines are built-in, while others are under development or available as plug-ins through third parties. The two main Render Engine choices that come built-in with Blender are Eevee and Cycles, with Eevee as the default. Blender's third built-in render engine – Workbench – is more practical than aesthetic and, for this reason, is not often used other than for setting up computationally complex scenes before switching to another Render Engine for final renders.

complex scenes before switching to another Render Engine for final renders.

The difference between the Blender Render Engines and which choice is most appropriate depends on our scenario, compared with the purpose and capabilities of each. In Blender, the active Render Engine is set via the Render Properties tab at the very top of the Properties panel, as we discovered in *Chapter 6, 3D Surfaces,* when we baked textures for our Pedestal object. Texture baking - only possible using the Cycles Render Engine – is an example of the kind of specialized function that may dictate the appropriate Render Engine choice.

The capabilities of the active Render Engine are generally unseen in the **3D Viewport** unless the **Rendered Viewport Shading mode** is enabled via the spherical icons in the upper right of the 3D Viewport. When the **3D Viewport** is set to the **Rendered Shading mode**, the viewport will simulate the properties of the currently selected Render Engine for preview purposes. Displaying the rendered view like this can

significantly slow down computer performance or obscure certain visual elements in complex scenes. A common workflow to overcome this is to toggle the **Rendered mode** on and off as needed or to split the **3D Viewport** into one view with **Rendered Shading mode** enabled and one with **Solid Shading mode,** as shown in *Figure 10.1*:

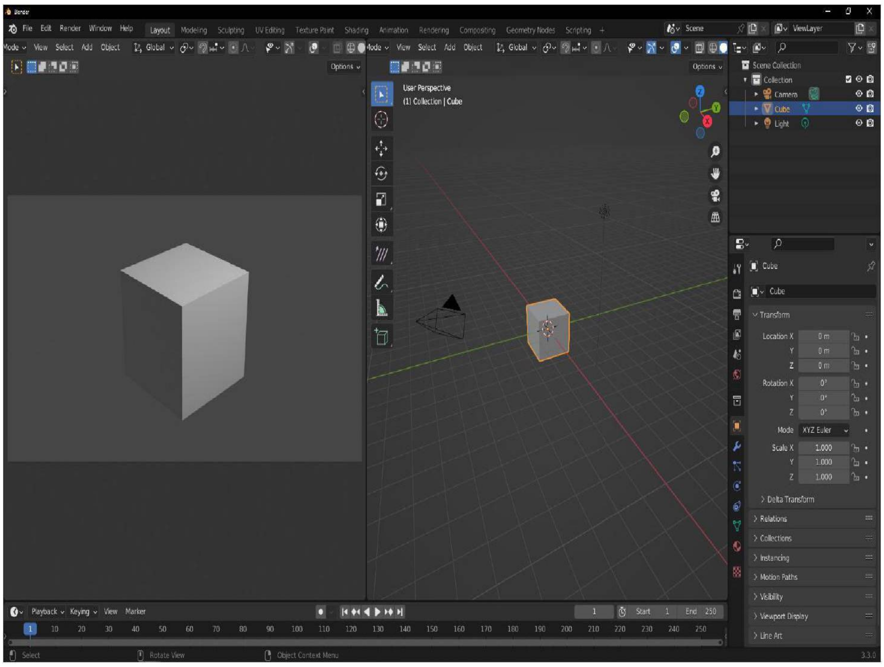


***Figure 10.1:** A split 3D Viewport with Rendered Shading viewed through a camera on the left, Solid Shading on right*

Note: The 3D Viewport will also show overlays that do not appear in renders unless Show Overlays is disabled, meaning that the icon appearing as two overlapping circles, one solid and one transparent, is not highlighted.

Rendering styles

One of the main considerations when choosing the appropriate Render Engine is the desired rendering style. Some Render Engines, such as Blender's Cycles, are purported to do a better job at "realism" at the cost of longer render times and a more complicated setup. A scene that looks great in Eevee, for example, rarely looks just as good after switching to Cycles without additional lights and adjustment of settings.

Learners often ask experts what the "best" choices are among options that affect a render's overall style, like Render Engines, Lighting setups, Texture types, Materials, and Shader effects. It may be tempting to think of the available rendering styles as a binary choice, such as realism or fantasy, shaded or flat, but in truth, there are unlimited possibilities for a style that often intersect.

As Veer Sharma explained in his Game Developers Conference 2021 Visual Effects Summit talk, "Shader Sauce: How to Use Shaders to Create Stylized VFX," there are many aspects of style that can be incorporated into 3D work. Style is not just a particular type of shading, a polygon budget, or a color palette. Style is also communicated through proportion, silhouette, physical properties like translucence and reflectivity, movement, and even sound. Each of these attributes of style can contribute to the desired result. For example, a realistically textured rubber ball will take on a very unrealistic look if it is shown landing on the ground with a brick-like thud.

It is helpful to know that Cycles are often recommended for producing more "realistic" effects such as reflections, refraction, and caustics in transparent objects like glass and liquid or light scattering in translucent objects like fog, wax, and skin. On the other hand, the increased computations involved in more "realistic" light calculations can diminish realism in a render overall, for instance, by introducing graininess into the resulting image or by requiring such long calculations that other aspects of the final result become impractical to address. Often, touches of realism must be balanced through some other process like denoising, which does not always result in improved image quality.

Developers of new software and hardware are constantly pushing the boundaries of so-called "physical accuracy" achieved through ever more complex computations and display specs. This subject is a source of frequent misunderstandings, causing anxiety especially among new 3D creators. Out of a desire to create in the most accepted way and achieve mainstream looks, it is tempting to assume that more (calculations, pixels, memory, frames, and so on.) is always better. The pursuit of increased realism produces the dreaded "uncanny valley" effect: where something appears so close to real that the inevitable imperfections stand out more than if the look were clearly stylized.

Fortunately, unreal appearances have many aesthetic and practical uses. Sometimes, diverging from reality is necessary to create a recognizable, repeatable, personal style. Entire communities of Blender developers are devoted to the advancement of NPR, or "non photo realistic" rendering techniques – unencumbered by the constraints of realism – further validating stylized art as a legitimate design choice.

Conversely, while it is commonly assumed that realism is more difficult to achieve, many stylized effects are more computationally intensive, and the constraints may be more daunting. Handling transparency in gasses and liquids without blending can be tricky, for example. In either case, it is helpful to acquire a habit of asking whether the benefit of any approach will be impactful enough on the viewer to justify additional time or effort required to achieve it. This is a process called "budgeting," that we will explore further in *Chapter 11, 3D in Production*.

3D Cameras

In real life, objects are visible to humans because photons moving through space bounce off of surfaces and then toward our eyes, which then transmit a signal to our brains. While we may imagine the unseen, we cannot see objects that do not reflect any light. The amount and angle of light bounces affect how much of an object we can see and whether the surface appears brightly lit, diffusely lit, or shadowed. Surface properties such as roughness, metallicness, and translucency affect whether we can see reflections or the transmission of light as well.

As we have touched on many times before, it is because all 3D software simulates this real-life behavior of light in various ways to display – or "render" – 3D objects, that the appearance of 3D objects can vary just as widely. To capture 2D images or videos (a series of images) of 3D creations, we must first choose a viewpoint, and then we must set parameters to indicate what we want captured in the view. In 3D, we use virtual cameras and lights to do this.

Note: Renders can be "unlit" and yet still capture visible 2D representations of 3D objects, or even utilize MatCaps as we saw in Chapter 5, Poly Modeling Extras. More often though, creative placement of 3D lights produces desired results.

Added to the scene just like a 3D object via the Add menu, Cameras and Lights in Blender are akin to gizmos and Empty objects, in that they do not have components like vertices, edges, or faces. Therefore, 3D Cameras and 3D Light sources are not rendered in images or videos themselves. However, like gizmos, 3D Cameras and Lights both have adjustable properties, and visible overlays in the 3D Viewport to assist with their positioning and animation.

Note: 3D renders typically work through a virtual camera and use virtual scene lighting, however the Viewport Render option in Blender simply renders everything we see in the 3D Viewport depending on the currently selected Overlays and Viewport Shading Mode.

Exercise 10.1

For this exercise, we will set up a scene to render the Pedestal object from previous exercises. To begin, we will practice copying content from one instance of Blender to another, to take advantage of the default scene's camera and light.

1. First, let us open a new instance of Blender, and leave the default **Camera** and **Light** in place.

Note: If we have changed our Blender defaults, we can navigate to File | Defaults and choose Load Factory Settings to restore them.

2. Delete the default **Cube**.
3. To set up any 3D scene to render, we first choose a Render Engine via the **Render Properties tab** of the **Properties panel**. In this case we will use the default, which is Eevee.
4. Next, we will open a second instance of Blender, by pressing and holding the Shift key, then clicking the Blender icon, either in our **Windows Taskbar**, or in the **Windows Start menu**.
5. In the second instance of Blender, navigate to **File | Open**, and navigate the **Blender File View** to open the Pedestal project from *Chapter 7, 3D Surfaces*.

   Alternately, we can open a new instance of Blender and simply add a **Mesh | Monkey** in the scene to practice with.
6. In **Object Mode**, select the **Pedestal object** to be copied in the **3D Viewport** or the **Outliner**, then with the mouse cursor over the **3D Viewport**, press Ctrl + C to copy the selected object to the clipboard.
7. Next, we will navigate to the first instance of Blender still open on our desktop. Then, with the mouse cursor over the **3D Viewport** in this Blender project, press Ctrl + V to paste the copied object into the scene.

   Alternately, we can right click in each – first the source, then the target 3D Viewport – and choose **Copy Objects**, then **Paste Objects** respectively, to perform the same operations.

Note: By default, the scene Camera is already set to Active, but a newly added Camera needs to be set Active before it can be used as the viewpoint for renders. To set a new Camera to Active, first select it, then under the View menu at the top left of the 3D Viewport, select Cameras, and Set Active Object as Camera.

8. With an **Active Camera** in the scene, to see the scene through the camera's "frustum" or **front view plane**, press the Number pad key **0** as shown in *Figure 10.2*.

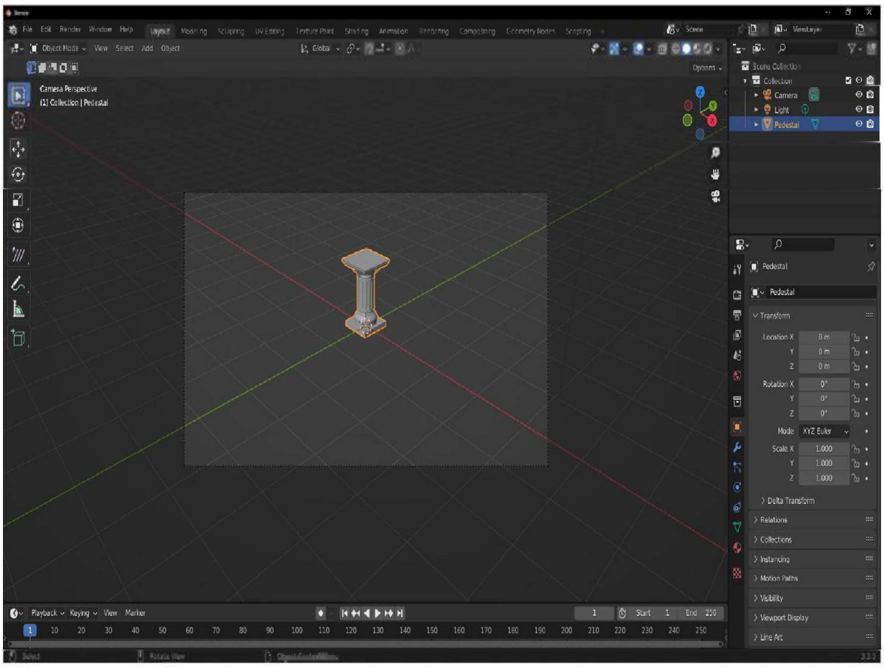


Figure 10.2: Pasted Pedestal object from earlier exercise, framed within the view plane of the Active Camera

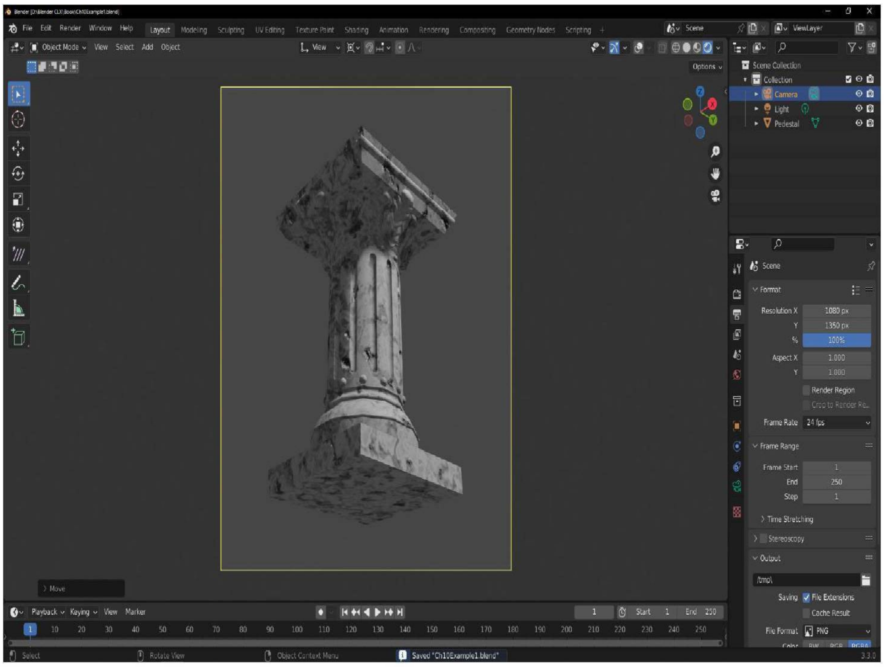
The pedestal could be framed better in portrait orientation with less empty space around it, so we will learn to adjust the **Camera** and **Output parameters** next.

9. First, we will open the **Output Properties tab** of the **Properties panel**, with the icon appearing as a printer printing an image.
10. From here, we will set the **Resolution** to 1080 by 1350, which are common dimensions for the portrait orientation used for the web or print.

    Next, we will move the **Camera** closer to the **Pedestal**, while maintaining the current 45 degree – or "isometric" – view.
11. Press the N key to open the **Side bar menu**, and from there, select the **View tab**.
12. Under the **View Lock section** toward the middle of this panel, enable **Camera** to **View** by clicking the check box.

    The Camera's view plane is surrounded by the dotted line, and we can use viewport controls to position it.
13. Scroll the mouse wheel forward to zoom the **Camera** closer to the **Pedestal**.
14. Press G, then Z, and move the **Camera** upward until the whole **Pedestal** is framed.

15. Disable the **View Lock** by deselecting the check box, and close the **Side bar** menu by pressing **N**.
16. From the top middle of the **3D Viewport**, change the **Transform Orientation** from **Global** to **View**, and change the **Transform Pivot Point** to **3D Cursor**.
17. Press **R** and **X** to constrain rotation of the **Camera** on the x view axis and rotate the camera downward to create a dramatic upward view of the **Pedestal** from below, as shown in *Figure 10.3*:



Saved "Ch10Example1.blend"

Scene

Format
Resolution X 1080 px
Y 1350 px
% 100%
Aspect X 1.000
Y 1.000
Render Region
Crop to Render Re...
Frame Rate 24 fps

Frame Range
Frame Start 1
End 250
Step 1
Time Stretching

Stereoscopy

Output
/tmp\
Saving ✓ File Extensions
Cache Result
File Format PNG

3.3.0

Figure 10.3: Pedestal object from previous exercises shown in Rendered Viewport Shading mode

Beyond this point, additional **Camera settings** such as **Focal Length** and **Depth of Field** can be adjusted by selecting the **Camera object** from the **Outliner** and opening the **Object Data Properties tab** of the **Properties panel**, with the icon appearing as a movie camera.

3D lights

Just as important as the 3D Camera and its parameters, 3D lighting determines how we see what we see in rendered images or videos. In Blender, there are multiple valid approaches to lighting a scene, including adding and positioning **Point**, **Sun**, **Spot**, and **Area lights** for dramatic effect. In our Pedestal scene, we now have a single **Point light**, which can be changed into any of the other light types through the light's **Object Data Properties tab** in the **Properties panel** with an icon appearing as a light bulb. Not all 3D lighting involves an actual 3D Light though.

By default, the background in a Blender scene is a flat, medium grey and does not contribute much to the ambient light in the scene. A range of effects can be achieved simply by changing the **Strength va**lue of the **Background**, or by changing the **Background Color** via the **World Properties tab** of the **Properties panel**. Replacing the default Color background with a special image called an HDRI is another way to add natural ambient lighting, and even the look of distant scenery and reflections.

Tip: HDRI images are tricky to create from scratch, which is why HDRI photography is a whole industry unto itself, but there are many online sources for HDRI images. Some are available for free, some for donation, and many for purchase on stock 3D asset marketplaces. Some artists even create HDRI images of 3D scenes to produce fantasy lighting or to reproduce natural lighting that is difficult to capture in real life.

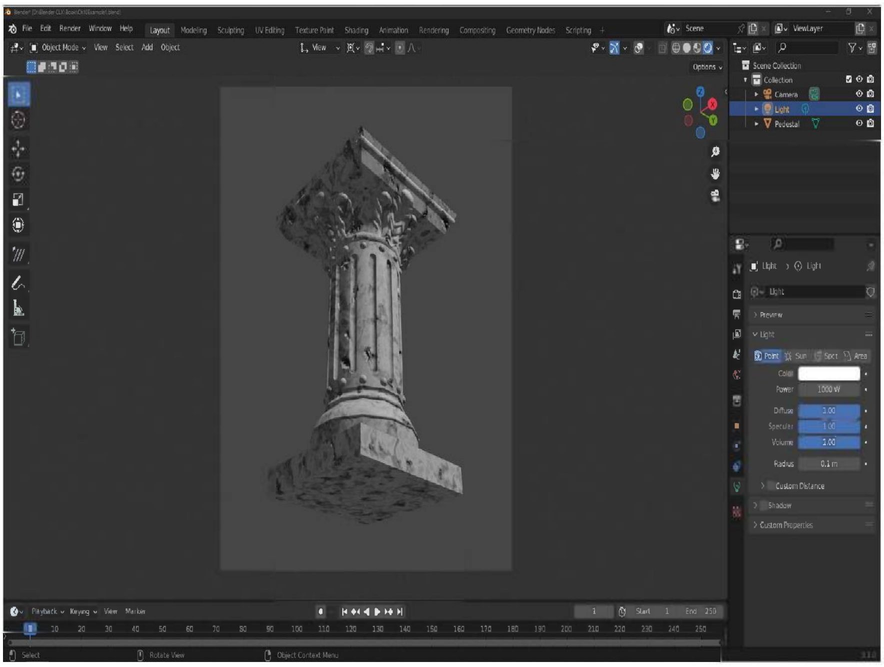
Blender's 3D Lights share many similarities with lights in other 3D software. Point lights have a limited range that is spherical. Spotlights also have a limited range but are cone shaped. Area lights take on a planar shape, which helps mimic bounded light sources like windows. Limiting the range of a light helps constrain the computations needed to render their effects on a scene, making rendering faster. Sun lights (akin to **Directional lights** in other 3D software) are unlimited, which makes them great for adding ambient light where limited lights don't reach.

Another method of adding the appearance of light to a scene is through enabling **Emission** in Materials on 3D objects, combined with enabling and adjusting Bloom via the **Render Properties tab** in the **Properties panel** to make 3D objects glow. Emission and Bloom increase the believability of objects modeled to look like lamps, lightbulbs, neon tubes, illuminated signage and such.

3D Shadows

Shadows are also an essential component of lighting, emphasizing depth and detail through contrast. To see the dramatic impact of shadows, with the scene Light selected, in the **Object Data Properties tab**, disable **Shadow** by unchecking the check box. While the result is arguably less realistic, removing the **Shadow** from the **Light**

allows the ambient light in the scene to reach the detail under the **Pedestal cap** as shown in *Figure 10.4*.

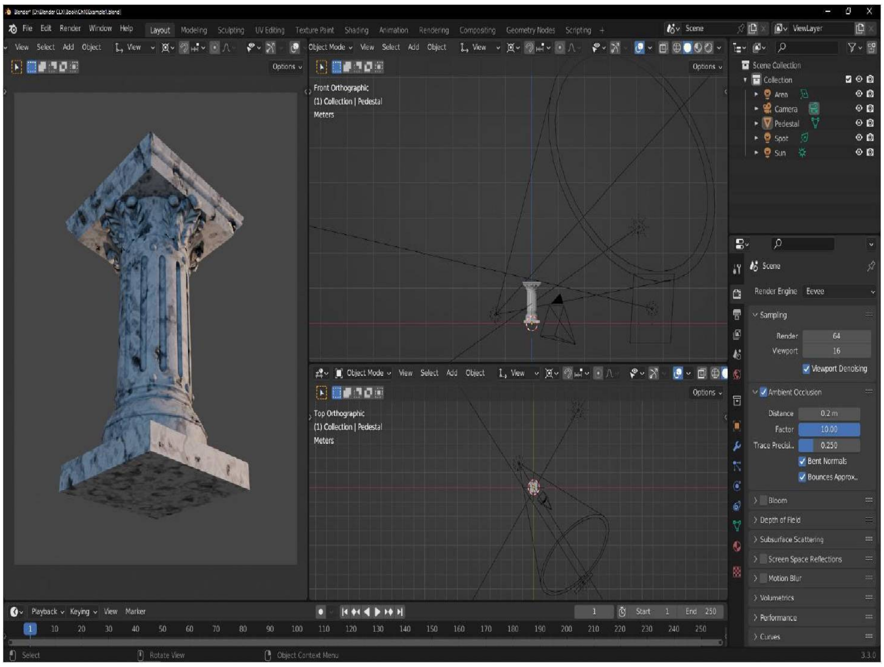


Color
Power 1000 W
Diffuse 1.00
Specular 1.00
Volume 1.00
Radius 0.1 m
Custom Distance
Shadow
Custom Properties

Figure 10.4: Pedestal shown in Rendered Viewport Shading mode with Point light's Shadow disabled

While this works well enough for a quick render, the common approach to achieve more professional results is to set up additional lights that mimic a real-world photo shoot. The standard "three point" lighting setup can be easily achieved in Blender through manual placement of lights: one at a natural angle with the warm color of sunlight, another cooler colored fill light to bring detail out of shadowed areas, and a third behind the subject to provide a "rim" light as shown in *Figure 10.5*.

Tip: Here is a situation where splitting the view so we can move lights around while seeing the result through the camera comes in handy.



Render Engine Eevee
Sampling
Render 64
Viewport 16
✓ Viewport Denoising
✓ Ambient Occlusion
Distance 0.2 m
Factor 10.00
Trace Precisi... 0.250
✓ Bent Normals
✓ Bounces Approx...

Figure 10.5: Pedestal lit by 3 lamps with Ambient Occlusion enhancing shadows

In addition to the shadows cast by objects in the scene, we can also enable and increase the **Ambient Occlusion** (shadows that accumulate wherever perpendicular surfaces meet) via the **Render Properties tab** of the **Properties panel**, with the icon appearing as the back of a digital camera, as shown in *Figure 10.5*.

While far from final, the last technique for lighting 3D scenes that we will mention here is combining textures with lights, also called "gobos" and "cookies," to change the basic round shape of light into patterns by only allowing light to pass through the lighter parts of a greyscale image. Masking light with texture this way creates natural looking patterns, like the shade cast by leaves of a tree or light rippling under the surface of water.

Note: Adding textures to lights requires use of the Cycles Render Engine and can be accessed through the Shader Editor by selecting a light and enabling Use Nodes for it in the Shader Editor viewport.

Exercise 10.2

1. To preview how our rendered image will look, let us enable the R**endered Viewport Shading mode**, and disable **Show Overlays**, from the upper right cluster of icons in the top of the **3D Viewport**.

2. Next, we will change one of the **3D Viewports** into the **Compositor Editor Type**, by selecting **Compositor** from the far left drop down menu at the top of the **3D Viewport**.

3. Tap **N** to close the **Side bar menu**.
4. At the top of the **Compositor viewport**, enable **Use Nodes** by clicking the checkbox, and enable **Backdrop** by clicking the **Backdrop** button in the upper right of the **Compositor viewport**, leaving it highlighted.
5. Choose **Add** from the top of the **Compositor viewport** or press **Shift + A**, then click **Search** and type in the word **Ellipse**. Click the **Ellipse Mask option**, and then click in the **Compositor viewport** to place it.
6. Search again in the **Add menu** for the word **Blur** and add a **Blur node** into the **Compositor**.
7. Next, search the **Add menu** for **Alpha Over** and place that node into the **Compositor**.
8. For the last node, we will search the **Add menu** for **Viewer**.
9. Press **F12** to render a preview of the image to work with, then close the **Blender Render viewport** by pressing the **X** in the upper right-hand corner.

Next, we will wire up all our new nodes.

10. First, we will click the circle to the right of the Ellipse Mask's Mask output and drag the wire to the **Image input** on the bottom left of the **Blur node**.
11. Then, we will click the circle to the right of the **Image output** on the right side of the **Blur node** and drag that into the **Fac input** on the upper left side of the **Alpha Over node**.
12. We will click the circle to the right of the **Image output** of the **Render Layers node** and drag that wire into the **Image input** on the lower left side of the **Alpha Over node**.
13. The open Image input on the middle left side of the **Alpha Over node** has a box we can click to choose a color for the vignette. We will set this to solid black for this exercise.
14. Lastly, we will click the circle to the right of the **Image output** on the **Alpha Over node**'s upper right side and drag two wires from it, one into the **Image input** on the upper left of the **Composite node**, and one into the **Image input** on the upper left of the **Viewer node** as shown in *Figure 10.6*.
15. At this point, we will need to adjust the **Width** and **Height** of the **Ellipse Mask**, as well as the X and Y values of the **Blur node**, to achieve the vignette look that we desire.

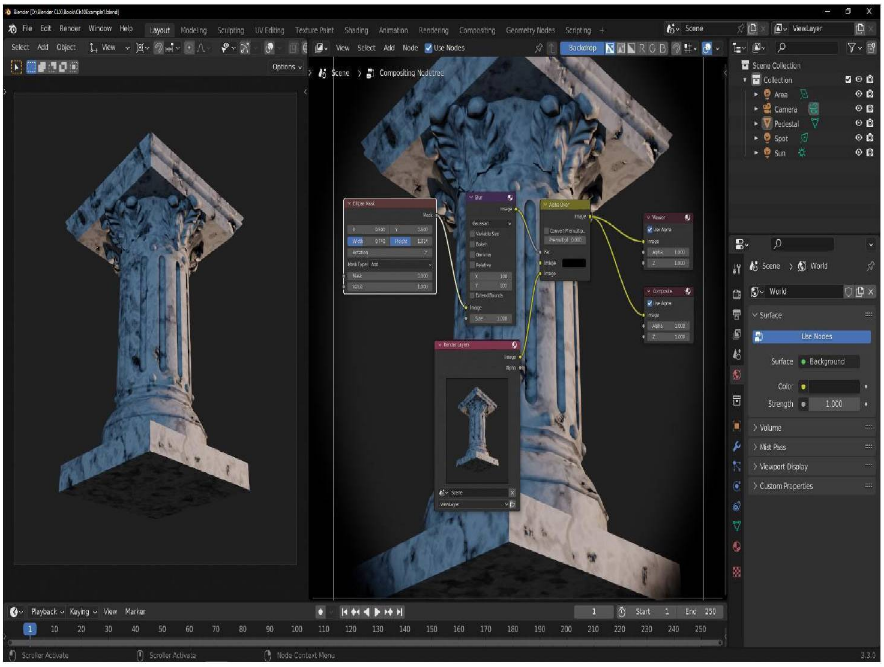


Figure 10.6: Pedestal image composited with a blurred Ellipse Mask, creating a vignette effect

16. Finally, we can press the **F12** key to render the image at full resolution as shown in *Figure 10.7*, where we can save it to our hard drive.

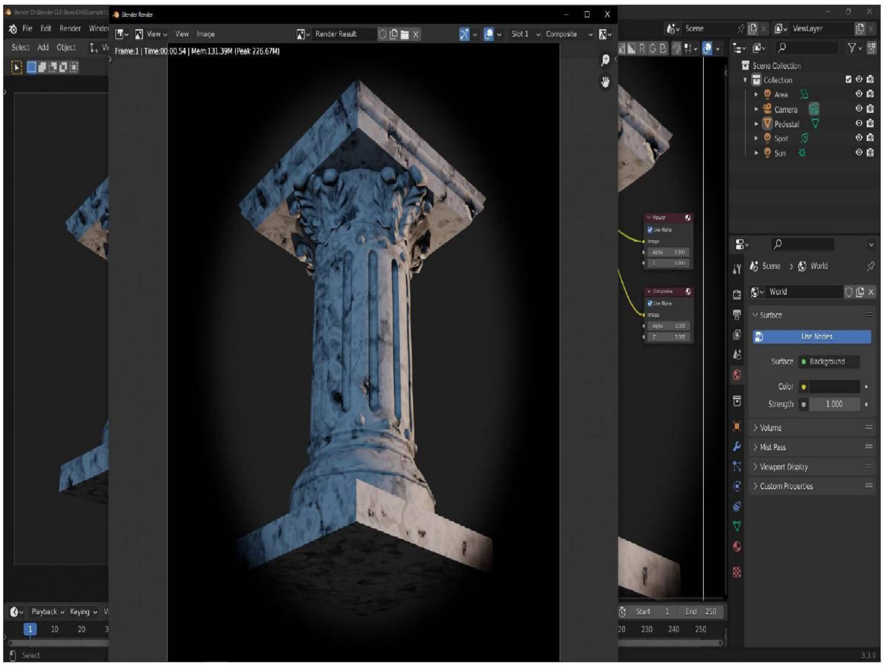


Figure 10.7: Final Pedestal render shown in Blender's Render window, suitable for graphics, portfolios, and marketing

Editing images and video

Though we learned how to edit Textures applied to a 3D model via Blender's Texture Paint Workspace in *Chapter 7, 3D Surfaces*, it is important to understand the distinction between **Texture Editing** and **Image Editing**. The **Editor Type Image Editor** in Blender can be accessed through the upper left dropdown menu of any viewport **Editor Type**, but should be understood to have limited functionality.

Image Editing in Blender refers to changing the 2D images of 3D objects after they have been rendered. While Blender's **Image Editing viewport** allows for drawing annotations and viewing some scopes, such as the Histogram, external image editors are better suited for post-processing beyond the Compositor. Conveniently, an eternal image editor can be added to Blender's interface via **Edit** | **Preferences** | **File Paths** in the **Applications section**. The assigned external editor can then be accessed from the **Image Editor viewport** under the **Image menu** in the top left, as shown in *Figure 10.8*.

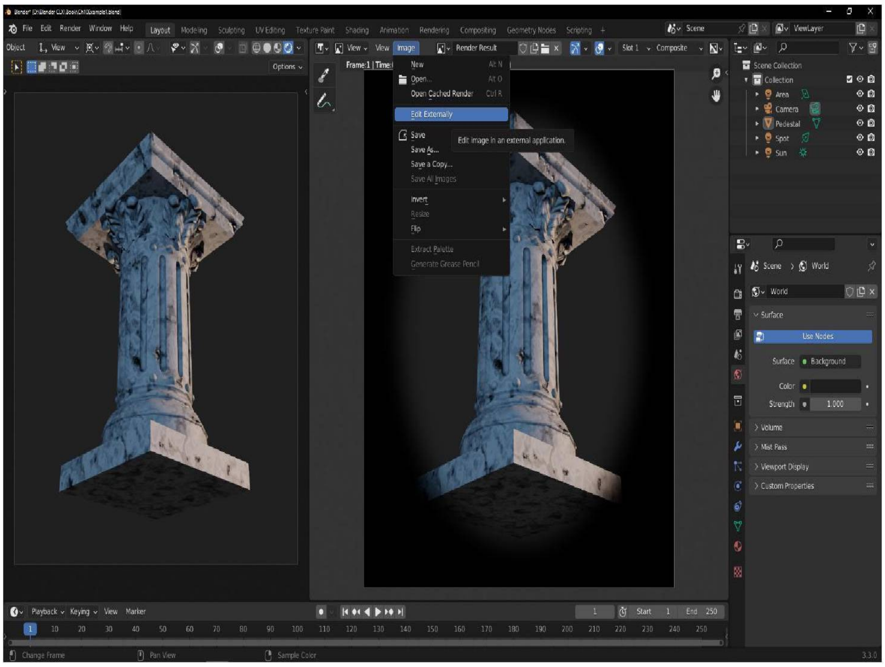


Figure 10.8: Accessing an external image editor via the Image Editor viewport to edit a render

In addition to offering compositing and easy access to external image editing tools, an underappreciated feature of Blender is its full-featured video editing, which is extremely convenient for animation pipelines that feature 3D content. Accessed by adding the **Video Editing tab** to the **Workspace tabs** at the very top of the Blender application via the plus sign at the far right, the **Video Editing Workspace** provides a **Preview viewport** and a Sequencer, which allows multiple Animations to play at different points in a video.

Like many of Blender's advanced capabilities, much of the Video Editing functionality is beyond the scope of this text. However, it is helpful to know that additional features like a full-featured 2D Animation pipeline and Motion Tracking can also be found in the **VFX Workspace tab**. Motion Tracking allows for the incorporation of 3D content into any video, using markers that move along with objects in the video, which can, in turn control the placement, rotation, and scale of 3D objects to make them seamlessly match the scene.

Conclusion

In this chapter, we have begun to put all of the pieces together – from navigating the Blender interface, to modeling, texturing, and animating 3D objects – and now we can finally prepare these creations for viewing, sharing, and even generating an income. In the final chapter, we will discuss exporting and using 3D models directly in the production of other 3D products like games and interactive applications.

Essential shortcuts and hotkeys

In this chapter, we used the following additional Shortcuts and Hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc

Scale	S
Rotate	R

Shortcut	Hotkey
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show/Hide Side Bar	N
Show/Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Toggle Armature Edit Mode and Pose Mode	Ctrl + Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F

Set Active Object as Camera	Ctrl + Number pad + 0
(Look through) Active Camera	Number pad 0

***Table 10.1:** Essential Shortcuts and Hotkeys*

Points to remember

- Assembling 3D assets in a scene is much like the process of setting a stage for theatrical productions, including props, lights, and cameras.
- Different Render Engines have different capabilities which dictate which choice is the best suited for our desired result.
- When it comes to rendering, more realistic is not always better.

Questions

1. What are Blender's two main Render Engines?
2. What is something only one Render Engine can do that the other cannot?
3. How does Texture Painting differ from Image Editing in Blender?
4. Where can we find the Video Editing Workspace?

CHAPTER 11
3D in Production

Introduction

In this final chapter, we will learn how to extend and apply our new Blender 3D skills to become sought-after creators in our fields of interest with the potential to earn life-sustaining incomes.

Structure

This chapter wraps up essential knowledge for job seekers who wish to use Blender 3D for fulfilling and lucrative work in 3D content creation. Topics to be covered include:

- Best practices of highly valued 3D creators

- Budgeting performance and design considerations
- Optimizing objects and workflows
- Preparing 3D objects for production
- Complementary tools for 3D production
- Next steps: where to go from here

Objectives

Learning how to create 3D content that is functional and appropriate for the production of marketable 3D products is the key to becoming a sought-after 3D creator because earning a reputation as a conscientious, reliable creator cultivates relationships with clients and employers who will pay us and recommend our work again and again. After reading this chapter, we will be ready to employ our knowledge of Blender and 3D basics for pursuing any number of rewarding and profitable creative endeavors.

Best practices

So far, we have begun each chapter and exercise of this book by examining the high-level considerations to make before diving in. We practiced this step repeatedly because one of the most critical components of working with 3D in a production environment is to devise a strategy for getting from start to finish before launching into work. This preparation helps ensure that our time working is used efficiently, that the impact of our work is maximized, and that missteps are minimized or avoided entirely.

Time spent is not just money, as we have frequently acknowledged; it is also momentum. Losing work, wasting work, and repeating work are all demoralizing mistakes that can chip away at the crucial motivation to continue moving forward to reach a goal or may even dissolve the trust that a client has invested in our work.

Production

Production refers to working toward any vision or end product as a whole rather than its parts. Production might refer to the development of games or augmented, virtual, mixed, and extended reality products. Production might also refer to app development; a product configurator or 3D visualization; short or long films for the web, television, or the big screen; renders for social media or a magazine spread;

additive or subtractive manufacturing; or it might even involve a groundbreaking invention that's never been seen before.

Production of products that contain 3D elements involves one or multiple processes –also known as "workflows" – which ideally blend together into one desired result. Even if we are working iteratively by making prototypes, testing, then revising and testing again, or diving in and letting creative impulses take our work where it feels right, it is usually very beneficial to define a clear goal, such as making a game or making a portfolio piece.

Getting started

While some form of prior planning is critical to production, for the sake of creativity, it is often also important to dive in while enthusiasm is high. Getting started in 3D production can take many forms. A brainstorming session with or without teammates where multiple options are listed or discussed without judgment can be immensely helpful to get the wheels of production turning. Even brainstorming can benefit from some structure, such as someone to describe the process, someone to start and stop the session, someone to take notes, or an agreed-upon time frame for conducting the session, and then a plan for how to sort and evaluate the results.

Shared understanding

Once a direction for production is identified, creating mood boards or collages can help pull together a themed look and necessary reference material. Creating concept art or paying concept artists for their work can have the same effect of unifying a team or client and creator around a shared understanding of the desired outcomes.

As we have seen with our Pedestal exercises, sketches, and story boards – even if they feature stick figures – can also provide the framework for a thoughtful production plan. It is also possible to sketch in 3D. Blender is a fantastic tool for sketching in both 2D and 3D and can even be used in some VR headsets for an immersive, first-person view of our blossoming creations. Planning in 3D this way is often a more effective way to communicate a concept than through 2D or words by giving collaborators and clients a more thorough perspective of our ideas.

Physical prototypes can also be extremely effective at communicating concepts that are difficult to verbalize. Using dice, miniatures, and other pieces from board games; pencil and paper; scale models or "maquettes" made from clay, wood, or other materials; posable figurines; music and sound effects: All sorts of real-world objects and media evoking all of the human senses can help solidify indescribable or vague ideas.

Asset lists

From these solidified concepts, lists of assets are more easily made. An asset list is an estimate of the individual items that are needed to complete a project or some phase or critical part of a project. For 3D, an asset list might include meshes, textures, animations, effects, graphics, audio clips, code, and even scenes or "levels" (as in a game) that need to be created or curated, set up and arranged. On very large projects, asset lists are usually divided by scenes or similarly logical grouping to keep them manageable.

Creating asset lists makes it easier to divide work amongst different creators and/or into smaller, less daunting milestones. Creators can start by reviewing an asset list and then check off each asset as it is completed. Asset lists can be recorded on a notepad, a spreadsheet, an outline, a flowchart, a wireframe, a whiteboard, or even a project management board. All of these tools can be used individually or together to record and highlight the necessary steps to get from the beginning to completion of a "shippable" or marketable product.

Rough drafts

The next stage of production is creating rough drafts. Again, Blender is exceptional for blocking out 3D – such as the process of grey boxing that we first discussed in *Chapter 4, Polygonal Modeling* – using primitive models or basic sub-object components to map out a concept that is meant to be refined further. Grey boxing involves quickly placing primitive objects of the approximate size and general shape of the meshes planned for the end product so that 3D creators and collaborators, employers, and clients can validate the composition and test functionality such as line of sight or pathing.

Refinement

With a rough draft complete, it is time to begin refining. Sometimes in the midst of production, we will continue to build on rough drafts. We may extrude, bevel, inset, subdivide, and sculpt a 3D draft or even use primitive elements as-is. Other times, we may replace the rough elements entirely. We may also use our initial concept as a low poly target and add further detail to a duplicate to be baked back onto the low poly mesh as we did in *Chapter 6, 3D Surfaces*.

Starting over

Occasionally it becomes obvious that our plans are going in the wrong direction entirely, so we may choose to start over again from scratch. Having begun with broader, more general concepts at the planning stage makes it easier to pivot midway and iterate as needed. It can be scary, but being able to catch an idea that is not working and scrap it early on is an incredibly valuable skill for production, saving vast amounts of time, effort, and expense, as well as morale and momentum.

Budgeting

Working on larger projects often means working within budgets. In 3D production, there are a variety of budgets to keep in mind.

Priorities

Since human effort takes time, and every human creator requires nourishment, shelter, rest, hygiene, relationships, and recreation to sustain that effort, time is perhaps the most obvious production resource with a limited budget to manage. Oftentimes, the time limit on production is the money available to sustain the human effort. Just as every human has a finite amount of time, every project also has a finite lifespan, regardless of whether it ends in completion, a pause, or abandonment. Even clients who have seemingly endless resources will expect a return on their investments.

For any project to bring in more money than it costs to produce, certain milestones must be completed, usually within a defined amount of time. Those milestones might occur before, after, or coinciding with an event where visibility and enthusiasm for the product are high, until a self-sustaining feature is complete or even until a specific pool of money runs out.

Completing production milestones within deadlines is easiest when tasks are "triaged," meaning that the relative priority of available tasks is estimated and weighed. Then, time and effort are devoted to the highest priority tasks first. Getting a character walking and idling, for instance, may take priority over flashier animations. Triaging tasks is tricky, as the pros and cons of any task are relative to the unique goals of any project. Sustaining creators through profit might be the top priority for one team or project, while achieving positive social impact or adhering to an artistic vision may be more important to another.

Working within a time budget is often easiest when the entire production is broken into smaller deadlines or milestones. A very large project with uncertain prospects for funding will often benefit from an approach called the "vertical slice" or MVP (for the minimum viable product), where the least effort possible is expended on key components to give a general sense of what the larger product would be like if it were expanded or "scaled" outward. MVPs and vertical slices are typically produced in a format that can be tested for feedback. The vertical slice of a game, for example, might include rudimentary systems for loading into and moving around the game world with only a few playable levels with rough art or even grey-boxed scenes so that play-testers can verify whether they would pay to see more.

Budgeting time for production also involves assessing and constraining the scope of the project. With as many creative possibilities that 3D presents, the temptation to add more and more detail or complexity to a project is constant. Working in 3D often inspires creators to enter a state of "flow" where time seems to stop and progress is unimpeded. But overcomplicating one aspect of a 3D production is often done to the detriment of another aspect. Taking longer on one part means less time for another. Keeping the "scope" or practical limits of the project in mind and restraining the urge to indulge in more than was planned and budgeted for is the key to reaching completion. Another highly valued skill.

Performance

As we have touched on in many chapters, another common budget to consider is the limitations of target software and hardware. Software and hardware budgets are usually thought of in terms of the maximum allowable for a whole project or scene, whether that be file sizes; the number of polygons; the number and resolution of textures; the number of materials; or some combination of all of these. Software and hardware budgets also include less obvious metrics such as shader complexity; the number of objects visible on screen at once; overlapping and transparent objects; the number of keyframes in animations; and even whether animations continue while offscreen or not.

A critical component of performance budgets is appropriateness for purpose. Not all 3D objects are made equal. In terms of purposes that impact performance in 3D, there are generally two main categories:

- Moving
- Static

Interactivity, animation, and rendering – even to 2D – are all situations where 3D object data is being transferred. In that sense, the 3D object's purpose is moving. Once rendered, engraved, or 3D printed, or any state in which the amount of detail can no longer be changed, the 3D object's purpose may be considered static.

As a general rule, lower resolutions are more appropriate for the performance of moving 3D objects, while higher resolutions are more appropriate for giving static 3D objects adequate detail. Games tend to run better with lower poly meshes, while 3D prints tend to turn out smoother when subdivided more, for example. However, the increased processing power, effort, and time required during the creation, transfer, and conversion (all movement) of a digital 3D object into a static format should not be dismissed. Striking the right balance by making sure the result justifies the requirements is a challenge and another highly valued skill.

Not only can software and hardware limitations affect the resolution of 3D assets they can handle, but they also impact the quality of the end product. For example, cheaper 3D printers use larger filaments. Since a highly subdivided mesh will not result in a smooth print if the target 3D printer has this limitation, it might not be worth the extra time and effort to create a very high poly mesh, to begin with. Another example is the fact that necessary lighting, materials, effects, and animations to produce a desired look are not always available in a target platform such as the web or mobile devices. Not every production target is capable of handling 3D objects made with Blender as beautifully as Blender can, so at times it is necessary to scale back.

The performance budget considerations for the software and hardware used in production as well as those being targeted are so numerous that it is impractical to

try and learn or commit them all to memory. In general, the best practice is to first anticipate that these limitations exist and then research the specific limitations of the software and hardware that are needed for each project before production begins.

Design

Design is an often overlooked but equally crucial area of production where a budget should be planned and adhered to. Budgeting for impact, for example, means considering which 3D elements are the most critical to the user experience and prioritizing them first. In practical terms, this might mean a small object, such as a pocket watch should not be as detailed as a much larger dresser that it sits on. That is, unless the pocket watch is meant to be a focal point, functional, or examined closely. Budgeting detail proportional to its impact is important at all levels, from individual objects and their components to screen space, entire scenes, and even projects as a whole.

No discussion of design budgets is complete without reviewing three key concepts:

- Modularity
- Reusability
- Cohesiveness

Modular 3D design means planning each 3D object to be an interchangeable part of a system that can be arranged in multiple ways. Building blocks are a very basic example of modular design. A set of toy blocks often consists of assorted cylinders, cubes, arches, and prisms that can be stacked and aligned in infinite configurations. In 3D, modular designs usually also feature a texturing technique called seamless tiling, where separate pieces have edges matching so well that the separation cannot be detected, as shown in *Figure 11.1*:

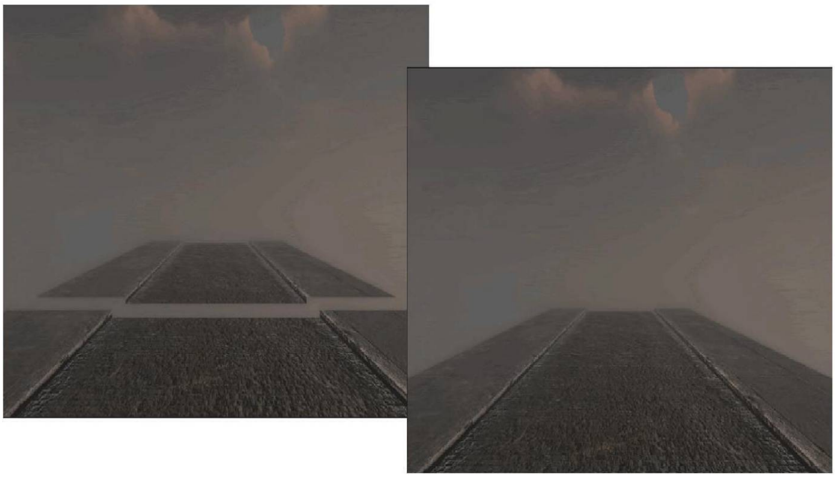


***Figure 11.1:** A modular road and sidewalk, separated on the left, snapped together and seamlessly tiling on the right*

Modularity in 3D also takes advantage of a feature of many 3D applications call "snapping" as we first practiced in *Chapter 2, Installation and Interface*. This continuity is achieved by creating each modular piece in a specific unit of measure established ahead of time or in regular subdivisions of that unit. Sometimes modularity is injected retroactively by cutting large pieces into smaller interchangeable sections, like separating walls, floors and ceilings from a room, or the skin, clothing, and accessories from characters.

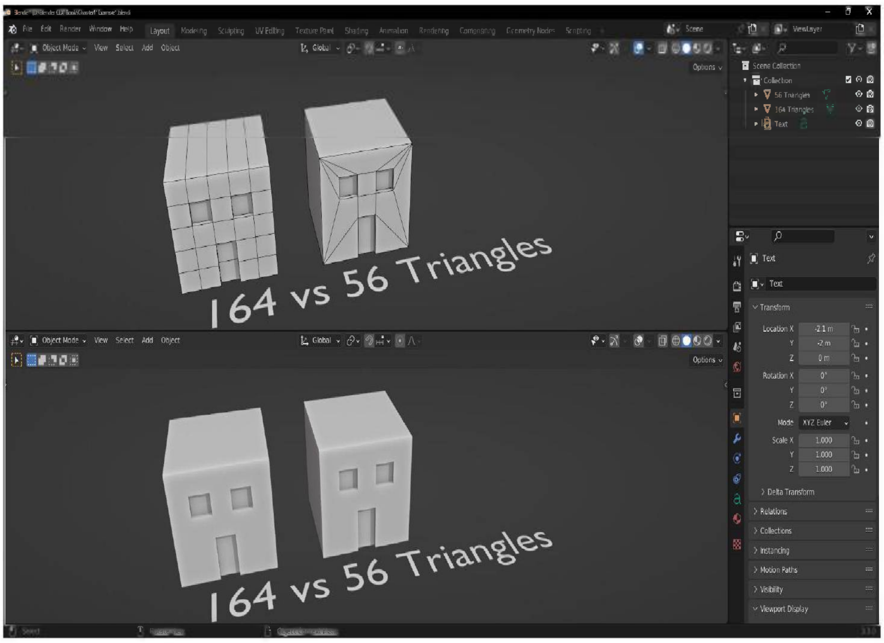
Designing 3D objects for reusability is the key to a modular design. Making a base – such as a turbulent particle effect – generic enough that a new feature such as a different material, could change it from smoke, to fog, to fire, is a very practical example of reusability. It is smart to save duplicates and make templates of frequently created objects or even object components. From the extruded and inset and beveled plane that could be a window, door, picture frame, or panel on electronics to entire scenes with lights, camera, and background set up to our preference: Anything we or our teams make repeatedly can form the basis of our own timesaving and mistake-preventing asset library.

Finally, the figurative glue that holds together reusable modular parts is cohesive design. Aside from using shared textures and a common unit of measure so that parts literally come together seamlessly, other elements that provide cohesion include color schemes, themed surface treatments such as shared materials and shaders, and repeated shapes or motifs in trims, or a pattern that reappears in every piece. Unifying effects can also be applied through compositing or post process, like **Color Grading** and **Gradient Maps**, **Look Up Tables**, **photo filters** and other global or screen based visual effects.

Optimization

The performance of software used during production and finished 3D products depend on the efficiency of the 3D objects within them and workflows used to create them. For example, applications which display 3D creations will run slower, the more polygons any given 3D object is made of. Optimization is the process of actively preventing or fixing these slowdowns.

Optimizing 3D objects involves strategically omitting or removing unnecessary data, from unnecessary geometry like edge loops as shown in *Figure 11.2*, to artifacts of construction like middle seams left over from mirror operations. The optimal approach to optimization is to leave 3D objects as simple as possible while still maintaining their intended form and function, whether that's decorative, as a focal point for attention and interaction, or as a distant prop or filler. Please refer to the following figure:



Location X -2.1 m, Y -2 m, Z 0 m
Rotation X 0°, Y 0°, Z 0°
Mode XYZ Euler
Scale X 1.000, Y 1.000, Z 1.000

Figure 11.2: *The same 3D shape, unoptimized on the left, and more optimized on the right*

A simple method for reducing unneeded geometry from 3D objects in Blender is to first enable Auto Merge as shown in *Figure 11.3* by pressing the icon in the upper right-hand corner of the **3D Viewport** in **Edit Mode** appearing as two vertices with zigzag lines between them.

Note: Working with Auto Merge enabled means that any two vertices occupying the same space will become welded together, so it is important to toggle this functionality off when that result is not desired. It is possible to separate inadvertently welded selections of vertices, edges, or faces by choosing Selection under Mesh | Split from the top of the 3D Viewport.

With **Auto Merge** enabled, we can select vertices that are not supporting critical structural edges like the boundaries of the windows and door in *Figure 11.3*, then we can activate **Edge Slide** by pressing G + G, to slide vertices along their edges into the nearest corners, wherever the move and merge won't cause unwanted twisting or overlap of surrounding geometry. Retopology, whether through automation or performed manually this way, and as we performed on the sculpted elements of our

Pedestal, is an important method of mesh optimization. Please refer to the following figure:

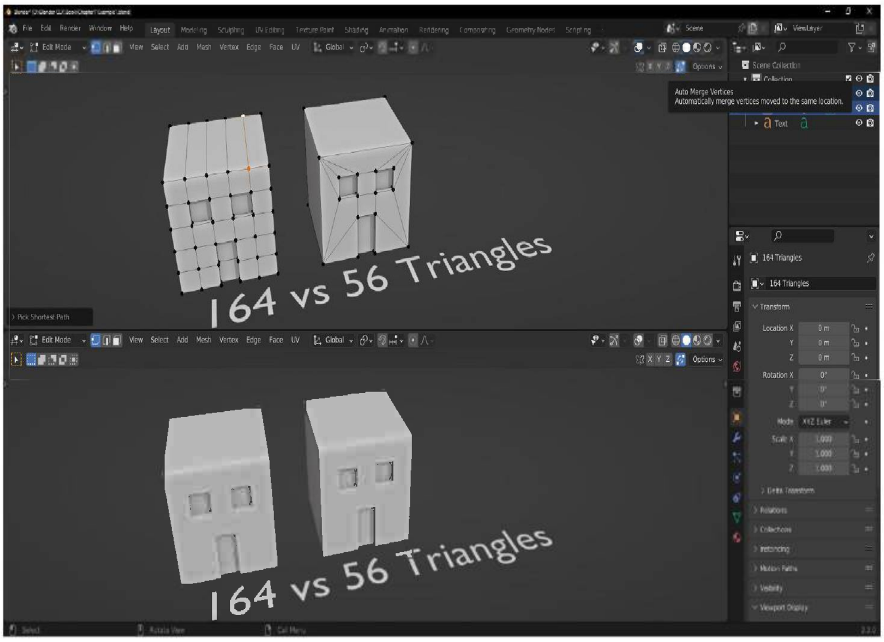

Auto Merge Vertices
Automatically merge vertices moved to the same location.

Figure 11.3: *In Edit Mode, Auto Merge enabled*

The poly count of the cube house example could be reduced even further by deleting

The poly count of the cube house example could be reduced even further by deleting unseen faces, but simply **Auto Merging** all the vertices that are not supporting the structure or silhouette brought the cube house in *Figures 11.2* and *11.3* from 164 triangles to 56. This might not seem like a big savings for one 3D object, but since many duplicates of a given mesh may appear in a scene, along with tens, hundreds, or maybe thousands of other elements of a 3D project, time spent optimizing even one asset can quickly add up to significant savings in overall performance.

Tip: Wait to optimize meshes until their design is finalized to make UV unwrapping and texturing easier but remember that triangles are not problematic in and of themselves. In fact, all 3D objects are constructed of triangles behind the scenes as we learned in Chapter 4, Polygonal Modeling. Manual triangulation of a mesh, and controlling which direction any quads are triangulated, can actually preserve the silhouette of a mesh, and prevent texture warping and unwanted shading artifacts.

LODs

Using duplicates temporarily in our workflows as stand-ins for high detail, as we did to bake sculpted detail onto the optimized Pedestal trim, is one way to keep individual 3D objects optimized. But lower poly duplicates can also be used to optimize entire scenes or a project. In games and other interactive applications,

lower poly duplicates left in the same place as the original and swapped in and out to optimize the scene view at run time are called **LODs**, for **Level of Detail**.

Most game engines and a few other real time 3D applications have dedicated LOD systems that automatically swap in lower poly duplicates per object at certain distances from an active 3D camera, as long as the LODs are set up properly for the application. Setting up LODs in Blender for applications with LOD systems usually involves duplicating a 3D object 2-3 times and then progressively reducing the resolution of each of the duplicates in the same location as the original.

As much as possible, the goal of creating LODs is to maintain the same silhouette and shading as the highest quality mesh so that the swap between LODs is imperceptible. Effective LODs may be decimated, an algorithm where neighboring vertices are welded together, or they may just have unnecessary data removed such as back faces that will never be seen or animations that don't need to play at a distance.

Once LODs are prepared they must be named, grouped, and exported with the original – frequently labeled LOD – following the conventions dictated by the target software's requirements. Applications which have LOD systems – such as game engines – typically list the requirements for LOD setup in their documentation and those requirements should be followed precisely for best results.

Conventions

Conventions

3D objects are not the only candidates for optimization in production. Optimizing workflows involves following established conventions and using the tools that have been agreed upon. Conventions are patterns established to save time, prevent errors, and make future work and collaboration easier. In order for production to flow seamlessly, it is important that conventions are established, documented, and then followed so that previously completed work does not have to be searched through or edited later on in production.

An example of a commonly used convention in 3D production is file naming, such as rules for the prefixes, suffixes, characters, capitalization, and syntax to identify files, objects, or component's purpose – like LODs – or to control their order in alphabetically or numerically sorted lists. Conventions are often established for object data too, such as UV map sets, or Texture map types like the **`Pedestal_BaseColor`** and **`Pedestal_Normal`** maps we baked in *Chapter 7, 3D Surfaces*, as well as Materials, UV sets, and even Armatures and their bones.

Other workflow optimizations

Another workflow optimization is keeping versions of all the software used on a project consistent between files and among team members to avoid surprise file conflicts. Following an established project organization structure ensures that it is

clear where to place and find files too. Even naming objects and data inside 3D files in the same way that 2D artists often name layers – even if the file is not meant for others to see – and structuring the contents of 3D files clearly, using collections and categorization so they're easy to navigate in the future or by other team members, are important workflow optimizations.

In general, following the best practices we have explored earlier in the chapter, including budgeting, prioritizing, employing modularity, making 3D objects reusable, and sticking with the project's established units of measure is also key to optimal workflows.

Preparing 3D objects for production

As we have seen and practiced, preparing 3D objects for production begins with investigating the requirements, devising a plan for production, and employing thoughtful design throughout. Once we have finished creating and optimizing 3D objects in Blender, there are a few more steps involved to incorporate them into external applications.

Note: Some external applications such as Unity and Sketchfab allow for the

Note: Some external applications such as Unity and Sketchfab allow for the import of entire BLEND files, which is very convenient but may not be optimal for performance of the end product, since BLEND files contain many more objects and much more data by default than is usually needed or desired in a game or interactive application.

Importing and exporting

While Blender is a fully featured 3D program that can be used at every stage of production for many products like renders and video, many 3D pipelines also involve other applications besides Blender. Moving 3D objects from Blender to another piece of software, or even bringing 3D objects created in other software into Blender, involves importing and exporting. More than just saving a selected object as a particular file type, the processes of exporting 3D for importing into other applications present us with multiple options for how that 3D object will behave outside of the source application. Therefore, it is important to learn as much as possible about the requirements of any target software before exporting.

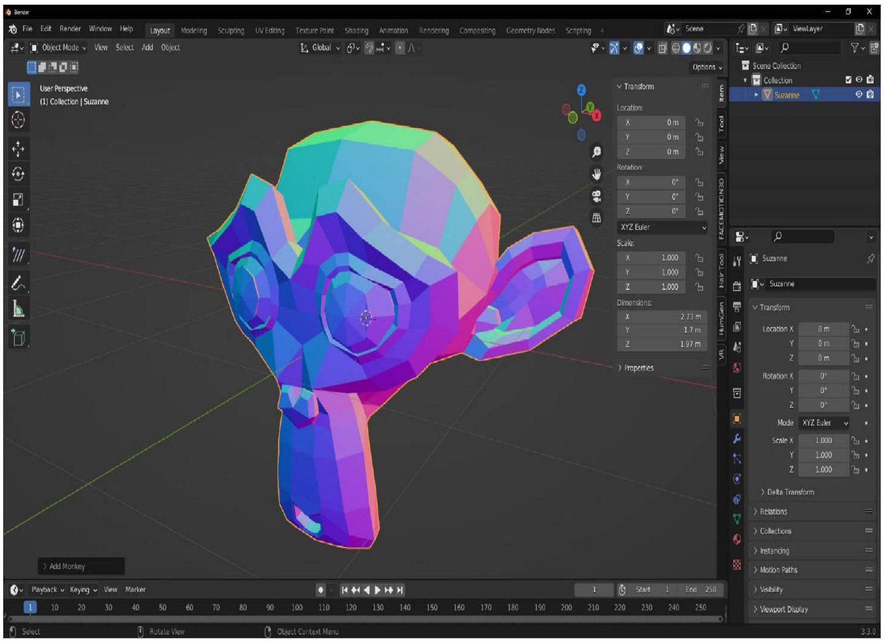
One major consideration for transferring objects from any 3D application to another is the base units for transforms. Mismatched units between two applications can cause endless confusion. For example, we might export an object that is 2 meters high and wide in Blender and import that into another application in which it appears as 2 centimeters or 200 meters instead. These mistakes can be difficult to troubleshoot because an improperly scaled object may be so small that it is not visible, or so large that the viewport renders inside it, also making it invisible. An object exported with

different units from the target software may import at the correct scale, but come in rotated under the ground, or its origin may off, placing it miles away in 3D space.

Separate 3D applications do not keep track of the base transform units employed by every possible source software, and they cannot, because 3D creators also have infinite flexibility to alter individual 3D objects relative to the source application, relative to the target application, and even relative to other 3D objects in both. Complicating things even further, unapplied transforms in the source software may not carry over to the target software, so that even when the correct export and import settings are used, imported objects may not be in the expected location, at the expected scale, or facing the expected direction.

Tip: To avoid endless confusion and hours spent troubleshooting, remember to always Apply transforms via Ctrl + A in Object Mode before export from Blender.

The best practice is to stick to one base unit of measure in both applications, for any single project. This means making any new object in the same scale, using the same rotation for facing forward, and keeping the origins of all objects at the center of the world. Then determine the appropriate export and import settings to maintain those units between applications, either from documentation, or through testing both export and import, using an object of known dimensions with a distinct rotation and origin, such as the Monkey head Suzanne as shown in *Figure 11.4*. Once they are confirmed, save the correct export and import settings as presets, making any future exports for the project error-free. Please refer to the following figure:



Transform

Location:

X 0 m
Y 0 m
Z 0 m

Rotation:

X 0°
Y 0°
Z 0°

XYZ Euler

Scale:

X 1.000
Y 1.000
Z 1.000

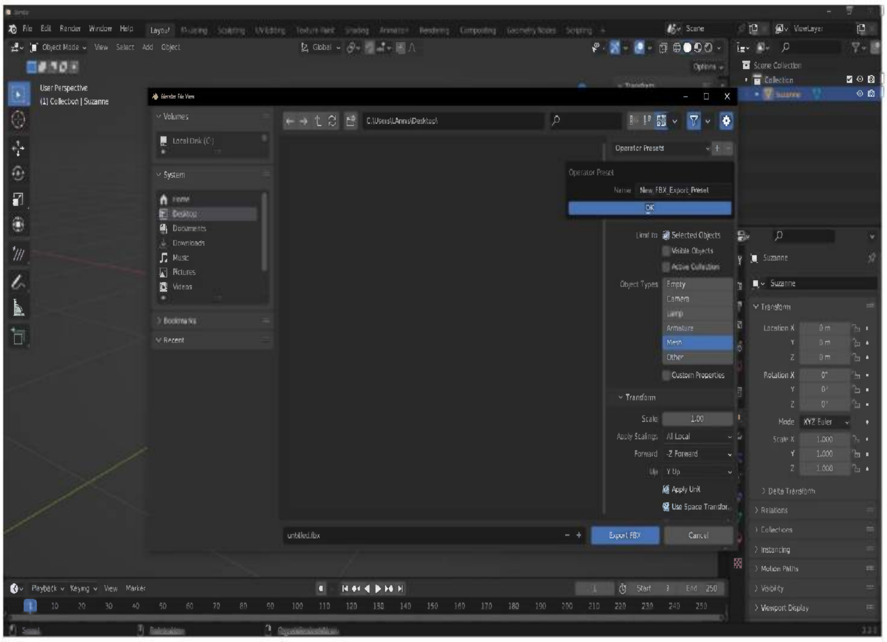
Dimensions:

X 2.73 m
Y 1.7 m
Z 1.97 m

Figure 11.4: Suzanne model shown with dimensions

In Blender, exporting begins with selecting the objects to be exported via the 3D Viewport or Outliner, otherwise nothing will be exported or everything in the scene will be exported together. Next (assuming we have applied transforms as needed), we navigate to the upper-most menu on the left-hand side of the application, and under **File** | **Export**, choose the desired file format such as FBX or OBJ. Supported file formats are dictated by the target software and can usually be found in the application's documentation. Blender can export to many popular formats, but if a particular format is not available for export by default, chances are it can be enabled via the **Edit** | **Preferences** | **Add-ons menu**.

Every file type presents a unique set of export setting sections, specific to the target file format. Export settings for the FBX file format, for example, contain the sections **Include**, **Transform**, **Geometry**, **Armature**, and **Bake Animation**, as shown in *Figure 11.5*. Each of the available sections of any file type's export settings contains parameters which control features of the resulting exported file like which of the available objects is to be exported, how to handle transform units, or whether to retain information about normals. Please refer to the following figure:



C:\Users\LAnna\Desktop\

Operator Presets

Operator Preset

Name: New_FBX_Export_Preset

OK

Limit to: Selected Objects

Visible Objects

Active Collection

Object Types: Empty, Camera, Lamp, Armature, Mesh, Other

Custom Properties

Transform

Scale: 1.00

Apply Scalings: All Local

Forward: -Z Forward

Up: Y Up

Apply Unit

Use Space Transform

untitled.fbx

Figure 11.5: Navigating the Blender File View to export Selected Objects to FBX format and save an export Preset

In some cases, the default export settings are just fine, but often for the sake of optimization should be changed to prevent the inclusion of unnecessary objects and data which bloat the size of the exported file by deselecting the **Empty**, **Camera**, and **Lamp** options for example. Saving the preferred export settings as an **Operator Preset** via the **Blender File View** as shown in *Figure 11.5* can save a lot of time as well as prevent many mistakes in lengthy productions.

Finally, objects to be exported must be named. Separate objects selected in Blender and exported together as the same file may appear in external software as a parent object with several child objects, which is sometimes desired and may even be necessary for external programs such as game engines where parts of an object need to stay related but able to be reconfigured. Please refer to the following figure:

Note: Keep in mind that actions working on objects nested in this way may impact performance, such as when a script locates a child object and disables its visibility. So, if this hierarchical structure is not necessary to the function of an object, it may be better to join objects in Blender first.

The first time an object is exported from Blender the name "untitled" will be suggested by default, but this name should be updated to avoid overwriting other files of the same name in the target folder.

Note: Blender can be scripted, and add-ons exist, to enable bulk export to various file formats with the automatic naming of separate objects according

to their Outliner names. However, since managing file versions and preventing accidental overwriting of files with the same name is a complex issue, we will advise investigating version control, discussed later in this chapter, before deciding on the most appropriate approach to use per project.

Pipelines and tools

In 3D production, the term "pipeline" refers to all of the tools and processes used to create the final product. One or more pipelines for one project, such as graphics and coding, may be kept separate or "siloed" in an effort to keep them optimized, but this practice is the subject of much debate in a fascinating field called **Dev Ops** (a combination of **Development and Operations**). Most 3D pipelines work a bit like an assembly line, with specific logical steps taken in sequence. In 3D production, 3D software like Blender is often used early in the pipeline, and when one component of the final product is completed there, it is moved along to another stage of production.

Development of a 3D game, for example might involve level (or scene) design and reference gathering from which 2D concepts are drawn and/or 3D grey boxes are assembled. Next, asset lists may be made from the finalized concept, from which art requirements are defined and tasks assigned such as 3D modeling, texturing, and animation. 3D objects would then be created based on those requirements, which then move along to a target software such as a game engine. Sometimes animations are also recorded during or after this stage. Once imported into a game engine, an art pipeline may continue with setup of 3D objects and their associated materials, textures, animations, and other components such as code to allow interactivity, and end with the arrangement of the assets into the previously designed level.

Project management

Managing production pipelines is a very important task and is sometimes done by a single person assigned to that role. Other times managing production is done by many people assigned to manage specific portions of a pipeline or different pipelines entirely. On a small team, production management may be handled directly by the same people who perform the tasks. It is also possible to produce 3D content without any structure or plan. This is more feasible on solo projects, but it is very difficult to keep a project moving forward without at least an outline of plans for the beginning, middle and end. Without a plan it is very easy to waste time and lose momentum.

Large teams often use software specifically designed for project management, and every team has unique needs in terms of the functionality such software provides. A quick internet search will uncover tens if not hundreds of project management tools, each boasting unique productivity features. While learning how project management software is generally used can help anyone become a valuable member of a team, it is usually necessary to gain a deeper understanding of just one tool that is approved for use on a specific project. Even if we never have to determine requirements or estimate tasks for other people to perform, it is still very helpful to understand how to interact with any form of project management that might be in place.

Version control

Almost equally as important as project management, version control (also called source control) refers to software that backs up work on a production and keeps track of changes so that mistakes can be reverted if needed. Many options for version control exist, but as 3D creators it is important to know that 3D assets (and their associated files like textures), referred to as "binaries," are often the largest files in a project. Unlike code, binaries are not easily compared for differences or branched and merged.

Like project management, the applications used for version control all have some high-level similarities, but many peculiarities unique to specific pipelines. In some cases, 3D creators on production are expected to make regular backups of their work, locally or on a shared device, in lieu of version control. On projects where there is code, such as with games and interactive application development though, version control is often used for both.

Since most version control solutions are made for backing up code rather than art, version control is often set up by programmers on a team and can be quite daunting for 3D creators to interact with. It is often helpful to follow an experienced programmer through submitting changes to a project with version control and take notes for future reference of the necessary commands and proper sequence of steps.

Documentation

The last of the production tools we will discuss is documentation. It may seem strange to describe documentation as a tool, but for 3D production, documentation is an absolutely critical one. Documentation is how all of the previous concepts we discussed are communicated, remembered, and followed. Documentation in 3D production can take on many forms, depending on the type of project.

The development of games, for example, is sometimes guided by one overarching document containing everything from a synopsis of the game play, the genre, art style, and target audience, to the core mechanics. This is typically called the **GDD** (for **Game Design Document**.) Other forms of documentation for 3D productions include the Wiki format – which is like a digital book with chapters – flowcharts, slide decks, wireframes, and collages. Entire services exist to host and organize all the assorted documents of many types of production.

A compelling case can be made, however, for drastically simplifying and limiting the number and size of production documents. As critical as documentation is as a roadmap and communication tool, documents can also easily become too lengthy and numerous for any one person to follow. Instead of recording every element of production through text and images, some say it is wiser to stick to one-page documents that are easy to grasp, and simply update them on a regular basis as needed.

Complementary tools

Throughout this chapter and previous others, we have referenced external software and hardware that might complement the production of 3D content in a more general way. Without promoting one tool or suite of tools over the others, we'll wrap up this section by highlighting the categories of third-party tools that are most useful to the production of 3D content with Blender.

As we learned early on, every industry is making use of the engaging, immersive power of 3D, but some primary destinations for 3D creations made with Blender rise to the top. Game engines may be the most universal of those. Not just for making games, but also for educational applications, architectural visualizations, configurators, and cinematics; game engines allow interactivity with 3D creations and make it faster and easier to publish 3D content by providing a set of tools so that people who want to make interactive 3D content don't have to code entire applications from scratch.

A number of versatile game engines are available to download on the web for free, and most only charge a fee if and when your game product makes many thousands of dollars. Evaluating whether to use one game engine or another involves identifying a project's needs and then comparing the available options. Some game engines are

better suited to innovative game play while others excel at one specific style. Some game engines are more non-coder friendly, but many have devoted and responsive communities to answer questions.

Other 3D tools that complement Blender are applications for remeshing and retopology, high-resolution sculpting, file format converters, photogrammetry tools, and 3D viewers. Hardware for the production of 3D content is a category all its own, including render farms, 3D printers, and 3D cutting tools, called additive and subtractive manufacturing, respectively. While Blender is capable of performing many of the same tasks, external tools can often extend creations made in Blender or make them accessible to a new audience.

[illegible]

The second primary category of tools that complement 3D content are 2D tools, like image and video editing and compositing software. Creating 3D from vectors or graphics from 3D (renders, curves, or some combination) makes editing illustrations, icons, and UI elements easier, when changing perspective is just a matter of moving a 3D camera. Tools for layout and markup, like whiteboards and collage tools are invaluable at the concept stage. Other categories of complementary 2D tools for 3D include software for advanced texture and material creation, which enable everything from procedural generation to photogrammetry-based images, as well as the automation of natural-looking wear and weathering.

Perhaps the last most important category of complementary tools for 3D job seekers is online portfolios, social media sites, and link aggregators. While cheap and easy options do exist for building personal websites, without a specialty in designing and marketing them, making personal websites stand out among the billions of existing websites is so difficult that even the free ones are not often worth the time and effort to build and maintain. Websites and web applications that are known for the specific type of content that we , as 3D creators, specialize in are much more valuable tools for networking, finding clients, and advertising the services we want to offer.

Next steps

As we wrap up this book's final chapter, the Blender Foundation has just released Blender version 3.4, introducing lots of great quality of life improvements like performance boosts to sculpting, painting, and UV editing, as well as a few changes to advanced functionality. As always, we can check out the Release Notes page at **https://www.blender.org/download/releases** for details.

We haven't even touched on so many advanced features of Blender making waves in 3D like Geometry Nodes and AI plugins, or the exciting advancements still on the horizon like a non-destructive, node-based texturing system. So, what's next?

Finding a niche

While 3D generalists are valuable in every industry, it can often be more fulfilling and profitable to specialize in at least one area of focus. Finding a compelling niche to explore can be tough. As we discovered in *Chapter 1, Features of Blender 3D*, every industry is using 3D, from entertainment to serious games; film, television, and marketing; science, medicine, and manufacturing; education, and training. Within each of those industries, a number of sub-specialties also exist, including character modeling, environment and prop modeling, surface design, animation, lighting, graphic design, UI, product display, visualization, clothing design, manufacturing, even instruction.

While considering various industries and specialties to explore, it is also important to evaluate the actual work involved, whether it is full-time employment, part-time, a contract or commission, consulting, or even a side gig or hobby. Talk to people actually working in a niche for some perspective.

Whenever there is a potential customer in the equation, which is generally how money is made, work in 3D is never just sitting at a desk and creating from our imaginations. For freelancers, administrative tasks like bookkeeping, licensing and other legal concerns, billing, taxes, contracts, and the tracking of time, due dates and deliverables are often part of the bargain. Interruptions, repetition, lack of privacy, corporate regulations, and personality clashes are all potential factors of corporate work. An unfortunate reality of modern work is that neither choice offers true financial stability or security.

Another viable option is selling 3D creations on stock asset markets, of which there are currently hundreds. Starting out, ala carte asset sales may not be dependable enough to live off of, but most 3D creators have assets lying around unused that could be just the thing someone else needs for their project. Evaluate stock asset marketplaces by finding out how much of a percentage they keep of sales and find out what the process is for bundling, uploading, and posting an asset for sale. Some do all the work for creators who only have to upload, while some require hours of preparation like labeling and custom graphics. Finally, it is smart to make sure that the markets we choose are where buyers search for the type of 3D assets we create.

Creative freedom versus guidance, repetition versus learning opportunities, consistency versus excitement, flexibility and accommodations versus regulations and restrictions, interpersonal dynamics versus isolation, interruptions versus isolation: All of these qualities come in different amounts with each of the available modes of work, and the experience of these as positive or negative varies from person to person. Not to be cliché, but there is no right answer because the pros and cons of each choice are highly subjective and based on our personal needs.

The good news is that while previous generations expected to pick a career for a lifetime, work in technology like 3D is constantly evolving. Projects span months to years these days, more often than decades. Plus, everyone is new to the latest developments, so no one is ever “behind,” and a bad career move is no longer a life sentence.

Conclusion

Hopefully, this text has provided a solid foundation from which to further explore the infinite world of creating in 3D. Refer back as needed for that "one technique" or a shortcut that was helpful, but also don't be afraid to move on from here. 3D is progressing faster than the speed of light, and the developers of Blender – 3D creators just like us – are not just striving to keep up but also leading the way.

As 3D creators, we benefit from finding and joining communities, patronizing content creators, bookmarking websites, and subscribing to channels that provide guidance and education in our areas of greatest curiosity and inspiration. Good luck to us all, and we'll see you there!

Essential shortcuts and hotkeys

In this chapter, we used the following shortcuts and hotkeys:

Shortcut	Hotkey
Mouse Select	Left-click
Deselect	Left mouse click away from the 3D object
Cancel Action	Right mouse click
Vertex Select (in Edit Mode)	1
Edge Select (in Edit Mode)	2
Face Select (in Edit Mode)	3
Clear Selection	Left mouse click away from 3D objects
Select All	A
Box Select	B
Circle Select	C
Lasso Select	Ctrl + Right mouse click and drag
Undo	Ctrl + Z
Redo	Ctrl + Shift + Z
Delete	X or Delete
Escape	Esc
Scale	S

Shortcut	Hotkey
Rotate	R
Grab (change position)	G
Snap Movement	Ctrl + M
Snap Rotation	Ctrl + R
Snap Scale	Ctrl + S
Show / Hide Side Bar	N

Show / Hide Side Bar	N
Show / Hide Tool Bar	T
Toggle Edit Mode and Object Mode	Tab
Toggle Armature Edit Mode and Pose Mode	Ctrl + Tab
Extrude (in Edit Mode)	E
Edge Menu (in Edit Mode)	Ctrl + E
Repeat Previous	Shift + R
Add (Objects) Menu	Shift + A
Adjust Last Operation	F9
Select Edge Loop (in Edit Mode)	Press and hold A, then left-click an edge in the loop
Select Linked (in Edit Mode)	Press L
Select Similar Menu	Shift + G
Revert all Movement (in Object Mode)	Alt + G
Revert all Rotation (in Object Mode)	Alt + R
Revert all Scale (in Object Mode)	Alt + S
Normals Menu (in Edit Mode)	Alt + N
Join Selected Objects	Ctrl + J
Hide Selected	H
Unhide All	Alt + H
Loop Cut (in Edit Mode)	Ctrl + R
Loop Slide (in Edit Mode)	G + G
Bevel Selected (in Edit Mode)	Ctrl + B
Duplicate Selected	Shift + D
Fill Selected (in Edit Mode)	F
Set Active Object as Camera	Ctrl + Number pad + 0
(Look through) Active Camera	Number pad 0

***Table 11.1**: Essential Shortcuts and Hotkeys*

Points to remember

- Budgeting resources well is a highly valued skill in 3D production.
- Optimization is important for 3D and workflows alike.
- The field of 3D is vast and expanding.
- Blender is developed by its users, making it uniquely poised to keep up the pace.

Questions

1. What is an example of 3D production?
2. What are three useful tools for production?
3. What are some pros and cons of employment and freelancing?

Index

Blender 3D for Jobseekers

DESCRIPTION

Learning how to create in 3D is a daunting and lengthy process, no matter which software is used. If you are a beginner or an aspiring 3D creator who wants to get familiar with the vast capabilities of Blender 3D, then this book is for you.

Beginning with an overview of Blender's capabilities and immediately launching into the installation and navigation of Blender's interface, this book will help you become comfortable with thinking and working in the 3D space. Next, core concepts are de-mystified, clarifying the difference between polygonal modeling and sculpting, and when to choose one approach over the other. Once you are comfortable with creating 3D models, this book will teach you how to create and manipulate 3D objects, scenes, and experiences.

By the end of the book, you will be prepared to begin fulfilling creative work making products that are in high demand in the vast, lucrative market of 3D.

KEY FEATURES

- Understand the capabilities of Blender 3D and how to get started.
- Get familiar with the fundamentals of 3D creation, from modelling to production.
- Learn how to use Blender professionally to become a sought-after 3D creator.

WHAT YOU WILL LEARN

- The features, installation, and navigation of Blender 3D.
- Understand core 3D concepts like poly modeling and sculpting.
- How to work with textures, materials, and shaders in 3D.
- An introduction to animation, effects and simulations.
- How to render images and video of 3D creations.
- How to use Blender for professional 3D work.

WHO THIS BOOK IS FOR

This book is for beginners and experienced 3D professionals who want to use Blender 3D for modeling, animating, and rendering their models.

BPB PUBLICATIONS
www.bpbonline.com

ISBN 978-93-5551-811-8
9 789355 518118