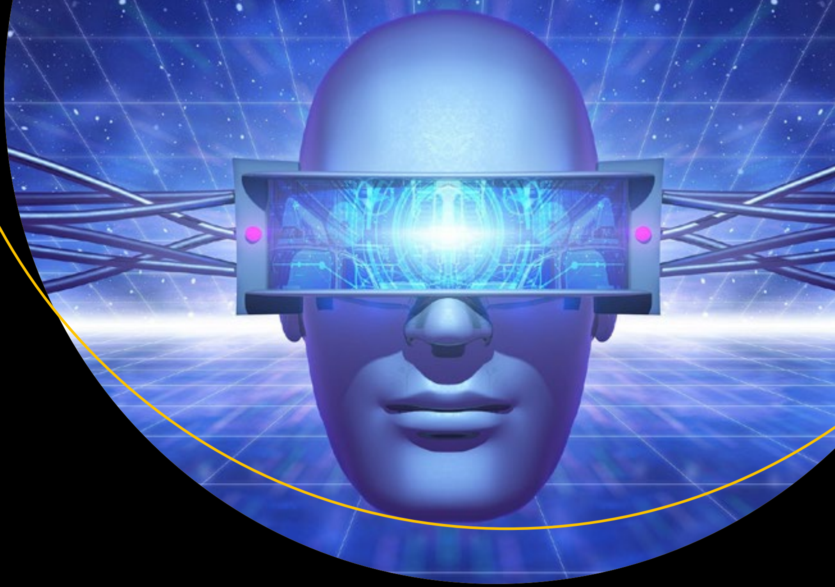


VR Integrated Heritage Recreation

Using Blender and Unreal Engine 4

Abhishek Kumar

Apress®

VR Integrated Heritage Recreation

Using Blender and Unreal Engine 4

Abhishek Kumar

Apress®

VR Integrated Heritage Recreation

Abhishek Kumar
Varanasi, Uttar Pradesh, India

ISBN-13 (pbk): 978-1-4842-6076-0 ISBN-13 (electronic): 978-1-4842-6077-7
https://doi.org/10.1007/978-1-4842-6077-7

Copyright © 2020 by Abhishek Kumar

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Spandana Chatterjee
Development Editor: Laura Berendson
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6076-0. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To mom, dad, Prof. B. K. Prasad, Usha Sinha, and my beloved wife, Alka, for 11 fantastic years of marriage and many more to come. To my daughter, Rishika Ryan, and my son, Shivay Singh Ryan. I love you all.

Table of Contents

About the Author

Dr. Abhishek Kumar is an Assistant professor at School of Computer Science and IT, JAIN (Deemed to be University) and former Assistant Professor at Banaras Hindu University, BHU. He is an Apple Certified Associate, an Adobe Education Trainer, and certified by Autodesk. He is actively involved in course development in animation and design engineering courses for various institutions and universities.

Dr. Kumar has published a number of research papers and covered a wide range of topics in various digital scientific areas (including image analysis, visual identity, graphics, digital photography, motion graphics, 3D animation, visual effects and Immersive technologies). He holds seven patents in the fields of design and IoT.

Dr. Kumar has completed professional studies related to animation, computer graphics, virtual reality, stereoscopy, filmmaking, visual effects, and photography from the Norwich University of Arts, the University of Edinburg, and Wizcraft MIME and FXPHD, Australia. He is passionate about the media and entertainment industry, and he has directed two animation short films.

Dr. Kumar has trained more than 50,000 students across the globe from 153 countries (including India, Germany, United States, Spain, and Australia). His alumni have worked for national and international movies, including *Ra-One, Krissh, Dhoom, Life of Pi,* the *Avengers* series, the *Iron Man* series, *GI Joe 3D, 300, Alvin and the Chipmunks, Prince of Persia, Titanic 3D,* the *Transformers* series, *Bahubali* 1 & 2, *London Has Fallen, Warcraft, Aquaman 3D, Alita,* and more.

About the Technical Reviewer

Bhuvnesh Kumar Varshney is an experienced senior artist with a demonstrated history of working in the VFX industry for the past 13 years. He is Computer Science graduate and having the specialization in 3D graphics, visualization, animation and has an interest in Python programming.

His key interest is Creature animation for VFX and gaming.

He has worked with the world's leading animation and visual effects studios as a team-lead and senior artist, including for Technicolor, DQ Entertainment, Maya Digital Studio, and Anibrain Studio.

Currently, he is working at Yash Raj Films as a senior creature animator.

He has worked on many blockbuster movies, TV series, and commercials, including *How to Train Your Dragon, Penguins of Madagascar, Barbie, Mickey & the Roadster Racers, Tell It to the Bees, Show Dog, Budweiser, Skoda, Norm of the North, War, Mardani-2,* and many more.

His projects have been awarded Oscar, Film Fare, and Zee Cine awards for best VFX.

He is currently working on the upcoming Bollywood movies titled *Prithviraj* and *Shamshera* from Yash Raj Films.

You can reach him via email at `bhuvnesh3danimator@gmail.com`.

Acknowledgments

It gives me immense pleasure to express my deep gratitude to my mentor, *Prof. Saket Kushwaha, Prof. Alok Kumar Rai* and my PhD supervisor, *Dr. Achintya Singhal*, Associate Professor, ***Banaras Hindu University***. Prof. Saket Kushwaha and Prof. Alok Kumar Rai has been the inspiration for this endeavor; without his encouragement, support, and guidance, this book would not have been possible.

I would also like to thank Bhuvnesh Varshney (TR), Spandana Chatterjee (AE), Divya Modi (CE), and Laura Berendson (DE) for the initiation to publish this book. Their helpful comments and suggestions resulted in numerous refinements and corrections that improved its quality. I am also grateful to iNurture and Prof. M N Nachappa, Head, School of Computer Science and IT, JAIN (Deemed to be University) for his encouragement and support.

A special thanks to *Dr. Vijayakumar Varadarajan*, professor at the University of New South Wales, Australia, and *Dr. Kalyana C. Veluvolu*, professor at Kyungpook National University, South Korea, for providing unconditional support.

Introduction

The aim of this book is to teach readers the fundamental principles needed to understand and create architectural pre-visualization of historical locations using digital tools. We will explore all the foundational aspects of 3D design visualization and VR integration using software that is actively used by the industry. We will use Blender to create VR-ready assets by modeling and unwrapping them. Then we will use Substance Painter to texture the assets that we created. We will also learn how to use the Quixel Megascans library to acquire and implement physically accurate materials to the scene. Then we will import all our assets into Unreal Engine 4 and re-create a VR integrated heritage that can be explored in real time.

A huge part of the world's history has been lost to wars, invasions, and natural degradation. Using VR technology and game engines, we can re-create digitally what has been lost. This can revolutionize the way we can preserve history digitally. Hopefully, after reading this book, you will be equipped to be a part of this heritage-preservation effort.

This book is suitable for beginners in the field of GeoICT (Geospatial Information and Communication Technology). Archaeologists, computer graphic engineers, game technology specialists, and so on, will all find the book extremely useful.

What you will learn:

- How to digitally preserve ruined historical locations.
- How to create high-quality optimized models suitable for any 3D game engine.

- How to texture assets using Substance Painter and Quixel Megascans and keep them historically accurate.
- How to use asset integration with game engines.
- How to create visualizations with Unreal Engine 4.

CHAPTER 1

Introduction

This book introduces the production pipeline you'll need to re-create destroyed or dilapidated historical monuments using 3D technology and then present them in VR (virtual reality) using a VR-integrated game technology. Throughout this book you will be presented with stages of the creation process, followed by a step-by-step guide on how to proceed with them. We go over each stage of the development process and cover the challenges you'll face and how to overcome them.

- If you are a beginner, this is a good place to learn some basics about the software packages and understand how to handle them.
- If you are an experienced artist, this book will teach you how to expand your skills and complete a practical project.

Developing and designing a complete project is hard and takes a lot of patience and dedication. Without a proper reference or guide, it is easy to get lost and give up. This book aims to document the important steps of the creation process so that you can understand the steps you must take to complete your projects.

This book is divided into multiple chapters, each covering a step of the development process in a systematic manner. It is very important that you go through this book sequentially and complete each chapter fully before proceeding. Each chapter will give you insight into the topic that it covers.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_1

You must translate the written instructions of the chapters into physical actions that you perform on your computer. It is easy to make mistakes during that process, so it is important to stay vigilant. If you do this correctly, you will have a good portfolio piece to show or personal work that you can admire. Knowing the full process of creating a large and complex asset from start to finish, and then integrating it into a game engine, opens lots of potential job opportunities for you. It not only shows that you have good planning skills but also that you can execute those plans well. I hope you are excited about the journey that you are going to take. Let's get started!

Re-Creating History Using Computer Graphics

History tells us who we are and how we became what we are today. It is important to understand the origins of our culture and remember those who were responsible for laying the foundations of modern society. Ancient art and architecture are the cultural symbols of our past and studying them reveals a lot about the lives and thought processes of the people who constructed them. Each monument, sculpture, and painting tells a story of that era. Not only that, they also indicate how sophisticated construction methods and advanced tools were used. It is our responsibility to preserve these priceless artifacts in whichever way possible.

A lot of our prestigious heritage currently lies in ruins. Once bustling cities are now abandoned and decaying. Many empires that once thrived have completely disappeared; only their echoes remain in the form of the relics they left behind. The main purpose of history reconstruction is to re-create these places and structures in the form they had in their glory days. Computer graphics can be used to reconstruct ruins that are dilapidated and cannot be reconstructed physically, sometimes because it's cost-prohibitive to do so. So, using modern modeling, texturing, and rendering techniques, we can bring these ancient monuments back to life, to their former glory. Game engines can enhance this undertaking by introducing new elements, like the ability to explore the constructed environment in real time. This not only increases the viewer's immersion into this artificially created world, but also allows them to see the monument from different perspectives and truly appreciate it. This could revolutionize the way in which we teach history and perhaps draw more people into studying it.

For the book's project, we will consider the monuments from one of the oldest cities in the world, Varanasi. It's the center of India's vibrant heritage. The multifaceted culture of Varanasi formed as a result of a long historical process. The intricacies of its arts and crafts, so essential to the upkeep of a stable economic system, are deeply rooted in the practical skills and technological know-how of yesteryear. In addition, the spiritual and religious ideologies inherent in this city have undergone successive developments to reach a height that makes India so special (see Figure 1-1).

Figure 1-1. *Old monuments of India*

To appreciate the rich heritage of any region or country, we need to be aware of its history. The history is weaved on the discovery and reconstruction of the remnants left by people of the past. These remnants, found in the form of buildings and objects, become the main source of heritage and history. It is imperative to communicate the history of these ancient remains to laypeople and children. A reliable means of imparting information about heritage, particularly of the tangible nature, is visual presentation, which is easy to grasp and interesting. Drawings and visual reconstruction of old structures by computer graphics is one of the recommended means. With the aid of computer graphics, it may be possible to re-create the original form and nature of such important, but dilapidated, monuments.

Virtual Reality (VR) Integration

VR is a very hot and trending subject in the digital industry. VR headsets are getting more and more affordable by the day and people find the immersiveness that they provide very appealing. The use of VR is not restricted to the entertainment industry; it's used heavily in architectural visualization, education, medical training, military simulation, and more. The architectural visualization industry (ArchViz) heavily uses VR technology to create highly immersive architectural walkthroughs for clients. This allows clients to explore scenes in depth, rather than simply seeing things on a flat screen.

You can download free ArchViz demos from `http://oneirosvr.com/download/`. This will give you a good idea why it is preferred and so much in demand

Another great application of VR is for medical education, or even just education in general. The education industry has been looking for a more fun and intuitive way of imparting knowledge onto students. VR provides something that a plain classroom setting cannot provide, education with a fun aspect. Watching a human anatomy class in 3D or seeing a VR walkthrough of the Pantheon in history class is exciting and memorable. Since the experience is so engaging, the student is much more likely to remember the information.

Another big reason that the educational field is slowly adopting VR is because it is more convenient than actually travelling to a different country or bringing in some complex or fragile thing to the classroom and experiencing it. Students can see or experience something that is far away or impossible to bring into the classroom using VR.

Military simulations use VR to train recruits in various kinds of scenarios. This is not limited to combat simulations, but includes various kinds of emergency situations. This can be something like a ship or a submarine that has suffered hull breach and is slowly sinking. Or it could simulate a combat aircraft attack. These are just some military applications of VR. This approach can also be extended into training other emergency responders. From simulating burning buildings to natural disasters, VR training simulations are used by many departments across the world.

VR use in video games is nothing new and is now widely known. After the release of *Half-Life: Alyx* by Valve software, VR gaming gained huge popularity and many more people understood its immense potential. There is now a huge demand for VR developers, as people are very much interested in playing highly immersive VR games. With the rapidly advancing technology, the day is not far off when VR games will have an audience base as large as traditional gaming media.

The future of VR is bright. In this day of rapidly evolving technology, it is important not to get left behind. The main purpose of this book is to give you all the information you need to kick start your journey into the field of VR architecture design and game development.

Topics Covered

This book is an introductory course aimed at aspiring artists who want to expand their horizons into the digital creative industry. The main purpose of this book is to lay out each stage of the development process for making a complete scene from scratch and explore these stages in detail. Each chapter covers a stage that falls into the production pipeline of asset development. From preparation to execution, you can expect full documentation of the steps that you must take to get to completion.

After brief introductions in Chapters 1 and 2, we come to Chapter 3, where we discuss where to gather the needed resources. Then in Chapter 4, we start with design visualization, where we decide what we are going to make and determine the plan of action.

In Chapter 5, we finish blocking out our asset, which is a very important step, as it allows you to move and modify the shapes without many problems. Next, we start detailing the model and go over the common modeling techniques used in the industry.

In Chapter 6, we proceed to unwrapping our models. Many people dread this process, but we will go over this in detail so you can understand how to efficiently unwrap your models. Despite being arduous in nature, unwrapping is an important step in the process of asset design, so everyone must understand it properly.

In Chapter 7, we will use the Substance Painter to see how we can create PBR materials for our assets. You'll learn how to texture them while finding the balance between realism and design. We will use external websites to get assets so you don't have to mix them using the power layer system of the Substance Painter. We will then convert this created material into "smart material," which we can easily reuse and repurpose for other meshes. That way, you don't have to create material from scratch every time you bring in a new model.

In Chapter 8, we learn how to create foliage assets for our project. We will use external tools to help us with this process, as the industry prefers specialty designs when creating foliage assets. We will use a tool called TreeIt to create trees and use Quixel Megascans to access photorealistic, 3D-scanned grass assets that have unmatched quality.

In Chapter 9, we export our assets from the respective tools to Unreal Engine 4. We cover the export settings and the proper formats for exchanging files. We also see how to create lightmap UVs, which are very important when using Unreal Engine 4. This is a special type of UV created in the 3D modeling application.

After exporting assets, Chapter 10 shows you how to import these assets into Unreal Engine 4. We learn how to manage and organize imported assets. We also learn how to use the Properties Editor window to modify our assets.

In Chapter 11, we see how to set up the materials for our imported assets. We learn to use the Materials Editor and understand how the node-based workflow of the material editor works. We see the use of commonly used nodes and learn the different types of material setup. We will, of course, start with a very simple material setup to understand the concepts. After that, we will create a complex material setup that allows us to make some modifications to our materials directly inside Unreal Engine 4. We will also work with master materials and instanced materials and see how each differs and learn when we should use each one.

In the final chapter, Chapter 12, we jump into C++ and see how we can program our game to work in virtual reality. A VR headset is recommended to follow along in this chapter, but even if you don't have one, you can still learn the basic concepts and see the programming side of the process. It is important to know at least the basics of programming so that you can understand how C++ works and how it is used to develop games.

CHAPTER 2

Introduction to the Software

Let's start by taking a quick look at the software that we are going to use throughout this book. We will explore each product in more detail as we proceed through the creation process. In this chapter we will cover the basic functions of each software product and discuss where in the creation process they are used. We will also see what their initial interface looks like. In further chapters, we will explore their basic features.

Blender

Blender is an open source 3D-creation suite that is a complete modeling, unwrapping, texturing, rigging, and animation package absolutely free of cost. You can acquire it by simply going to `blender.org` and downloading it. Learning Blender is also quite easy because there is a huge number of tutorials available on the Internet. Blender is used widely by Indie film and Indie game developers, not only because it's free, but also because it has all the features required for creating high-quality assets. The basic concepts of digital art apply universally to all content-creation tools, so if you learn one, you can use the same methods in other applications as well. Figure 2-1 shows Blender's interface when you first start it up.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_2

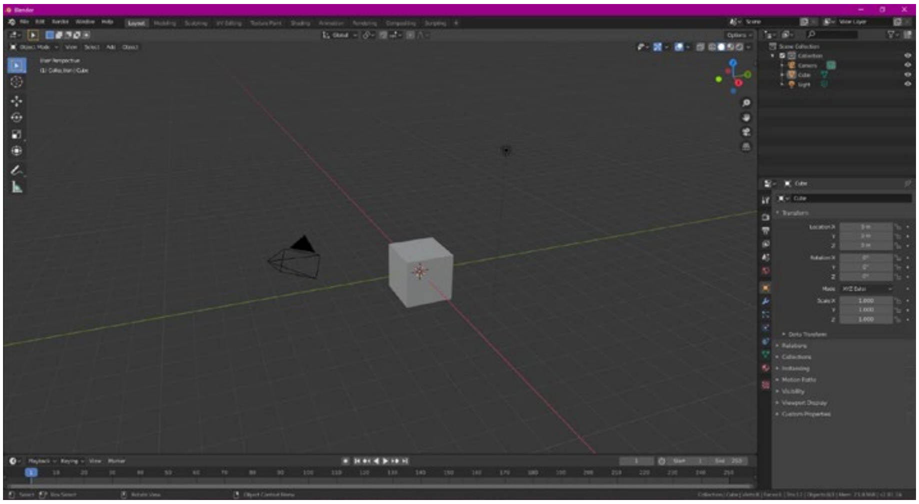


Figure 2-1. *Blender's basic interface*

Substance Painter

Substance Painter is the industry standard tool for texturing assets. It has a highly flexible procedural workflow and easy-to-use tools that make texturing anything simple. Substance Painter is used by all kinds of studios, both AAA (Triple-A) and Indie, in all kinds of fields, including game design, film production, product design, and architectural visualization. It is very important to know how to use this tool if you're interested in working for any production house. Figure 2-2 shows Substance Painter's interface when you first start it up.

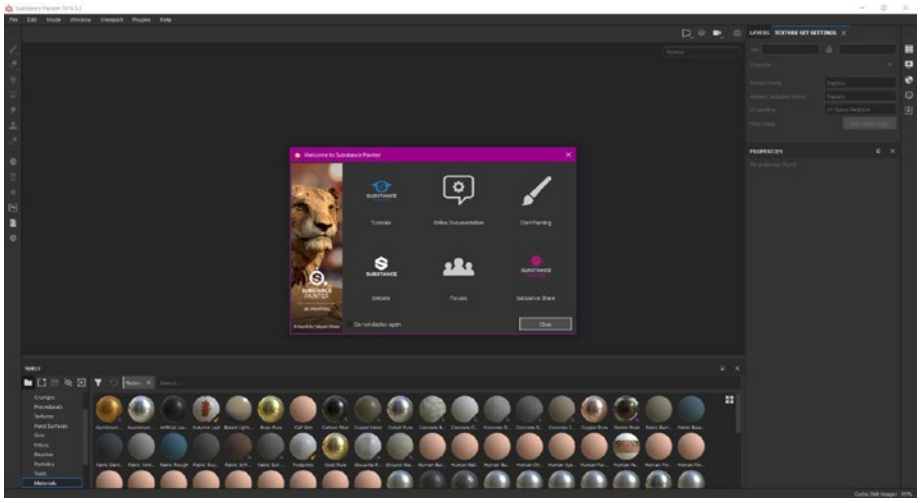


Figure 2-2. Substance Painter's interface

Quixel Megascans and Bridge

Quixel Megascans is the largest scanned library in the market. These scans include materials and 3D models that have been directly scanned from real-world objects and places, which means they are extremely realistic. The materials are physically accurate and the models are highly detailed, and best thing is that it is free to use for Unreal Engine 4 users. Megascans is used by AAA studios and production houses for both films and games, as nothing is more realistic than scanned real-world objects. Best of all, you can access all of these absolutely free of cost. Figure 2-3 shows the Quixel Bridge interface when you first start it up.

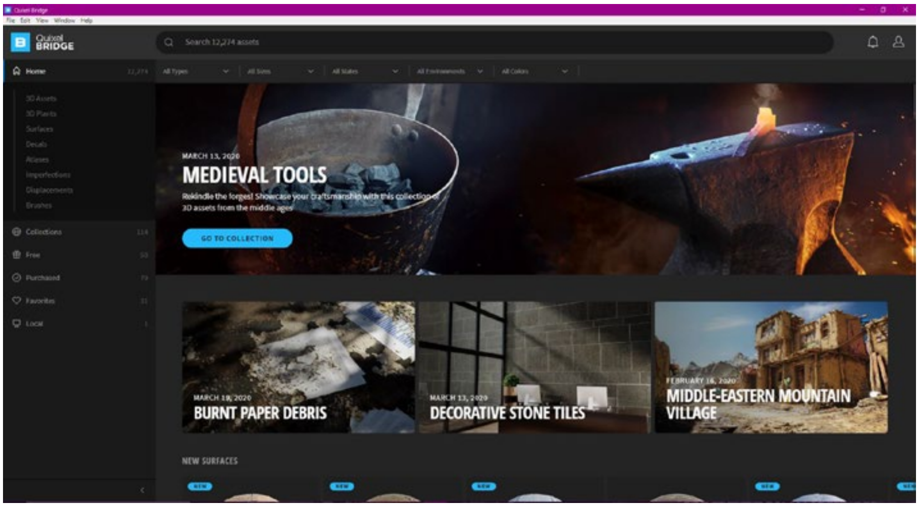


Figure 2-3. *Quixel Bridge's interface*

Unreal Engine 4

Unreal Engine 4 (UE4) is a game engine that allows you to build games using assets and codes and deploy them on any platform. UE4 is an industry standard game engine as well. AAA developers use Unreal Engine to create modern games. If you are interested in working in the game industry, a portfolio in Unreal Engine 4 is a must. Figure 2-4 shows Unreal Engine 4's interface when you first start it up.

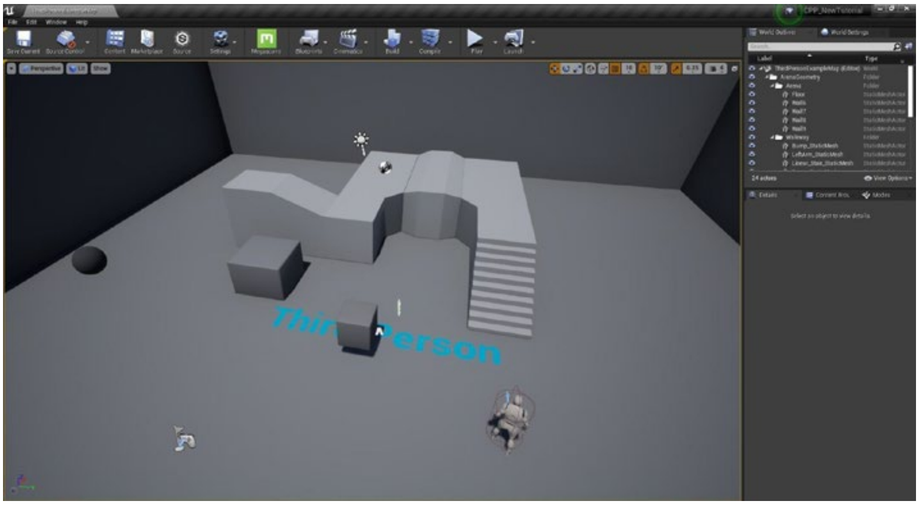


Figure 2-4. *Unreal Engine 4's interface*

This was a short introduction about the software that we will be using for the project in this book. I suggest you visit YouTube and watch some basics on how to use all these programs. In the next chapter, we will explore how to acquire the various resources needed for this project.

CHAPTER 3

Acquiring Resources for the Project

In this chapter, we learn how to acquire some essential resources for the book's project. This project requires a lot of textures and 3D models, not to mention all the reference images that you'll need for proper guidance. You'll need some extra textures and assets that have already been created. This is an optional step and you can certainly create everything manually from scratch. But you may find that you need extra help when it comes to projects like this one. Especially when it comes to textures and materials. Not everyone has the resources to go to distant places and capture textures using an expensive camera. That's where the websites mentioned in this chapter come into play.

Sources for Textures

You will need a lot of textures and materials for this project and there are a lot of ways that you can get them. For one, you can create some yourself by taking photos and editing them in a program like Substance Alchemist. This is a time-consuming process and usually not recommended unless you want something very specific that isn't available anywhere else. Alternatively, you can go to online texture libraries, which provide textures and materials for purchase (some of them are even free). The paid ones are usually better than the free ones, for many reasons.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_3

Check out these free texture libraries:

- `www.texturehaven.com`
- `www.cgbookcase.com`
- `www.texture.ninja`
- `www.cc0textures.com`

Check out these premium texture libraries:

- `www.gametextures.com`
- `www.quixel.com/megascans` (free for use with Unreal Engine 4)
- `www.poliigon.com`
- `www.textures.com` (freemium/lower resolution textures available for free)
- `source.substance3d.com`

Access to Substance Source comes with a Substance Suite subscription. So, if you have a Substance Painter subscription, you won't need to purchase additional resources. Also, Quixel Megascans is free to use with Unreal Engine 4 (which is our engine of choice). That means we won't need to spend any extra money on textures. However, if there is something that you can't find in your library of choice, you can certainly explore the other ones.

Sources for 3D Models

After you have created all the main assets you need, you might find that you need some extra stuff to populate your scene. In that case, you can get models created by other artists, some for free and some for a price. It is totally okay to use additional assets to polish your scene or add extra details after you have put in 90% of the effort by yourself. Alternatively, you can even sell your creations on these websites or perhaps distribute them for free if you want to. The following websites offer 3D models for download, either paid or for free.

These websites offer 3D models for free:

- `www.clara.io`
- `www.archive3d.net`
- `www.free3d.com`
- `www.grabcad.com`

These websites offer premium 3D models for purchase and provide some free assets for download as well:

- `www.turbosquid.com`
- `www.cgtrader.com`
- `www.sketchfab.com`
- `www.cubebrush.com`

One final note about acquired assets: Remember that if you are using assets that you acquired, be sure to give credit to the creators by mentioning their name along with a web link to the download. Never claim work done by others to be yours! In the next chapter we are going to start blocking out our scene.

Collecting References

Let's now work on what we are going to create. It is very important to plan the steps before proceeding to the actual creation process. We need lots of information about what we are going to create as well as reference images related to it. Gathering references is an essential part of the pipeline, as we need a very solid idea of every detail that must go into the structure or scene that we are re-creating in order to make it historically accurate. Reference images can be captured manually by going to the site and clicking on high-resolution photos or can be gathered online from various websites.

The following websites provide images for free:

- `www.pixabay.com`
- `www.pexels.com`
- `www.unsplash.com`
- `www.freepik.com`

To gather references, you can use Google Images. There are also some alternative search engines that search for images with Creative Commons (CC) licenses. You can use a search engine like `https://ccsearch.creativecommons.org/` to search for CC licensed images. But again, always remember to credit the authors.

Let's begin by searching for "Ancient India" in the websites mentioned previously. We will use the reference images to better understand the architecture style of the era and find the key features present in them. We need to see what kind of details were added, what materials were used for construction, what colors were used for painting, etc. Download as many images as you want, as you need as much information as you can get. You also need to see the structures from as many angles as possible. Once you have everything, store them all in a single folder so they are easy to find.

One good way of working with references is to use a program called PureRef. You can download it from `www.pureref.com` and install it on your PC. Once the installation is done, just drag and drop your references into the canvas of PureRef. PureRef will automatically create the best collage of the provided images on its canvas. You can adjust the images manually as well. The best thing is that this software is completely free.

Creating a Basic Plan

We will now decide what our scene is going to be. First of all, we will look at our reference images to determine what is clearly visible and can be re-created easily. For the purposes of this book, we will create a small place of worship and surroundings, reminiscent of the ancient times. The goal is to re-create a site that has been completely lost and reconstruct it based on the information that remains. This method can be used to reconstruct ancient cities, villages, religious temples, and other historically important places. According to this plan, the PureRef canvas should look something like Figure 3-1.

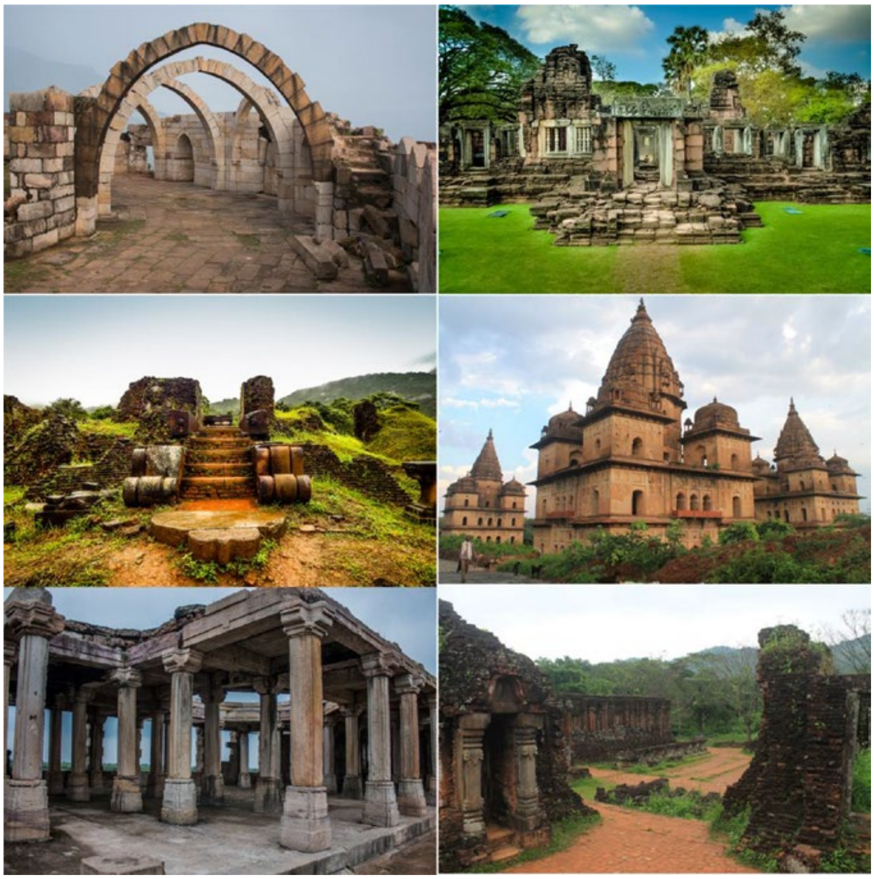
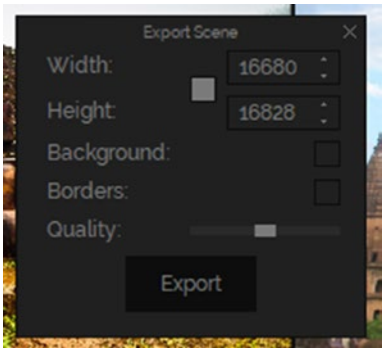


Figure 3-1. *References arranged inside PureRef*

Now, as you can see, there is a lot to digest in the reference images. But we are not going to make every single thing that we see; rather, we will pick parts of details from each image and combine them to form one cohesive structure. The rest just gives us the idea of the architecture.

Once you are happy with your reference board, you can export it as a very high definition image based on the resolution of the images imported. You can then use this image as your reference without needing any software to open it. To do this, first right-click on the canvas and then choose go to Canvas ➤ Optimize. Alternatively, you can use the shortcut Ctrl+O to do the same thing. This will crop your canvas around your images and remove the extra regions. After that, right-click again and go to Save ➤ Export ➤ Scene, or press Ctrl+E. This will open a new export configuration window, as shown in Figure 3-2.


Export Scene

Width: 16680

Height: 16828

Background:

Borders:

Quality:

Figure 3-2. *Export configuration window*

The resolution of the image being exported can be very high depending on the number of imported images and their resolution, and you can also manually enter resolution values as well. Once you are happy with the settings, click on Export.

We have our reference images ready, so we can proceed with the next step. A simple layout drawing showing where everything is supposed to be will be a great reference when creating the scene in 3D. Such a drawing also helps you identify what all needs to be made for the scene. This drawing will be a top-down layout of what we plan to achieve. This layout can does not need to be too detailed. It is merely for determining whether the layout you planned looks interesting and works okay.

The layout drawing can be very simple and should guide you when you're working on the scene. This drawing should be based on the ideas derived from the reference images. When creating real locations that have a very specific layout, a plan has to be made according to that. Maintaining historical accuracy is the most important thing.

Now that you have planned the scene, we can proceed to creating something called `White-Box Scene`. Basically, we will create very simple geometric shapes and send them to UE4 and then create a basic scene with them. This will be done to get a reference for scale as well as for testing how our layout looks. For this, we will use Blender to model the assets and Unreal Engine 4 to create the terrain and set up the scene.

Let's first create a list of assets that we need. Again, these will be very basic models whose only purpose is to help you prototype the scene. Also, because we will only be prototyping, we will keep the number of assets to a minimum. The following are the models that we need for this white-box scene:

- Main temple model
- Temple's additional sub-structures
- Rocks (four variations)
- Tree model (two variations)

In the next chapter we will see how we can block out the scene using all the data that we have collected.

CHAPTER 4

Design Visualization

In this chapter, we are going to learn the basics of the software we will use for heritage re-creation. After that, we will create a white box level using Blender and UE4. We will also learn some important hotkeys that everyone must know when working with this software.

Basics of Blender

Let's start with the basics of Blender. It is important to get a good grasp of modeling before you can continue making the advanced stuff. Blender is a shortcut-based software program and you need to memorize lots of shortcut keys if you want maximum efficiency in Blender.

Let's start with the basic navigation controls. Click and hold the middle mouse button to rotate your viewport. Hold Shift and the middle mouse button to pan your view. Use the middle mouse scroll to zoom in or out. Alternatively, you can hold the Ctrl key and the middle mouse button to zoom in and out more smoothly.

Currently we are in Perspective view. You can change view mode by pressing and holding the ~ (tilde) key (it's just above the Tab key) to open the View Pie menu. Use the View Pie menu to switch between different orthographic views. To switch to Perspective view, press 5 on the number pad or click the middle mouse button and drag to tumble the view.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_4

Next, let's learn how to work with objects. Press Shift+A to open the Add menu. This gives you a list of objects that you can add to your scene. Go to Mesh and choose Cube. This will add a cube to your scene. Click to select the cube. Right now, we are in the Object mode, where we can individually select any object that we want and move it around. The following shortcuts can be used in Object mode.

Hotkey	Function
G	Move
R	Rotate
S	Scale
A	Select all
X	Delete selection
Shift+D	Duplicate
H	Hide selected
Alt+H	Unhide all
Ctrl+J	Join selected objects
M	Move selection to collection
Ctrl+S	Save file
Ctrl+Z	Undo
Ctrl+Shift+Z	Redo
Ctrl+Spacebar	Toggle Minimize/Maximize editor area

You can press the Tab key to go to the Edit mode of an object so that you can select its individual elements, like vertices and faces, to edit them. When in Edit mode, some functionality and shortcuts change. The following table lists important shortcuts in Edit mode.

Hotkey	Function
E	Extrude
I	Inset selected faces
F	Create face/edge
K	Use knife tool
O	Enable Proportional Editing
L	Select linked vertices
Shift+L	Deselect linked vertices
Ctrl+N	Recalculate normals
P	Separate to new object
U	Unwrap mesh

You can access these commands from the toolbox using the mouse, but I recommend you memorize the shortcuts because it'll make your work efficient and fast.

There are some other things that you need to know as well:

- If you select an edge loop, you can slide it along its plane by double-tapping G rapidly.
- You can press Z while in Object mode to switch to Wireframe mode so that you can see things more clearly.
- When you extrude any face, you can hold the Alt key to enable even extrusion.

- You can hold Ctrl when performing any transformative action to enable snapping respective to the tool you are using.
- You can press and hold Shift to make your action slower so that you have more accuracy when performing any transformative action.

There are certainly more things to learn, but it is better to learn them along the way as you are doing things. Try not to memorize everything at once. It is easy to forget things or mix things up. It is important to keep practicing to bring fluidity and accuracy in what you do.

Basics of Substance Painter

Substance Painter is an industry standard texturing tool. It requires a lot of practice to master this tool and use it to its potential. As a beginner, you need to know a lot of things to get you started on this journey.

Let's start with basic navigation. To rotate the view, you need to click and hold Alt+left mouse button (LMB). To pan the view, press Alt+MMB (middle mouse button). Use the mouse scroll wheel to zoom in or zoom out of the view. Alternatively, you can use Alt+RMB (right mouse button) to zoom in and out more smoothly. Press Alt+Shift+LMB to snap the camera to different view modes, such as the Front, Side, and Top views.

The following table lists more shortcuts that you should know to use Substance Painter effectively.

Hotkey	Function
]	Increase tool size
[	Decrease tool size
P	Color Picker tool
L	Enable Symmetry
D	Enable Lazy Mouse
F4	Toggle between 2D/3D view
F5	Switch to Perspective view
F6	Switch to Orthographic view
C	Display next material channel
M	Display material
B	Display next mesh map
1	Select Paint tool
Shift+LMB (left mouse button)	Draw straight lines

The others you will see while you work on this project. Try to learn the shortcuts by using them frequently in Substance Painter. That's the best way to learn.

Blocking Out the Scene

Let's now start blocking out the scene. Blockout is an important step toward creating a complete scene. A blockout allows you to experiment with your ideas with a lot of flexibility and decide what looks good and what works well.

Blocking Out the Main Temple in Blender

Let's now see how to create a scene with Blender. We will be working with the latest version of Blender, which as of writing this book is 2.81a. However, you can use any version, as most of the basic concepts will remain the same. Once Blender has been downloaded and installed, launch it. You will be greeted by its default welcome window, as shown in Figure 4-1.

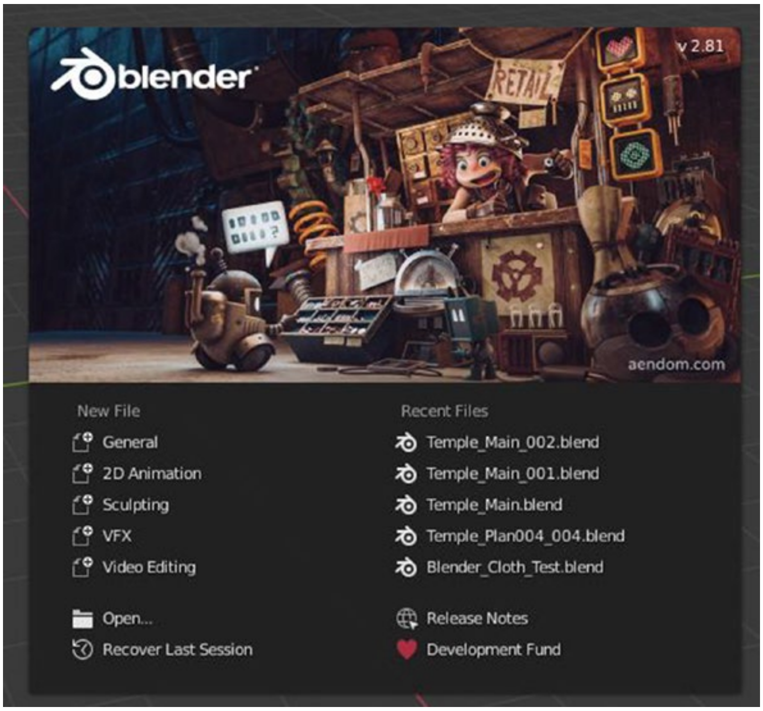


Figure 4-1. *Blender welcome window*

This window lets you create a new project by clicking on one of the options under the New File heading or open an existing project by clicking on the options under Recent Files or Open.

Since you are creating a new scene, either click on General or click anywhere outside the welcome window to close it. Then you should see the default startup file of Blender, which has a Cube, a Light, and a Camera already placed in the scene. We usually don't require these things when we start creating a scene. So, you can select everything by clicking and holding down the left mouse button and then drawing a selection around all the objects in the scene. Press X on the keyboard and confirm the delete. You can also use the Delete key.

You can now save this empty scene as a startup file so that you don't have to select and delete the unwanted objects every time you want to create a new scene. To do this, go to File ➤ Defaults ➤ Save Startup File, as shown in Figure 4-2.

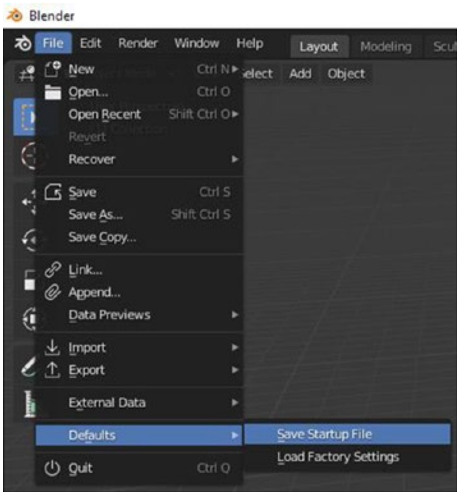


Figure 4-2. *Saving the startup file*

Now whenever you create a new project, you will have an empty scene to start your work from scratch. This way you will save some time.

Next, let's import the character reference model into the scene. This will give you an idea about how tall or wide you need to build the temple. It is very important that you work with references because only then can you create things realistically. Unless you have a proper reference, you can't judge whether your objects have the correct dimensions.

Once you have downloaded the human reference model, put it in a folder where it is easy to find. See Figure 4-3. To import it inside Blender, click on File ➤ Import ➤ FBX.

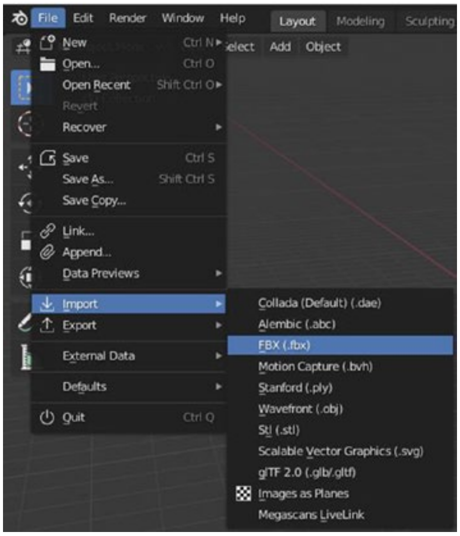


Figure 4-3. *Importing files*

A new File Browser window will open, allowing you to navigate to the location where you saved the character model. Select the file in the File Browser and click on Import FBX. The character should appear in the center of the screen. This character is scaled according to the size of the default Unreal Engine character. This will ensure that the scale of the models stays consistent between Blender and UE4.

Let's start creating the basic shapes first. Press Shift+A to open the Add menu, then go to Meshes and choose Plane. This should add a plane where your 3D cursor is, which by default should be in the center of your screen. Then press Tab on your keyboard to go to Edit mode and press 1 to switch to Vertex mode. See Figure 4-4. Now double-tap A to select everything and press A to open the Delete menu. Click on Edge Collapse.

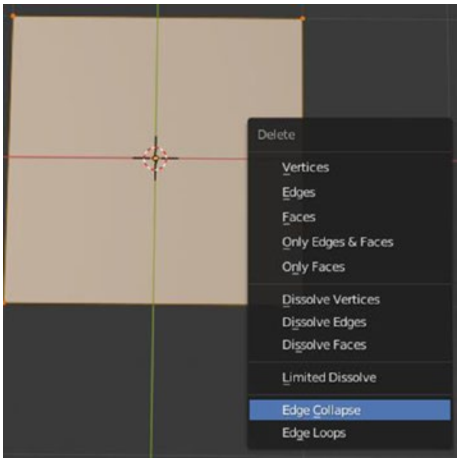


Figure 4-4. *Delete menu*

You will then be left with a single vertex. Click on the Snapping button on the top-center of your viewport, as shown Figure 4-5. Make sure Increment and Absolute Grid Snap are enabled.

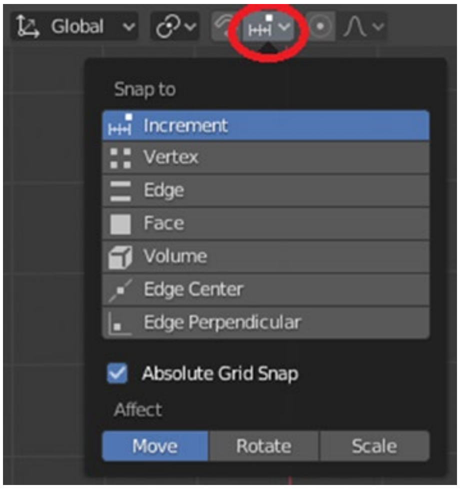


Figure 4-5. *Snapping options menu*

Press 7 on your number pad to switch to Top-Orthographic view. Select the single vertex while in Edit mode and move it along the Y-axis by six grid blocks, as shown in Figure 4-6. To do this, press G after selecting the vertex. This will allow you to move the vertex along all three axes. Then press Y to lock movement only along the Y-axis. While moving, hold Ctrl to snap it along the grid.

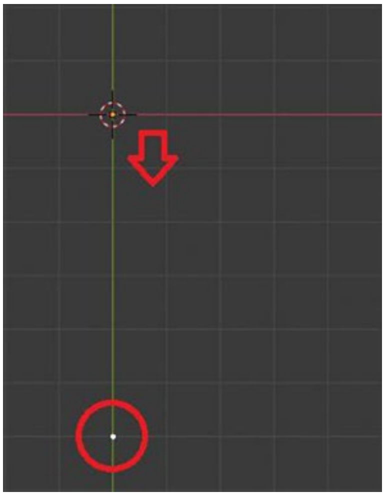


Figure 4-6. *Moving the vertex by six grid blocks*

Press E to extrude the vertex and move it along the X-axis, this time by pressing X to constrain movement only along the X-axis. Hold Ctrl to snap along the grid and move it two grid blocks right, as shown in Figure 4-7.

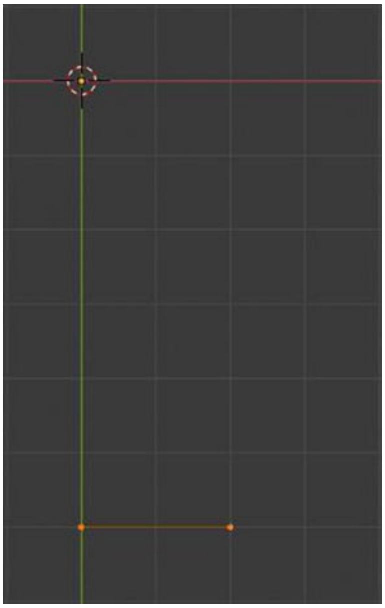


Figure 4-7. *Extruding the vertex by two grid blocks*

Repeat these steps to create the shape shown in Figure 4-8.

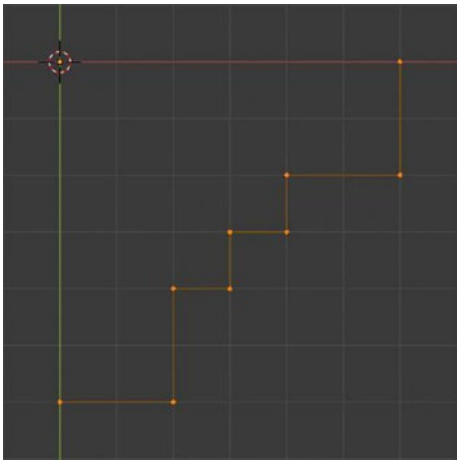
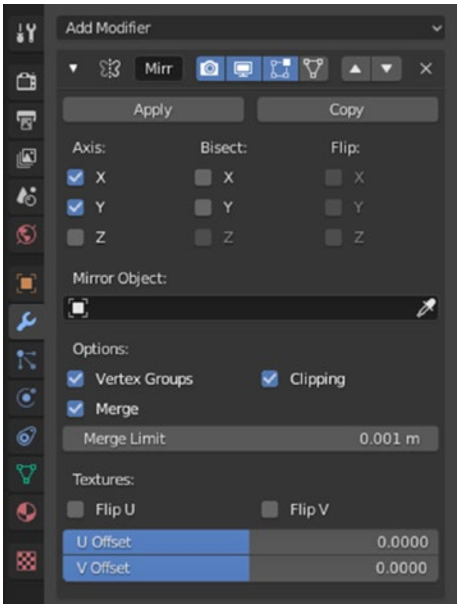


Figure 4-8. *Shape so far*

Ensure that the proportions of the shape are the same as that you see in the figure. It is very important; otherwise, you will have problems later.

We will now add a mirror modifier to the shape. To do this, go to the Modifiers tab and choose Mirror from the drop-down menu. Mirror Modifier should now appear in the modifier stack. You need to change a couple of its properties. Under Axis, ensure that both the X- and Y-axes are enabled, but the Z-axis should be disabled. Under Options, enable Clipping. Your mirror settings should look similar to Figure 4-9.



Axis: X ✓, Y ✓, Z ☐
Bisect: X ☐, Y ☐, Z ☐
Flip: X ☐, Y ☐, Z ☐
Mirror Object:
Options: Vertex Groups ✓, Clipping ✓, Merge ✓
Merge Limit 0.001 m
Textures: Flip U ☐, Flip V ☐
U Offset 0.0000
V Offset 0.0000

Figure 4-9. *Mirror Modifier settings*

Your shape should now look similar to Figure 4-10.

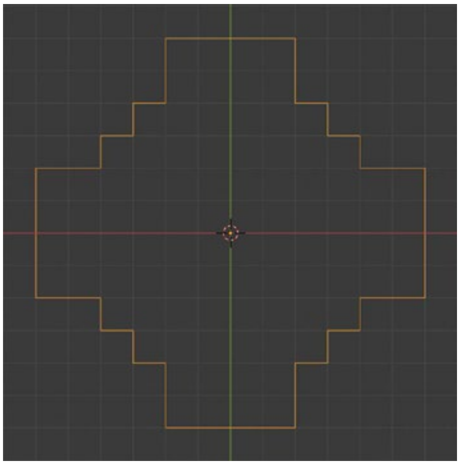


Figure 4-10. *Shape so far*

You now have the base shape. Switch to Front view by pressing 1 on the number pad. Then, while in Edit mode, select all the vertices by tapping A and extrude them on the Z-axis by 15 grid blocks while holding Ctrl. Your scene should now look like Figure 4-11.

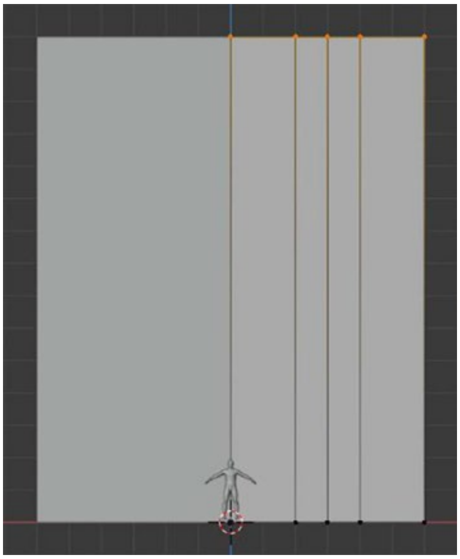


Figure 4-11. *Model so far*

Next let's block out the fractal shapes of the temple before we model it in detail. This will require some precise placement, so if any shape is not right or if any proportion is not precise, you will have a lot of problems getting the shape right.

You need to change the shape slightly before you proceed; see Figure 4-12. As you can see in Figure 4-12, we added two smaller steps to the shape. This will give some variety to the overall shape. To do this, simply select the faces at the extreme ends and delete them. Then, from the top perspective view, carefully draw the smaller corners in the same way in which you created the larger ones.

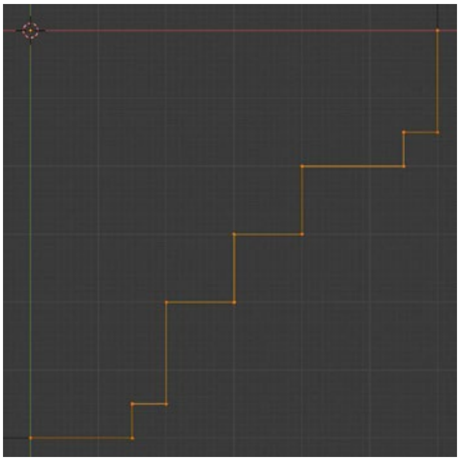


Figure 4-12. *Adding some detail to the shape*

We will now add a cube to the scene and set its size to 1m. Place the cube like you see in Figure 4-13.

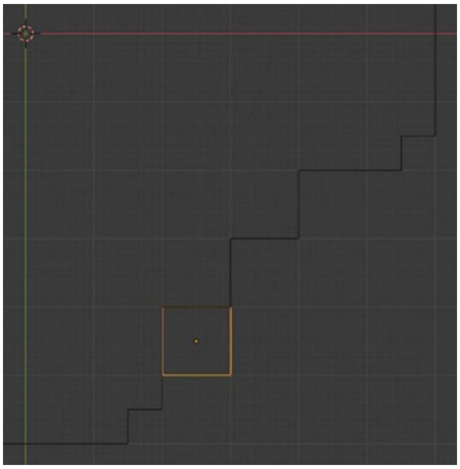


Figure 4-13. *Placing the cube*

Duplicate the cubes and place them as shown in Figure 4-14.

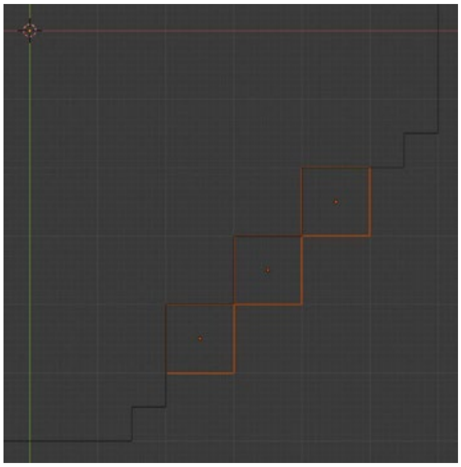


Figure 4-14. *Placing the cubes*

Next, go to the Edit mode while having all the cubes selected so that you can edit them all at once. Select the top faces and extend them along the Z-axis until they are the same height as the main structure that you created.

Next, delete the top and bottom faces of these three cubes. Another thing to note here. We created these three column structures just to be placeholders for now. We will add details to just one of them and then simply duplicate and mirror it. But while we are blocking out the silhouette of the temple, we will edit them all at the same time.

Next, go to the Edit mode of one the pillars and press Shift+R to initiate the Insert Edge Loop tool. Then add 4-4 edges vertically on both of the visible faces, as shown in Figure 4-15.

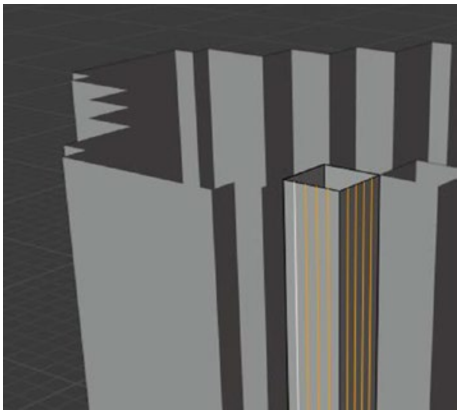


Figure 4-15. *Adding edge loops to the pillar*

Then select the two center edges on both the faces and press X to bring up the Delete menu. Choose Dissolve Edges from the list. Select the two faces enclosed between the remaining edges and, in the Tools menu, click and hold the Extrude tool to open a submenu. Choose Extrude Along Normals from this list, as shown in Figure 4-16.

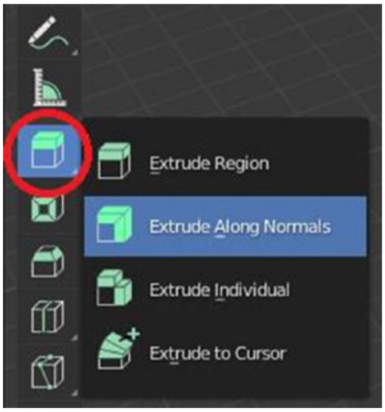


Figure 4-16. *Choosing the extrude method*

Click and hold the Extrude tool gizmo that is present near the selected faces and drag your mouse to start extruding. Then you can manually adjust the extrude amount until it is roughly 0.1 or type `0.1` on your keyboard to set the final amount. Delete the top and bottom faces created as a result of the extrude operation. Your result should look similar to Figure 4-17.

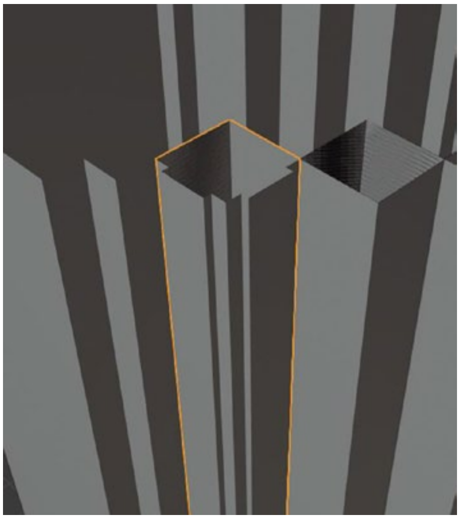


Figure 4-17. *Result after the Extrude operation*

You can now delete the other pillar pieces, duplicate the one you edited, and place them where you previously placed the other pillars.

Next, we will block out the top portion of the temple. We will start by creating a plane and collapsing all its vertices into a single vertex, like we did in the beginning. Place it as shown in Figure 4-18. You can place the vertex 2-3 blocks down, depending on how wide you want the shape to be.

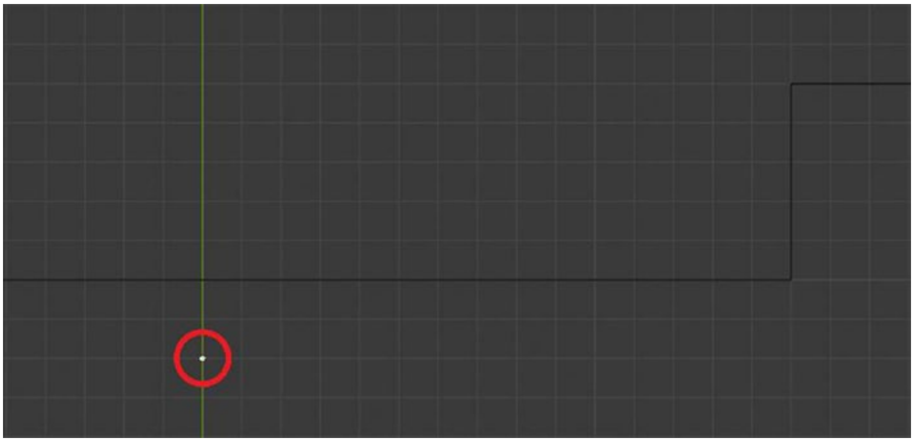


Figure 4-18. *Placing the vertex*

Select the vertex in Edit mode and extrude it by pressing E along the X-axis. We will extrude around the silhouette.

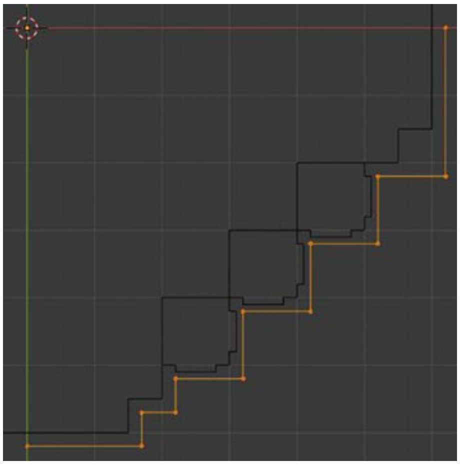


Figure 4-19. *Extruding the vertex*

Now switch to Object mode by pressing Tab. Make sure the shape that you created is selected, then right-click. From the menu that appears, choose Set Origin and, from the sublist, choose Origin to 3D Cursor. See Figure 4-20.

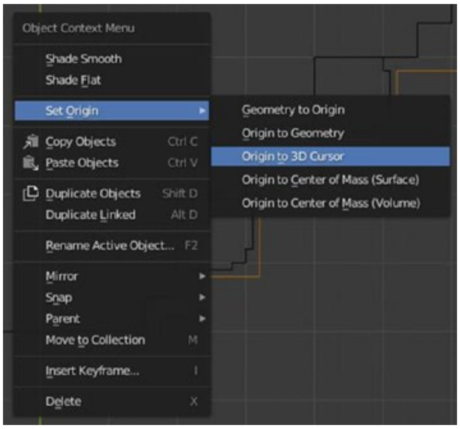


Figure 4-20. *Setting the origin*

Your 3D cursor should be in the World Origin by default, but if you moved it, you can press and hold Shift+S to open the Cursor Pie menu, hover your cursor over Cursor to World Origin, and release Shift and S. This will send your cursor back to the World Origin.

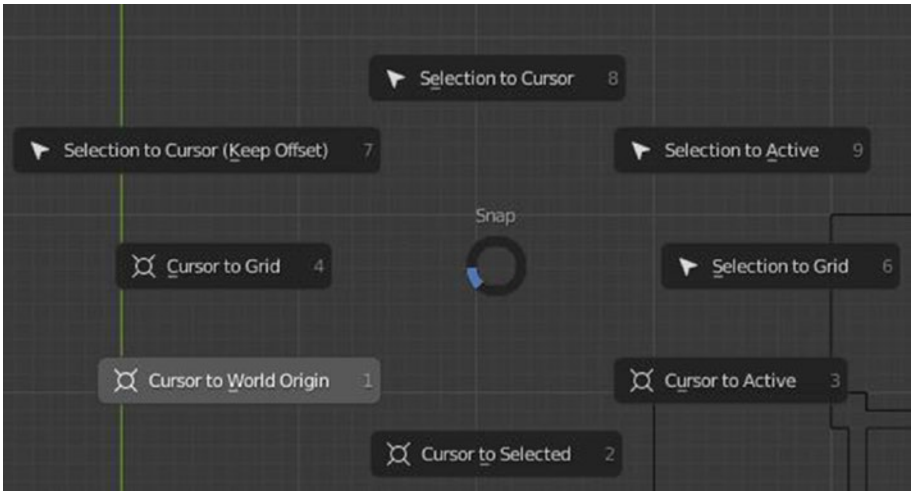


Figure 4-21. *Sending the cursor to World Origin*

Once the cursor is at the World Origin, you can set the origin of the object to be the 3D cursor. See Figure 4-22. After that, we will apply a mirror modifier to the shape and enable both X-axis and Y-axis as well as Clipping.

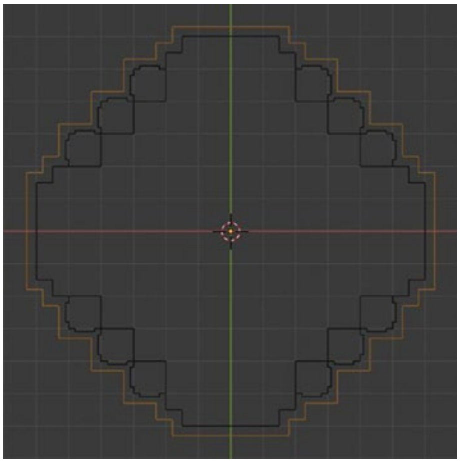


Figure 4-22. *Resultant shape after mirroring*

Now apply the mirror so that you can edit the shape properly. Go to Vertex Edit mode and dissolve the vertices at the center of the edges, as shown in Figure 4-23.

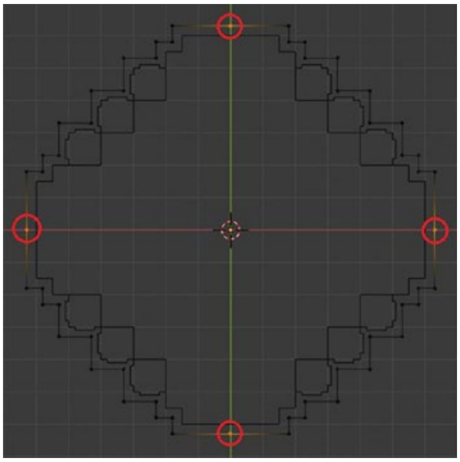


Figure 4-23. *Dissolve these vertices*

Now select all the remaining vertices by pressing A and switch to Front Orthographic view by pressing 1 on your number pad. In Front view, zoom in toward the vertices and extrude them up by pressing E. Move them up by 3-4 grid blocks. You can adjust the height later if necessary. See Figure 4-24.

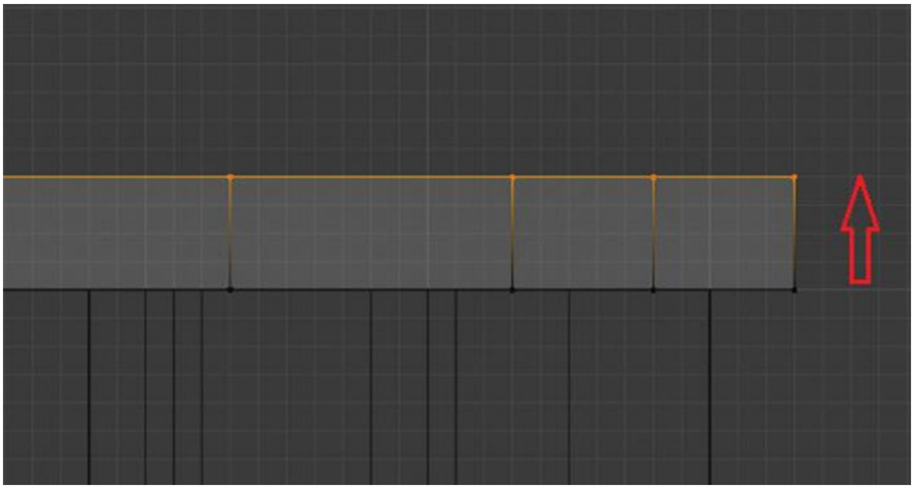


Figure 4-24. *Extruding the vertices*

After that, press F to fill the top border edge with a face. Next select a pillar structure and select the top edges, as shown in Figure 4-25.

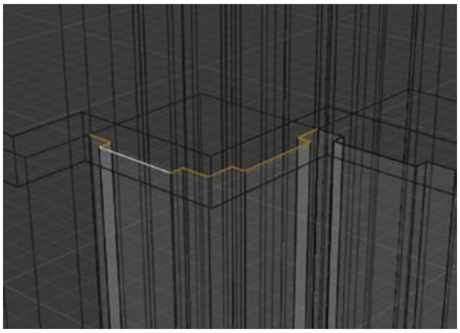


Figure 4-25. *Select these edges*

Go to Front view by pressing 1 on the number pad and press Shift+D to duplicate the selected edges and move them up. The best way to do this is by holding G to enable grid snapping. Place it just on top of the shape that you extruded up. Next, press P while the duplicated edge is selected to open the Separate menu and choose Separate by Selection. See Figure 4-26.

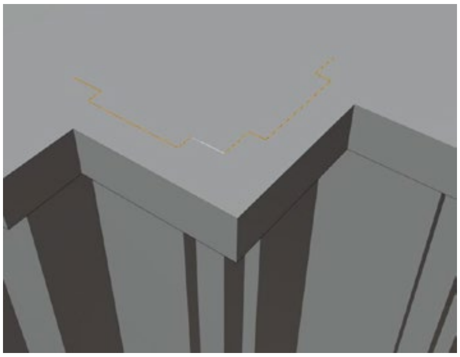


Figure 4-26. *Placing the shape*

Press Tab to go into the Edit mode of the shape and select the two wider edges at either side of the shape. Then right-click and choose subdivide from the list. This will add one extra vertex to both edges. See Figure 4-27.

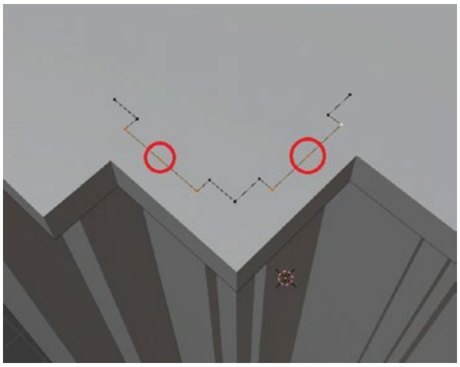


Figure 4-27. *Subdividing the edges*

Press 7 on the number pad to switch to Top Orthographic view. While in the shape's Edit mode, select the end vertex on the left side of the shape and extrude it along the Y-axis, until it is parallel to the end vertex on the right side. Then select both end vertices and press F to create an edge and join the vertices. Your final shape should look like Figure 4-28.

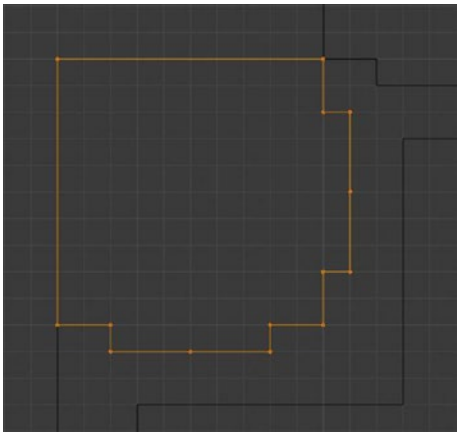


Figure 4-28. *Resultant shape*

Now select all the vertices of the shape and press F to fill in a face. Then Tab to Object mode, right-click and choose Set Origin. From the sublist, choose Origin to Center of Mass (Surface). Your shape may suddenly look strange because there is a mirror modifier carried over from the pillar shape from which you separated the shape. If there isn't a mirror modifier present, then add one now. Enable the X- and Y-axes as well as Clipping. To correct the strange shape, you should delete the vertices shown in Figure 4-29.

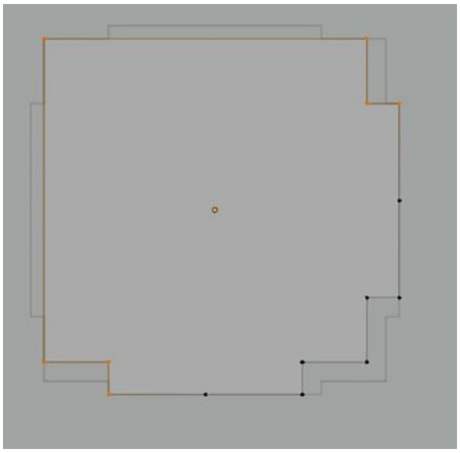


Figure 4-29. *Selecting the vertices for deleting*

After that, apply the mirror modifier and adjust the shape slightly to match the pillar in Figure 4-30. Your shape should look something like Figure 4-30.

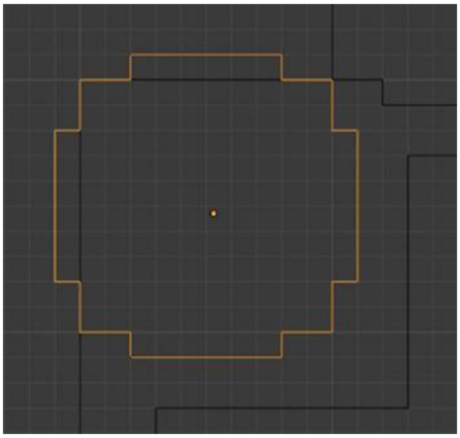


Figure 4-30. *Shape so far*

Switch to Front Orthographic view and go to the shape's edit mode if you are not in it already. Select everything. Delete the center face, as you won't be needing it. Then extrude it along the Z-axis, as seen in Figure 4-31.

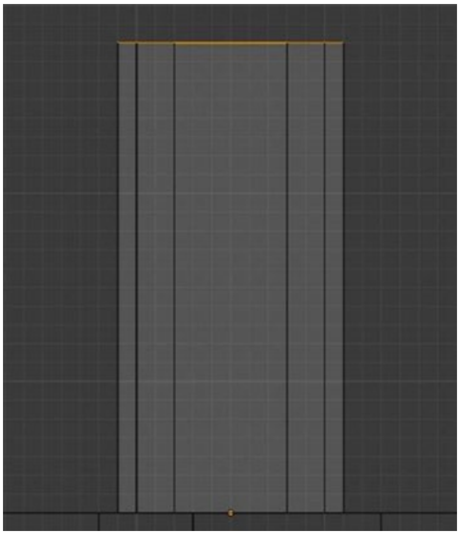


Figure 4-31. *Extruding the edges up*

Press Shift+R to initiate the Insert Edge Loop tool and then insert two edge loops. Press G twice quickly to start sliding the edges. Place the edges as seen in Figure 4-32.

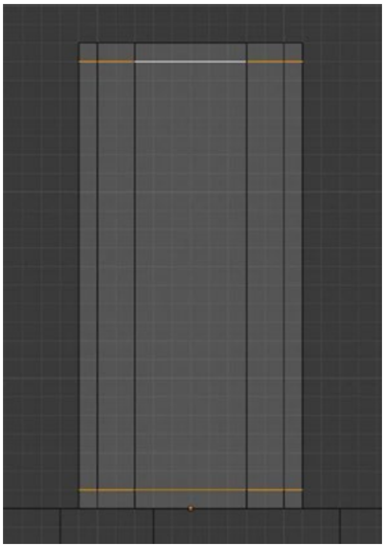


Figure 4-32. *Placing the edges*

Now press Shift+R to initiate the Insert Edge Loop tool again and insert five edge loops in the center part of the shape, as shown in Figure 4-33.

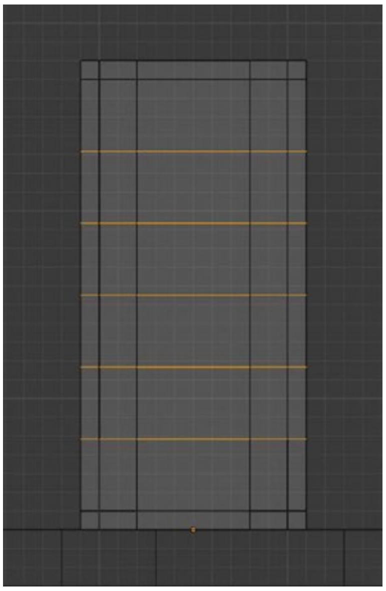


Figure 4-33. *Inserting five edge loops*

Switch to Vertex mode and select the five edge loops that you inserted. Press Shift+H to hide the unselected edges. Basically, this will isolate the vertices that you selected and allow you to work with them while not affecting anything else. Select the horizontal center edge, enable Proportional Editing by pressing O, scale your brush size accordingly, and scale out the vertices. The results should roughly match Figure 4-34.

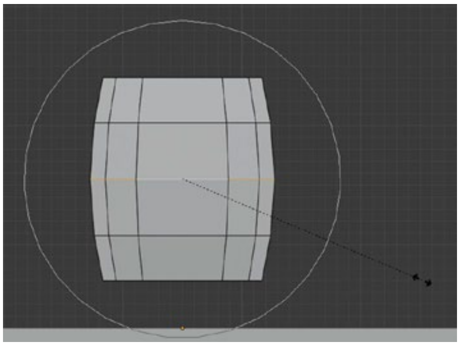


***Figure 4-34.** Scaling with Proportional Editing*

You can certainly adjust the scale to be the way you want. Once you are happy with the shape, press Alt+H to unhide everything. Adjust the shape with Proportional Editing turned off (press O again to turn off Proportional Editing). You can also add edges if you feel it is needed and modify the shape until you have something similar to Figure 4-35 (or something that you are happy with).

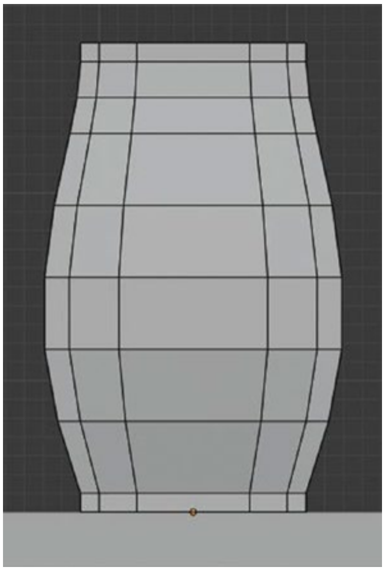
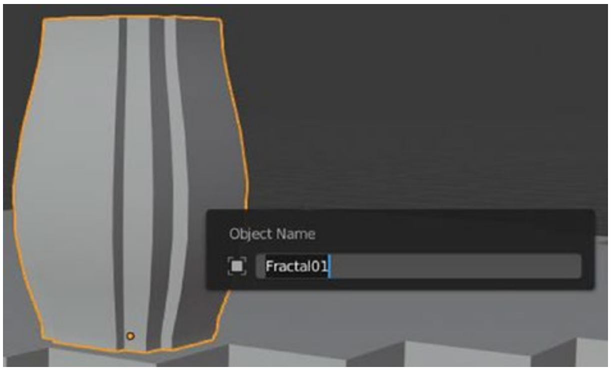


***Figure 4-35.** Final shape*

Once you're done, press Tab to exit Edit mode and go to the object of this shape. Press F2 to open the Rename box. We set the name of this object to `Fractal01`, but you can name it whatever you want. This will make it easy to identify it in the World Outliner window. See Figure 4-36.



Fractal01

Figure 4-36. *Renaming the object*

Figure 4-37 shows the World Outliner, which shows all the objects that you created. You can rename the objects by double-clicking its name in the Outliner. This is a very important step for asset organization, when there are lots of assets in the scene.

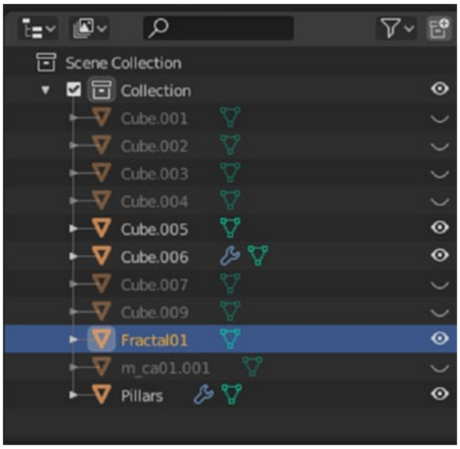

Scene Collection
Collection
Cube.001
Cube.002
Cube.003
Cube.004
Cube.005
Cube.006
Cube.007
Cube.009
Fractal01
m_ca01.001
Pillars

Figure 4-37. *World Outliner*

Once you have renamed your assets, it will become easy to find them, even when the scene starts becoming complicated. It is good practice to name objects based on what they resemble so they easier to identify. The Eye icon on the right shows whether the object is visible or not. Clicking on that icon will toggle the visibility.

Let's return to working on `Fractal01` now. As you may have noticed, the shape looks very jagged and the sharp edges are very prominent. To fix this, while `Fractal01` is selected in Object mode, right-click and choose Shade Smooth. See Figure 4-38.

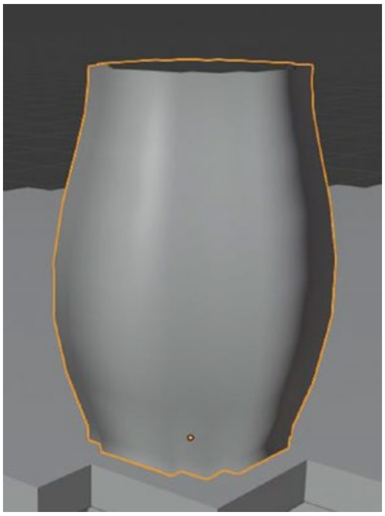
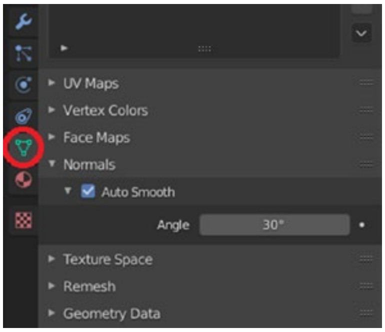


Figure 4-38. *Shading the shape smooth*

As you can see in Figure 4-38, the object looks extremely smooth now. You can fix this issue by enabling Auto-Smooth. To enable Auto-Smooth, click on the Object Data Properties icon in the Properties Editor. The icon is an inverted triangle formed by three interconnected dots, as shown in Figure 4-39.



☑ Auto Smooth

Angle 30°

Figure 4-39. *Enabling Auto-Smooth*

You can see that there is an Angle option under Auto-Smooth. This angle is used to control which edges are smoothed and which aren't. Any edge that has an angle of less than the value in the Angle option will be smoothened out. Any angle larger than that will be hardened. You can adjust this value to get the result that you want. For the case here, the default value works fine. Your result should now look similar to Figure 4-40.

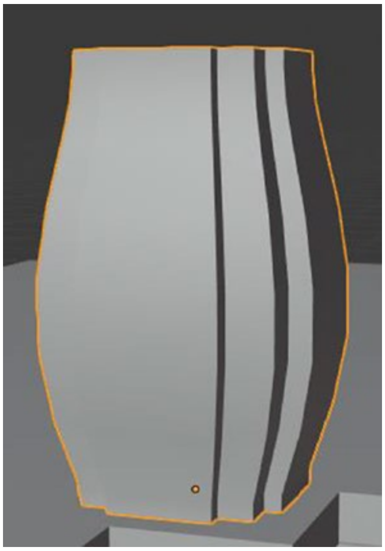


Figure 4-40. *Result after Auto-Smooth*

You have now made the object smooth procedurally, using Blender's automatic tools. But what if you want to do some manual intervention? You can mark edges sharp or smooth if you want to. Select the edges in Figure 4-41.

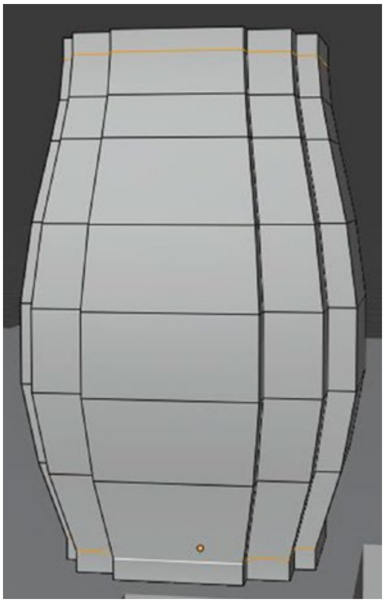


Figure 4-41. *Selecting the edges*

After selecting the edges, right-click; the Edge Context menu should appear. In the list, you should see two options called Mark Sharp and Clear Sharp. See Figure 4-42.

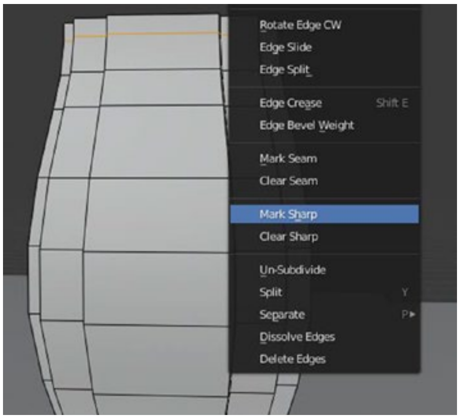


Figure 4-42. *The Mark Sharp and Clear Sharp options*

As their names suggest, the Mark Sharp option will sharpen the selected edges and Clear Sharp will clear any sharpness applied to the edges. So, with the aforementioned edges selected, right-click and mark them sharp. The selected edges should now be highlighted in blue, confirming that they have Sharpness applied to them. The result should look like Figure 4-43.

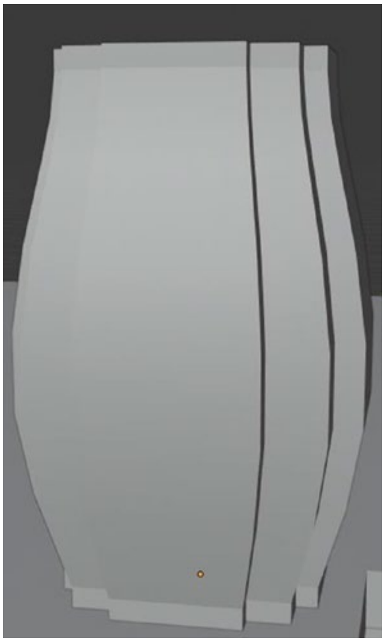


Figure 4-43. *Result after marking the edges sharp*

There is a point to note here. Clear Sharp only removes the sharpness effect that you manually applied by marking edges sharp. It cannot be used to smooth out edges by clearing sharpness from them. To make edges smooth, you need to increase the Auto Smooth angle.

Select the top edges of the object by Alt+clicking on one of them and pressing F to fill in a face. Duplicate that object twice and place them as shown in Figure 4-44.

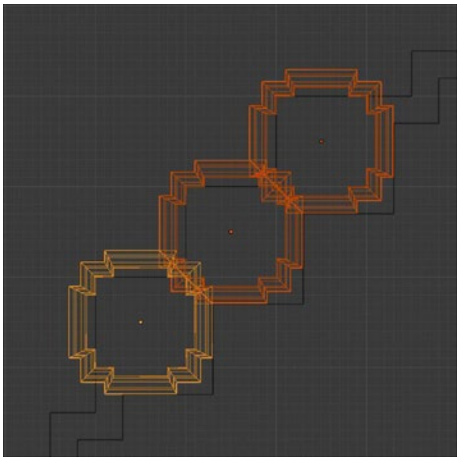


Figure 4-44. *Placing the duplicated objects*

Combine the objects by selecting all three of them and pressing Ctrl+J to join. Once they are joined, right-click and choose Set Origin ➤ Origin to 3D Cursor. Next, apply the mirror modifier to your object and enable both the X- and Y-axes mirror. See Figure 4-45.

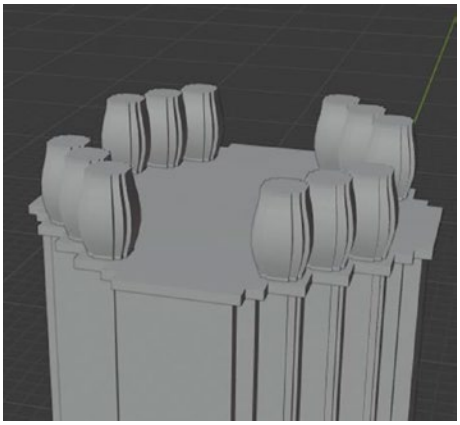


Figure 4-45. *Result so far*

You will now edit the platform on which the fractals sit. You need to add the mirror modifier back that you removed. Let's isolate the platform first by pressing / on the number pad and inserting some edge loops, as shown in Figure 4-46.

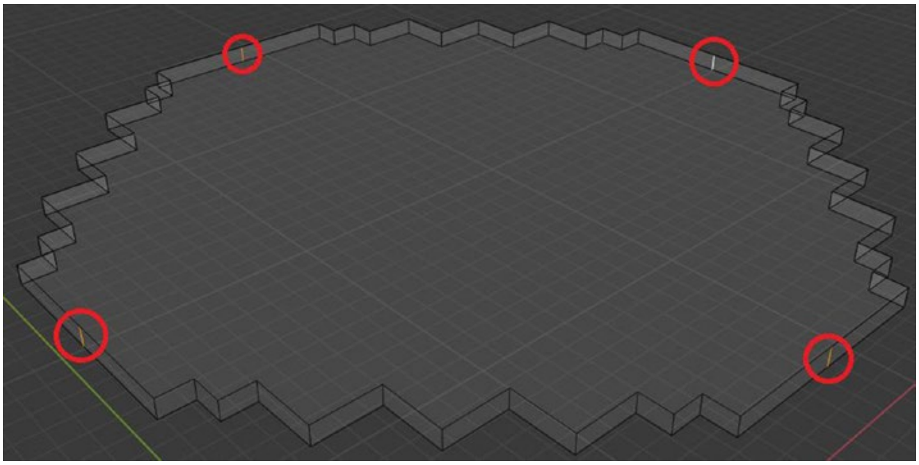


Figure 4-46. *Adding edges to the shape*

Next, select the top vertices opposite to each other and press J to connect them with an edge. See Figure 4-47.

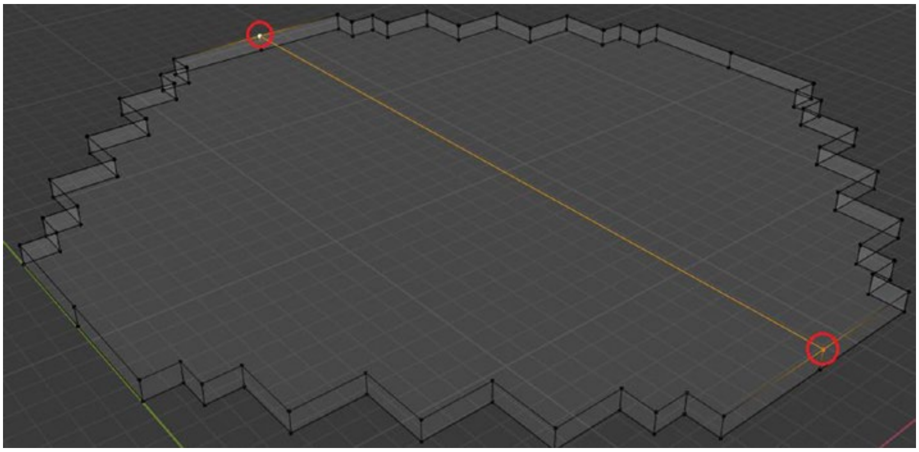


Figure 4-47. *Selecting and joining vertices*

Do the same thing with the other two vertices as well—select and join them. Now you have divided the model into four equal parts. You can delete three of the four parts and use the mirror on the remaining quarter to create a symmetrical model. Select the vertices shown in Figure 4-48 and delete them.

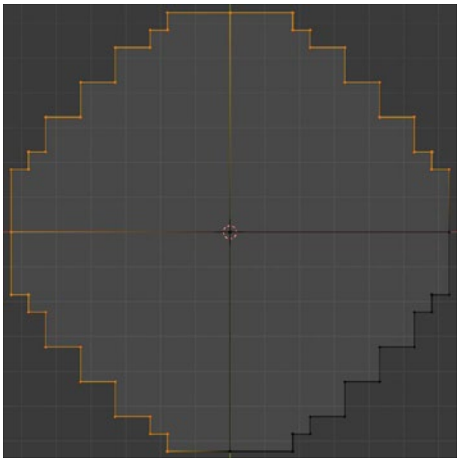


Figure 4-48. *Select these vertices and delete them*

After deleting the vertices, you can apply a mirror modifier with X- and Y-axes mirror enabled and Clipping enabled. With that, you should have the entire platform shape back, like it was before. Now, if you make any changes to the mirrored shape, it will reflect on the other parts as well.

Now you will bevel the vertices highlighted in Figure 4-49.

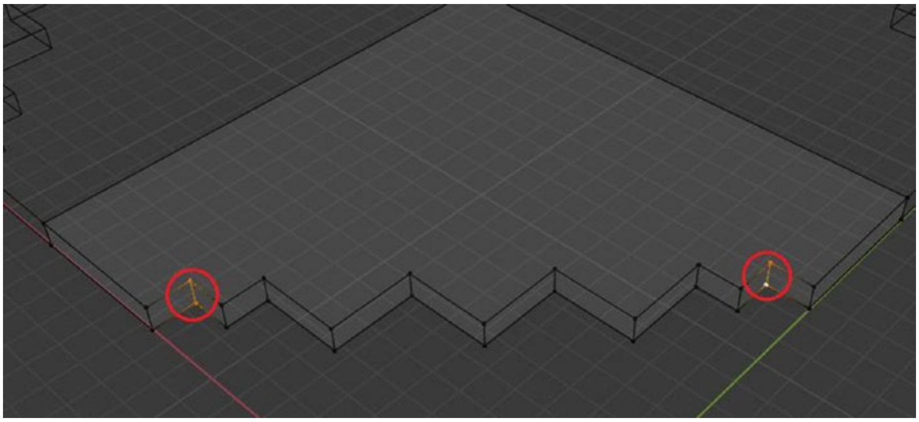


Figure 4-49. *Bevel these edges*

You bevel by pressing Ctrl+B after selecting the vertices/edges and increase the number of segments to two by scrolling the mouse wheel. You can see how many segments you are inserting at the bottom of screen as well as reflected on your mesh. See Figure 4-50.

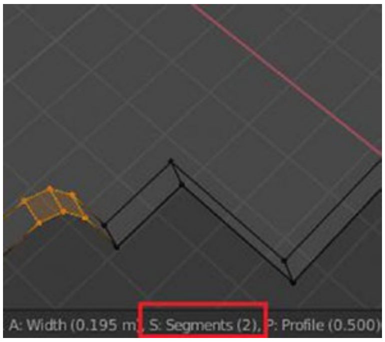

A: Width (0.195 m), S: Segments (2), P: Profile (0.500)

Figure 4-50. *Increasing the number of segments*

The beveling will make the edges rounded. So, we will manually adjust the edges of both corners, as shown in Figure 4-51.

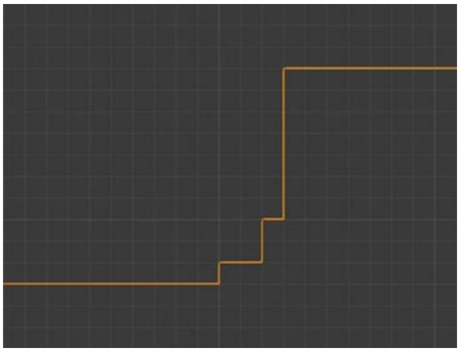


Figure 4-51. Adjusting the edges

To properly adjust the edges, zoom in closer and hold Ctrl to enable grid snapping. Next, you'll subdivide the central edge running along the Y-axis line. This will add a vertex to it. Select this new vertex, hold G, and move it along the Y-axis until it is positioned as shown in Figure 4-52. After that, hold Shift and click on the vertex just opposite it on the other side and connect them by pressing J.

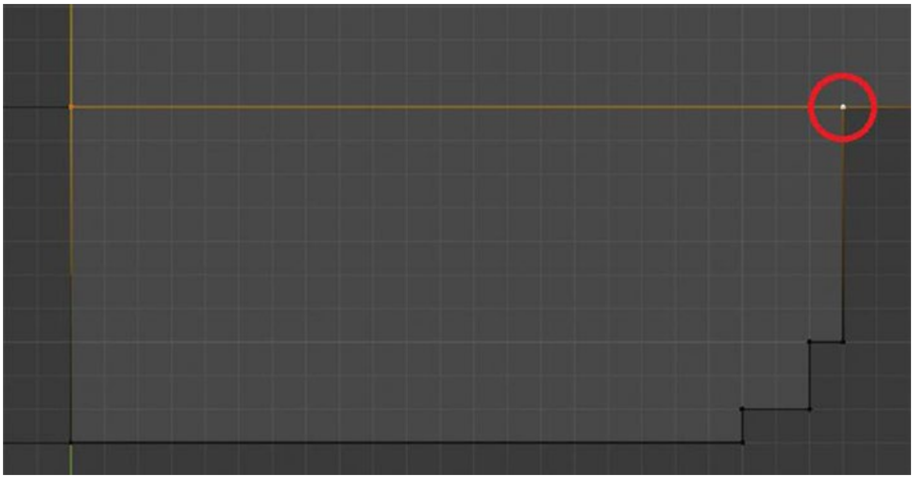


Figure 4-52. Positioning the vertex and joining it

Now select the face created by joining the two vertices and duplicate it by pressing Shift+D. After duplicating the face, right-click to reset its position back to where it was duplicated. Go to Front Orthographic view and, with the face selected, press G to start moving the face. While holding Ctrl, move it up by two and half grid blocks. Click to confirm.

While the new face is selected, press P to open the Separation menu. From it, choose Separate by Selection. Switch to Top Orthographic view and extend the back edge along the Y-axis, as you can see in Figure 4-53.

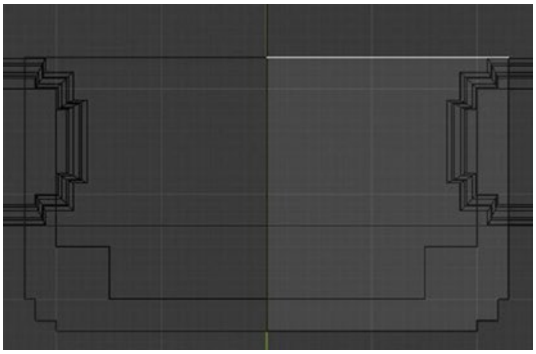


Figure 4-53. *Extending the edge backward*

Select the bottom faces and extrude them down until they meet on the platform object. After extrusion, the normals of the object maybe flipped. To see whether you have any normals facing the wrong way, click on the Shading menu drop-down icon. See Figure 4-54.

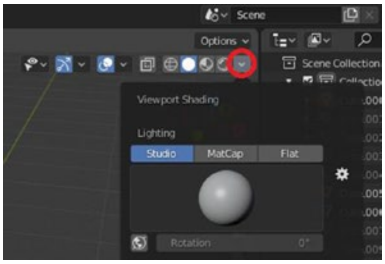
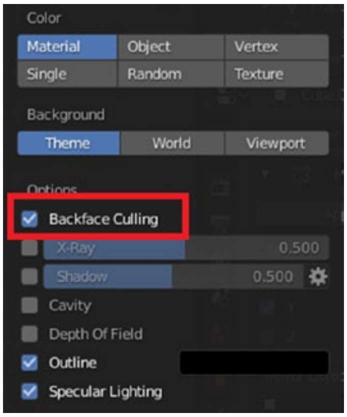


Figure 4-54. *Drop-down icon for the Shading menu*

Toward the bottom of that menu, you'll see an option called Backface Culling, as shown in Figure 4-55. Activate it by clicking on its checkbox.



Color

[✓ Material] Object Vertex

Single Random Texture

Background

[✓ Theme] World Viewport

Options

✓ Backface Culling

X-Ray 0.500

Shadow 0.500

Cavity

Depth Of Field

✓ Outline

✓ Specular Lighting

Figure 4-55. *Enabling Backface Culling*

Every face that has its normal facing outside will be visible and the face whose normals are inward will appear invisible. See Figure 4-56.

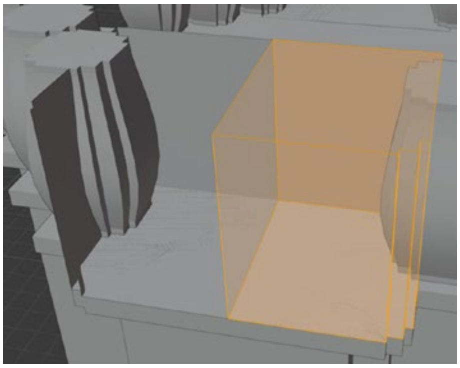


Figure 4-56. *Faces with normals facing the wrong direction*

To fix these inverted normals, Tab into the Edit mode of the object that you want to fix, select all its faces/edges/vertices by pressing A, and then press Shift+N. This will recalculate the normals and the inward facing normals will be fixed automatically.

Next, press Shift+D to duplicate and rotate along the Z-axis by 90 degrees. This way, you'll have the same shape on the other sides as well. See Figure 4-57.

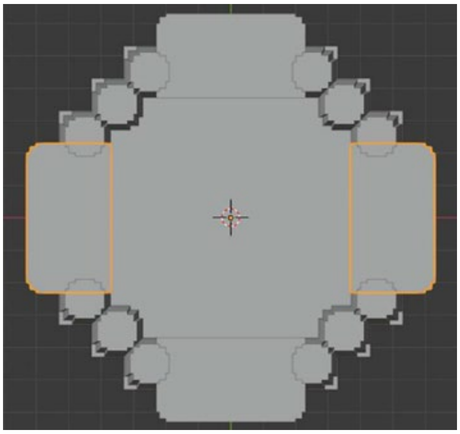


Figure 4-57. *Rotating the shape*

There is one more thing to do before you proceed. You should slightly adjust the shape in Edit mode to create a very small gap, as shown in Figure 4-58.

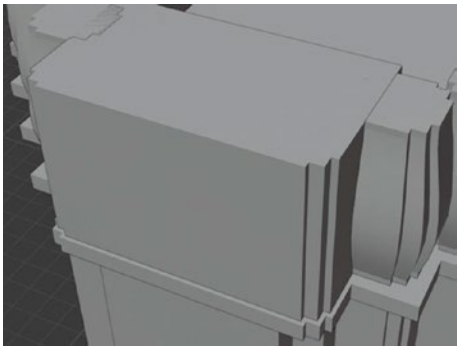


Figure 4-58. *Adjust the shape to create a tiny gap*

Adjusting this in Edit mode will keep the mirror modifier intact and the changes will be reflected on the mirrored side as well. Do the same with the duplicated shape as well. We will duplicate the shape again and move it to the Z-axis until the duplicated one is sitting exactly on top of the original. See Figure 4-59.

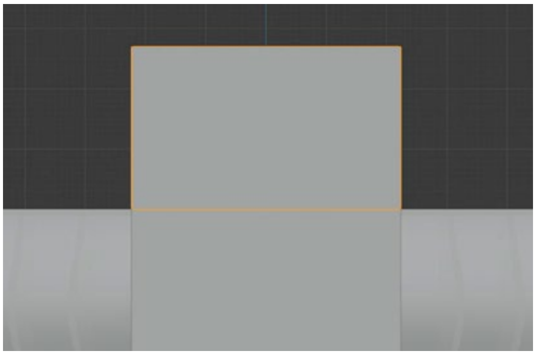


Figure 4-59. *Duplicating and placing the object*

Next, Tab into Edit mode, switch to Front Orthographic view, hold Z to open the Shading Pie menu, and choose Wireframe. Then select all the vertices on the right side of the object and move them toward the center. Then, from the Top view, move the vertices in the front toward the back. See Figure 4-60.

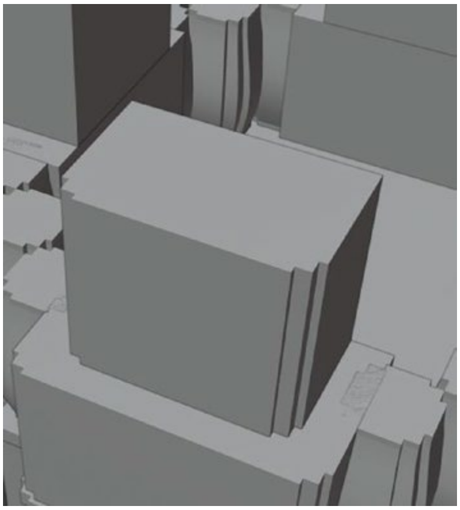


Figure 4-60. *Adjusting the shape of the object reference*

Now Tab into the object's Edit mode and extend it up, as you can see in Figure 4-61.

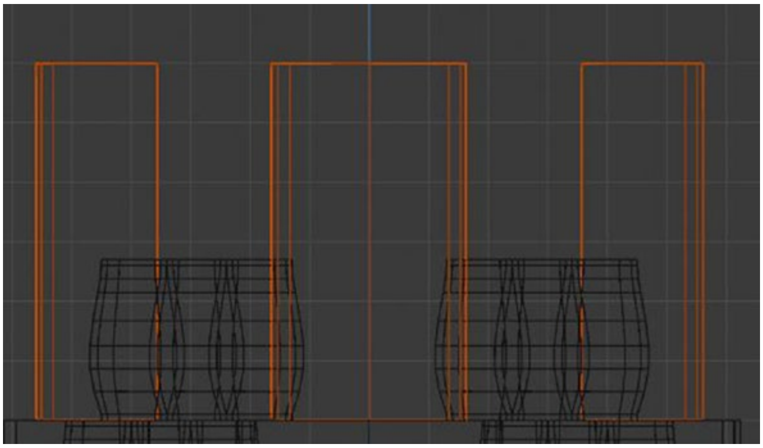


Figure 4-61. *Extending the shape*

Duplicate the shape again and place it slightly behind the original one, as you can see Figure 4-62.

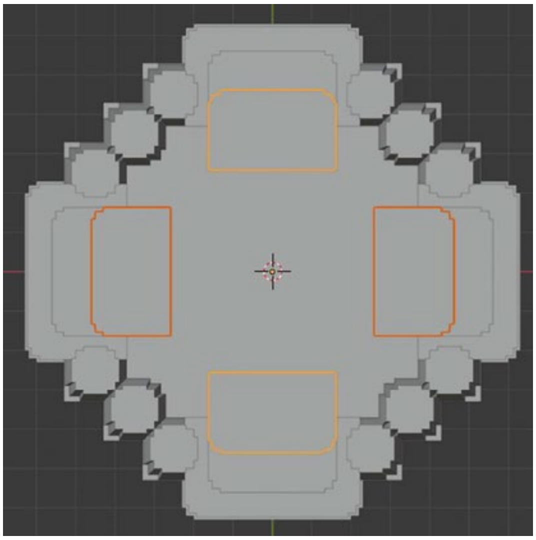


Figure 4-62. *Duplicating the object and placing it*

Make sure to do all the placing in Orthographic view and while in Edit mode so that mirror modifier is not affected and the mirrored copy is automatically in the correct place. If you don't duplicate while in Edit mode, you'll need to adjust the mirror modifier later, which will be a lot of trouble. Switch to Front Orthographic view and go to the Edit mode of the objects that you duplicated. Extend them up. See Figure 4-63.

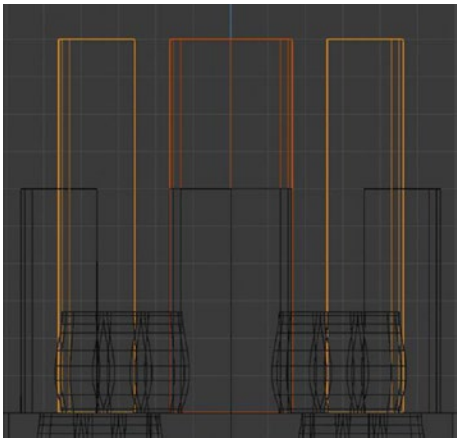


Figure 4-63. *Extending the shape up*

Now, using these shapes and human character reference, try to find a good size and shape proportion for these objects. Adjust the height and width of the objects so that they look good. Use reference images for more help. Next, Tab into the shapes' Edit mode and add edges by using the Insert Edge Loop tool. We will insert five edge loops in each. Beginning with the smallest ones, scale the edge loops that you inserted to create a shape similar to Figure 4-64.

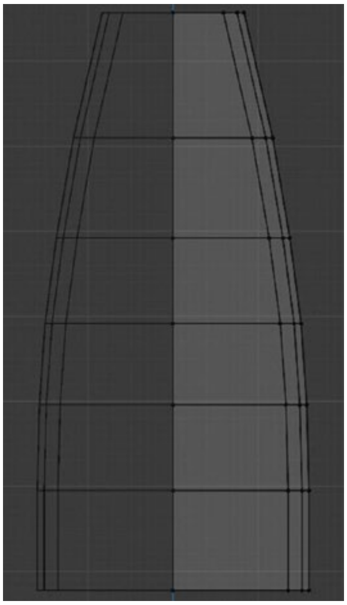


Figure 4-64. *Modifying the shape*

You can now duplicate this and place it around the temple. The result should resemble Figure 4-65.

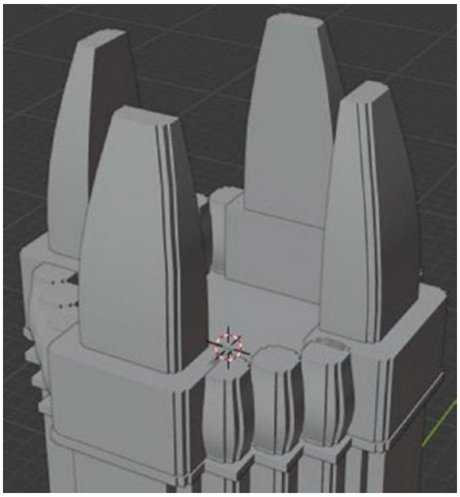


***Figure 4-65.** Duplicating and placing the object*

Let's duplicate it once again and scale it up globally. Then scale again along its width so that it is similar to Figure 4-66.

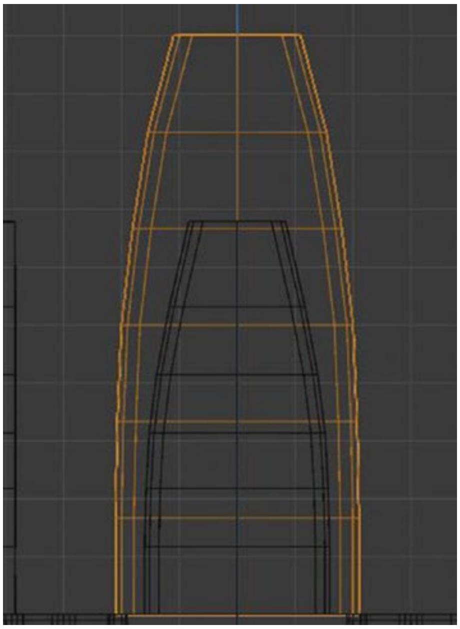


***Figure 4-66.** Resizing the object*

After scaling, select the object in Edit mode and move it back (while in Edit mode) slightly. See Figure 4-67.

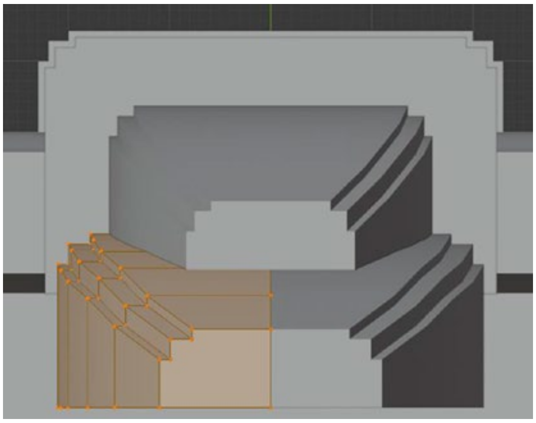


Figure 4-67. *Moving the shape back in Edit mode*

If you are happy with the shape, size, and location of the object, duplicate it around the temple, like you did with the previous object. See Figure 4-68.

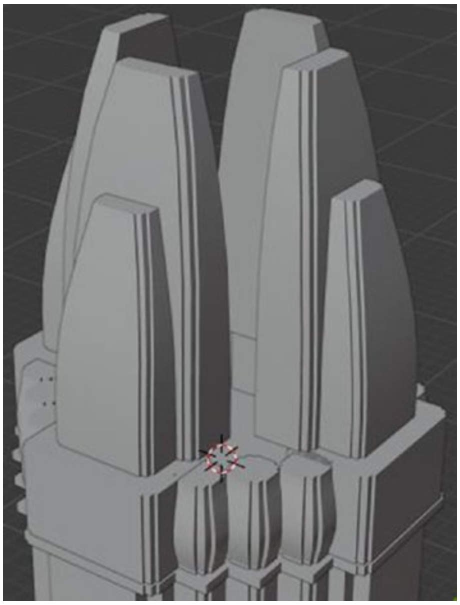


Figure 4-68. *Duplicating the shape around the temple*

Once again, take some time to adjust the dimensions of the temple if you think that it is needed. You are still in the blockout phase and therefore not committed to anything. The model is still relatively simple so it is easy to move things around and make adjustments if needed. When you start adding more details, it will become harder to make small changes due to the complexity of the model.

Let's proceed by adding the final piece of the blockout, the central spire. Add a cube by pressing Shift+A. Move the cube up until it is sitting exactly on top of the platform object. Then Tab into the Edit mode of the cube and select its bottom face. Press and hold Shift+H to open the Snap Pie menu, then hover over Cursor to Selection and release. The cursor should snap to the center of the selected face. Next, Tab out of Edit mode to Object mode. Right-click to open the Object context menu. From that menu, choose Set Origin ➤ Origin to 3D Cursor. See Figure 4-69.

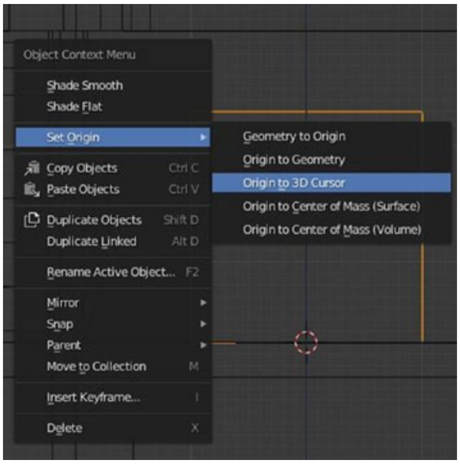


Figure 4-69. *Origin to Cursor*

After that, scale the cube along the X- and Y-axes. To do this, first initiate the Scale tool by pressing S and then press Shift+Z to constrain the scale along the X- and Y-axes. The same applies to other axes as well. If you press Shift+Y, the scale will be constrained along the X- and Z-axes. The same goes for Shift+X, which restricts scaling along the X-axis and allows scaling only along the Y- and Z-axes. This applies to the Move and Rotate tools as well. See Figure 4-70.

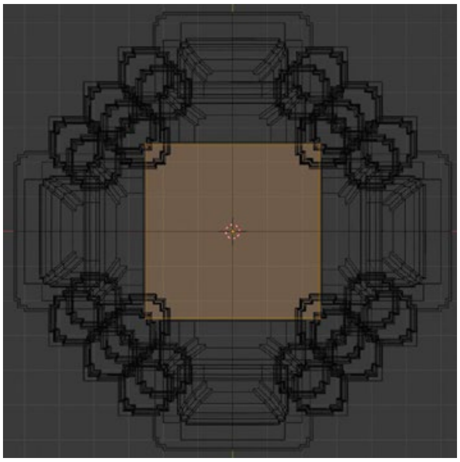


Figure 4-70. *Scaling the cube*

Switch to Front Orthographic view, select the top face of the cube, and extend it up, like you see in Figure 4-71.

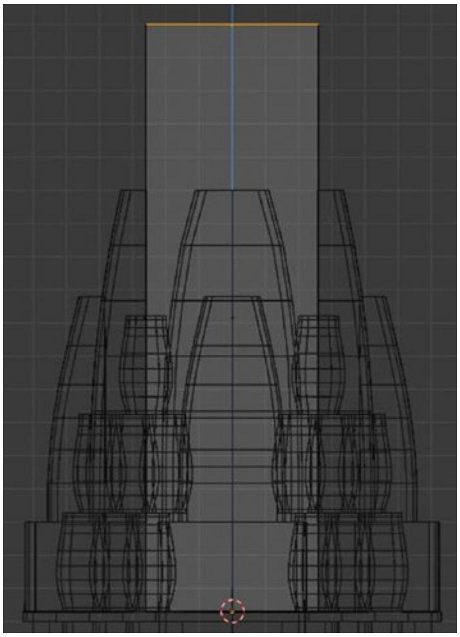


Figure 4-71. *Extending the top face of the cube*

Now you will add an edge loop to the center of the cube by pressing Shift+R to initiate the Insert Edge Loop tool. Left-click to insert one edge and then right-click to confirm the addition of a single edge loop at the center. See Figure 4-72.

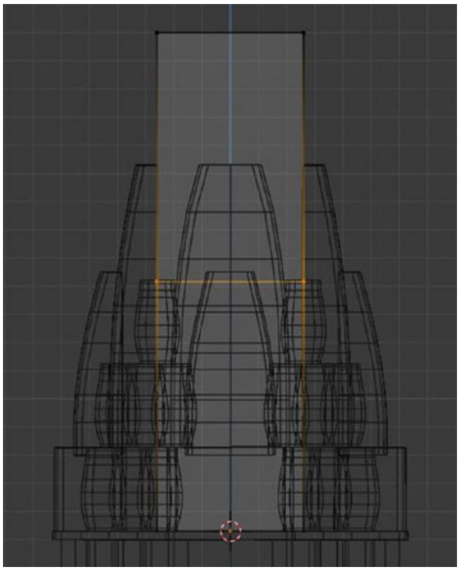


Figure 4-72. *Inserting an edge loop in the center of the object*

Next, you'll add another five edge loops just above the edge loop that you inserted before. You will then scale them individually to get the shape shown in Figure 4-73.

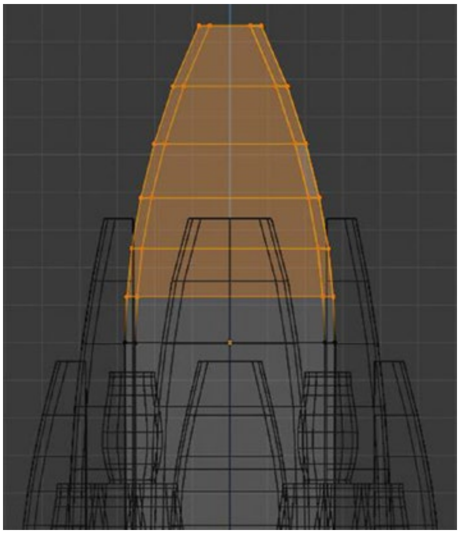


Figure 4-73. *Shaping the cube*

You can see that I have duplicated and placed the "fractal" shapes as well, and you should try to do that yourself. If you are having difficulty then, here is a quick walkthrough—Tab into the Edit mode of the original three that you placed. Duplicate them and move them up, like you see in Figure 4-74.

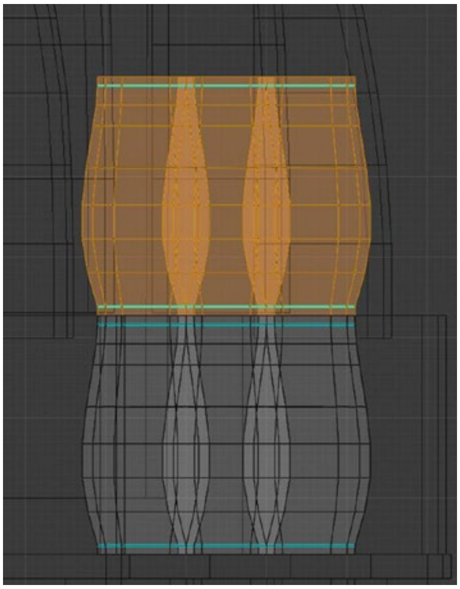


Figure 4-74. *Duplicating and placing the object*

Now switch to Top Orthographic view and move the duplicated shape along the Y-axis by a small amount, as shown in Figure 4-75.

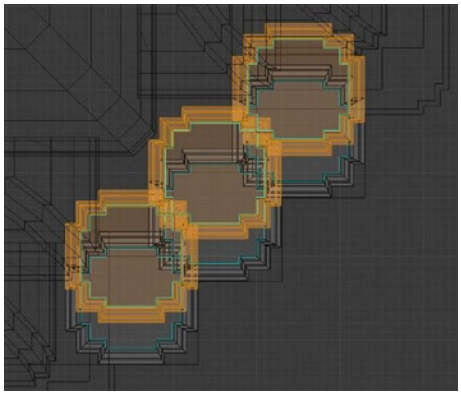


Figure 4-75. *Positioning the duplicated shape*

Then move along the X-axis by the same amount. Finally, select the central “fractal” shape and duplicate it up, and position it in a similar way.

The concludes the blackout part; now you can begin adding details to the blackout in an iterative manner to make them more realistic. Your final blackout should look similar to Figure 4-76.

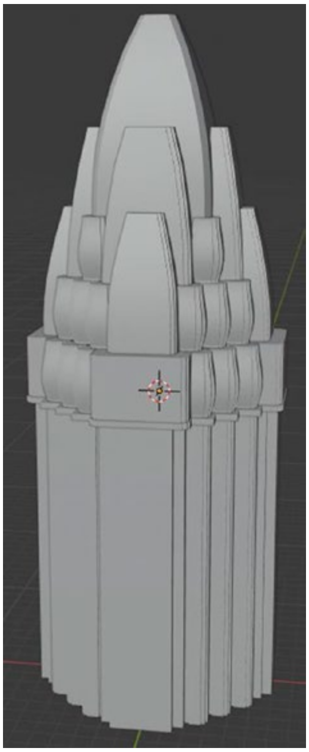


Figure 4-76. *Blockout final*

You can edit the shape and change proportions if needed. The blockout phase is ideal for making big changes, as you are not committed to anything yet. Feel free to move things around as required.

Blocking Out Additional Structures

We will now block out additional structures required for the scene. This is required for getting a good sense of scale and scope of the scene. We will be creating a simple scene with a few additional assets to make things simple and achievable.

Let's start by creating a ground plane for reference. Press Shift+A to open the Add menu. Go to Meshes and choose Plane. Before adding the plane, make sure your 3D cursor is in the World Origin. If it is not, then press and hold Shift+A to open the Snapping menu and choose Cursor to World Origin. After adding the ground plane, scale it up along the X- and Y-axes so that it is wider than it is longer. See Figure 4-77.

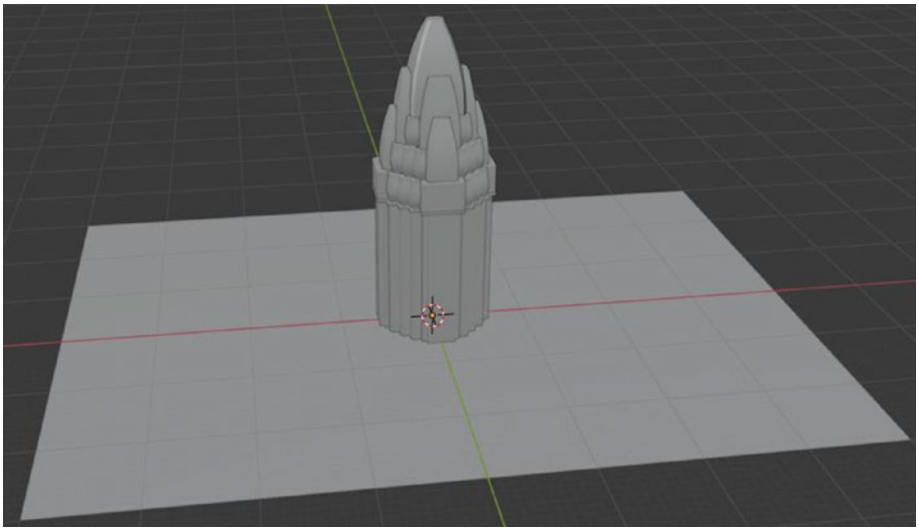


Figure 4-77. *Adding and scaling the ground plane*

Next, add a cube and scale it down to roughly half of its size. With this cube, you will start blocking some additional structures around the temple. We will extrude and scale cubes to create some structures around the temple. Look for some references to get a good idea about what to do. It is always beneficial to look at real-world references to get an understanding of how things are laid out. You will see lots of things that are easy to miss and forget, so don't just depend on your memory to create your scene. Figure 4-78 shows a simple addition added to the blockout.

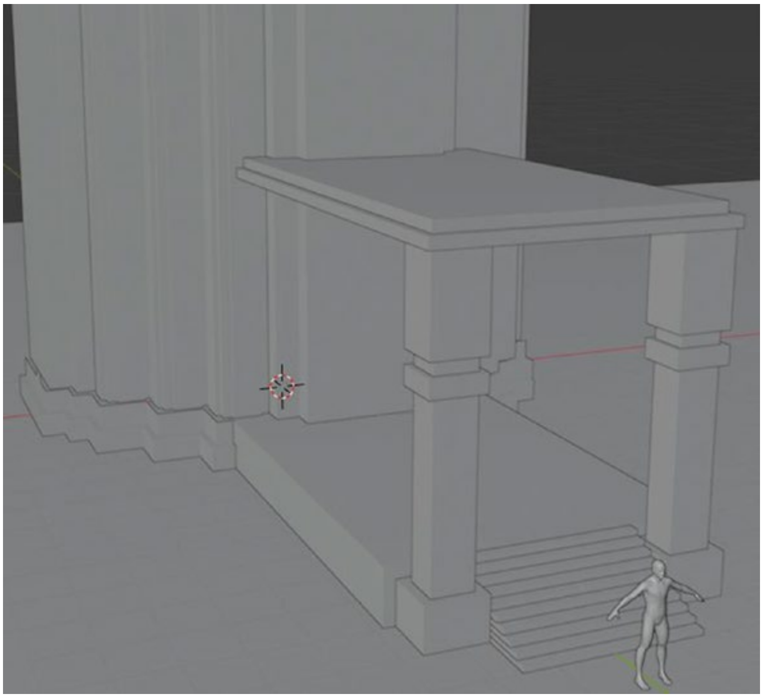


Figure 4-78. *Additional blockout structure*

Try to create something similar by yourself. As you can see, most of the things are simply created by scaling and positioning the cube that we created. As for creating the stairs, here is a quick walkthrough. Create/duplicate a cube and go to its Edit mode to modify its shape from the Top and Side views, as shown in Figure 4-79.

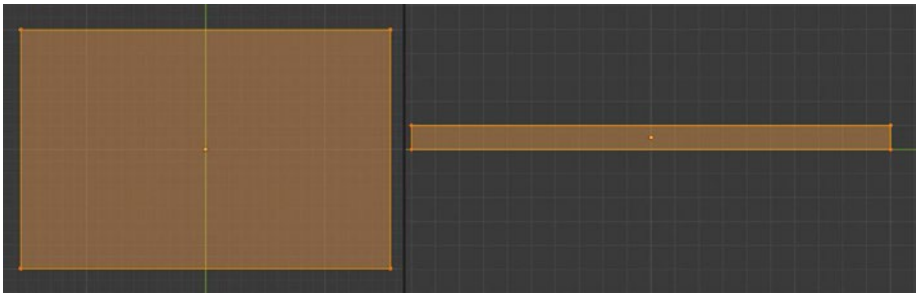


Figure 4-79. *Shaping the cube*

Now duplicate it multiple times along the Z-axis so that it forms a stack. To do this properly, first switch to Front or Side Orthographic view and then duplicate and move up while holding the Ctrl key to use snapping. After doing this once, press Shift+R to repeat the action. Stack up as many as you want according to the required height. See Figure 4-80 for a reference.

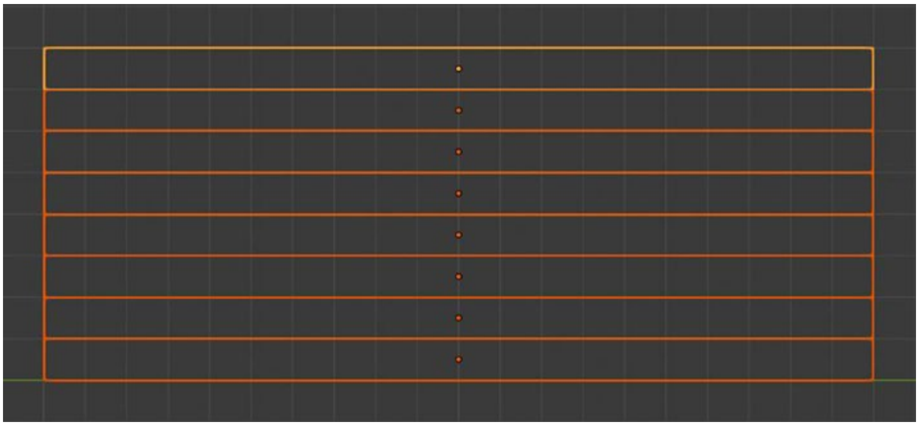


Figure 4-80. *Stacking the stair steps*

We will now go to the Edit mode of each one by one and move the front vertices back to give it the shape shown in Figure 4-81.

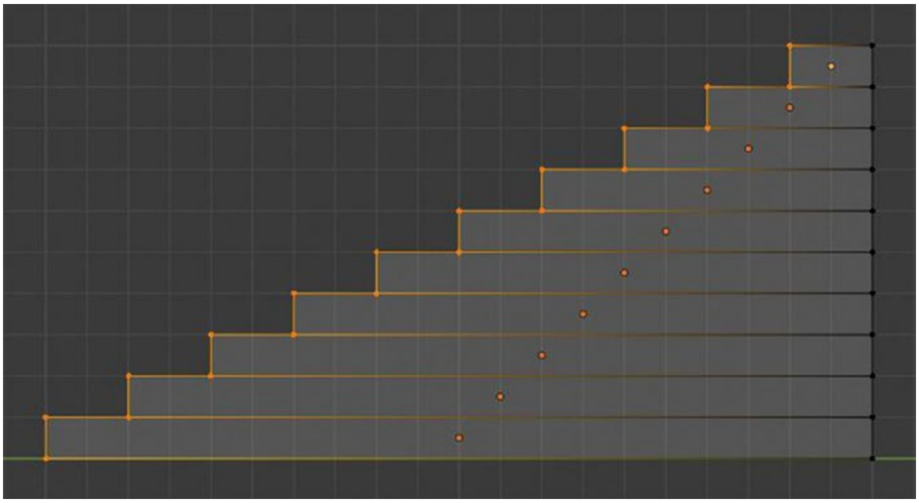


Figure 4-81. *Adjusting the shape of the stairs*

These shapes are the silhouettes of the final shapes and the details that they will contain. Making everything tileable can be hard and perhaps will require a little bit of trail-and-error until you find something that looks correct. The more complex parts of this will start popping up when we start detailing these models.

Let’s create some tree blockouts. Once again, we will begin with a cube. Tab into its Edit mode and try to trace the shape of a tree trunk. Use a real-world image of a tree as a reference. See Figure 4-82.

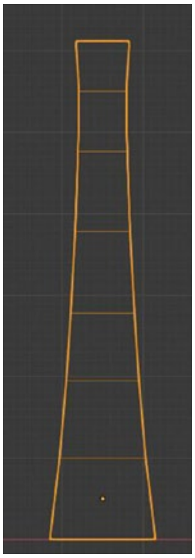


Figure 4-82. *Tree trunk blockout*

Next add another branch by duplicating and reshaping the main trunk. Then add a sphere on top to create the canopy blockout. Subdivide and reshape the canopy. See Figure 4-83.

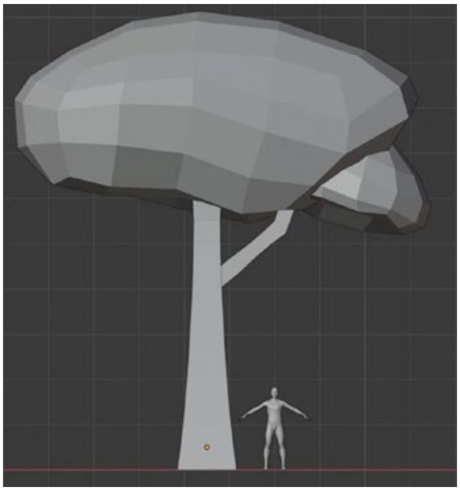


Figure 4-83. *Basic tree blockout*

This is a tree blockout that we will use as a height reference when we create our detailed tree. So, make sure that you have its height the way you want.

Next, let's create some rocks or boulder blockouts that we will scatter around the scene. These blockouts will be more detailed so as to somewhat represent the final form of the rocks. We will use these blockouts when sculpting to add details right on top of them. So, take some time to create these. Check out references of real rocks and try to block out their basic forms.

Add several subdivisions to the cubes so that you have some geometry to work with and deform as you please. Then Tab into the cube's Edit mode, enable Proportional Editing by pressing O, and start deforming your cube to make it look like a rock. See Figure 4-84.

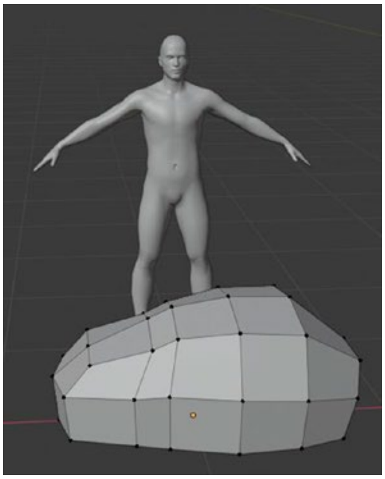


Figure 4-84. *Basic rock blockout*

You can modify this one rock and rotate and scale it around to easily create multiple variations. Rocks have very organic and random shape, so use your imagination. See Figure 4-85.

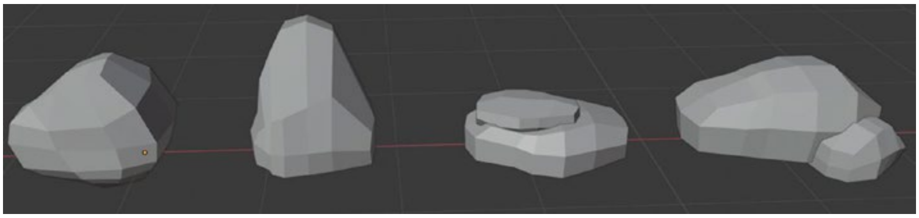


Figure 4-85. *Simple rock variations*

You can add further details when sculpting. You'll use Blender's sculpting tools for that. But for now, this completes the blockout phase; you will now start creating details for all of the assets.

In the next chapter, we will start to create an optimized, game-ready temple. These asset models can then be used within any game engine and VR integration.

CHAPTER 5

3D Design Visualization

In this chapter, we are going to start detailing the temple. We created the blocking for our model in the last chapter. Using that blocking as a base, we will start adding more polygons and shape the temple in a more realistic way. You should do the detailing stage after you are confident with the blocking stage. Once you have added more edges and perhaps used the Subdivision Surface modifier, the original will be very hard to get back. So make sure that you have finished the low-poly modeling stage before proceeding.

Detailing the Temple

We will start with the temple pillars. Select the pillar structures that you created and Tab into the Edit mode. Then select one of them and separate it. If you already separated the pillar models, you're ready for the next step, which is to create some edge loops, as you can see in Figure 5-1.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_5

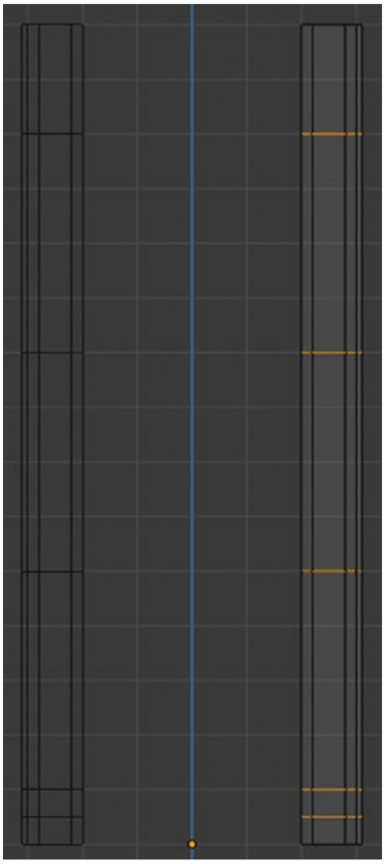


Figure 5-1. *Adding edge loops to the pillar*

Next, select the bottom two rows of the faces and extrude them out slightly, as shown in the Figure 5-2. Make sure that you are using the Extrude Along Normals tool and holding the Alt key while performing extrude to extrude evenly on all sides. Then delete all the extra faces that won't be visible, like the faces on the bottom of the pillar and toward the sides.

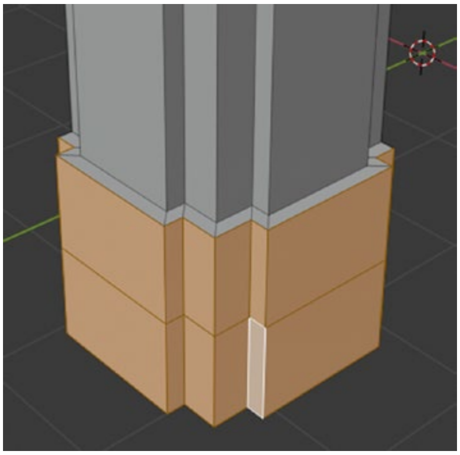


Figure 5-2. *Extruding the faces*

Now select the bottom and extrude it in a similar way. Make sure to delete all the extra faces that won't be visible. Figure 5-3 shows which faces to delete.

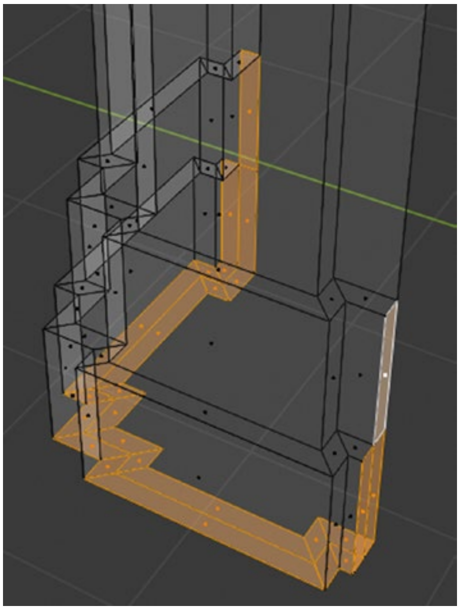


Figure 5-3. *Select and delete these faces*

It is important to delete any unnecessary faces to not only optimize the model but also save UV space when unwrapping. Optimization for game engines is the main reason that we remove the extra faces, to keep the vertex count to a minimum.

Next, you'll add some more edge loops, as shown in Figure 5-4.

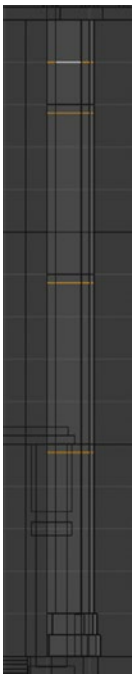


Figure 5-4. *Adding edge loops*

Next, select the faces enclosed by these edge loops and scale them out by a small amount. See Figure 5-5 for reference.

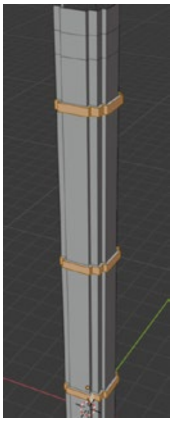


Figure 5-5. *Extruding the created faces*

The extruded faces create some unwanted faces on the sides, so make sure to delete them. Also, the extruded region may look a bit too narrow at the moment. Make sure to extend it further if necessary. In this case, we can extend the region down a bit to make room for details we'll add during the sculpting process.

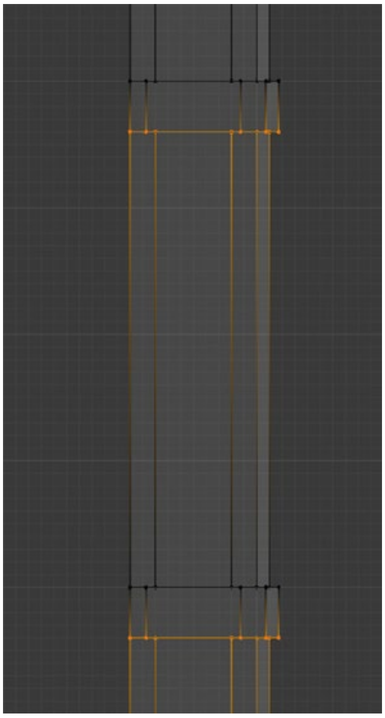


Figure 5-6. *Extending vertices down*

In Figure 5-6, you can see that we extended the vertices down by selecting them in the orthographic view. How much you extend them depends on the type of shape that you are trying to model. You can proceed to extrude the top two rows of faces, as you did with the bottom ones. See Figure 5-7 for reference.

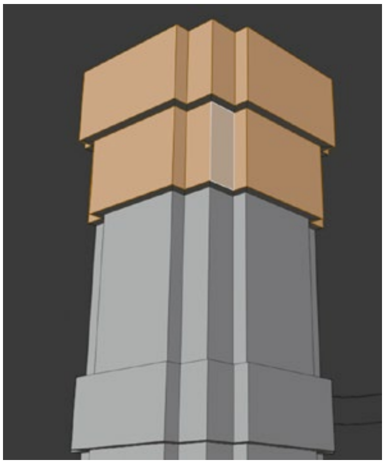


Figure 5-7. *Extruding the faces*

You can now duplicate this pillar piece and place it appropriately, preferably from the top orthographic view, as shown in Figure 5-8, so that you can see where you are copying it.

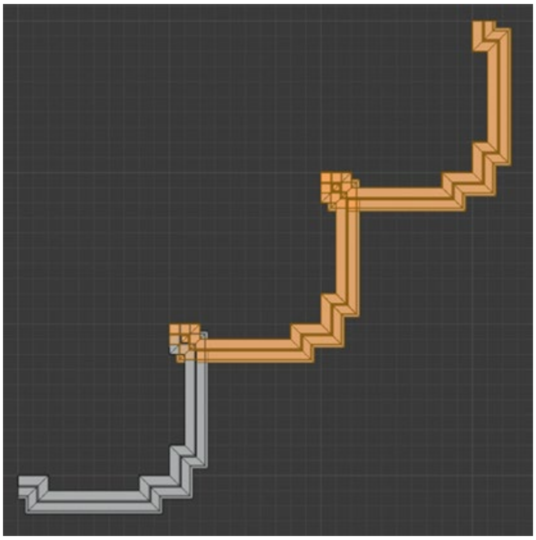


Figure 5-8. *Duplicating the pillar object*

Next let's add details to the main temple body. We will begin by adding a couple of edge loops to the body of the temple, as shown in Figure 5-9.

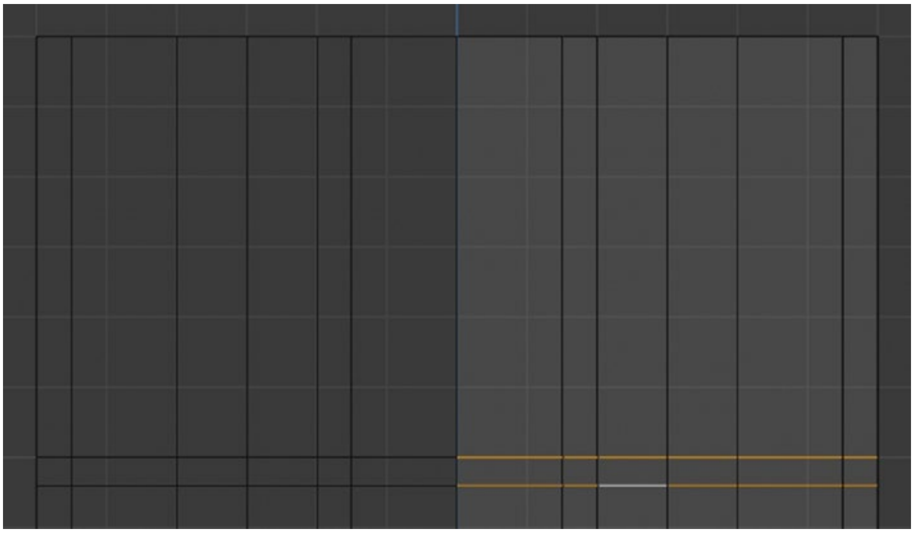


Figure 5-9. *Adding edge loops to the temple*

Let's select the inner faces shown in Figure 5-10 and delete them, as they won't be visible when we add additional geometry to them. Also, deleting them will prevent overlapping issues when we extrude our faces in the next step. It is good practice to remove unwanted geometry from your meshes, as it will keep them simpler.

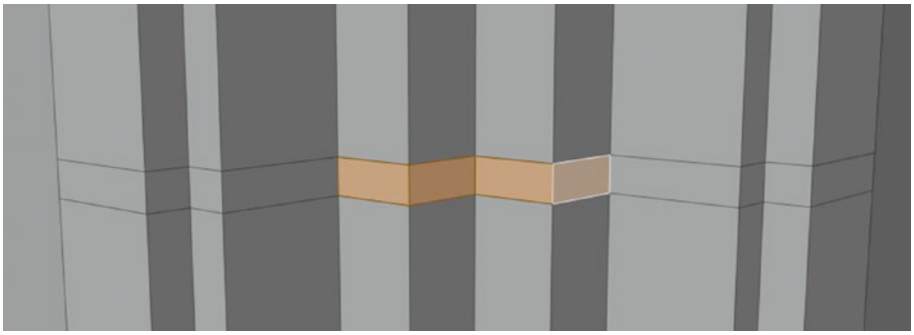


Figure 5-10. *Select and delete these faces*

Let's align these edges with the extrusion on the pillars. Select each edge one by one and, from any orthographic view, move the edges to align them with the extrusions on the pillars. Do this by pressing G and holding Ctrl to snap. Your result should be similar to Figure 5-11.

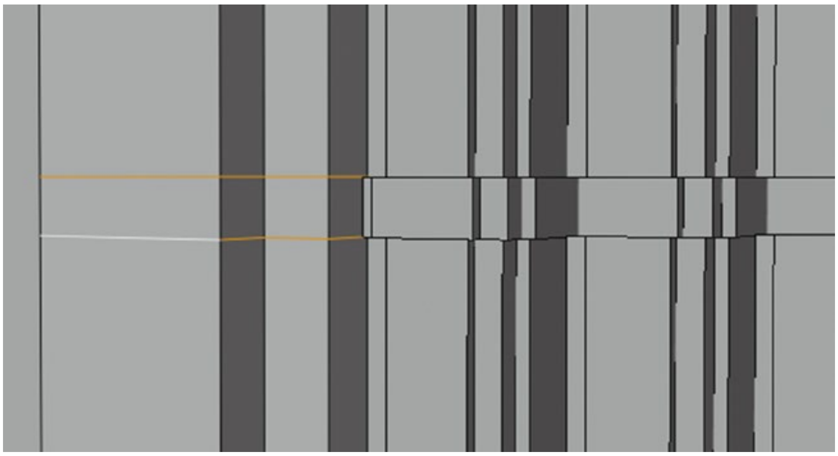


Figure 5-11. *Aligning edge loops with extrusion*

After that, select that entire row of faces and extrude them, as can be seen in Figure 5-12.

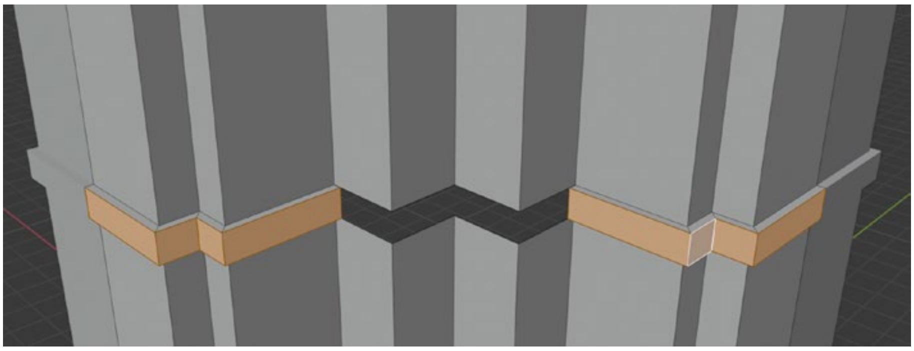


Figure 5-12. *Extrude the selected faces*

Now we will do the same thing, but a bit lower this time. We will add a couple of edge loops matching the bottom extrusion on the pillar, as shown in Figure 5-13.

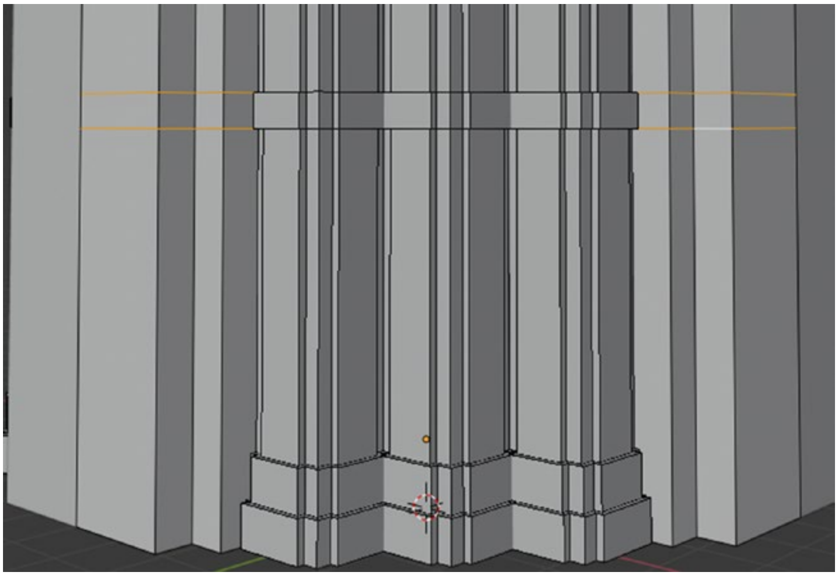


Figure 5-13. *Adding edge loops aligned to the extrusion*

Before we extrude, let's delete all the faces on our main temple body that are not visible. See Figure 5-14 for which faces to select and delete.

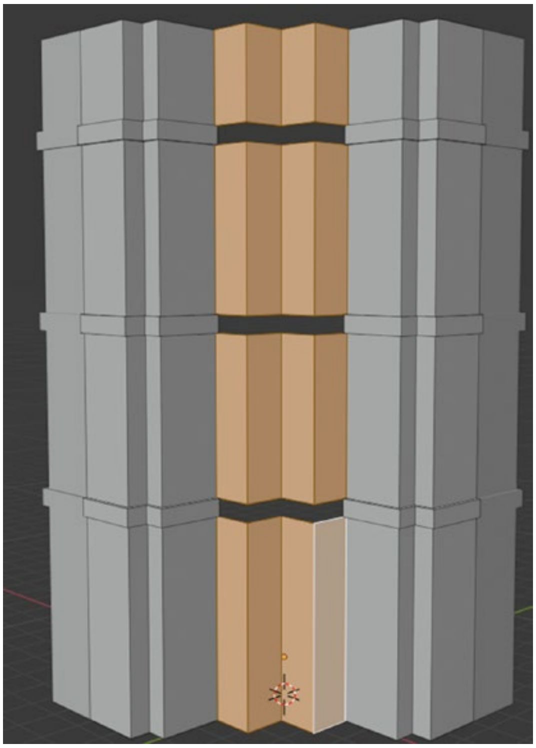


Figure 5-14. *Select and delete these faces*

Let's also add a couple of edge loops toward the top, aligned with the extrusions of the pillar. See Figure 5-15.

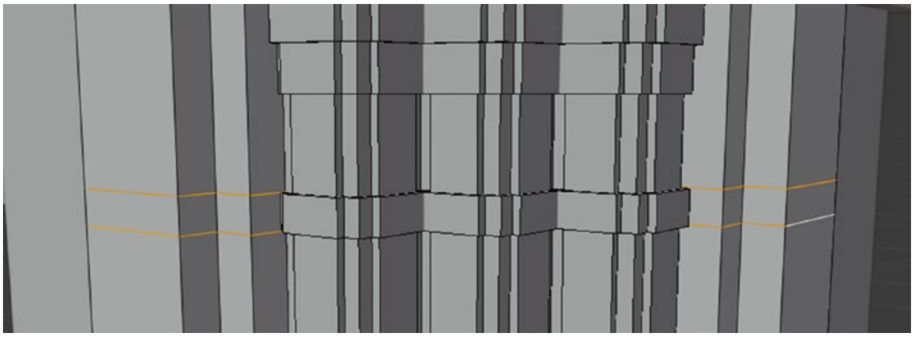


Figure 5-15. *Adding edge loops*

We will extrude all the faces enclosed by both the top and bottom edge loops. Your final output should be similar to Figure 5-16.

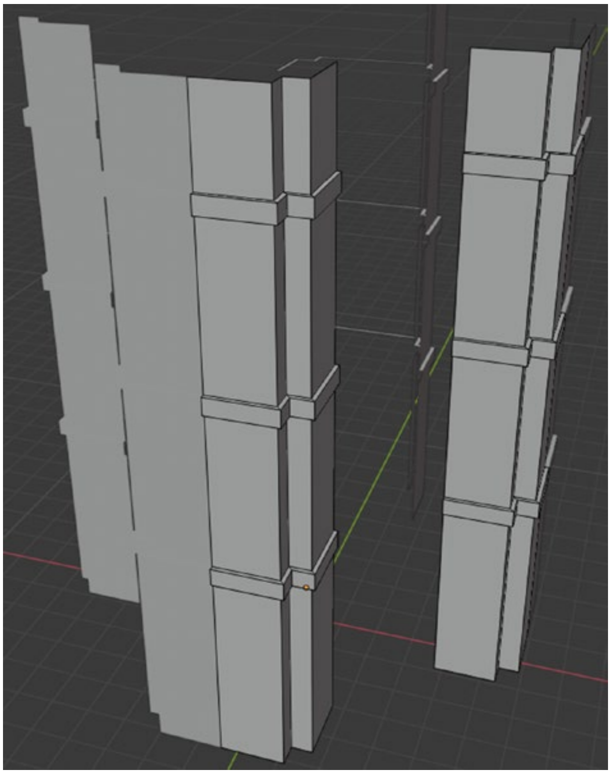


Figure 5-16. *Results after extrusion*

We will now create a door structure toward the bottom of the temple body. We will begin by performing an Inset operation on the face. To do this, first select the face and then press I to start the Inset operation.

Then type `0.36` (modify this amount of needed) to manually set the inset amount. You should have a result similar to Figure 5-17.

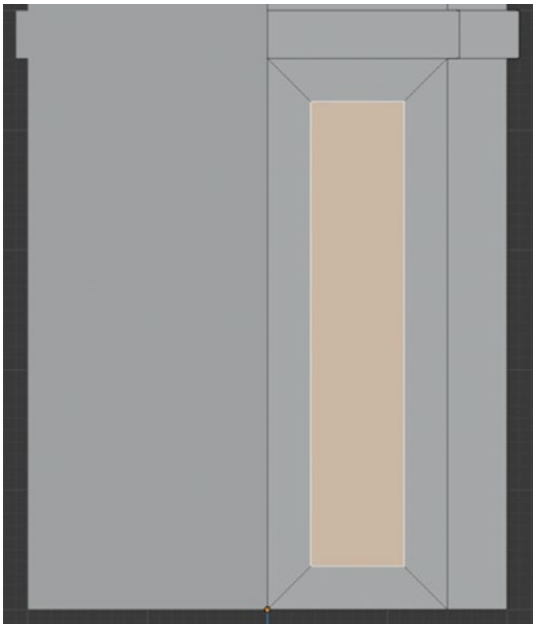


Figure 5-17. *Performing an Inset operation on the face*

Next, select the faces shown in Figure 5-18 and delete them to create a hole.

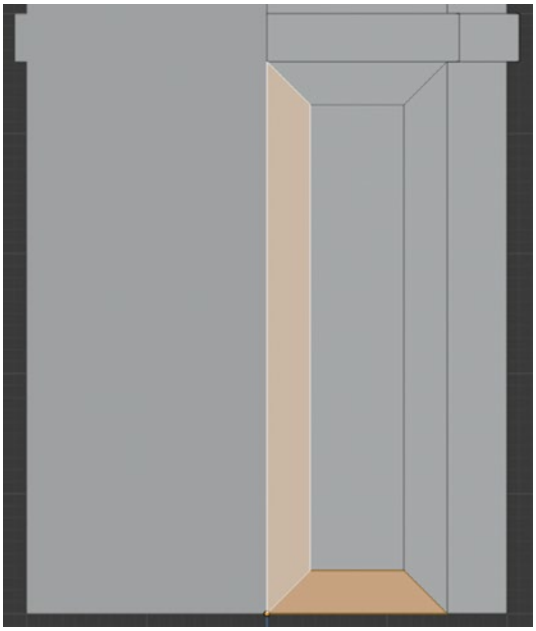


Figure 5-18. *Select and delete these faces*

After that, select the left edge and move it to the mirror center. Select the bottom edge to align it with the other bottom edges. This should get rid of the gaping hole.

Then feel free to adjust the position of the top edge of the door shape until you are happy with the height of the door. Your final output should be similar to Figure 5-19.

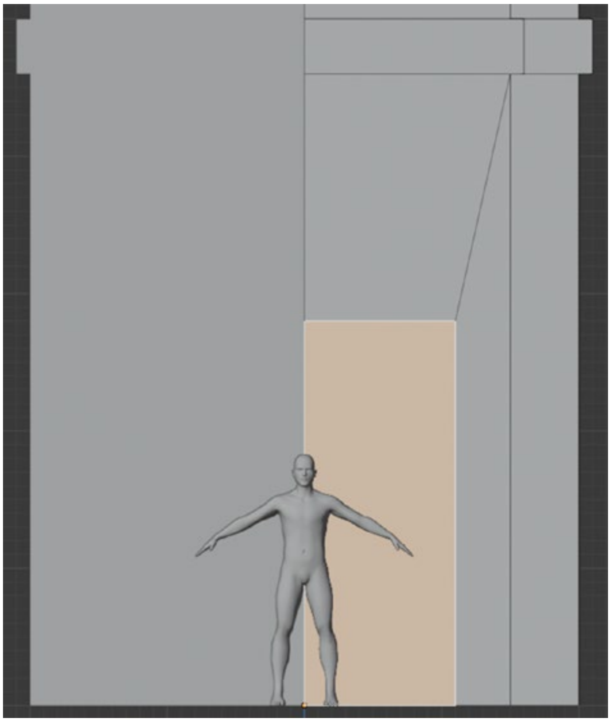


Figure 5-19. *Dimensions of the door*

Next, we will delete this face. Select the two boundary edges and duplicate them by pressing Shift+D. Right-click to reset their position to their original place. Press P to open the Separation menu and choose Selection to separate the shape based on your selection. See Figure 5-20.

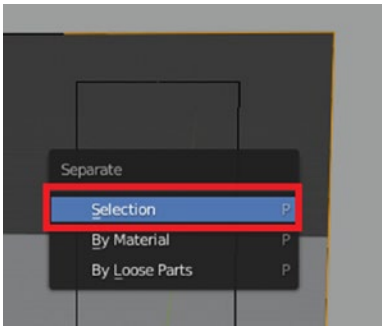


Figure 5-20. *Separating by selection*

Make sure to adjust the width of the door according to what looks suitable. After that, select the shape that you separated from the door and Tab into its Edit mode. Position it as shown in Figure 5-21.

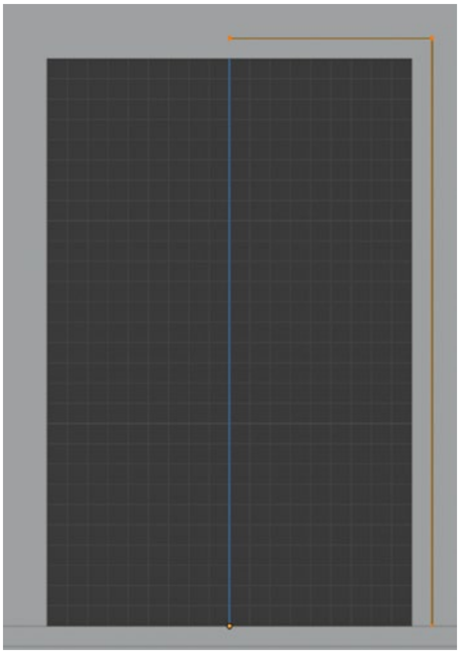


Figure 5-21. *Repositioning the vertices of the shape*

Make sure the edges of the shape are selected and extruded. Reshape the extruded region somewhat, as shown in Figure 5-22.

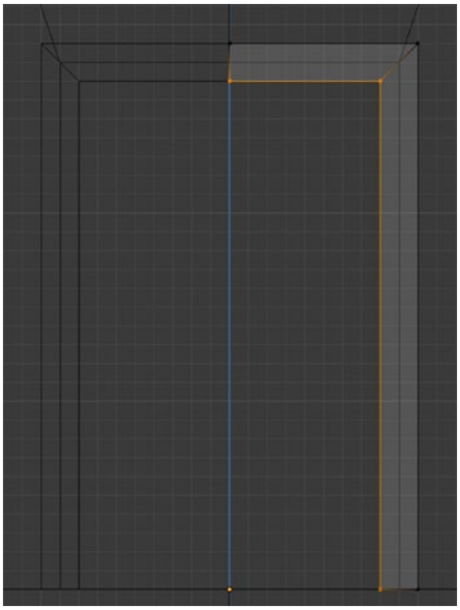


Figure 5-22. *Extruding the shape*

Next, select all the faces of the shape and extrude them out. See Figure 5-23 for reference.

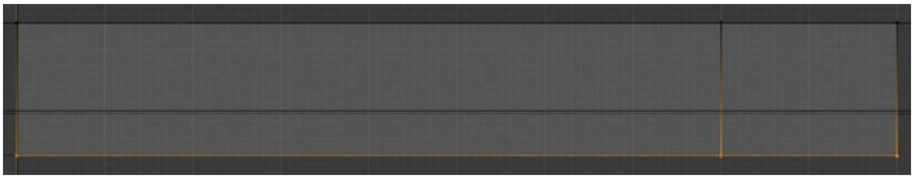


Figure 5-23. *Extruding the shape*

Select the back and bottom faces and delete them because they won't be visible. Now select the inner edge and extend it back, as seen in Figure 5-24.

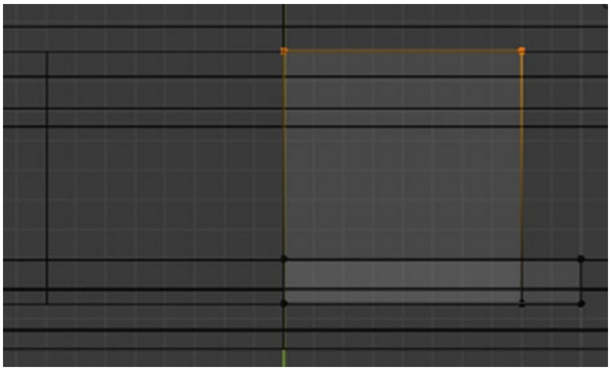


Figure 5-24. *Extending the edges back*

We will create the interior of the temple using this. But that is for a little later. First, we will flesh out the exterior some more before we proceed inside. We will now focus on the top of the temple, which is going to be extremely complex. We add more details to it.

Select the faces shown in Figure 5-25 and separate them by selection, by pressing P.

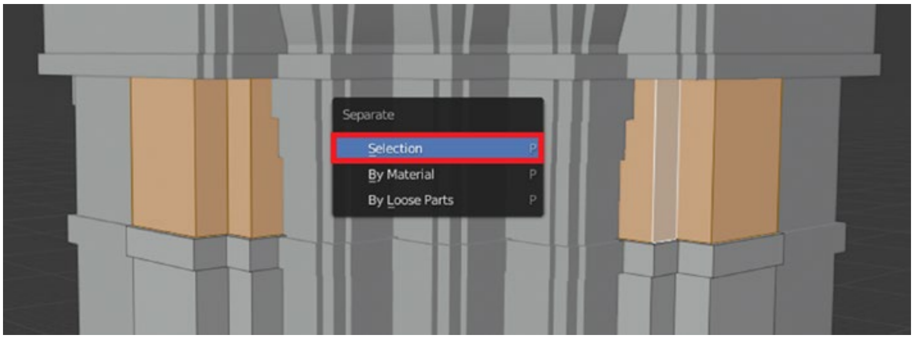


Figure 5-25. *Separate these faces by selection*

Next, add a couple of edge loops to both pieces of the separated shape, as you can see in Figure 5-26.

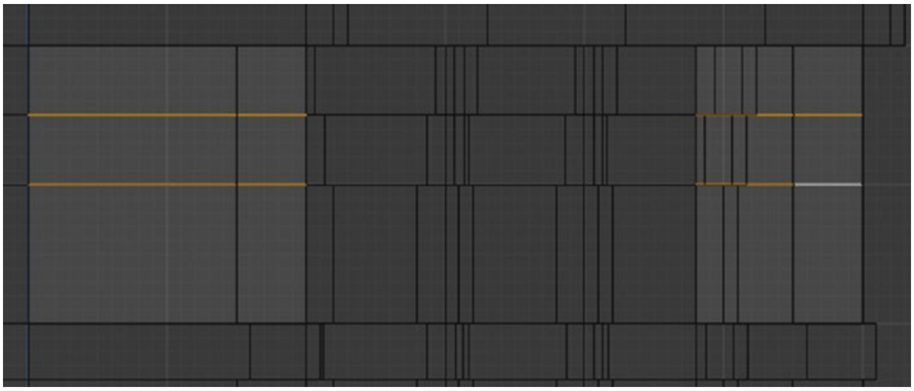


Figure 5-26. *Adding edge loops to the object*

Next, add one edge loop vertically to both the objects, as you see in Figure 5-27. You can eye the distance from the center and keep it to what you feel is okay. But keep enough space, as we will be extrude the faces. If there isn't enough space, then the faces will start overlapping.

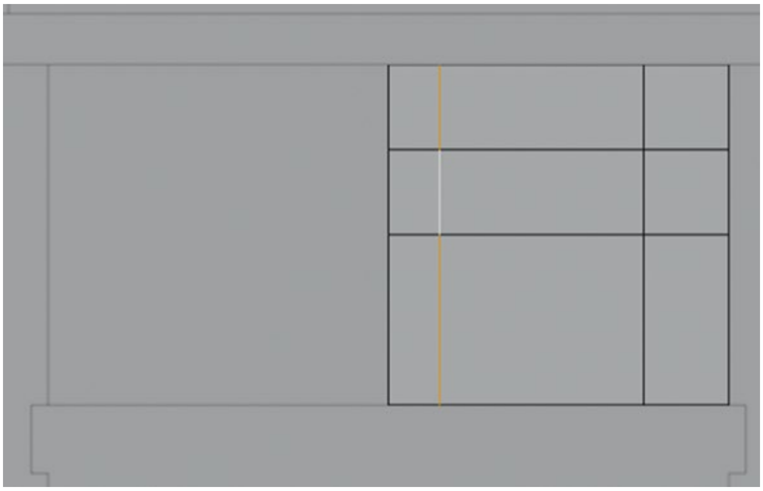


Figure 5-27. *Adding vertical edges*

Make sure you add edge loops to the other side as well. Next, select the faces created by inserting edges and extrude them inside, as shown in Figure 5-28.

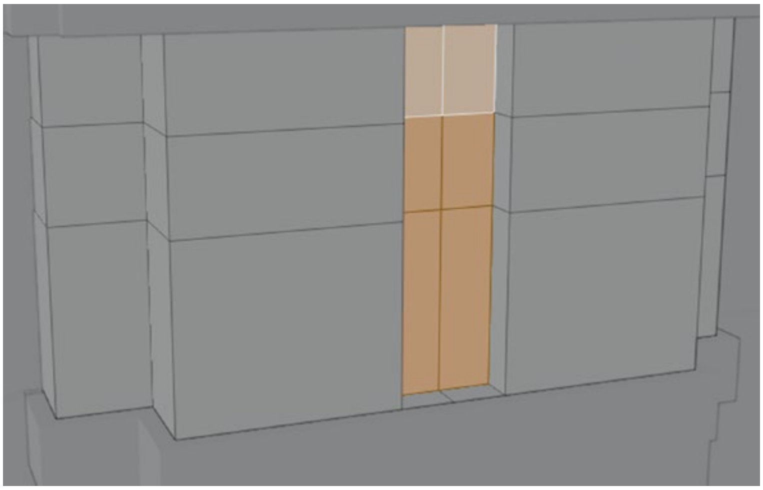


Figure 5-28. *Extrude the face inside*

Delete the unwanted faces created because of the extrude operation. Next, select the top two rows of faces and extrude them out, as shown in Figure 5-29.

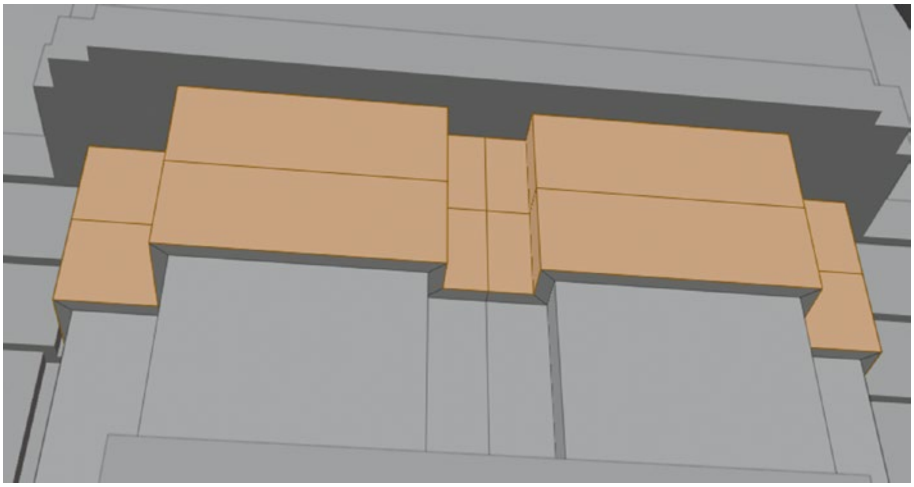


Figure 5-29. *Extrude the top rows of faces*

Select only the top row of the faces and extrude them out in a similar way. After that, add two edge loops vertically to both pieces, like you see in Figure 5-30. Make sure you add to both sides and align them to the grid to make them proportional. Hold the Ctrl key while moving the edges to snap them to the grid.

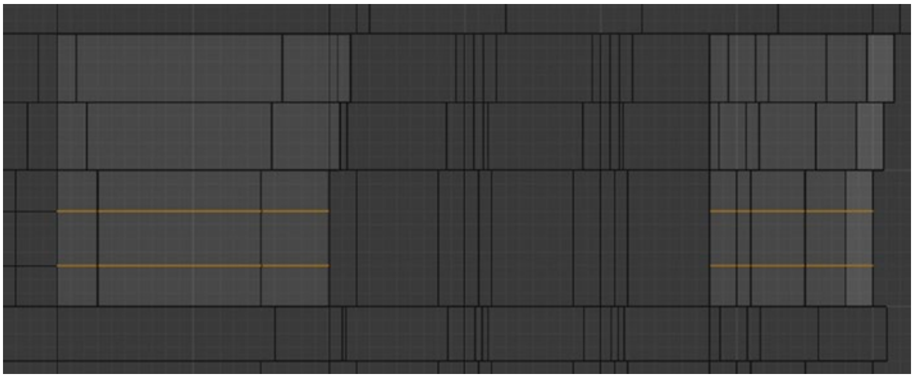


Figure 5-30. *Adding horizontal edge loops to the shape*

Now, select the faces enclosed between these edge loops and extrude them out. In the same way, add two edge loops to all the pillar pieces and extrude them out exactly as you did with the previous object. Your results should be similar to what is shown in Figure 5-31. Your results don't have to be exact, but something similar to this would be good.

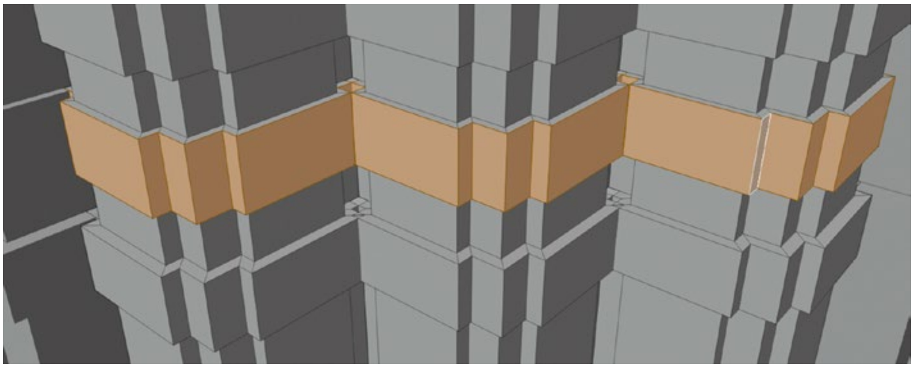


Figure 5-31. *Extrude the faces of the pillar*

Let's move along and start detailing the top structures. We will be detailing the "fractals" and other elements making up the spire.

Modeling Additional Assets

Select a fractal piece and make sure it's a unique separate object. We will create one and duplicate it all around. Separate it by selection if it is part of a larger mesh. When you have it as a unique object, Tab into its Edit mode and select its very top face. Enable Proportional Editing by pressing O on keyboard or by clicking on its icon. Then click on the Proportional Editing Settings drop-down menu and choose Sharp as the falloff. See Figure 5-32 for reference.

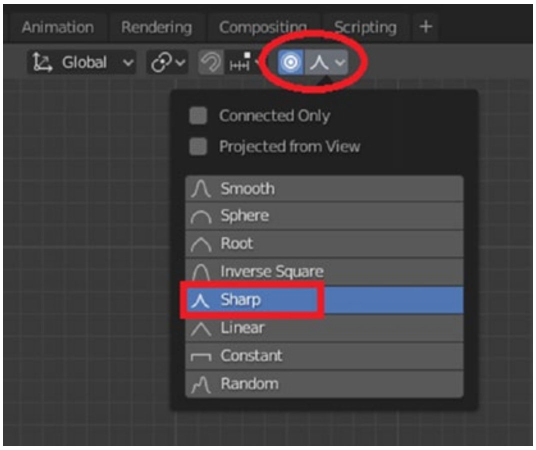


Figure 5-32. *Selecting the falloff mode of the Proportional Editing tool*

With the top face of the "fractal" selected, press S to initiate the Scale tool, then press Shift+Z to constrain scaling only along the X and Y axes. Increase the brush size of the Proportional Editing tool by scrolling the mouse wheel and then scale it down, as shown in Figure 5-33.

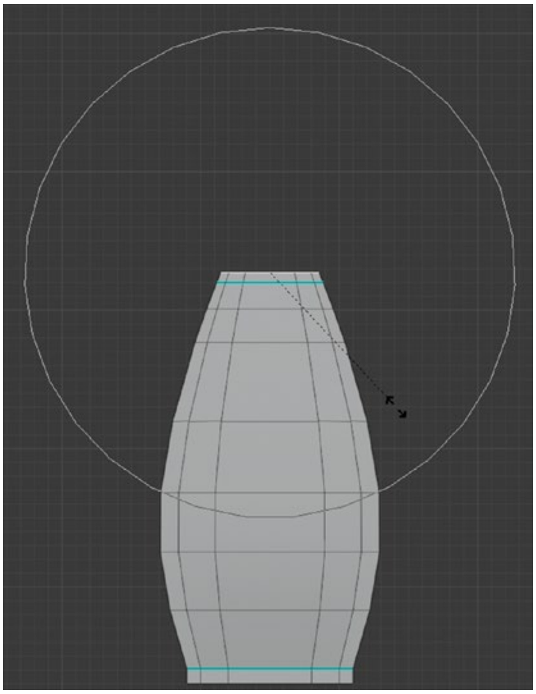


Figure 5-33. *Scaling the shape down*

Click to confirm when you are satisfied with the shape. Check out some reference images for guidance regarding the shape of the fractal.

Select the top face of the fractal, then press and hold Shift+S to open the snapping pie menu. From the menu, choose Cursor to Selected, as shown in Figure 5-34.

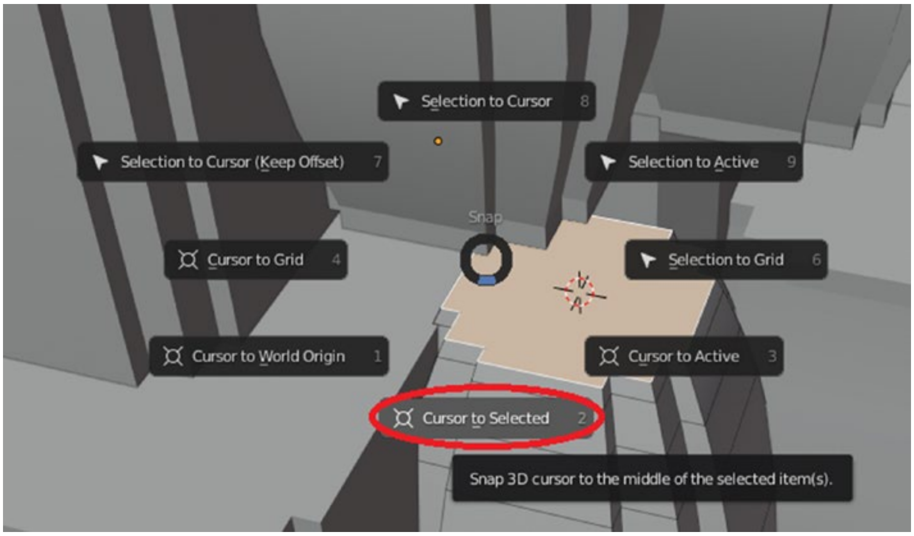


Figure 5-34. *Snapping the cursor to the selection*

Now, if you create anything, it will be placed on the exact top center of the fractal. Tab out of Edit mode so that you can add some objects. Press Shift+A to open the Add menu, and then go to Mesh and choose Circle. See Figure 5-35.

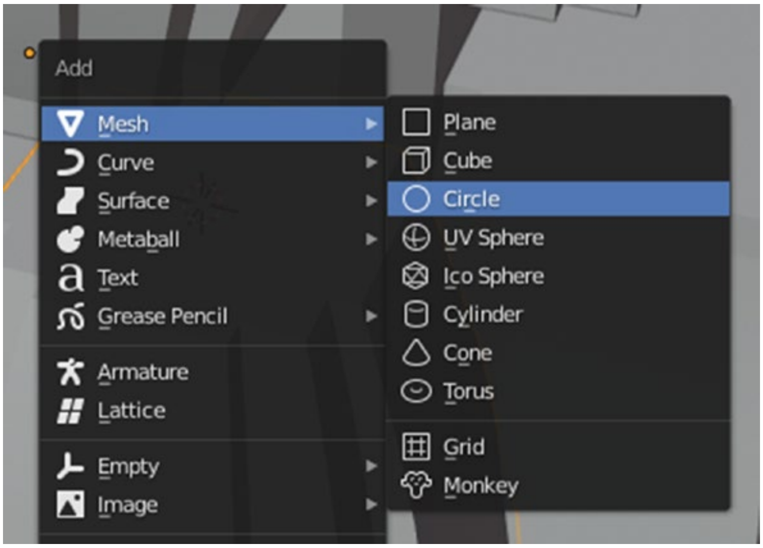


Figure 5-35. *Adding a circle*

We will extrude this circle to the top and create a design to extend the top of the fractals. Once again, check out some reference images to see what the top should look like. You can search for terms like "temple sketch" in Google images to find some 2D references.

Once you have found something usable, download it where you can easily find it. Switch to an orthographic view before importing the image. After importing the image, it will be aligned to the view camera. So, it is very possible to have an image imported in an awkward or undesirable angle. The next thing that you need to do is press Shift+A to open the Add menu. In the list, go to Image ➤ Reference (see Figure 5-36).

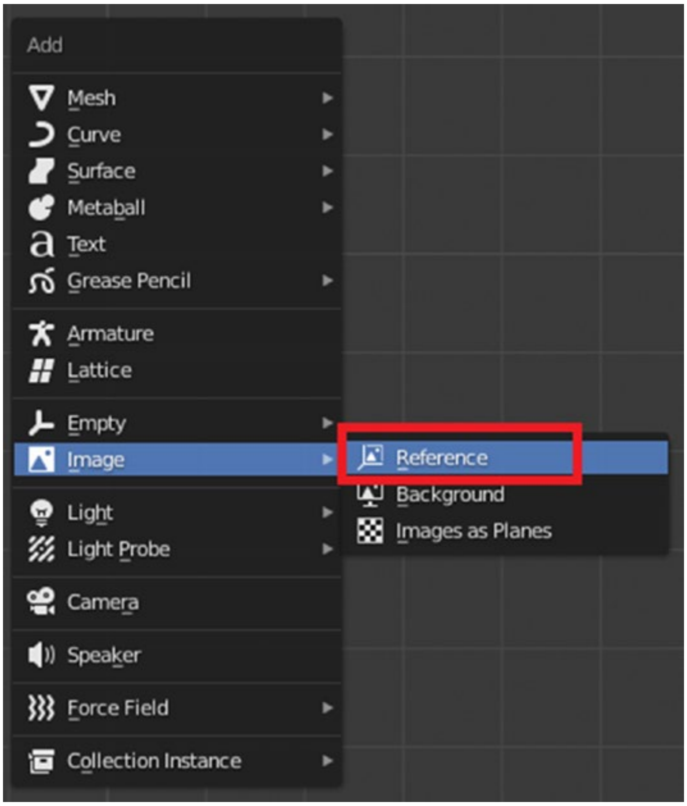


Figure 5-36. *Adding an image as a reference inside Blender*

I am going to use the image shown in Figure 5-37 as the reference for getting the shape. This image is from Wikimedia commons and can be found easily using a Google image search.

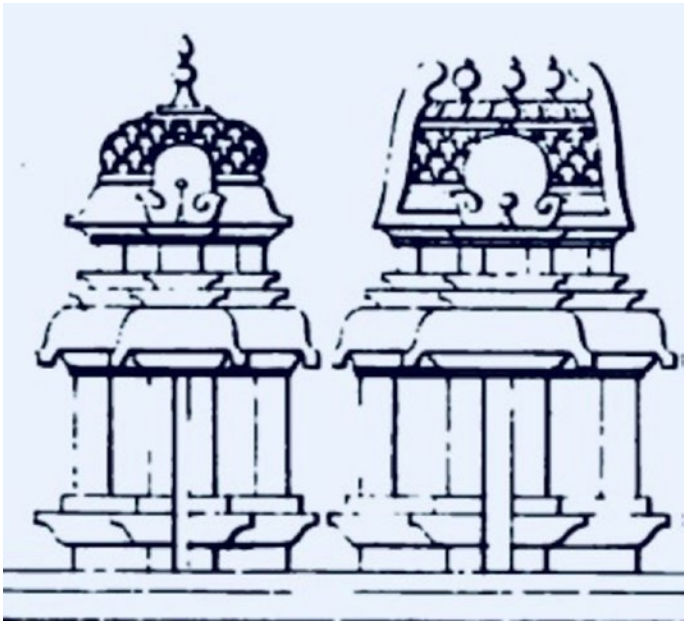


Figure 5-37. *Reference for a fractal shape*

Start by scaling your image so that you find it easy to work with. Then use a vertex to draw the shape you see in Figure 5-38 from the top, orthographic view.

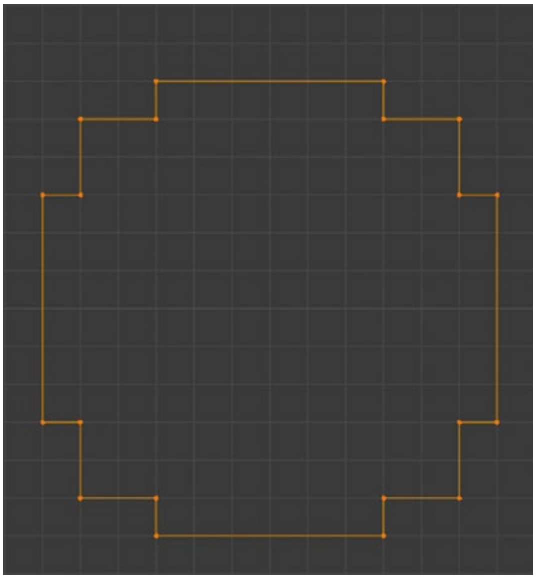


Figure 5-38. *Starting shape*

Next, switch to the front orthographic view and extrude your vertices up while tracing the shape that is visible in the reference image, as shown in Figure 5-39.

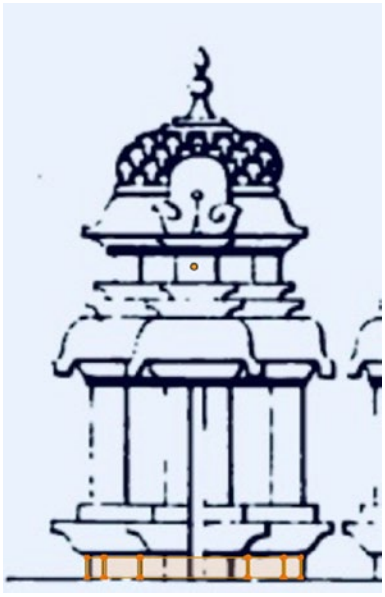


Figure 5-39. *Extrude the vertices*

We will once again extrude our vertices up, but as you may notice now, we will need to scale our vertices in order to match the shape. So, extrude and scale out several times until you have the shape traced out, as shown in Figure 5-40.

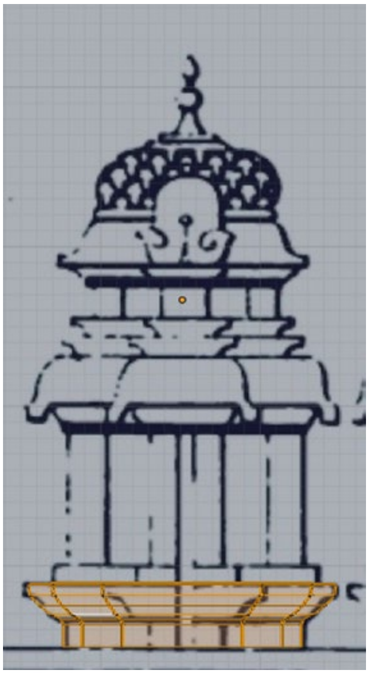


Figure 5-40. *Tracing the shape in the reference*

Continue extruding and scaling in order to trace the shape. Scale in and out and follow the topology of the shape in the reference image. Trace the shape as accurately as you want to. There is no compulsory requirement to have the exact same shape as the reference image, as you can see in Figure 5-41.

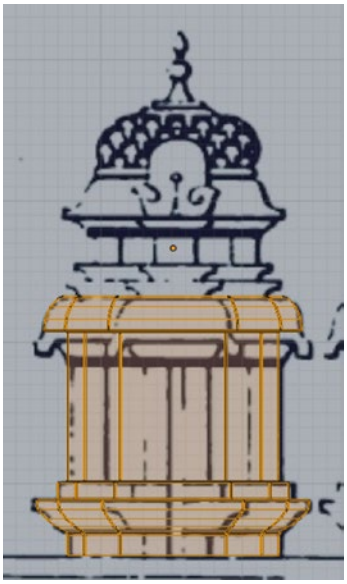


Figure 5-41. *Tracing the reference image*

Keep scaling and performing the extrude operation until you have reached the top of the shape in the reference image. Your final result should look somewhat similar to Figure 5-42.

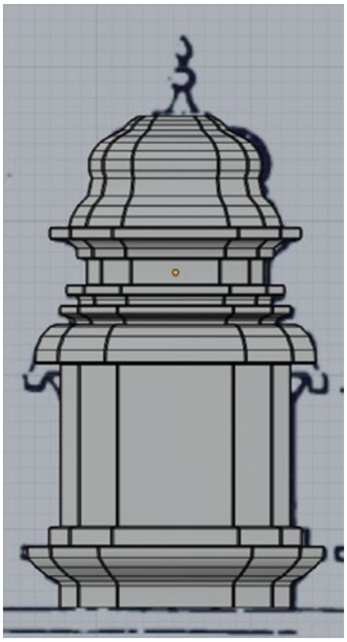


Figure 5-42. *Full trace of the shape*

Now duplicate the top border edge loop and right-click to open the Edge Context menu. From the menu, go to LoopTools and choose Circle from the list. If you don't see LoopTools then you don't have the add-on enabled. It is an external add-on that needs to be enabled from Preferences ➤ Addons. Extrude the rounded shape up and trace the top design in the reference image, as shown in Figure 5-43.

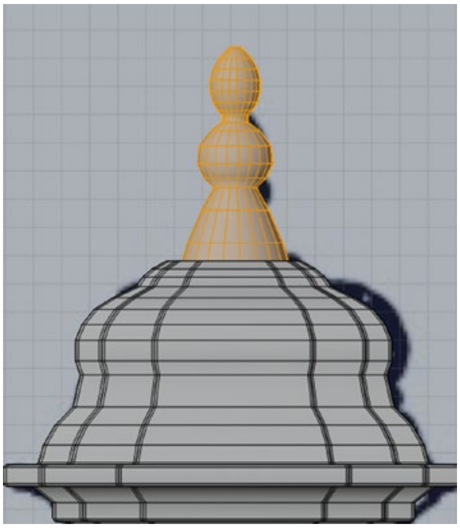


Figure 5-43. *Tracing the top design*

Once you're done with the shape, you can select and delete the reference image because you won't be needing it anymore. Place the object that you created as shown in Figure 5-44.

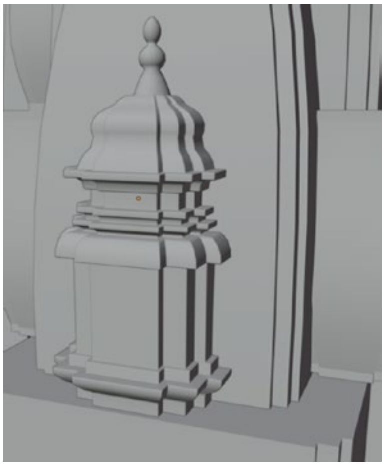


Figure 5-44. *Placing the fractal object*

Next, duplicate the object once and select the top faces shown in Figure 5-45. Press P and separate all the selected faces by selection. After separating the top, delete the bottom object consisting of the remaining faces.

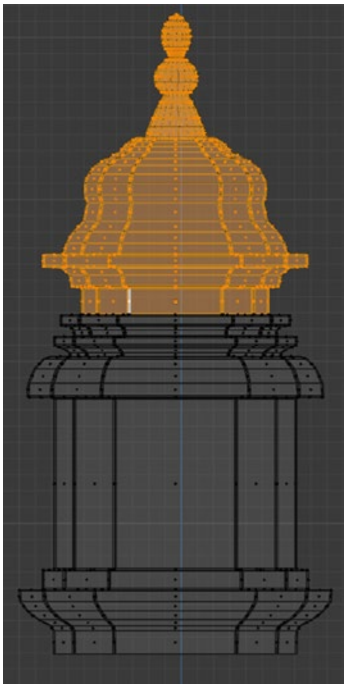


Figure 5-45. *Selecting and separating the faces*

Now place the Cap shape on top of one of the larger shapes, as shown in Figure 5-46.

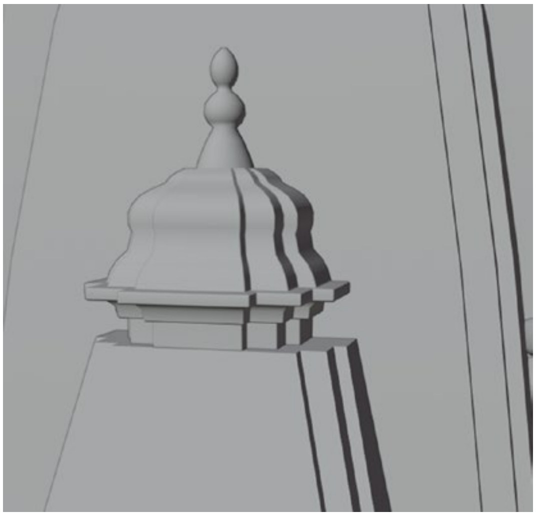


Figure 5-46. *Placing the cap shape*

We will also place the cap on top of the fractals. This will add some detail to them. We will also make some modifications to the fractals. Select the fourth edge loop from the bottom and bevel it (see Figure 5-47).

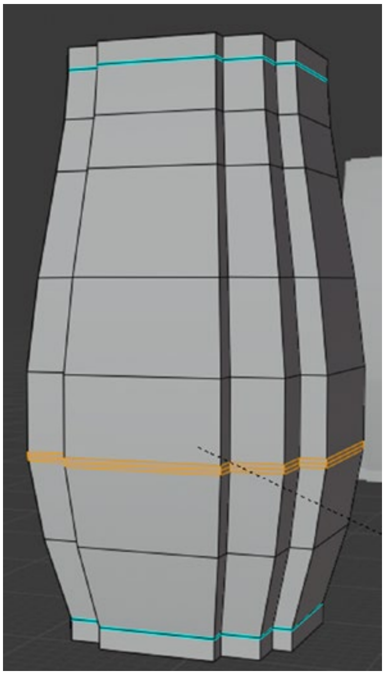


Figure 5-47. *Beveling the selected edge*

As you can see in Figure 5-47, you are beveling the edge with two additional segments and by a small amount. Next, select the middle edge of the bevel that you just created, then press Alt+S to start shrinking the selected edge loop. Shrink until the edge is pushing inside the mesh, something similar to Figure 5-48.

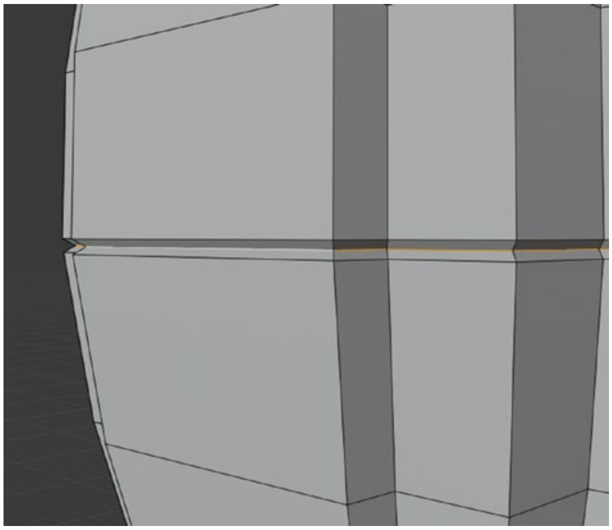


Figure 5-48. *Shrinking the edge loop*

This will break the flatness of the shape and give the fractals an interesting look. After this, duplicate the cap object and place it on top of the fractal object, as shown in Figure 5-49.

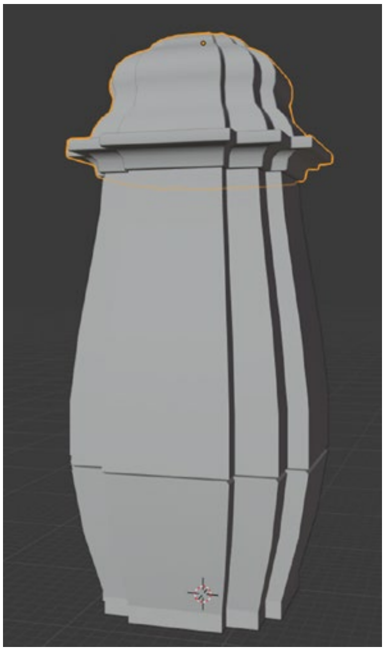


Figure 5-49. *Place the cap on top of the fractal*

That will be the final form of the Fractal object. Now we will duplicate the shape and replace the placeholder objects with the new detailed object. Your result should look something similar to Figure 5-50.

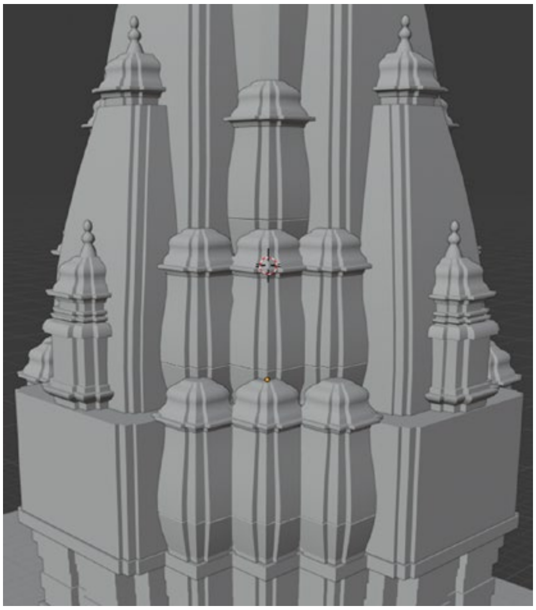


Figure 5-50. *Duplicating and placing the fractal*

Also duplicate the cap shapes and place them around the temple so that your final temple top is something similar to Figure 5-51.

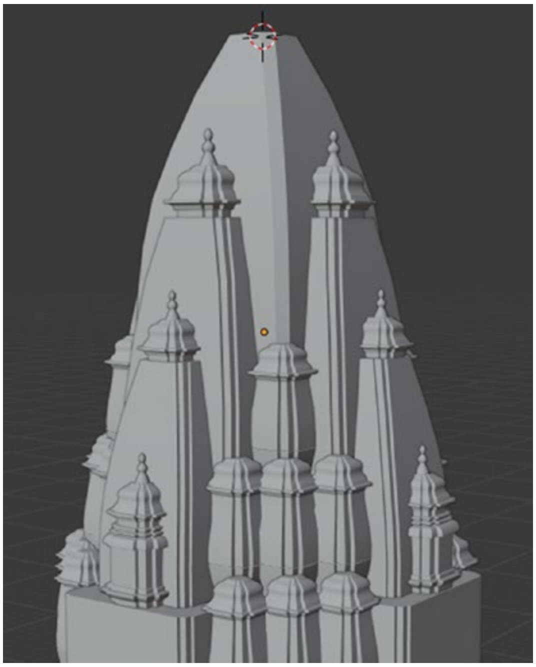


Figure 5-51. *The temple spire so far*

Now let's finish the top part of the temple by creating the spire. The spire design of the temple is just another geometric shape with lots of extrusions and bevels. We can once again use a reference image, as it is really important to know what a real one looks like. Start by creating a circle exactly on top of the top-most part of the temple, as shown in Figure 5-52.

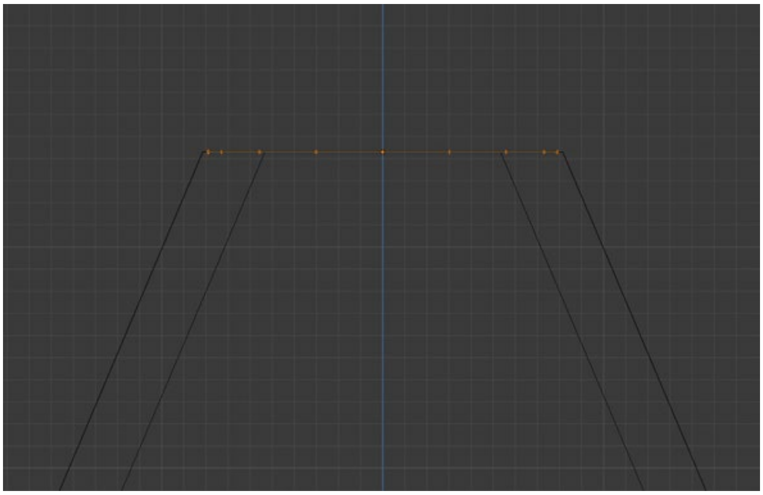


***Figure 5-52.** Adding a circle shape to the object*

Now extrude up and shape the object according to the reference image that you have. Figure 5-53 shows the final outcome of the shape.

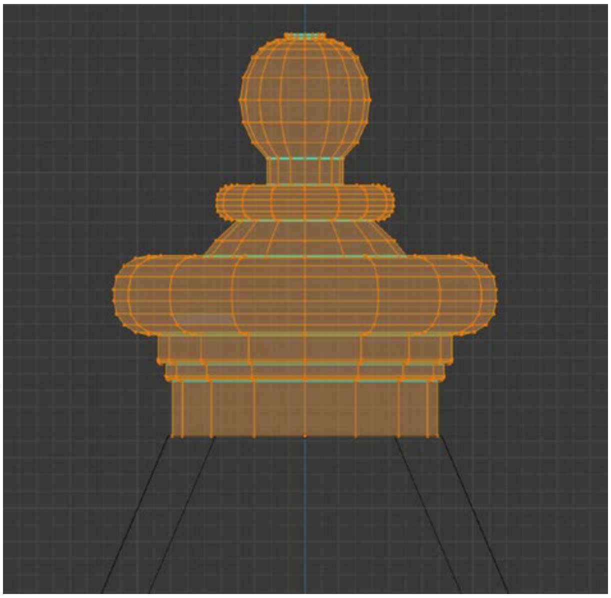


***Figure 5-53.** The final shape on top of the spire*

Detailing the Other Assets

Let's now work on the "mandapa" part of our temple. This is the bottom rectangular structure where people will gather and pray. We will start with the columns. Delete all the vertices of the top edge loop, leaving only the bottom-most edge loop. Bevel the corner vertices of the remaining edge loop by pressing Ctrl+Shift+B. Snap the vertices to the grid so that the form looks something similar to Figure 5-54.

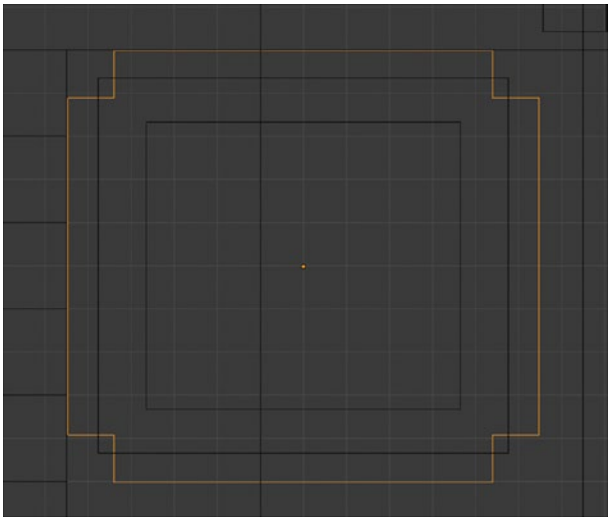


Figure 5-54. *Shape so far*

Now we will extrude this up and add an edge loop to the center of the shape. We can then extrude the top half of the shape inside, as shown in Figure 5-55.

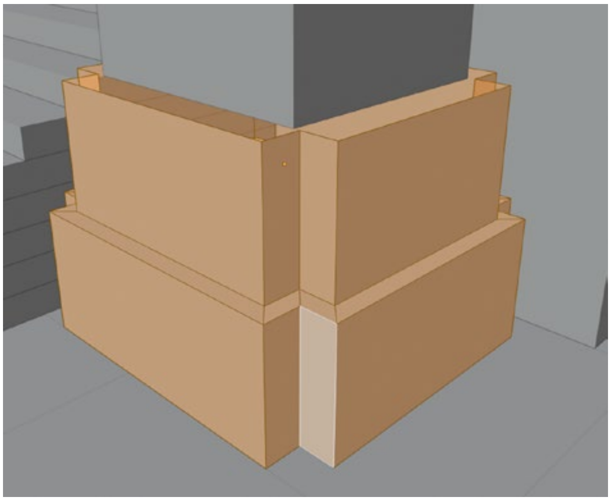


Figure 5-55. *Creating the base of the pillar*

Next, select the top vertices and join them by pressing J so that this looks similar to Figure 5-56.

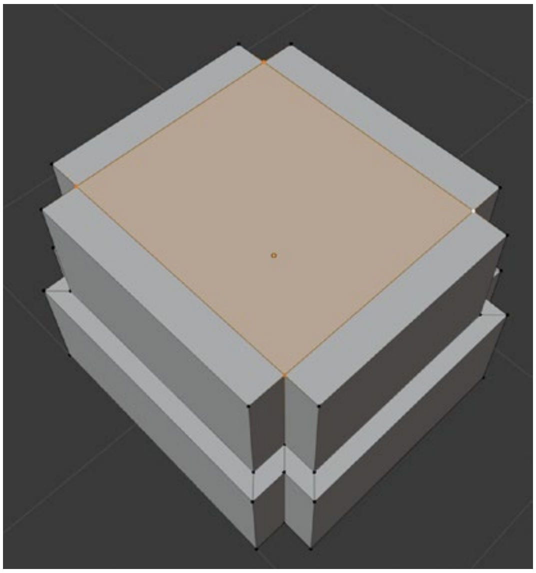


Figure 5-56. *Joining the top vertices*

Then add three edge loops to both sides of the pillar base by pressing Shift+R to initiate the Insert Edge Loop tool. Increase the loop count by scrolling the mouse wheel. The results should be similar to Figure 5-57.

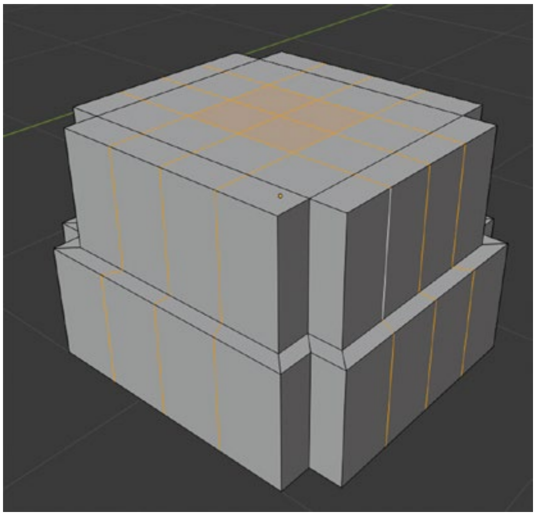


Figure 5-57. *Adding edge loops to the base*

Now select the central edge on both sides and dissolve them by pressing X. Then select the central faces and extrude them out. Manually adjust any face that doesn't extrude properly. Your result should look similar to Figure 5-58.

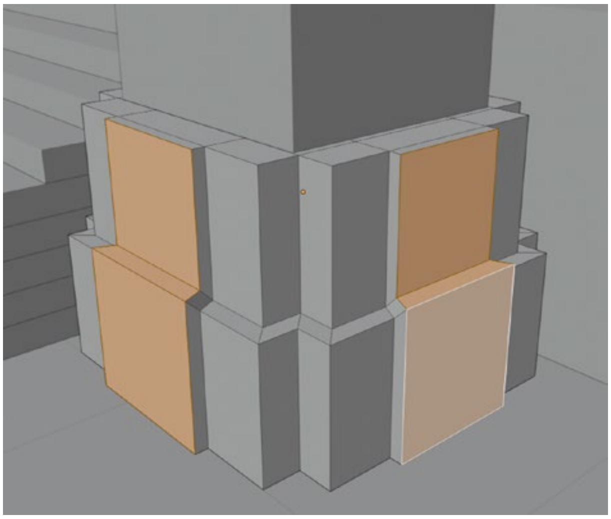


Figure 5-58. *Extrude the central faces*

Now you should delete all the top faces that are not part of the pillar base. Extrude the upper body of the pillar from the base itself. Delete all the faces shown in Figure 5-59.

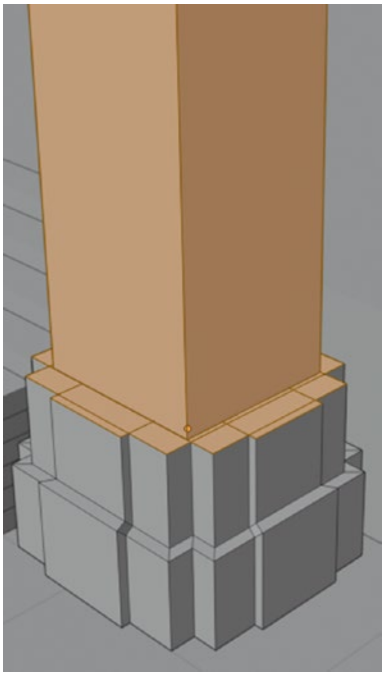


Figure 5-59. *Delete all the selected faces*

Next, select the top border edge of the pillar base and press F to fill in the face. This face will be one giant N-Gon at first. Next, select this new face and press I to start an Inset operation on the face. Provide a small inset amount and then scale down the face some more (see Figure 5-60).

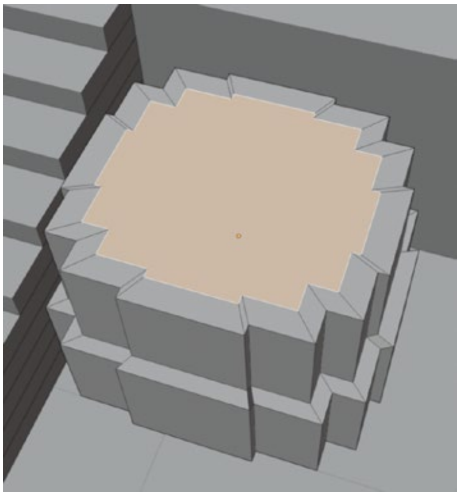


Figure 5-60. *Inset operation on the top face of the base*

Next, you need to extrude the selected face up, as shown in Figure 5-61.

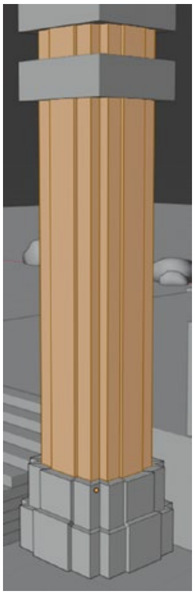


Figure 5-61. *Extrude the face up to create the pillar form*

This is how you detail the model; you should continue doing this for the rest of the assets. It is a pretty easy task once you know how it's done. After going through this chapter, you should have a solid idea on how to do this. In the next chapter, we will start unwrapping the model.

CHAPTER 6

Unwrapping the Models

In this chapter we are going to start unwrapping the models. One thing to note here—the unwrapping that you do will not be final ,as it will have to be readjusted when you start the texturing process. So, we will not be committing to anything as of yet. With that in mind, let’s start with a brief introduction to what unwrapping and texture mapping mean.

Basically when we unwrap a model, we lay out the 3D model and flatten it down to a 2D form so that the textures that we create can be mapped onto it. Blender has very robust and easy-to-use unwrapping tools. In Edit mode, you can press U to see all the unwrapping options available to you. There are a lot. In this chapter we are going to explore the important options and learn how to utilize them to unwrap the model.

You can click on the UV Editing layout tab (see Figure 6-3) to open the UV Editor window. Just under the menu bar you will see an editor-specific menu bar with all the options that you will need. The only one that you will use for your work in this chapter is the UV Editing menu. You will see how everything works later in this chapter. So with this brief introduction, let’s get started.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_6

Unwrapping the Temple

Let's first identify the different sections of the temple. What do I mean by that? In game design, things have to be highly optimized and efficient. We can easily divide the main temple body into three similar parts, as highlighted in different colors in Figure 6-1.

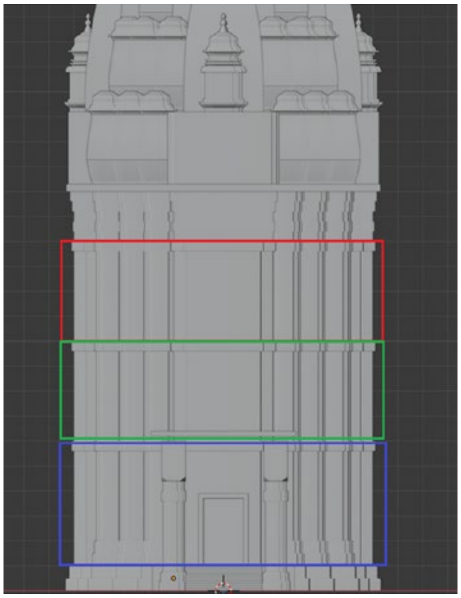


Figure 6-1. *We can divide the temple into three sections*

We can unwrap one section and then simply copy it above when inside an engine. The same thing can be done for other parts of the temple as well. Let's look at the temple's pillars. As highlighted in Figure 6-2 in different colors, the different parts of the temple are modular and one can be used to replace another.

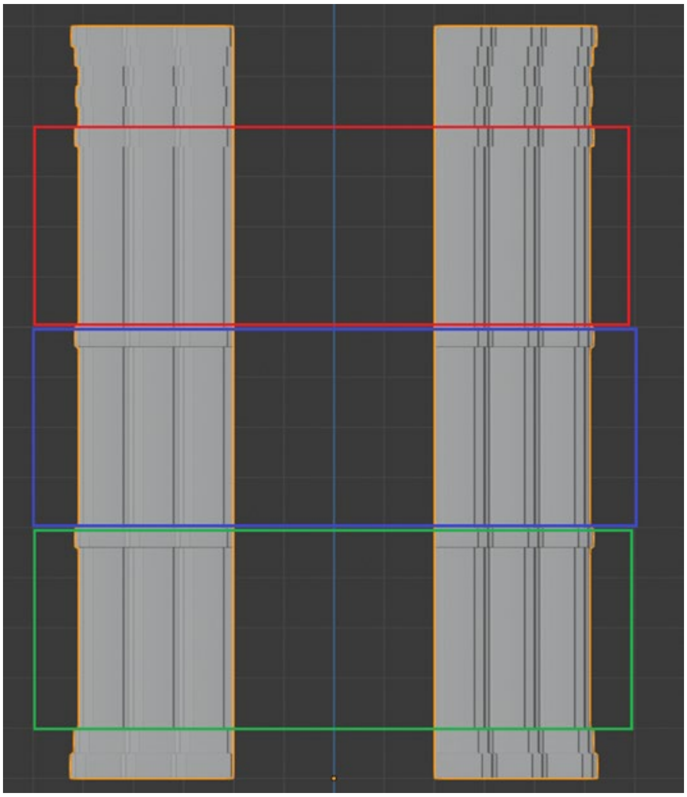


Figure 6-2. *Modularity of the pillars*

With that in mind, we can unwrap only one of the modular pieces and leave the others. They can simply use the date of the previously unwrapped modular piece. So, with that, let's begin with the temple body. Switch to UV Editing layout by clicking on the UV Editing menu option from the Layout menu on top of the screen (see Figure 6-3).

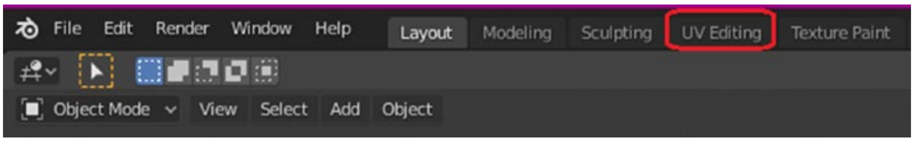


Figure 6-3. *Switching to UV Editing layout*

If you want to unwrap the model first, you need to assign some seams to some edges. To do that, first select an edge/edge loop in Edit mode and press Ctrl+E to open the Edge context menu. From that menu, choose Mark Seam (see Figure 6-4).

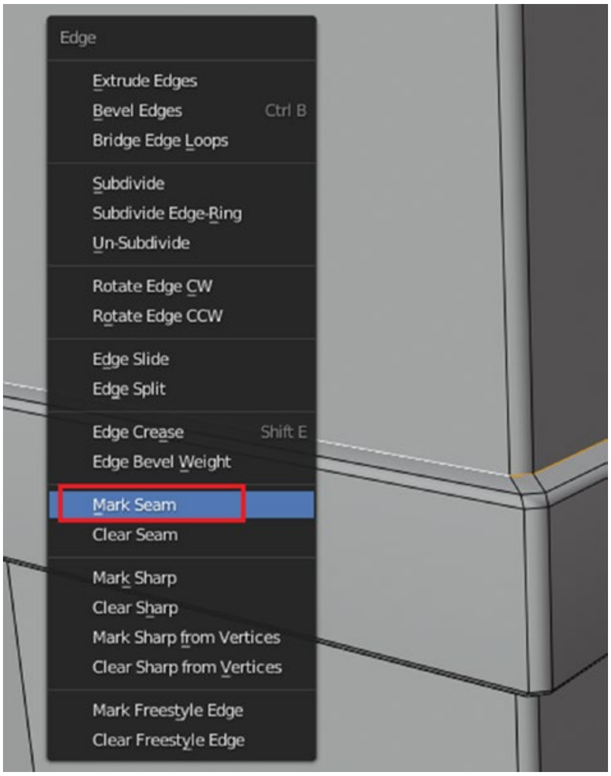


Figure 6-4. *Marking edges as seams*

Let's mark some edges as seams now. Whichever edge we mark as a seam, the UVs will split from there. Select the edge loop shown in Figure 6-5.

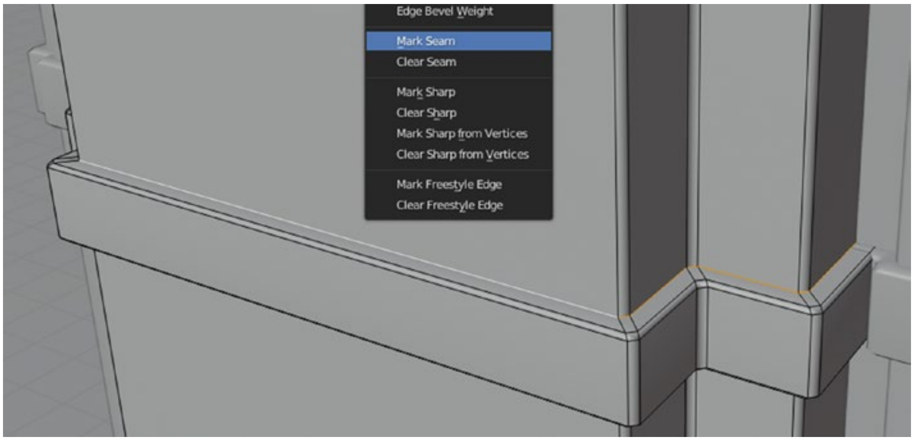


Figure 6-5. *Mark this edge loop as a seam*

Now press Ctrl+E and mark the selected edge loop as a seam. Now, in order to unwrap it, we need to select all the faces that we need to unwrap and press U to open the Unwrap menu. Select all the faces by pressing A and then press U. The Unwrap menu has a lot of options and we will use quite a lot of them. But for now, choose Unwrap. You can see your result in the UV Editor window (see Figure 6-6). It won't be good just yet, but this is the basics of unwrapping.

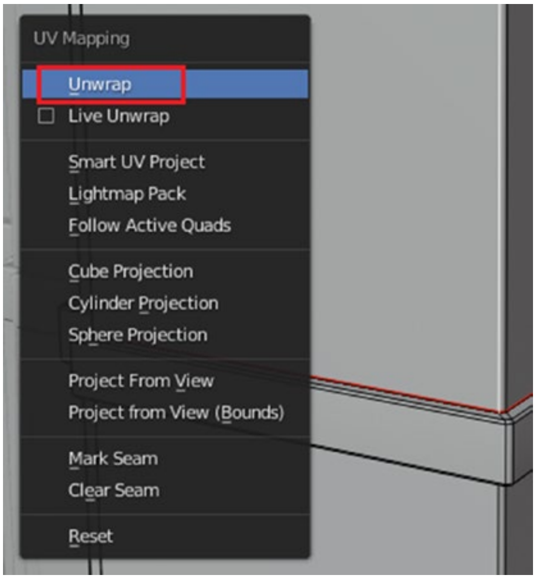


Figure 6-6. *UV Mapping menu*

Next, select the edges shown in Figure 6-7 and mark them as seams. And press U and choose Unwrap again to see what happens to the UVs.

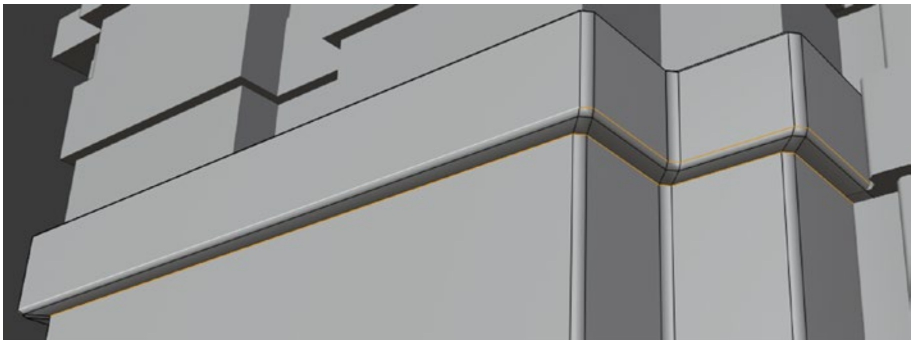


Figure 6-7. *Mark these edges as seams*

You'll notice that once you mark the edges as seams and unwrap them, the UV island becomes clearer and less messy. That is what our objective is, to create a proper 2D representation of our 3D model.

Let's continue. You will now select the edge loop shown in Figure 6-8 and mark it as a seam.

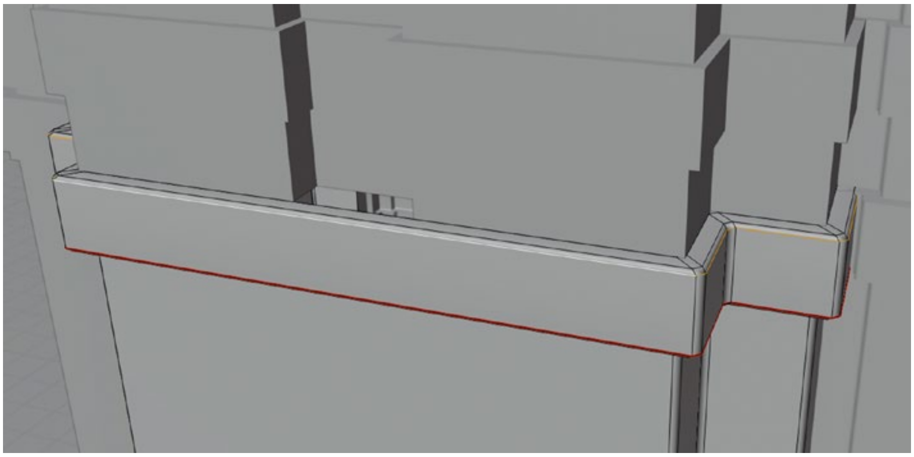


Figure 6-8. *Mark this edge loop as a seam*

Select the region shown in Figure 6-9 and press U to open the mapping menu. Then choose Unwrap. This will unwrap the whole selected region. This will be your first modular region. We can separate this part as a separate mesh. Then we can assemble a temple base right from this modular piece by duplicating this around. But that is for later.

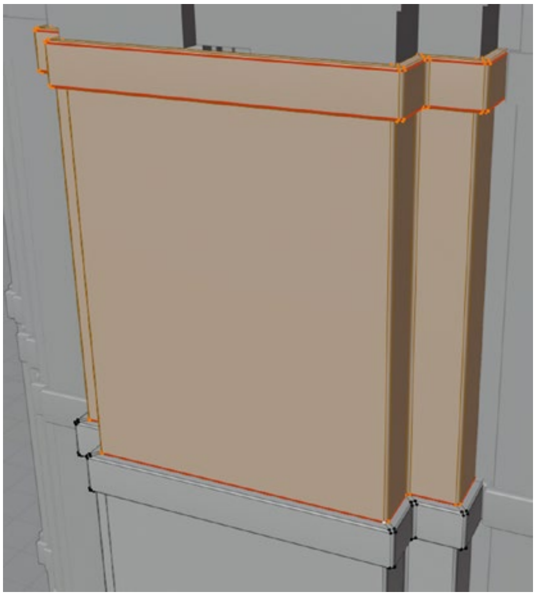


Figure 6-9. *Select this whole region and unwrap it*

Your final UV islands should look similar to Figure 6-10. Their arrangement and alignment might vary and that is not a problem.

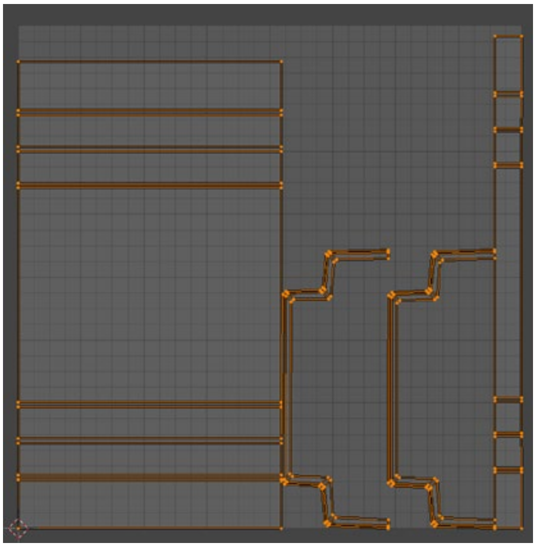


Figure 6-10. *UV islands so far*

We can rearrange the UV islands in whichever way we want. We can also resize the UV islands, depending on whether we want to give a certain region of the mesh a higher or lower texel density. It is not advisable to scale the UV islands arbitrarily, as that will cause more problems unless you have a solid idea of what you are doing.

We can move and rotate the UVs around as well. This will also cause alignment issues in the mesh, so it is better to modify the UVs when you have a good idea of the impact. As of now, we can make some minor adjustments to make the texturing work more easily (see Figure 6-11).

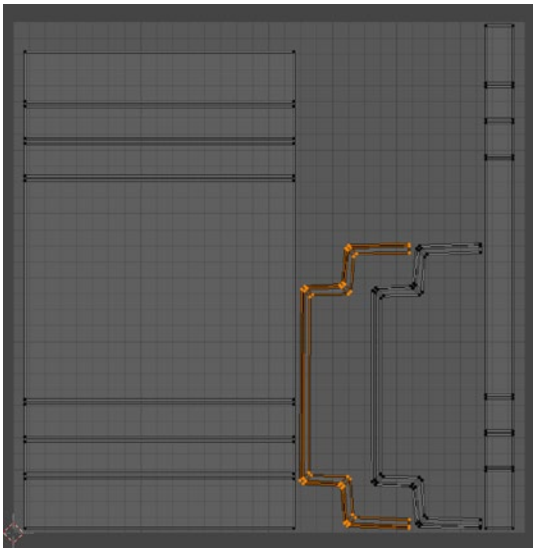


Figure 6-11. *Adjusting the UV islands*

As you can see, we made some minor adjustments to the UV islands by simply moving them around. We made sure that there is a gap between the Islands and that they are far from the boundary of the UV space. This will make your texturing work easier inside Substance Painter. You may also notice that there is a very slight distortion in two particular islands. But that will not affect the outcome, so we are not very concerned with them. You can now unwrap the bottom-most piece of the temple that has the gate (see Figure 6-12).

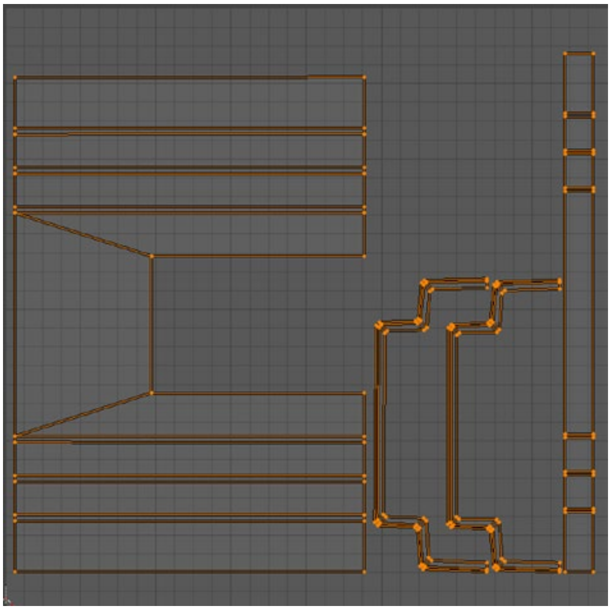


Figure 6-12. *Unwrap the bottom region of the temple*

We are now going to unwrap the pillars. Mark the two edges on the top as seams, as shown in Figure 6-13.

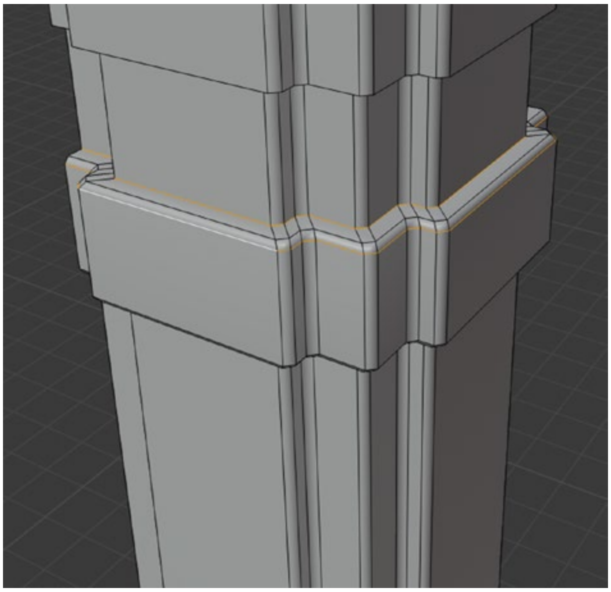


Figure 6-13. *Mark these edges as seams*

We will continue downward now. Select the edges shown in Figure 6-14 and mark them as seams.

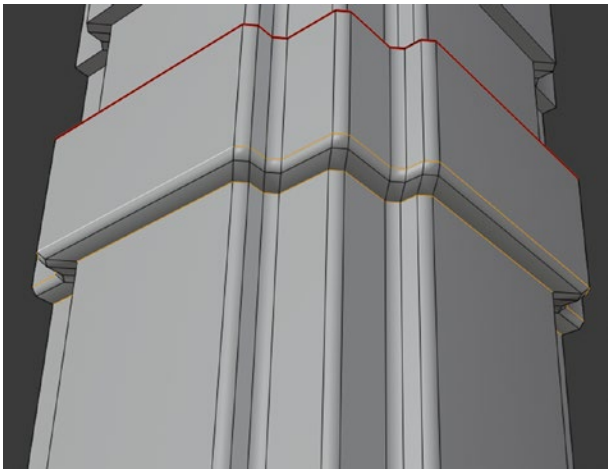


Figure 6-14. *Mark these edges as seams*

Come way down and mark the junction between the modular pieces as a seam.

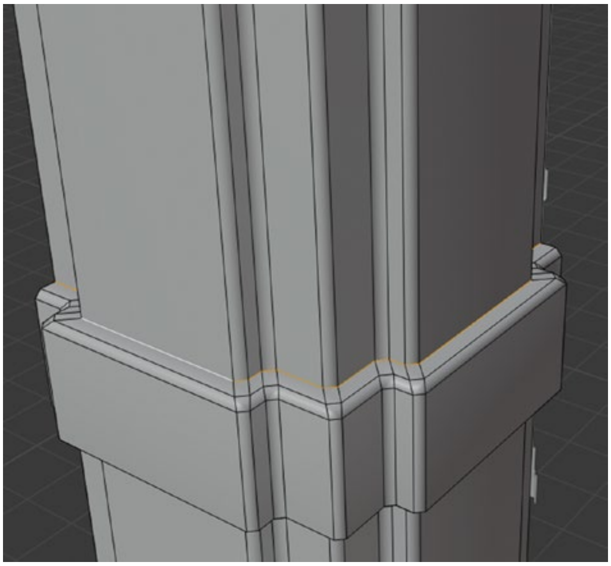


Figure 6-15. *Mark the junction edge as a seam*

Select all the faces shown in Figure 6-16 and press U. Then choose Unwrap.

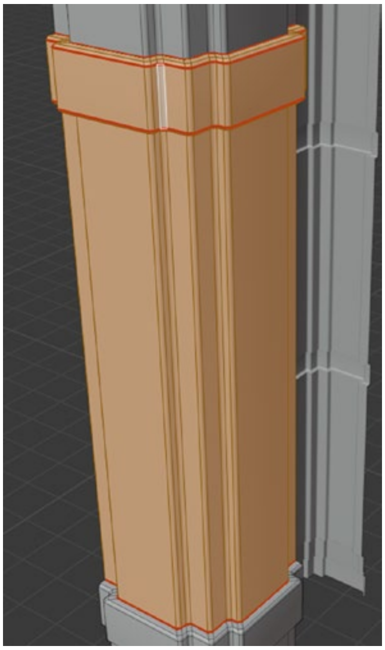


Figure 6-16. *Select all these faces and then unwrap them*

Your unwrap result should be similar to what is shown in Figure 6-17. The alignment and arrangement of the islands may vary. Move and rotate them to match the result shown.

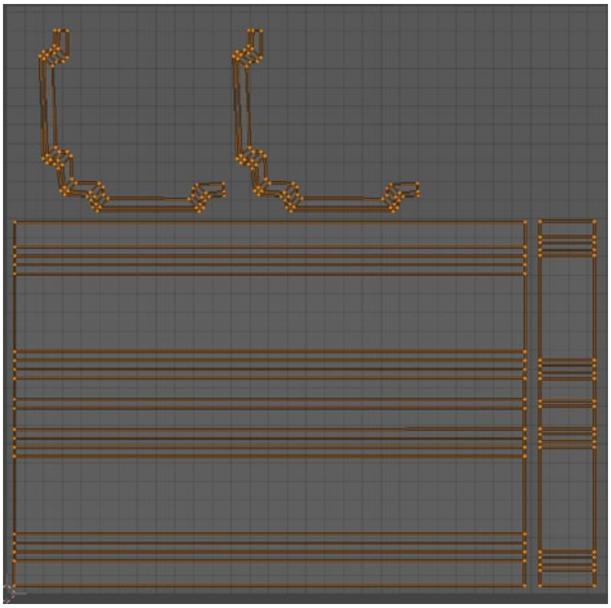


Figure 6-17. *Result of unwrapping the pillar*

Next, let's unwrap a fractal piece. The fractal piece can be easily duplicated and copied around, so you only have to unwrap it once. Unwrapping this will be simple. Just Tab into Edit mode and select an edge loop in the back that's not visible (see Figure 6-18).

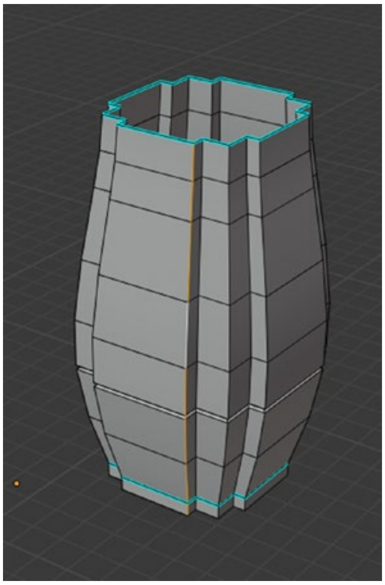


Figure 6-18. *Selecting an edge at the back*

Now select all the faces by pressing A and choose Unwrap. Your final result should be something similar to Figure 6-19.

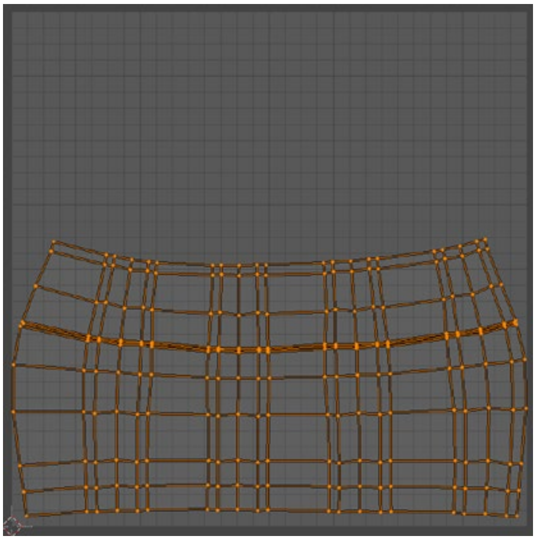


Figure 6-19. *Unwrapping a fractal body*

Next, let's unwrap the cap of the fractal. Select all the edges marked in red in Figure 6-20 and mark them as seams.

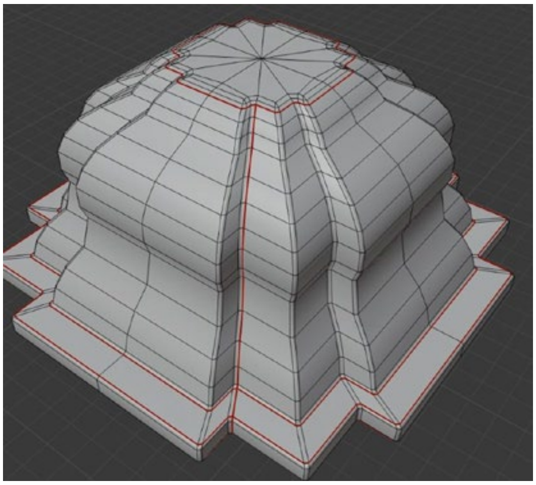


Figure 6-20. *Mark the highlighted edges as seams*

Also mark the bottom edges as seams, as shown in Figure 6-21.

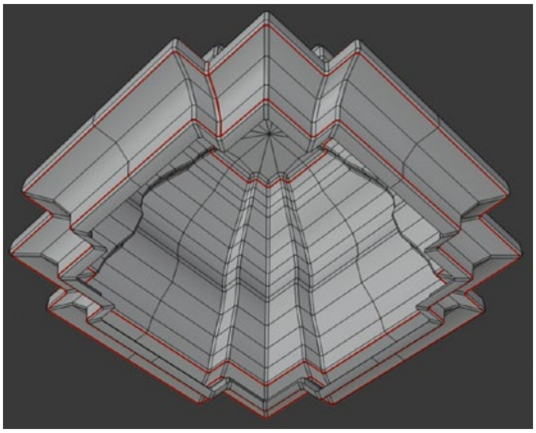


Figure 6-21. *Mark these edges as seams*

Finally, select everything by pressing A. Then press U and choose Unwrap. Your unwrapped result should be similar to Figure 6-22.

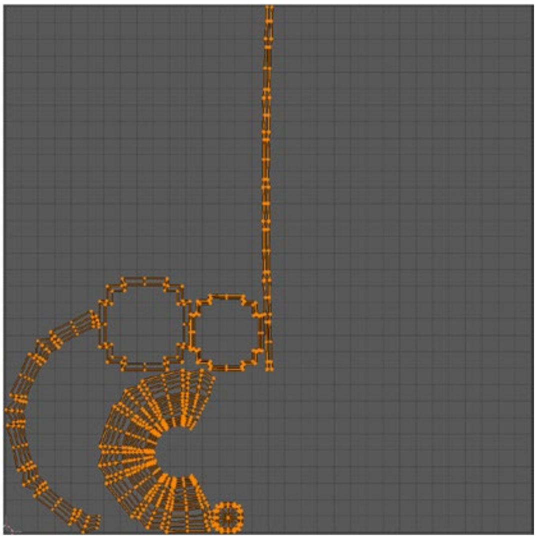


Figure 6-22. *Unwrap result*

As you can see, default algorithms of Blender do not do a good job of packing the UV islands. Manual intervention is required for proper results. But before we do that, let's combine the cap and body of the fractals. Then we will adjust both of their UVs to share space. You will have overlapping UVs and they will not be uniformly scaled. Select all the UV islands by pressing A. Then, inside the UV Editor window, choose UV ➤ Average Islands Scale (see Figure 6-23).

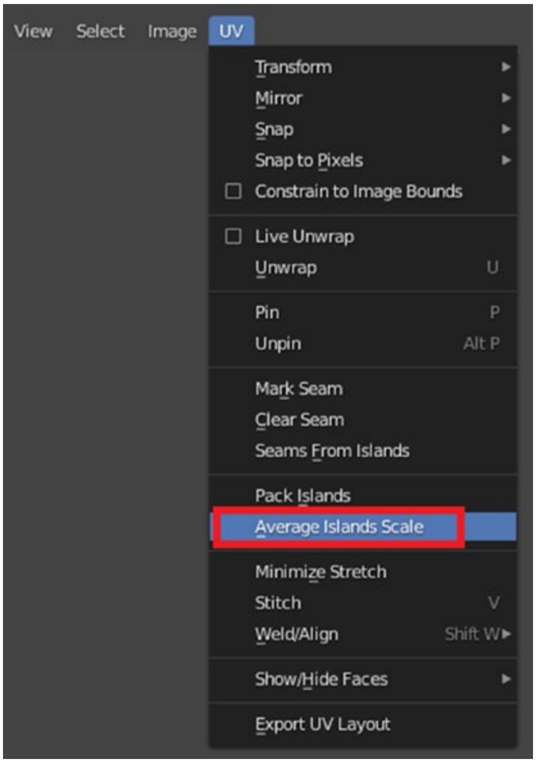


Figure 6-23. *Averaging the island scale*

Once again, with all UVs selected, go to UVs and choose the Pack Islands option, which is just above the Average Islands Scale option. Your UVs should now be unwrapped and packed, ready to be exported. Your final UVs should be similar to Figure 6-24.

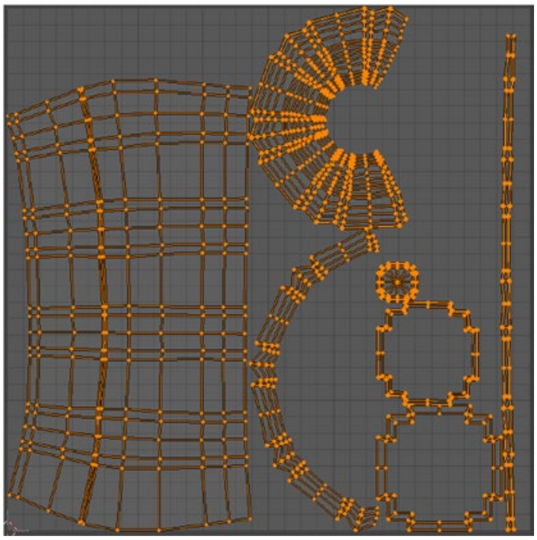


Figure 6-24. *Final UV layout*

So this is how you unwrap assets. Hopefully, you now understand how you can unwrap any asset. With this knowledge, go ahead and unwrap the remaining assets. In the next chapter, we will start texturing the assets.

CHAPTER 7

Texturing Assets Using Substance Painter

In this chapter, we will start texturing the scene in Substance Painter. Before we take our assets to Substance Painter, we need to ensure that we have unwrapped all the assets that need to be textured. When it comes to modular assets, we will texture only one and then duplicate it around.

Exporting Assets to Substance Painter

Let’s start by texturing the body of the temple. First we will see how to export assets from Blender to Substance and how to set everything up once they are inside Substance Painter. Select the top modular piece of the temple body. Then choose File ➤ Export ➤ FBX (see Figure 7-1).

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_7

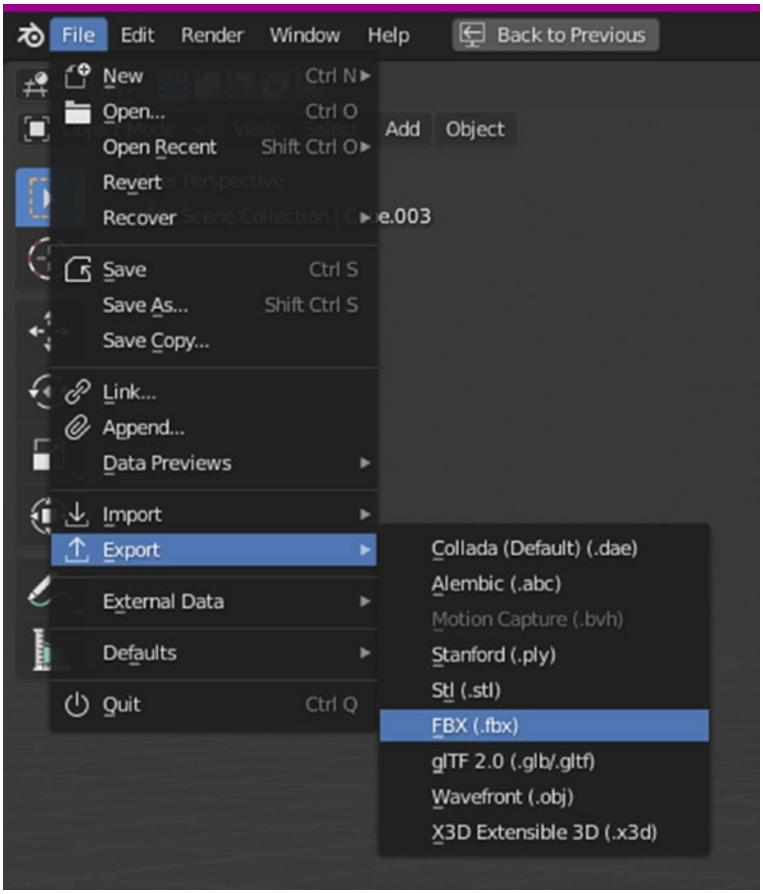
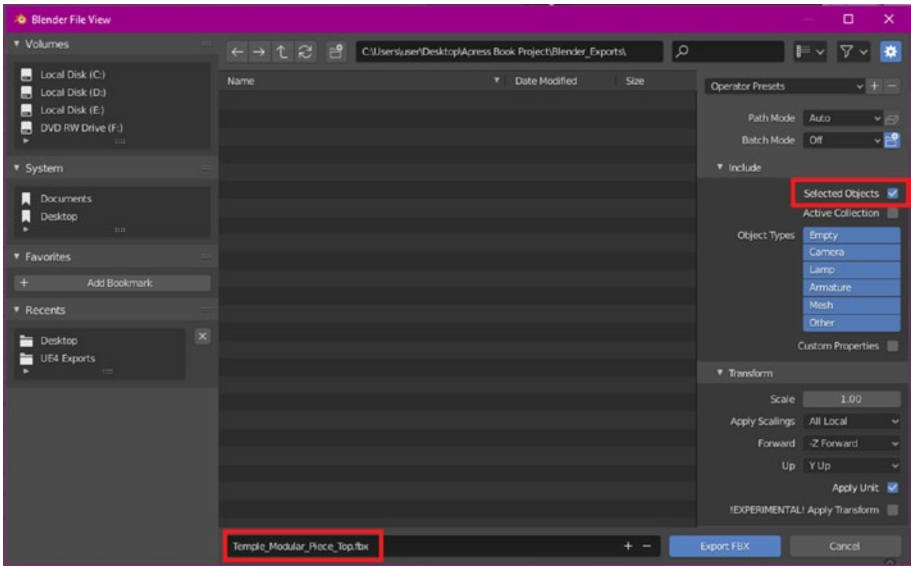


Figure 7-1. *Exporting an object as FBX*

A new Export Settings window will open (see Figure 7-2). In this new window, make sure to enter a proper name for your file so that it is easy to identify. Next, enable Selected Objects so that only the assets that you have selected will be exported. Finally, choose a directory where you want to export. It is good practice to keep a separate directory for Blender exports. This way the directory will be neater and files will be easier to find. Leave everything else set to the defaults.



C:\Users\user\Desktop\Apress Book Project\Blender_Exports\

Path Mode: Auto
Batch Mode: Off
Selected Objects ✓
Active Collection
Object Types: Empty, Camera, Lamp, Armature, Mesh, Other
Custom Properties
Scale: 1.00
Apply Scalings: All Local
Forward: -Z Forward
Up: Y Up
Apply Unit ✓
!EXPERIMENTAL! Apply Transform

Temple_Modular_Piece_Top.fbx

Figure 7-2. *Export window of Blender*

Once you are done, click on Export FBX. This will export your mesh to the destination you set. Next we are going to import this mesh into Substance Painter. Launch Substance Painter and click on File to open the File menu (see Figure 7-3). From the File menu, choose New. Alternatively you can press Ctrl+N to create a new scene.

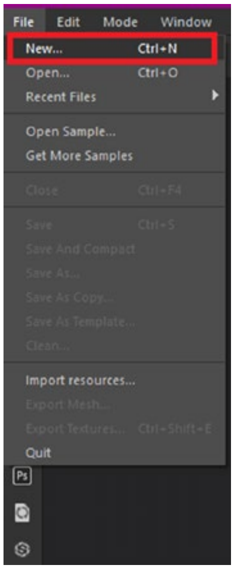
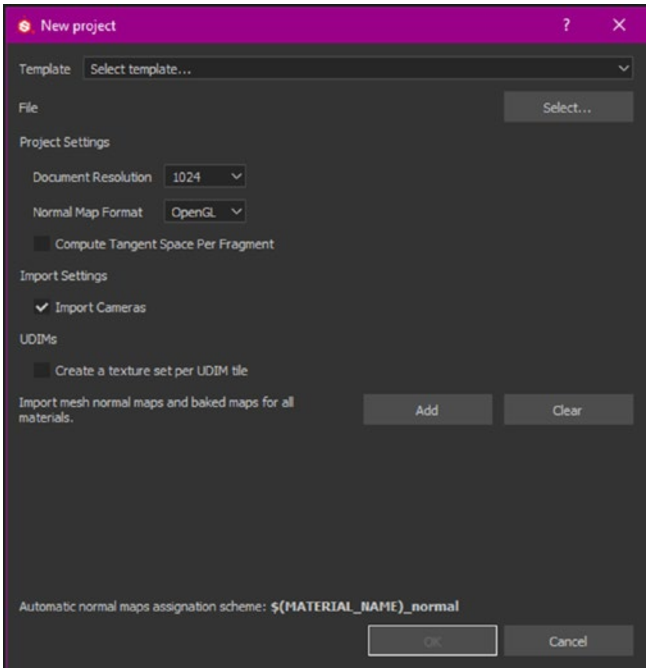


Figure 7-3. *Creating a new scene*

When you click on New, a new window should open to allow you to set up the new scene and import the required assets (see Figure 7-4).


New project

Template: Select template…

File: Select…

Project Settings

Document Resolution: 1024

Normal Map Format: OpenGL

☐ Compute Tangent Space Per Fragment

Import Settings

✓ Import Cameras

UDIMs

☐ Create a texture set per UDIM tile

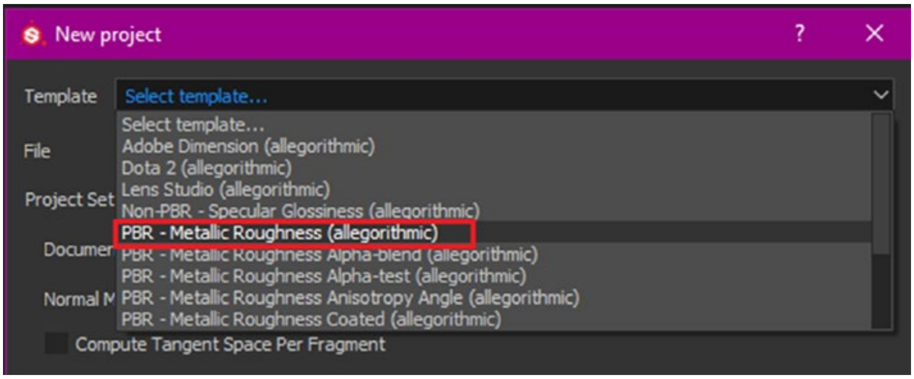
Import mesh normal maps and baked maps for all materials. Add Clear

Automatic normal maps assignation scheme: $(MATERIAL_NAME)_normal

OK Cancel

Figure 7-4. *New Project window*

Let's start from the top and select the template first. Click on the Template drop-down menu and choose PBR – Metallic Roughness (Allegorithmic) (see Figure 7-5).


New project

Template: Select template…

- Select template…
- Adobe Dimension (allegorithmic)
- Dota 2 (allegorithmic)
- Lens Studio (allegorithmic)
- Non-PBR - Specular Glossiness (allegorithmic)
- PBR - Metallic Roughness (allegorithmic)
- PBR - Metallic Roughness Alpha-blend (allegorithmic)
- PBR - Metallic Roughness Alpha-test (allegorithmic)
- PBR - Metallic Roughness Anisotropy Angle (allegorithmic)
- PBR - Metallic Roughness Coated (allegorithmic)

☐ Compute Tangent Space Per Fragment

Figure 7-5. *Choosing a template*

Next we will import our file. Click on the Select button just below the Template to launch the File Explorer window, which will allow you to choose a file you want to import. Choose the temple piece that we exported from the directory where it was exported (see Figure 7-6).

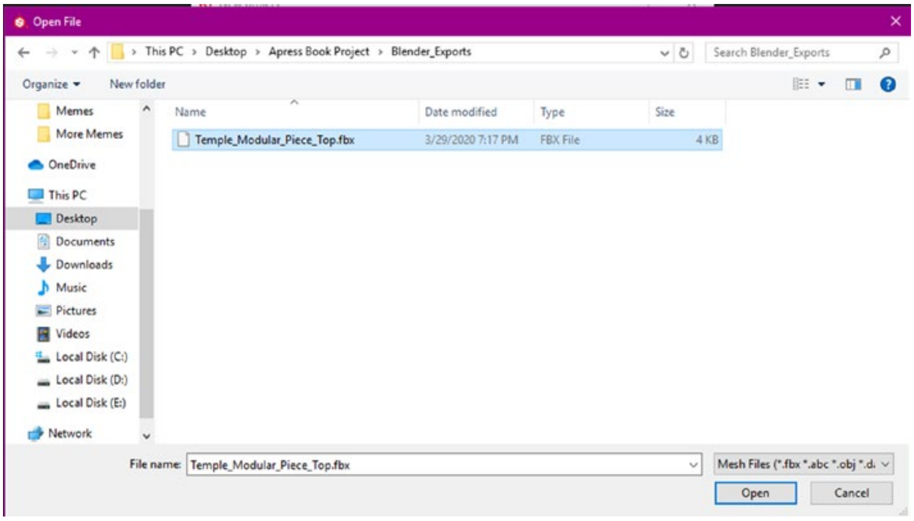
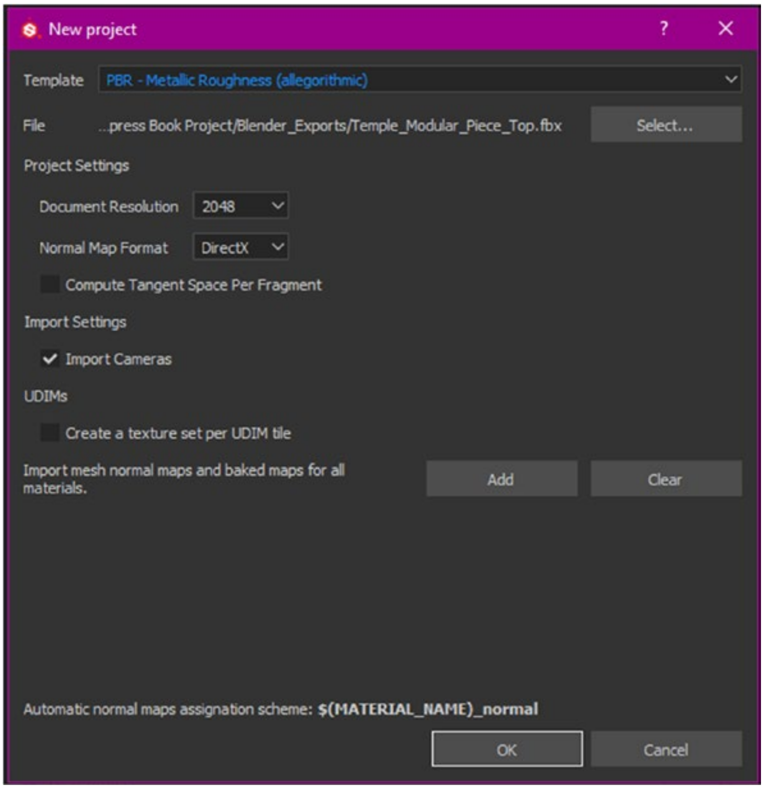


Figure 7-6. *File Explorer window*

Choose your file in the File Explorer window and click on Open to import it into Substance Painter. Next, we are going to change the document resolution by clicking on the drop-down menu and setting it to 2048. Then change the Normal Map Format to DirectX, as this is the format that Unreal Engine 4 uses. You can leave the rest of the settings to the defaults. Your final settings should look like Figure 7-7.


New project

Template: PBR - Metallic Roughness (allegorithmic)

File: ...press Book Project/Blender_Exports/Temple_Modular_Piece_Top.fbx

Project Settings

Document Resolution: 2048

Normal Map Format: DirectX

Compute Tangent Space Per Fragment

Import Settings

✓ Import Cameras

UDIMs

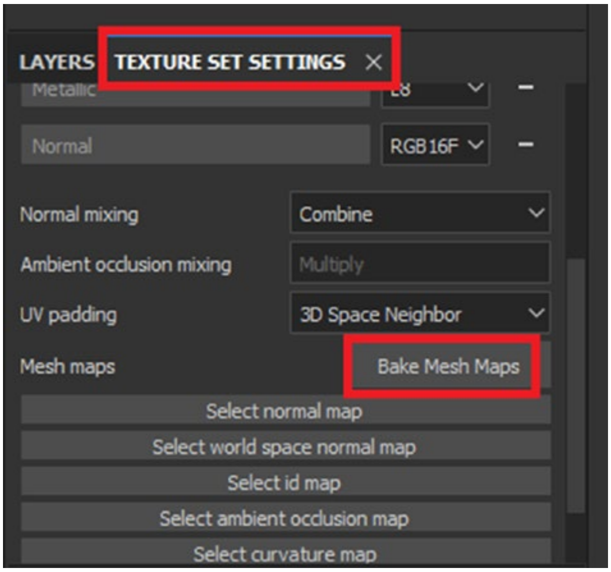
Create a texture set per UDIM tile

Import mesh normal maps and baked maps for all materials.

Automatic normal maps assignation scheme: **$(MATERIAL_NAME)_normal**

Figure 7-7. *Final settings for this project*

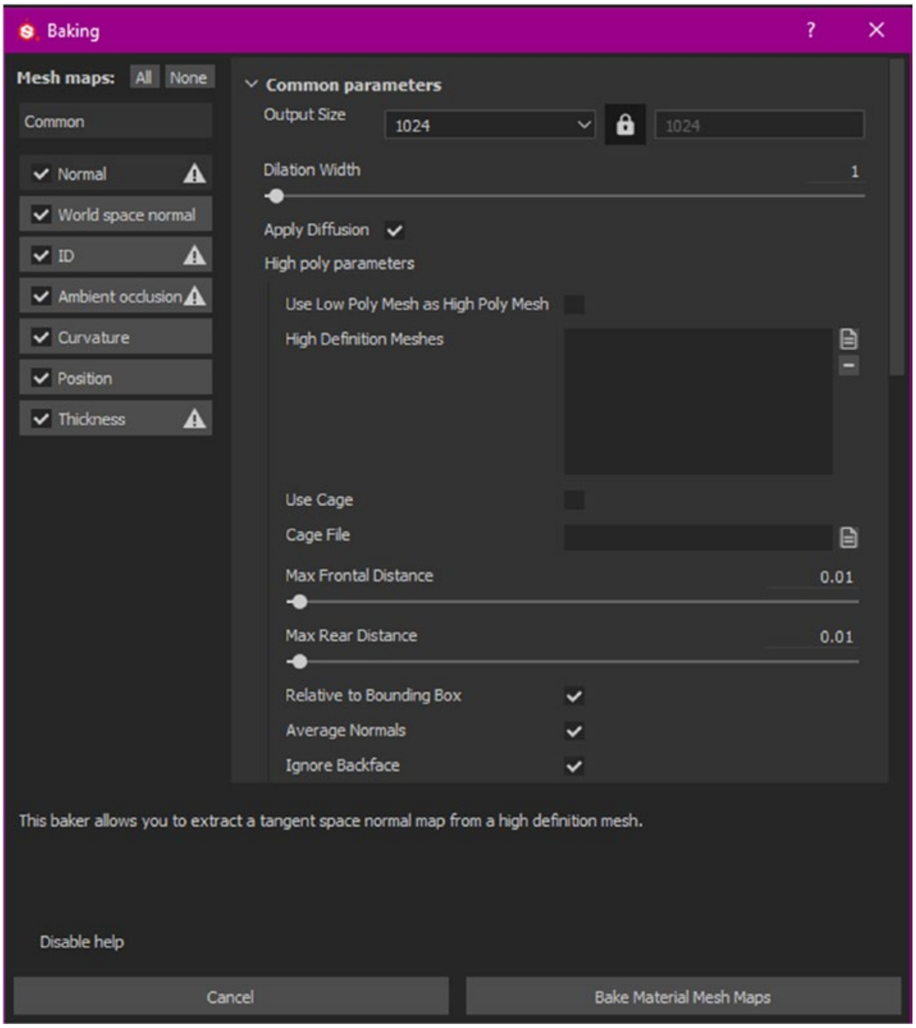
Once you click on OK, the file will be imported and visible on your screen. We can now work on this further. But before we begin texturing, we need to do some more things. We will now see how we can bake the mesh maps. Mesh maps are used for various types of calculations and for applying procedural effects on your mesh. As soon as you import anything into Substance Painter, the first thing that you do is bake the mesh maps. To bake maps, click on Bake Mesh Maps under Texture Set Settings (see Figure 7-8).



Normal mixing: Combine
Ambient occlusion mixing: Multiply
UV padding: 3D Space Neighbor
Mesh maps: Bake Mesh Maps

Figure 7-8. *Baking texture maps*

Once you click on the Bake Mesh Maps option, a new window will open, which will allow you to modify the Baking parameters (see Figure 7-9).



Baking

Mesh maps: All None

Common

- ✓ Normal
- ✓ World space normal
- ✓ ID
- ✓ Ambient occlusion
- ✓ Curvature
- ✓ Position
- ✓ Thickness

Common parameters

Output Size: 1024 / 1024

Dilation Width: 1

Apply Diffusion ✓

High poly parameters

Use Low Poly Mesh as High Poly Mesh

High Definition Meshes

Use Cage

Cage File

Max Frontal Distance: 0.01

Max Rear Distance: 0.01

Relative to Bounding Box ✓

Average Normals ✓

Ignore Backface ✓

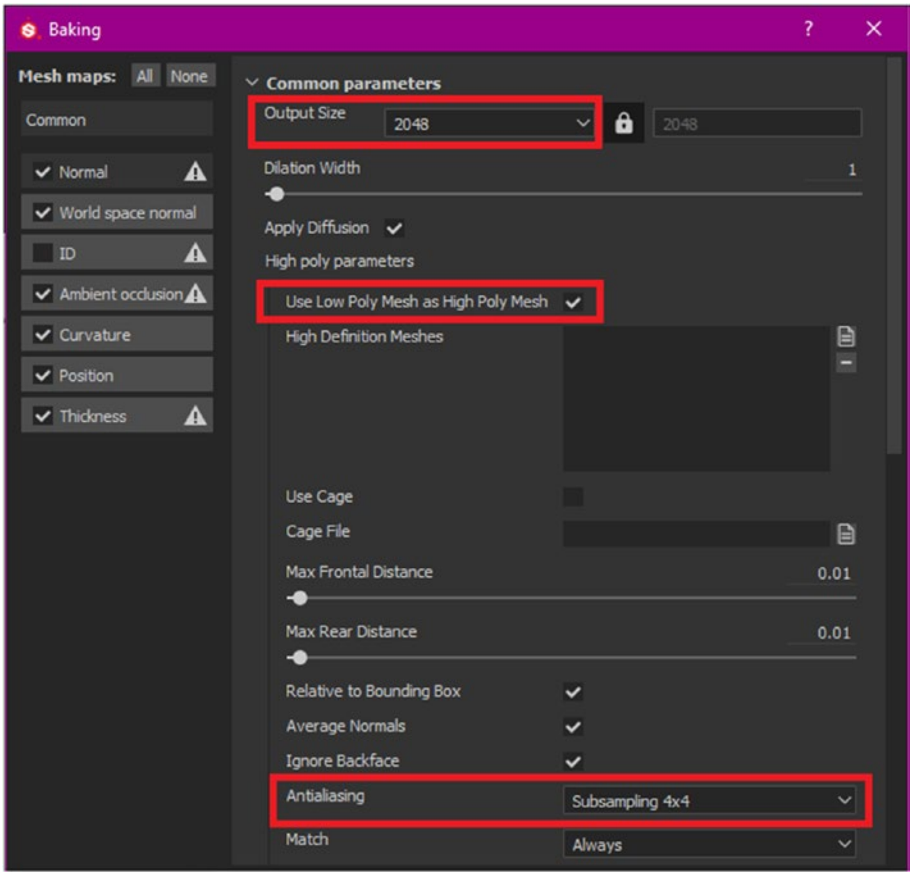
This baker allows you to extract a tangent space normal map from a high definition mesh.

Disable help

Cancel | Bake Material Mesh Maps

Figure 7-9. *The Baking window*

First, we will determine which maps we want to bake and which maps we don't want to bake. In this case, we can disable the ID setting by unchecking the checkbox to the left of the map name. We disabled ID because we haven't assigned material IDs to the object. We will also disable the Thickness baker, as we won't be needing a thickness map and disabling it will reduce the baking time. Next, we will modify some parameters. First of all, we will increase the output size from 1024 to 2048. Next, check the Use Low Poly Mesh as High Poly Mesh option and increase anti-aliasing to 4x4 (see Figure 7-10).

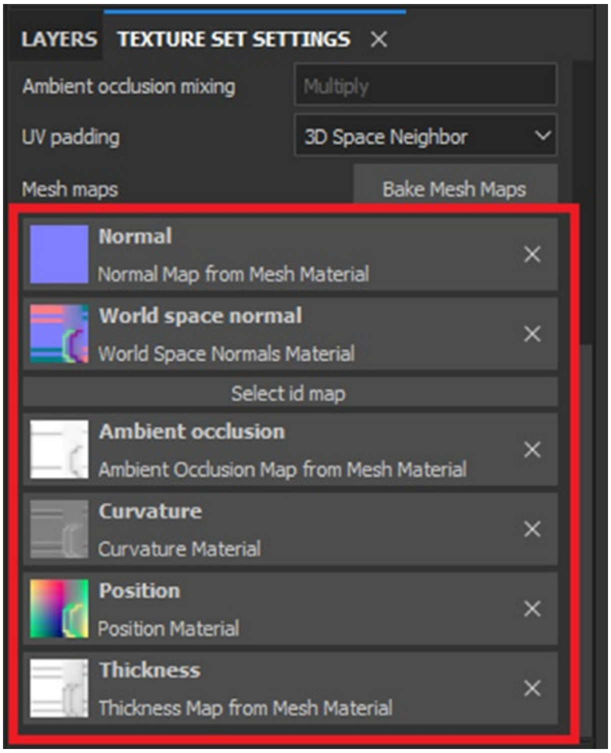


Mesh maps:
✓ Normal
✓ World space normal
ID
✓ Ambient occlusion
✓ Curvature
✓ Position
✓ Thickness

Output Size 2048 2048
Dilation Width 1
Apply Diffusion ✓
Use Low Poly Mesh as High Poly Mesh ✓
Use Cage
Max Frontal Distance 0.01
Max Rear Distance 0.01
Relative to Bounding Box ✓
Average Normals ✓
Ignore Backface ✓
Antialiasing Subsampling 4x4
Match Always

Figure 7-10. *Changing the appropriate parameters for the bake*

Next, click on Bake; the Substance Painter will start baking the maps. This will take some time, depending on the strength of your PC. Faster hardware will bake faster and vice versa. Once the bake finishes, you will see that the mesh maps have been automatically applied to their respective slots under Texture Set Settings, as shown in Figure 7-11.


LAYERS TEXTURE SET SETTINGS ×

Ambient occlusion mixing: Multiply

UV padding: 3D Space Neighbor

Mesh maps: Bake Mesh Maps

Normal — Normal Map from Mesh Material ×

World space normal — World Space Normals Material ×

Select id map

Ambient occlusion — Ambient Occlusion Map from Mesh Material ×

Curvature — Curvature Material ×

Position — Position Material ×

Thickness — Thickness Map from Mesh Material ×

Figure 7-11. *Mesh maps in Substance Painter*

Now you are ready to texture the mesh. You can start experimenting with the various materials that are shipped with Substance Painter. First, let's see where can you get more assets. Google Substance Share and open the official website at `https://share.substance3d.com/` (see Figure 7-12). Sign in using the same account that you used for downloading Substance Painter.

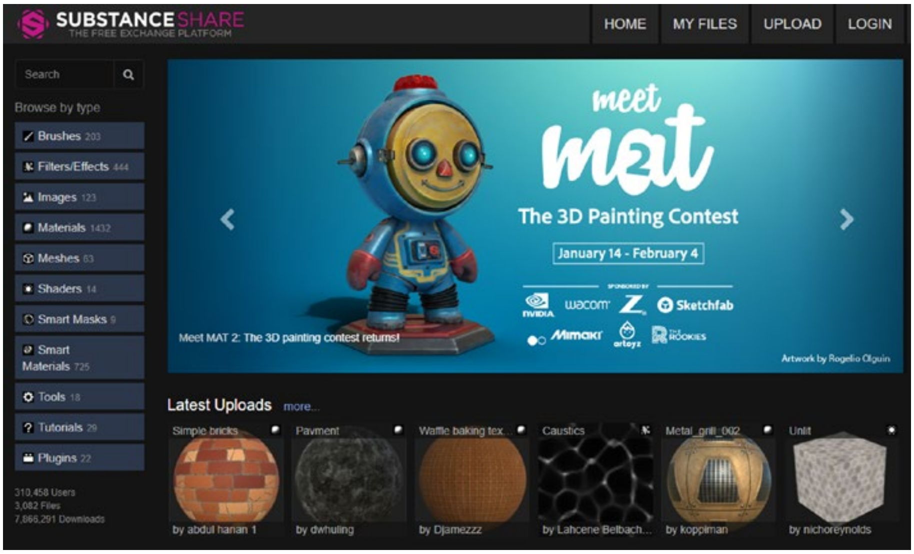


__Figure 7-12.__ Substance Share website

While you are there, use the search bar to search for some materials that can be used for temples, such as smooth rocks or bricks. See the reference to identify which materials will be suitable. I personally found the Aged Stone material quite interesting and we will repurpose it for use in this example. Download it (or any other material) by simply clicking on the Download button. Once the material is downloaded, it will normally be in a compressed archive. Extract it to a suitable destination. Then drag and drop the `.spsm` file into the Content Browser of Substance Painter to open the Import Resources dialog box.

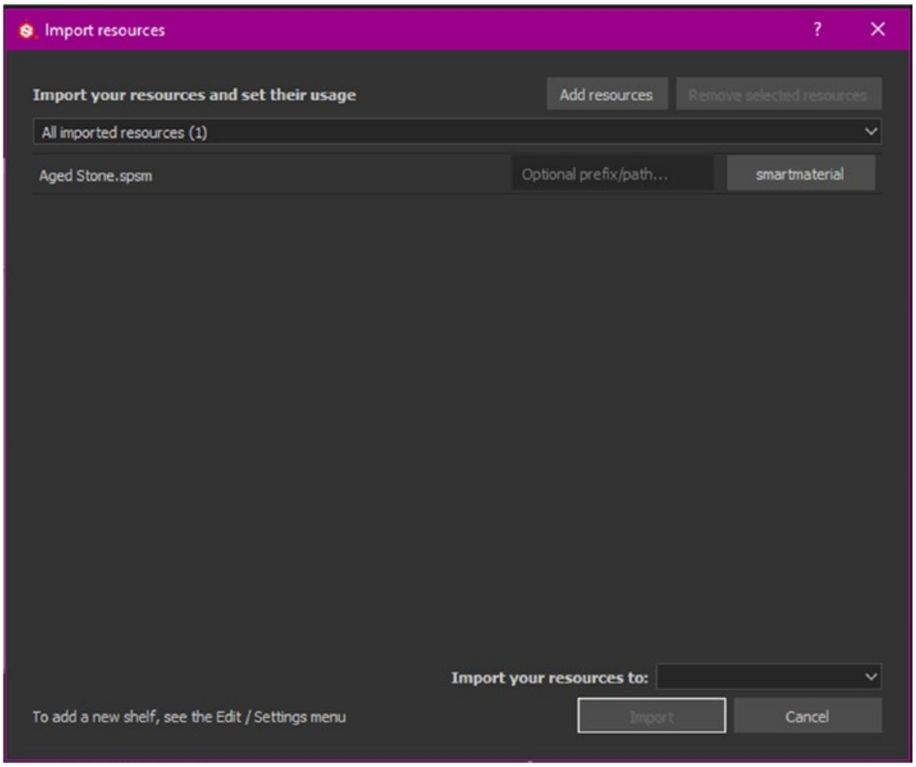

Import resources
?
Import your resources and set their usage
Add resources
Remove selected resources
All imported resources (1)
Aged Stone.spsm
Optional prefix/path...
smartmaterial
Import your resources to:
To add a new shelf, see the Edit / Settings menu
Import
Cancel

Figure 7-13. *Import Resources dialog box*

Based on the file type, Substance Painter automatically assigns the material a type. In this case, it is Smart Material. Next, you need to determine where you want the resource to be imported. Use the Import Your Resources To drop-down menu to set where to import your resource. There are three options: current session, project [project name], and shelf "shelf." Each will result in different outcomes.

- *Current Session* will import the asset and will keep it only as long as you are working on the session. As soon as you close the session, the asset will be deleted.

- Importing it to *Project* will keep it only for the project you are currently working on. When you create a new project, it won't be there. This can help reduce clutter.
- *Shelf* will import the asset permanently into your shelf and it will stay there for every session that you create. This allows you to use this asset in any number of projects.

In this case, import the resource to shelf, as shown in Figure 7-14.

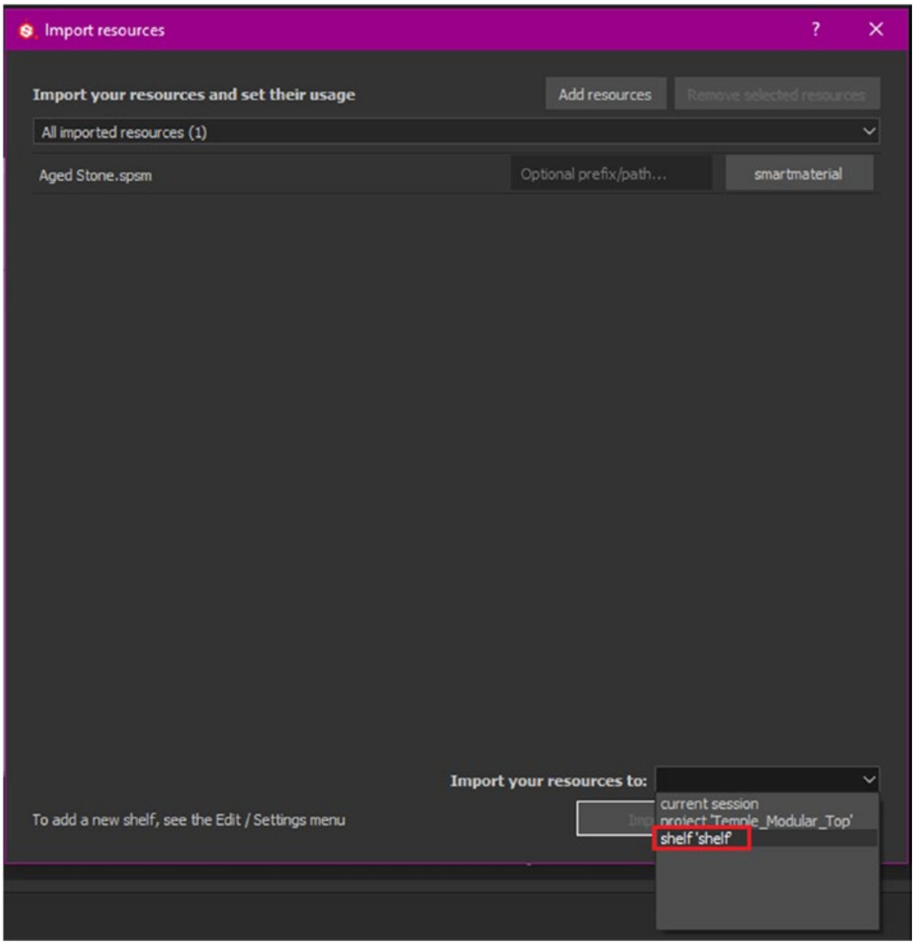

Import resources
?
Import your resources and set their usage
Add resources
Remove selected resources
All imported resources (1)
Aged Stone.spsm
Optional prefix/path...
smartmaterial
Import your resources to:
To add a new shelf, see the Edit / Settings menu
current session
project 'Temple_Modular_Top'
shelf 'shelf'

Figure 7-14. *Importing assets into the shelf*

We will be using the downloaded materials for nearly all our assets, so it is better if we import the asset into the shelf. This way, we won't have to keep importing it every time we create a new project. Once this is done, click on Import Your Assets(s) into Substance Painter's shelf. You will find the assets in their respective shelf type. Our asset was a Smart Material, so you can access it by clicking on the Smart Materials tab in the shelf. Materials and Smart Materials are very different from each other. We will explore the differences in detail as we work. For now, click on the Smart Materials tab to display all the Smart Materials that you have currently installed (see Figure 7-15). By default, Substance Painter ships with more than 100 Smart Materials, which is very useful.

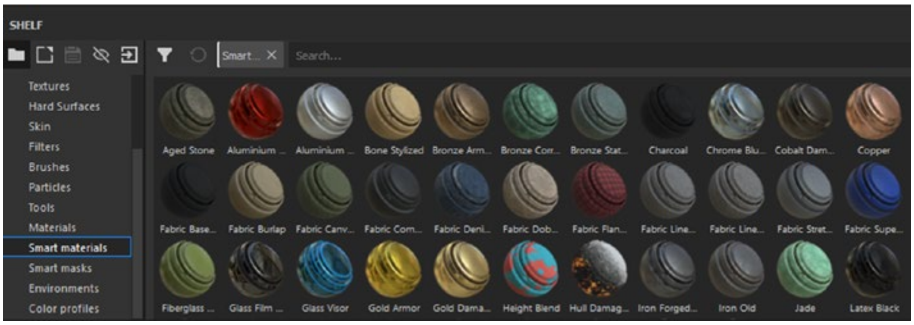
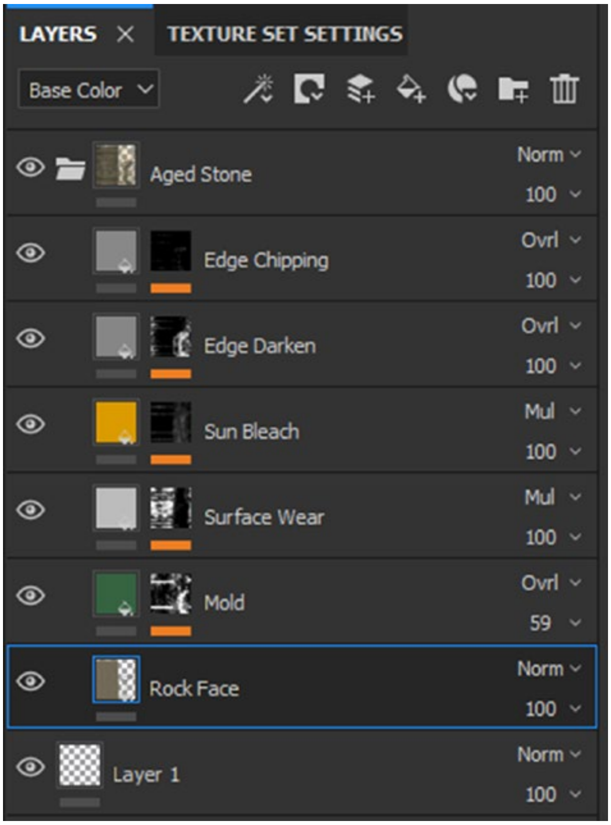


Figure 7-15. *The Smart Materials tab of Substance Painter*

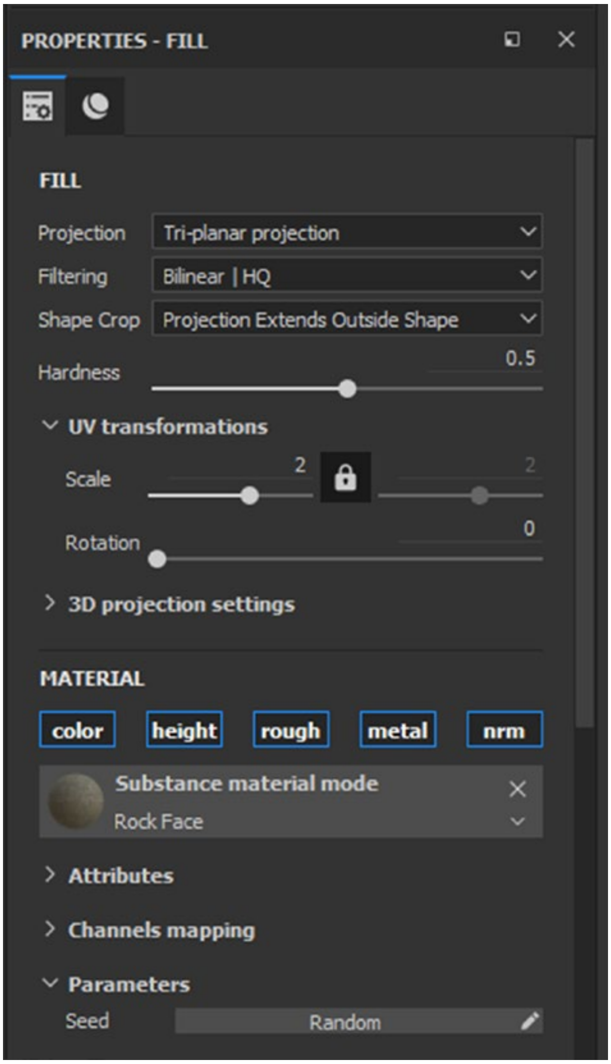
Scroll down to locate the asset that you downloaded. Let's apply that to the model. Drag the Smart Material from the shelf and drop it on the model. You will see that the asset becomes textured. Let's now take a quick look at the Layers and Layer Properties tabs. The Layers tab shows any materials, masks, and filters you applied to your mesh in the order they were applied, as shown in Figure 7-16.



Aged Stone Norm 100
Edge Chipping Ovrl 100
Edge Darken Ovrl 100
Sun Bleach Mul 100
Surface Wear Mul 100
Mold Ovrl 59
Rock Face Norm 100
Layer 1 Norm 100

Figure 7-16. *The Layers window*

Just below the Layers window is the Layer Properties window, which shows the editable parameters of the selected layer. You can modify these to change the effect the parameter has on your model (see Figure 7-17).



PROPERTIES - FILL

FILL

Projection: Tri-planar projection

Filtering: Bilinear | HQ

Shape Crop: Projection Extends Outside Shape

Hardness: 0.5

UV transformations

Scale: 2 2

Rotation: 0

3D projection settings

MATERIAL

color height rough metal nrm

Substance material mode

Rock Face

Attributes

Channels mapping

Parameters

Seed: Random

Figure 7-17. *The Layer Properties window*

We will explore all the effects in more detail as we texture our assets. As for now, I highly suggest that you play with the values and experiment to see what does what and what you can come up with. This may seem like there is lot to digest, but as you work more with Substance Painter, you will realize that it is not that complicated after all.

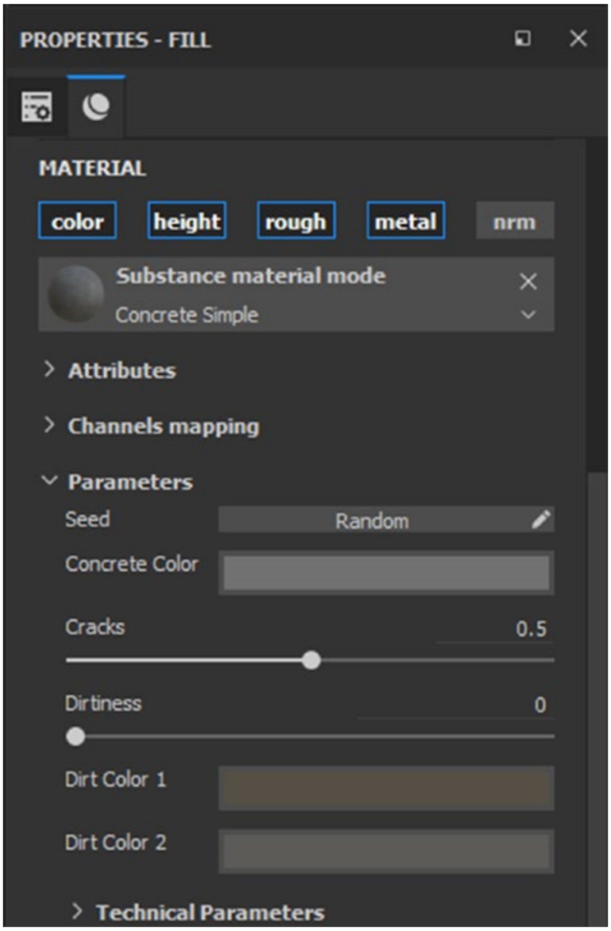
Texturing the Larger Structures

Let's begin texturing now. We have already applied a Smart Material to the first asset, which is the top part of the temple modular piece. Since we are not going to use Substance Designer to create our own materials, we will depend on materials that can be downloaded or that ship with Substance Painter. Mostly we have to download materials to get the result that we want. So look on the Internet.

We will now see the difference between a material and a Smart Material. This is best understood with an example. Here is a short explanation first.

- A *material* is created inside Substance Designer using nodes and graphs. When they are imported into Painter, they work as materials that can be applied to any mesh. How modifiable the material is depends on the parameters exposed by its creator. Otherwise, the material will just do its job of applying the texture data to the mesh, without many modifications available.
- A *Smart Material,* on the other hand, is created by stacking layers and effects to create a material. Every layer is modifiable and the effect can be changed to make the material the way you want it. This also makes modifying them more complicated. All effects in a Smart Material will automatically adapt according to the mesh on which it is applied, hence it is called a Smart Material.

Let's now see the examples of both. We will first apply a material to the object. Any material will do, as we are just trying the basics now. So choose any material from the shelf and drag and drop it on to the mesh or on to the layer stack (the effect will be same). Once it's applied, you will see the editable material parameters in the Layer Properties window. If you scroll down, you will see some parameters that you can edit to modify the material, but it will usually not be much. The modifications are limited (see Figure 7-18).


PROPERTIES - FILL

MATERIAL

color height rough metal nrm

Substance material mode
Concrete Simple

Attributes
Channels mapping
Parameters
Seed: Random
Concrete Color
Cracks: 0.5
Dirtiness: 0
Dirt Color 1
Dirt Color 2
Technical Parameters

Figure 7-18. *Material properties to modify their look*

As you can see in the Concrete Simple material, there are a couple of settings (Cracks and Dirtiness) that can be modified to alter the amount of dirt and cracks in the material. Next, let's play with Smart Materials and see what options we have for them. Select the material from the Layer Stack and click on the Delete icon to delete it (see Figure 7-19).

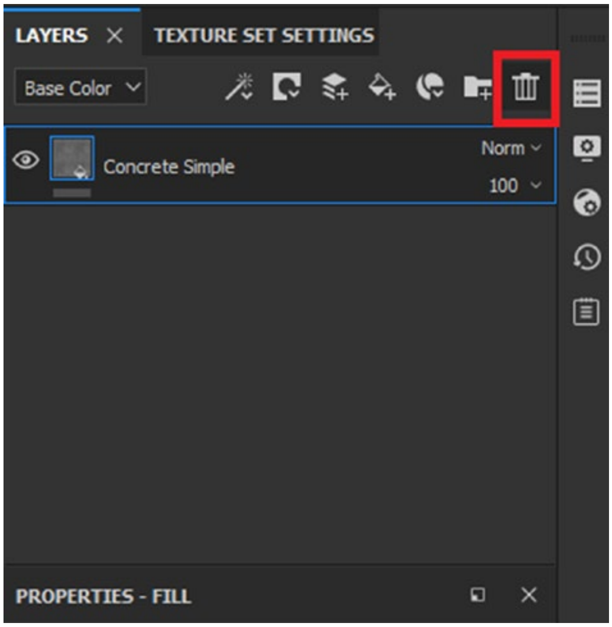


Figure 7-19. *Deleting the selected layer*

Now switch to the Smart Materials tab. Apply the Smart Material of your choice to the mesh. In this case, I apply the Bronze Statue Smart Material to the mesh. Once you apply a Smart Material, you will see that, instead of a single material layer, it's a folder that consists of several layers as well as some procedural effects that have been stacked together to create the final material (see Figure 7-20).

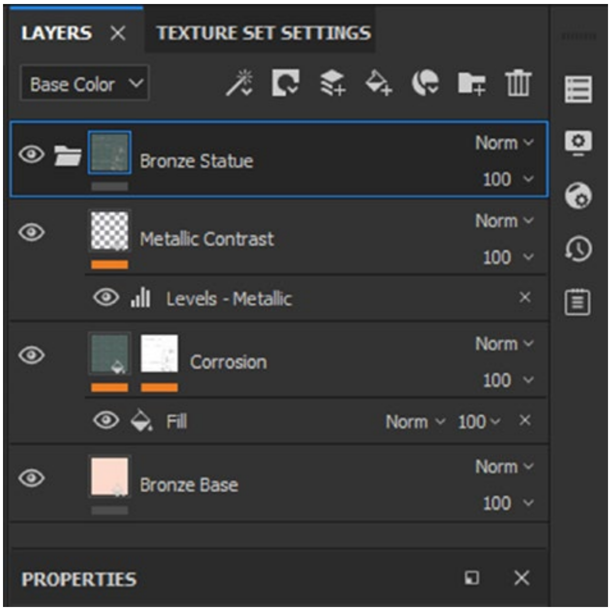


Figure 7-20. *Layers and effects stacked to form a Smart Material*

As you can see, there is a lot that can be changed to modify the look of the Smart Material, but it is also very complex and unintuitive. In this project, we will create our own Smart Material by stacking together several layers. That way, we can apply the same material to the whole temple without having to re-create the material every time we texture a new asset. Let's delete this Smart Material from the layer stack by selecting the folder that houses the Smart Material and deleting it

We can now start the texturing process. First we will need a stone material that will form the base for the material. In this case, I drag and drop a limestone asset that I found on Substance Source. You can either drop your material directly on the mesh or drop it in the layer stack. Either way, your model will be textured and the material properties will show up in the Properties window. You can also modify certain settings. The first thing that you can change is the texture projection type. By default, it is set to UV Projection. Click on the Projection drop-down menu and change it to Tri-Planer Projection (see Figure 7-21). Check out the difference that it makes. Pick the one that one looks better to you.

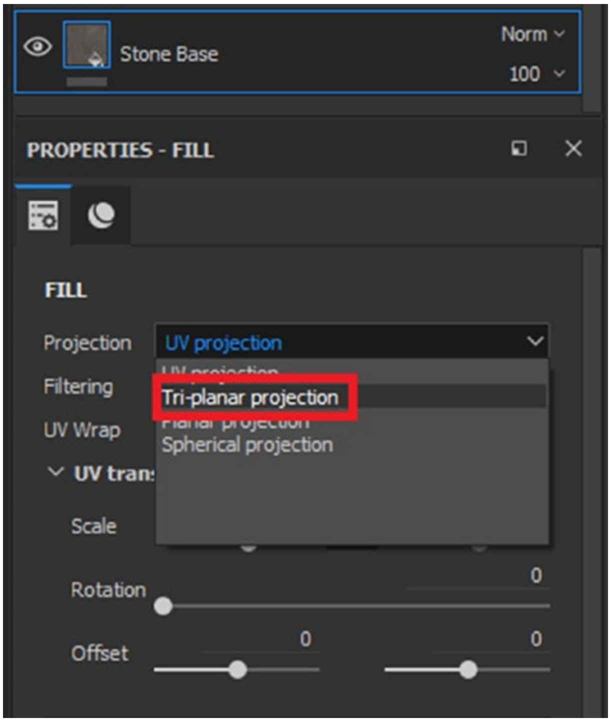
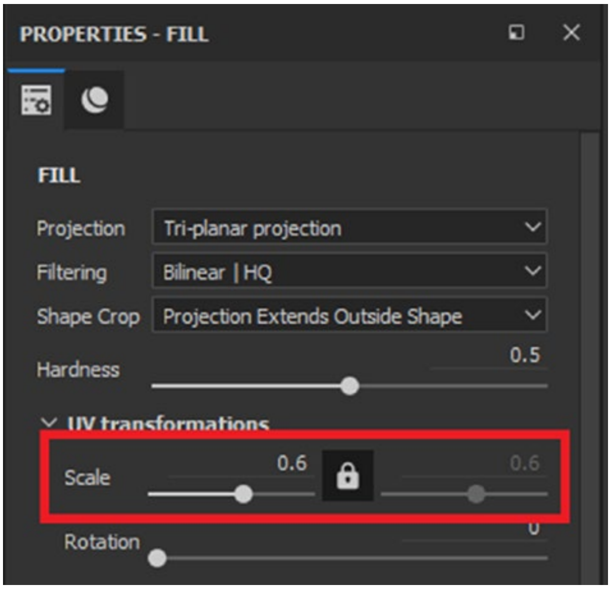


Figure 7-21. *Changing the texture projection type of the material*

There are a lot of options that you can change. Let's now move on to UV Transformations. You can change the tiling of the material by modifying the Scale value (see Figure 7-22).


PROPERTIES - FILL

FILL

Projection: Tri-planar projection
Filtering: Bilinear | HQ
Shape Crop: Projection Extends Outside Shape
Hardness: 0.5

UV transformations
Scale: 0.6 0.6
Rotation: 0

Figure 7-22. *Changing the tiling of the material*

Try increasing and decreasing the scale to find a suitable value that you like. You will notice that when you change the Projection to Tri-Planar Projection, the texture suddenly starts appearing more tiled. When you switch back to UV Projection, the texture will appear less tiled (in other words, it will be scaled down). So keep that in mind when changing your projection type.

Close the Texture Set List window because we won't be using it and it is occupying space unnecessarily (see Figure 7-23).

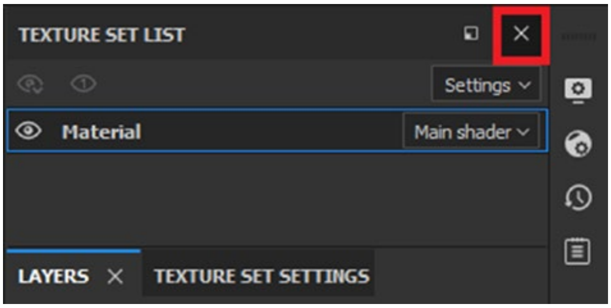
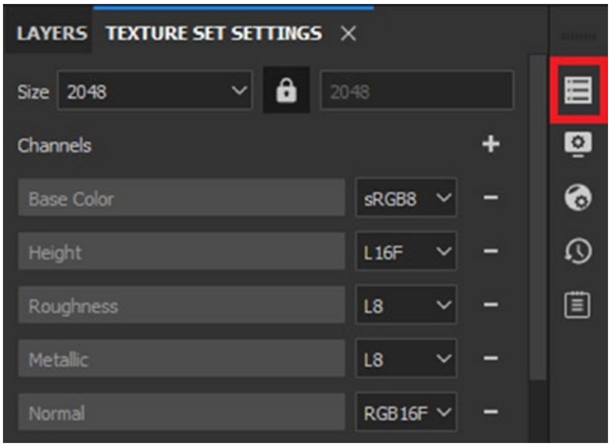


Figure 7-23. *Closing the Texture Set List window*

When you close it, you will see its icon appear on the right side of the window; you can open it anytime by clicking on the icon (see Figure 7-24).



Size 2048 2048

Channels

Base Color	sRGB8
Height	L16F
Roughness	L8
Metallic	L8
Normal	RGB16F

Figure 7-24. *Minimized Texture Set List icon*

You can extend the important windows so that you have more room to work with them. So let's proceed with texturing. In Figure 7-25, you can see the current result of the material applied to the mesh. You should try to achieve a similar result. With downloaded materials, you might not get exactly what you want, so it may be hard to make something that looks exactly similar.

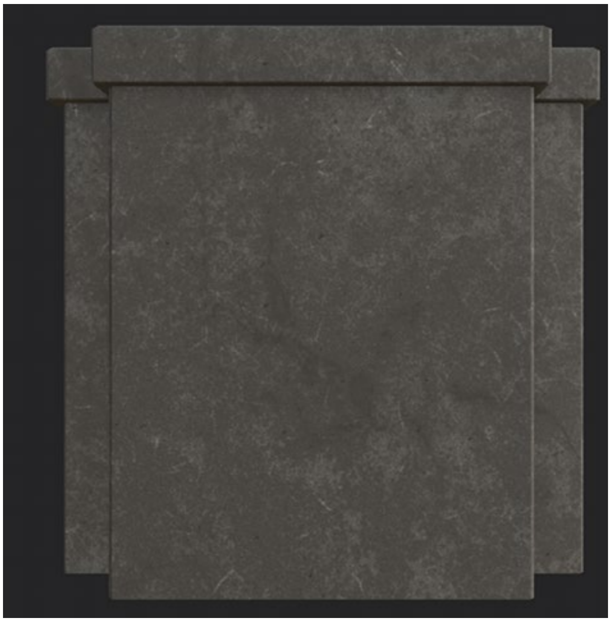
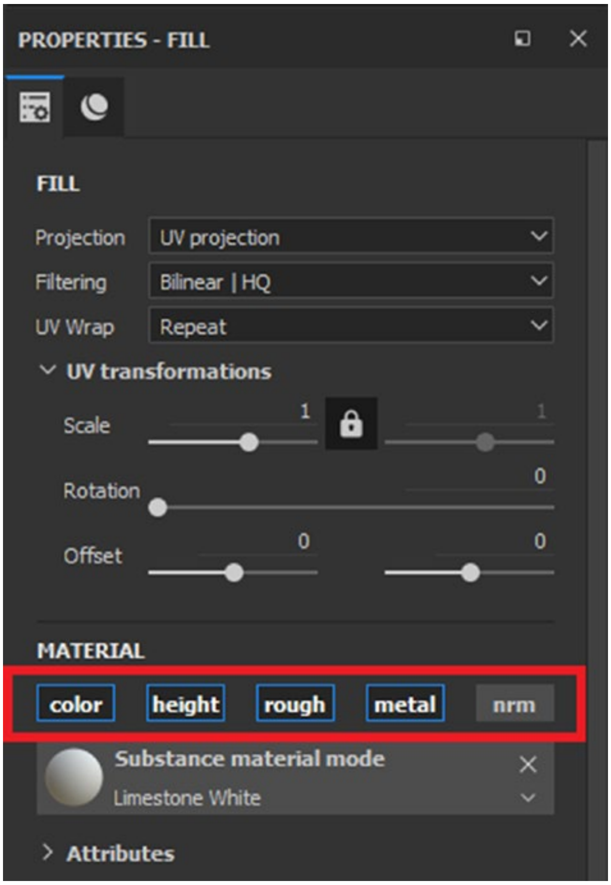


Figure 7-25. *Result of texturing so far*

You have created the base for the texture, so it's time to stack some more details on top of it to get the result that you want. Let's get a sandstone material and apply it to the top of the previous material. This material should be on top of your previous material in the layer stack. This layer will provide the primary rough sandy color to the mesh. In the Layer Properties window, scroll down and find the active channels. Disable the Rough and Metal channels by clicking on them, as shown in Figure 7-26.



PROPERTIES - FILL

FILL

Projection: UV projection

Filtering: Bilinear | HQ

UV Wrap: Repeat

UV transformations

Scale: 1 / 1

Rotation: 0

Offset: 0 / 0

MATERIAL

color height rough metal nrm

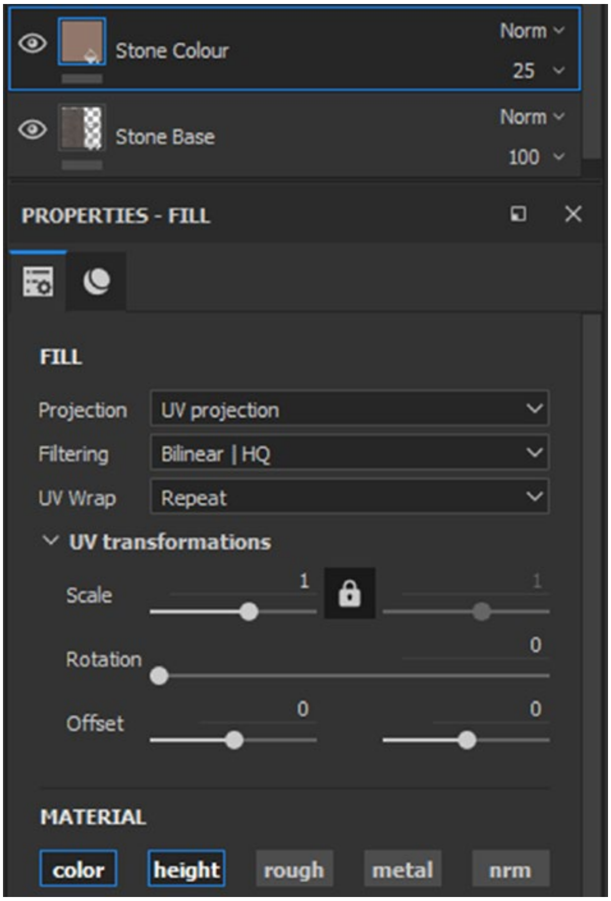
Substance material mode

Limestone White

Attributes

Figure 7-26. Material channels active/inactive

Active material channels are shown highlighted in blue, while the disabled channels are grayed out. Your setup should look similar to Figure 7-27.



Stone Colour — Norm — 25

Stone Base — Norm — 100

PROPERTIES - FILL

FILL

Projection: UV projection

Filtering: Bilinear | HQ

UV Wrap: Repeat

UV transformations

Scale: 1 / 1

Rotation: 0

Offset: 0 / 0

MATERIAL

color height rough metal nrm

Figure 7-27. *Setup of the materials*

Next, let's see how the material looks. Your outcome should be something similar to, if not the same as, Figure 7-28.

Figure 7-28. *Result so far*

Next, let's tone down the transparency of the color layer so that some of the details from the layer below it are visible as well. First make sure that you are in the Base Color mode in the Layer Stack window (see Figure 7-29).

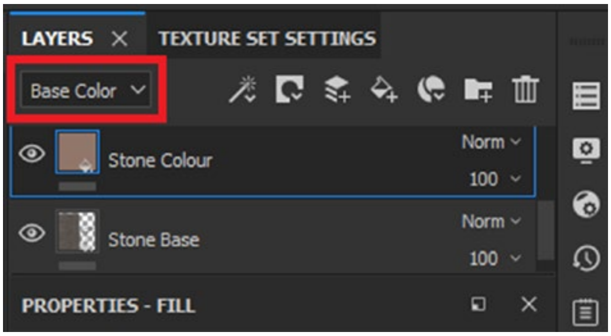


Figure 7-29. *Texture mode of the layer*

If you are in Base Color, then proceed; otherwise, click on the drop-down menu highlighted in Figure 7-29 and switch to Base Color. Next, we will reduce the transparency by using the transparency slider (see Figure 7-30) and tone down the transparency to something between 30-50 (or whatever looks good to you).

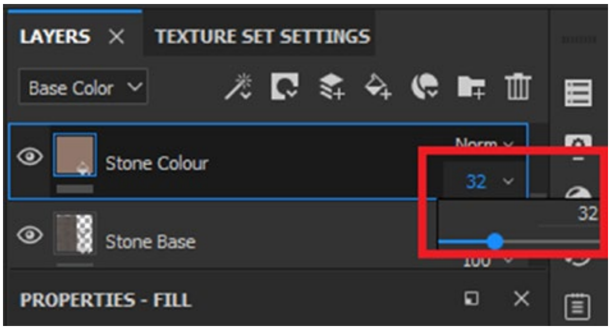


Figure 7-30. *Transparency slider*

When you reduce the transparency of the material, the material below it will start showing on the mesh. Your output should be similar to Figure 7-31.

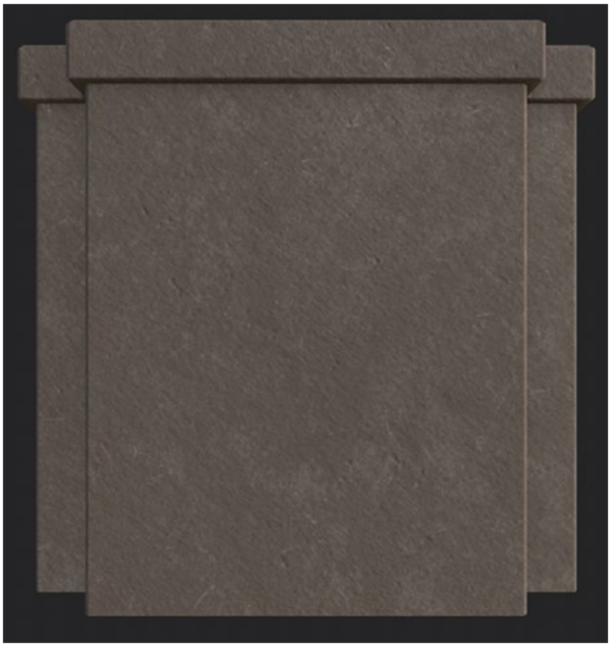


Figure 7-31. *Result so far*

As you can see in Figure 7-31, when we reduce the transparency of the color layer, lots of the damage and color clusters present in the layer below become visible. This will give it a better look than one clean material. The color patterns from the layer below break the flatness of the uniform color and give it a more natural, weathered look.

Next we are going to add another Fill layer to the layer stack by clicking on the Add Fill Layer icon (see Figure 7-32).

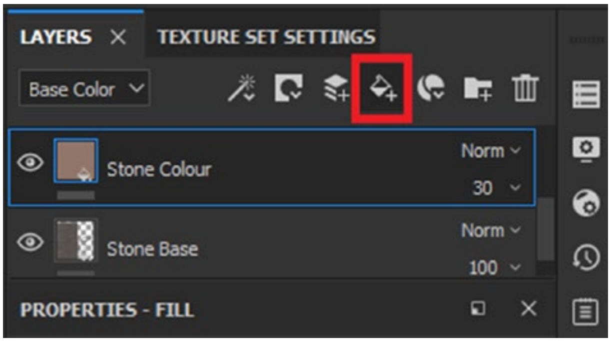


Figure 7-32. *Add Fill Layer icon*

Your new fill layer will be added and you will see its effect immediately on your mesh. A fill layer will be created with the default fill settings. You can see those settings by scrolling down in the Layer Properties window under the Material category. Here, we will change a few things. First, we will disable all channels except Base Color. Then we will change the base color to a deep gray color, as shown in Figure 7-33.

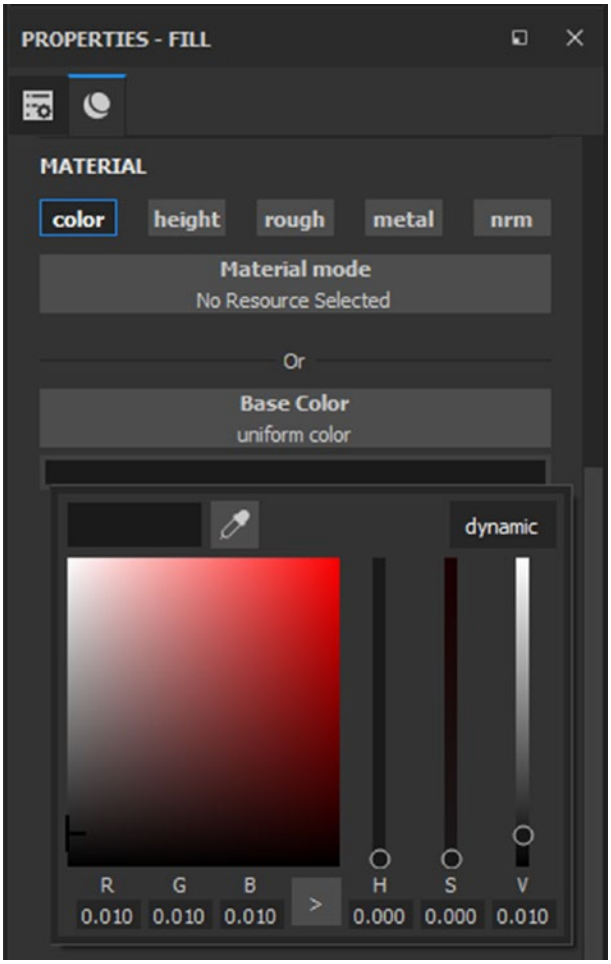


Figure 7-33. *Setting the color of the fill layer*

As you can see, the fill layer currently affects the whole mesh. But we want to limit its influence only on the extruded part. For that, we will use a mask. A mask is a grayscale texture that determines the transparency of the object/texture on which it is applied. Black represents completely transparent or invisible, while white represents completely opaque or visible. Black regions will not contain the masked texture, while white regions will. Right-click on the fill layer that you created and choose Add Black Mask from the list (see Figure 7-34).

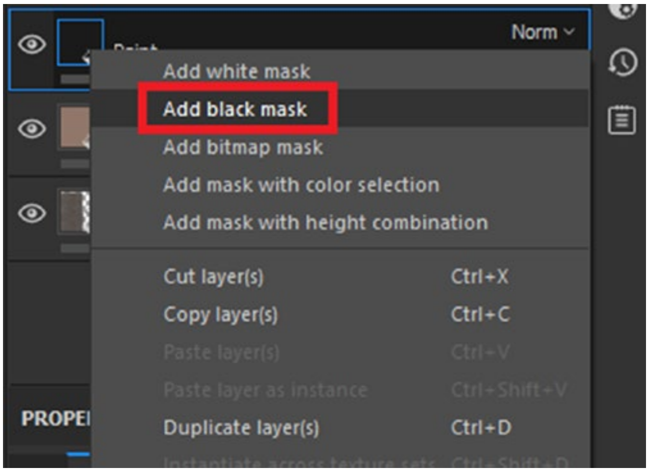
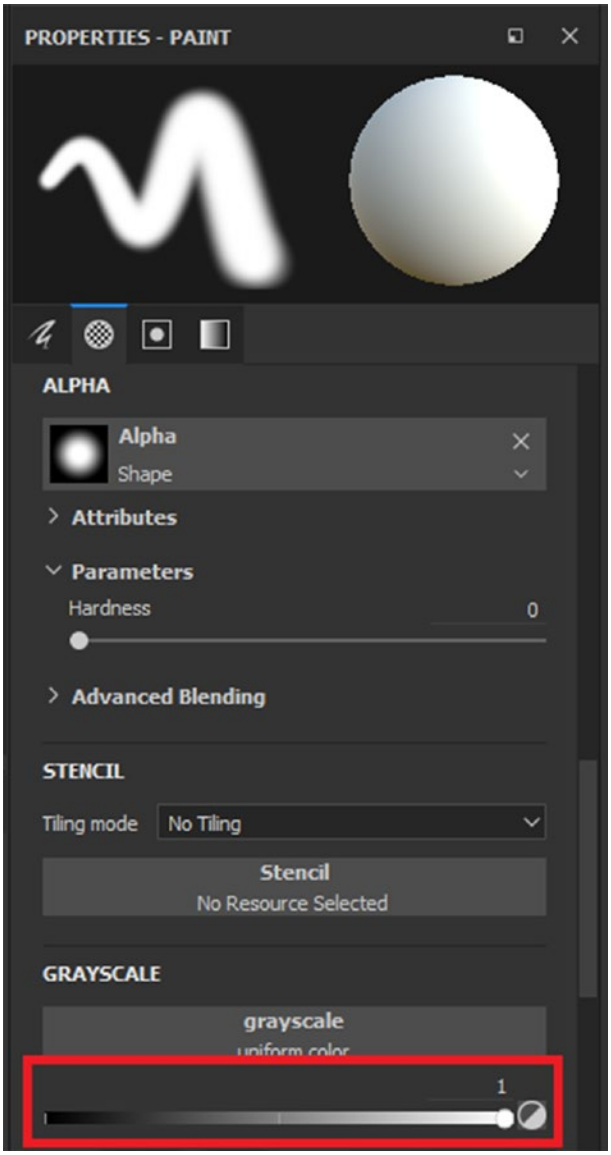

Norm
Add white mask
Add black mask
Add bitmap mask
Add mask with color selection
Add mask with height combination
Cut layer(s) Ctrl+X
Copy layer(s) Ctrl+C
Paste layer(s) Ctrl+V
Paste layer as instance Ctrl+Shift+V
Duplicate layer(s) Ctrl+D
PROPE

Figure 7-34. *Adding black mask to our layer*

You will immediately see that the effect of the fill layer disappears. The black mask has made everything in the layer completely transparent. Now, with the mask selected, view the Properties window. You will see mask-related settings on it now (see Figure 7-35). You can paint in or out the areas that you want to be affected by the mask. There are several ways to do this. First, ensure that the Grayscale slider is set all the way to white, as shown in Figure 7-35.



PROPERTIES - PAINT

ALPHA

Alpha
Shape

Attributes

Parameters

Hardness 0

Advanced Blending

STENCIL

Tiling mode: No Tiling

Stencil
No Resource Selected

GRAYSCALE

grayscale
uniform color

1

Figure 7-35. *Mask settings*

Now try painting on your mesh and see what happens. If you have set your Grayscale slider all the way to white, you will see that wherever you paint, the information from the paint layer starts showing up. To remove paint information, simply bring the Grayscale slider all the way back to black. Then when you paint you will be removing the painted information of the layer.

Next, let's learn about a more accurate way of editing the mask. We will use Polygon Selection tool to select which polygons we want our mask to affect. This way we don't have to manually paint the mask and worry about inconsistencies.

First, turn on the Polygon Selection tool while the mask is selected (see Figure 7-36).

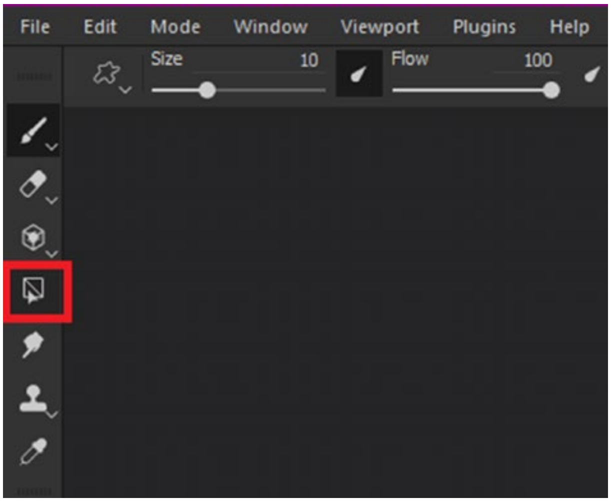


Figure 7-36. *Polygon Fill tool*

When you activate the Polygon Fill tool, you will see more settings related to it pop up in the horizontal bar, as shown in Figure 7-37.

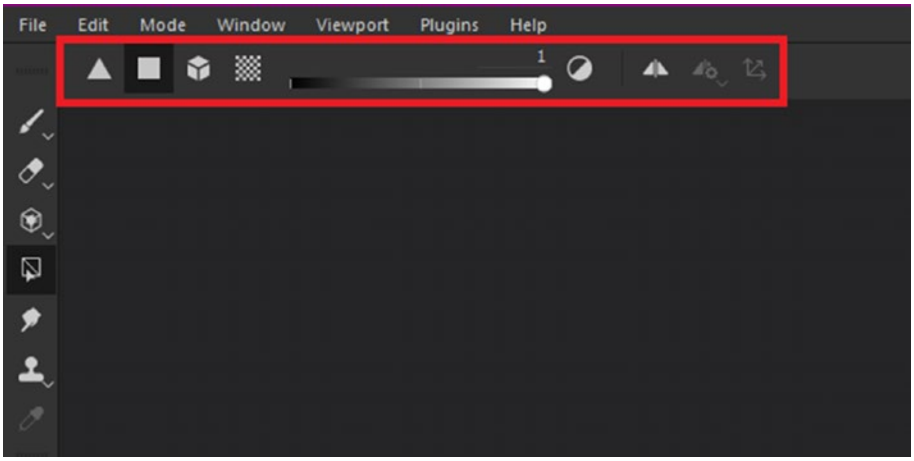


Figure 7-37. *Tool settings*

From left to right, the Polygon Fill tool options are:

- **Triangle Fill:** This fills the triangle present in the mesh.
- **Polygon Fill:** This fills an entire polygon at a time.
- **Mesh Fill:** This fills an entire connected mesh at a time.
- **UV Chunk Fill:** This fills all the faces that are contained in a single UV island.
- **Color Selection:** Using this you can change which color to fill. You can choose between grayscale values and black and white.
- **Invert Value:** This simply inverts the color value of the Color Selection tool.
- **Symmetry:** This toggles the symmetry, allowing you to make edits to one side and the edits will automatically carry over to the mirrored, opposite side.

Click on the Polygon Fill icon highlighted in Figure 7-38.

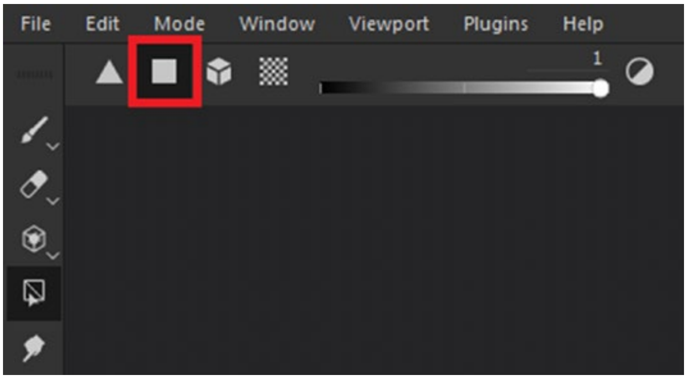


Figure 7-38. *Polygon Fill tool*

Once the tool is selected, your mesh will highlight all the selectable polygons. The wireframe on your mesh will help you identify what is selectable and what isn't (see Figure 7-39).

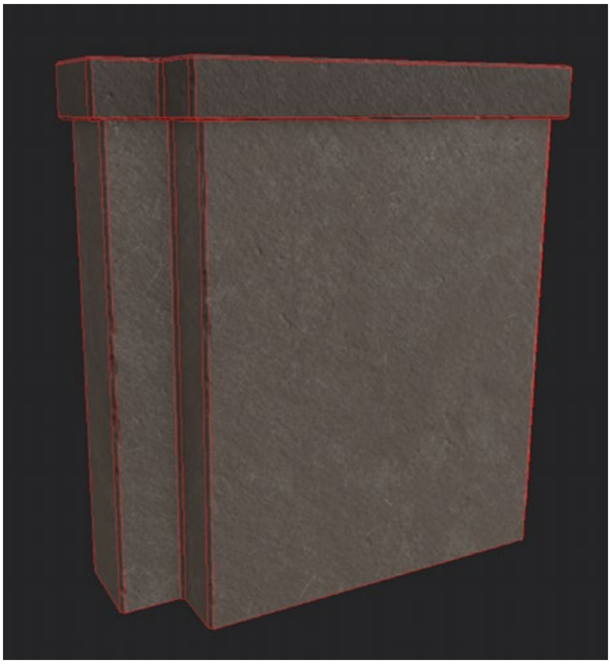


Figure 7-39. *Wireframe showing selectable polygons*

Try clicking on your mesh and you will see the material present on the layer. It will start appearing anywhere you click as long as the mask color is set to white. The opposite is also true; if your color is set to black, the paint information of the layer will disappear from any region where you click.

Try drawing a selection box by clicking, holding, and dragging. This will allow you to select a wider area where you want to apply the paint information of the layer. So let's use this information to apply the dark paint color to the extruded region of our mesh.

First make sure that the mask called Color Selection in the mask settings is set all the way to white; then draw a selection box similar to what is shown in the Figure 7-40. This will select the entire top region.

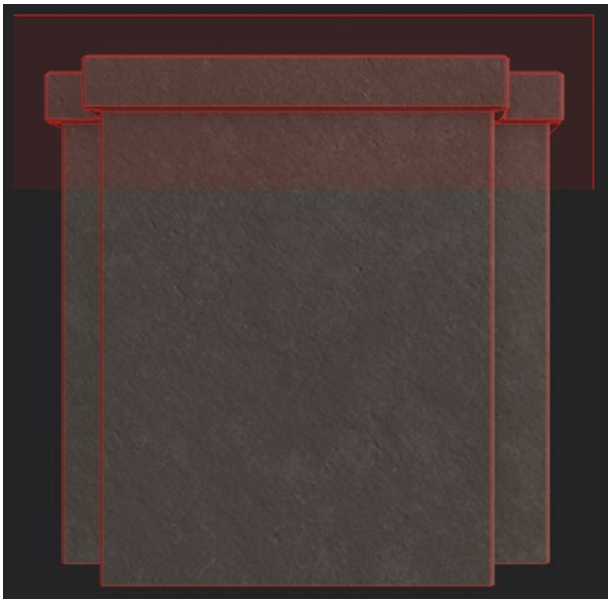


Figure 7-40. *Selecting using selection box*

Remember that this works for everything that is within the box. It does not matter whether it's visible or not, or behind some other mesh. So every face within this selection box will be selected regardless of its location and visibility. You will immediately notice that the layer's paint information is applied to all the faces.

Next, select the faces not to be affected by the paint layer. Set the color selection slider all the way to black and draw a selection box similar to Figure 7-41.

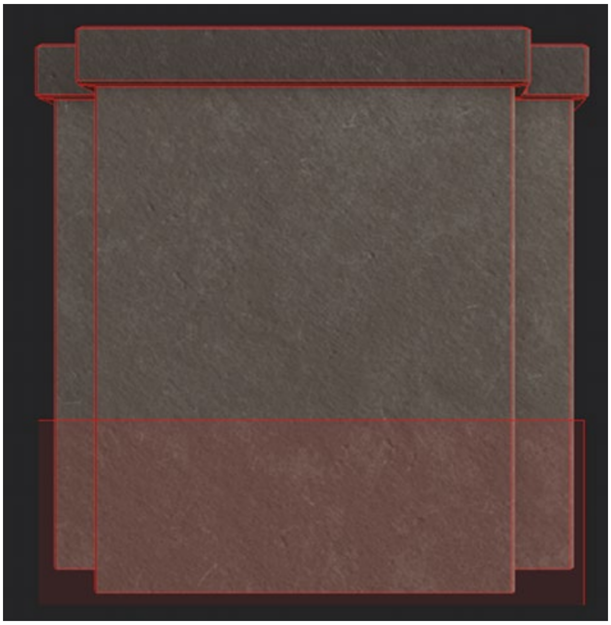


Figure 7-41. *Selecting the faces that you don't want painted*

Now you have painted the mesh in two different colors. This gives the mesh some variation. You can set any color you want for this piece; that's entirely your choice.

Next you need to add some more variation to this in the form of edge damage. If you check the edge of any real-world object, be it a table, box, wall, closet, etc., you will notice that most of the wear and tear happens on the edges of these objects (see Figure 7-42). This is probably because it is most exposed and the material is thinnest there.

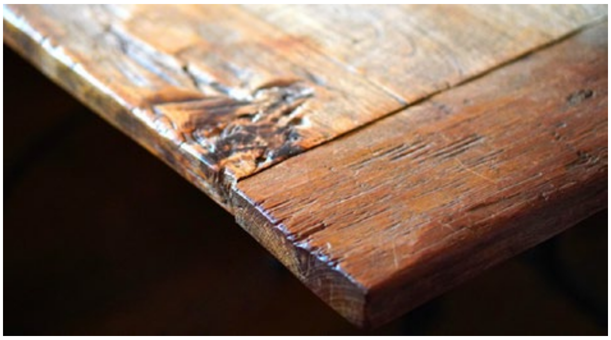


Figure 7-42. *Table showing edge wear (Image by Matthias Böckel from Pixabay)*

Figure 7-42 clearly shows edge wear on the edges of the table. This is a natural phenomenon. To add some realism to these textures, we will add some wear in Substance Painter.

So, the first thing that you need to do is create a new Fill layer just like we created before, by clicking on its icon. Disable all the channels except Color and Height. Set the color to a whitish gray, more on the whiter side, and set the height channel to `-0.5`. Your settings should look similar to Figure 7-43.

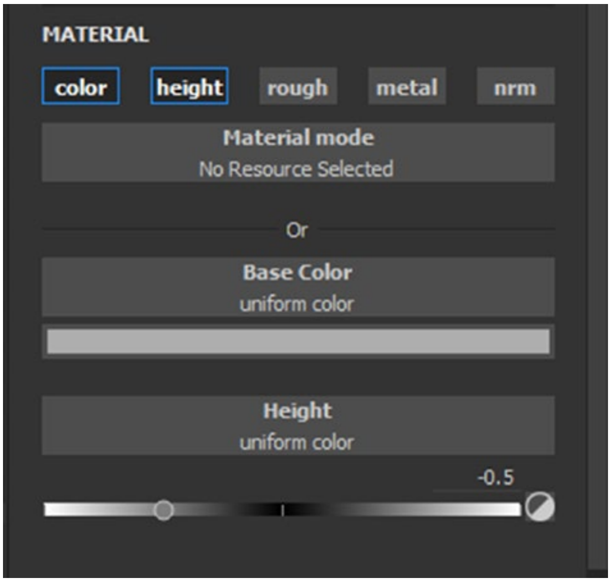


Figure 7-43. *Material settings for the edge wear layer*

As you can see, this fills up your whole object with the data on this layer. Right now, it may be hard to see what exactly the height channel is doing because it is working on the entire object and thus its effect is not comprehensible.

So, to see what is going on, we are going to add a black mask to this layer. You can do this by right-clicking on the layer in the layer stack. After adding the black mask, select the Brush tool in case you still have the Polygon Fill tool selected from the last session. To select the brush, click on its icon, which is highlighted in Figure 7-44.

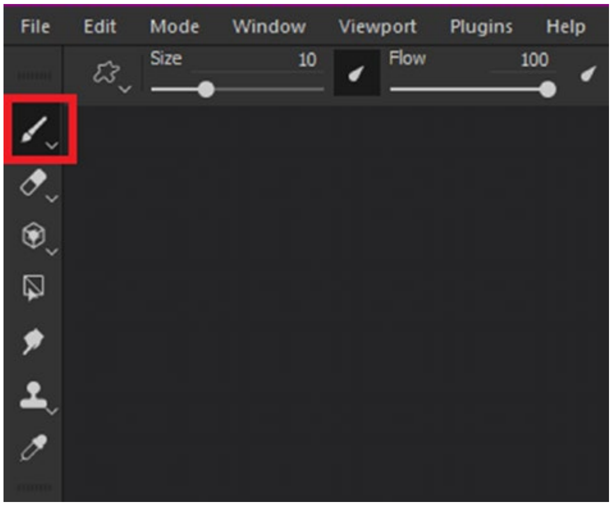


***Figure 7-44.** Brush tool*

Next, make sure that the color selection in the mask properties is set all the way to white and paint a few strokes on your mesh. You should have a result similar to Figure 7-45 if you have done everything right.

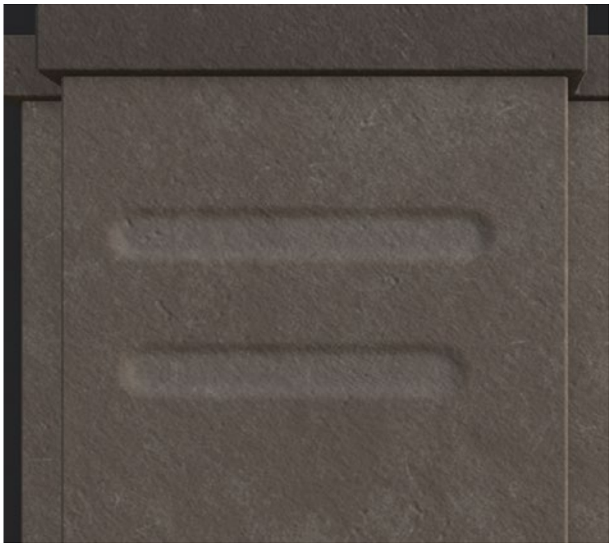


***Figure 7-45.** Effect of the fill layer*

As you can see in Figure 7-45, the paint strokes seem to dig into the mesh. This is exactly what we need. Remove the strokes by right-clicking on the mask and choosing Clear Mask (see Figure 7-46). This will reset the mask back to completely black.

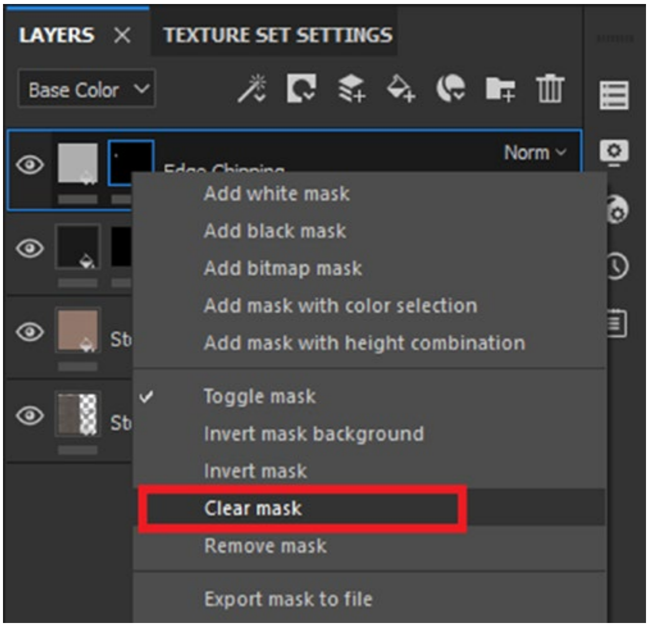


Figure 7-46. *Clearing the mask*

Now we will use a generator to procedurally create edge wear. In order to add a generator, you need to click on your mask container on the layer and then right-click after it is selected to open a menu related to the mask. Toward the bottom of this menu, you will find the Add Generator option (see Figure 7-47). Click on OK to add a generator to your mask.

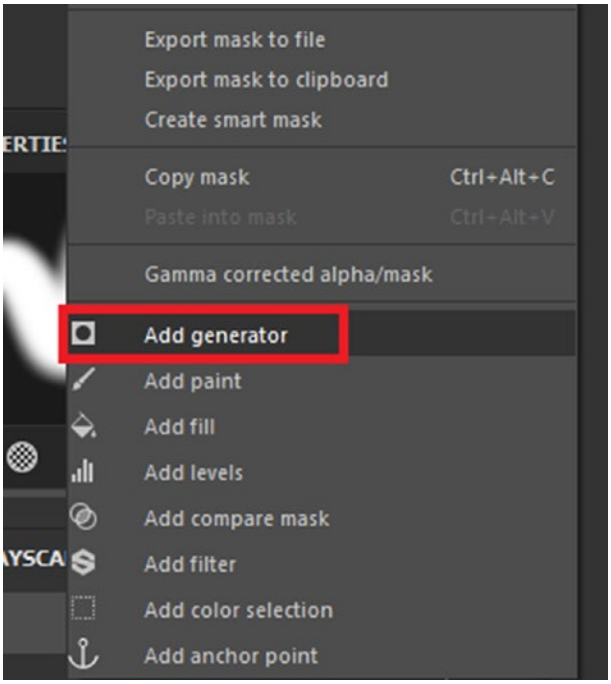


Figure 7-47. *Adding a generator to the texture*

Once the generator is added, you will see an empty generator slot in the Properties window (see Figure 7-48).

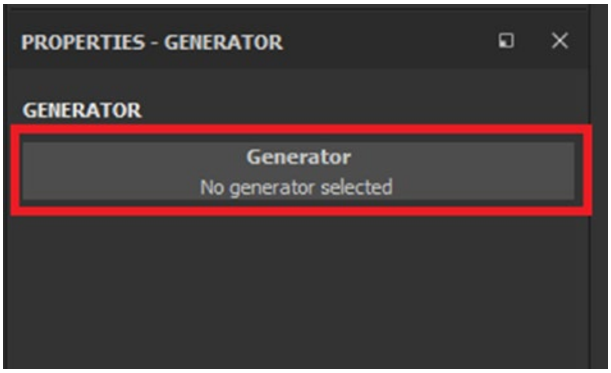


Figure 7-48. *Empty generator field/slot*

If you click on it, a new window will pop up giving you a menu of several generators that you can use for your work. Choose Mask Editor from this window (see Figure 7-49).

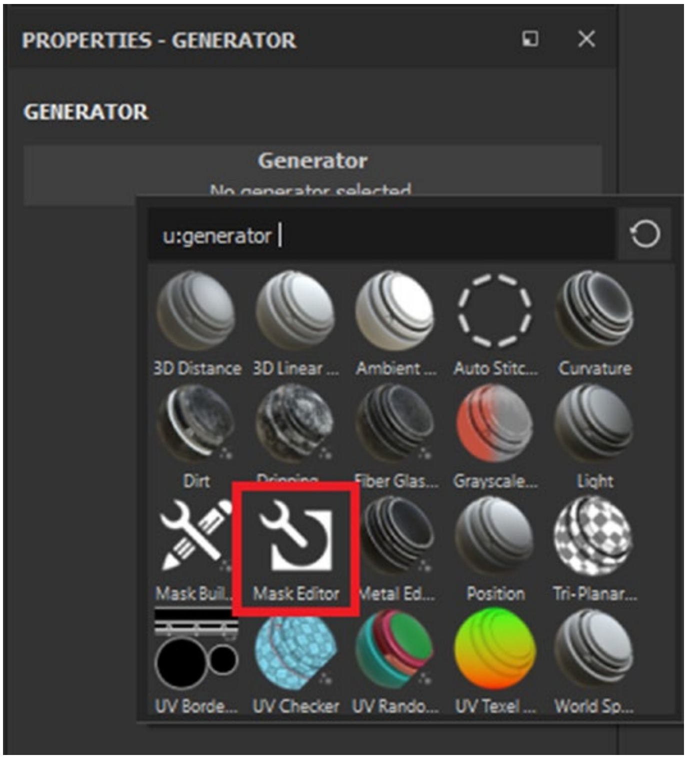
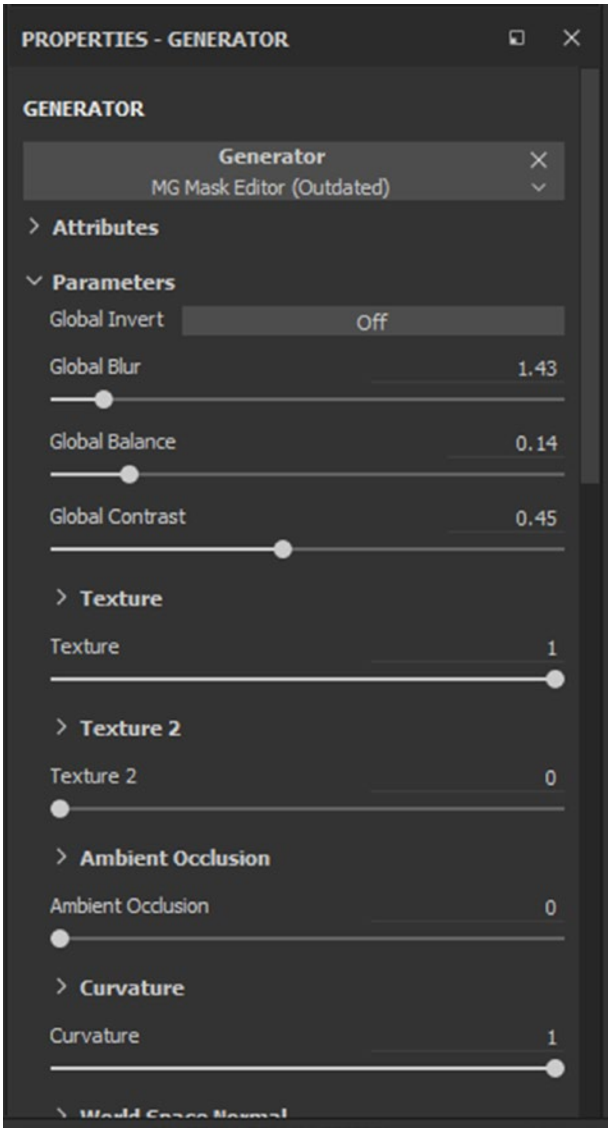


Figure 7-49. *Generator list*

When you add this generator, you will see that the Properties window will transform and all the parameters related to the Mask Editor generator will appear (see Figure 7-50). This new list of parameters will be large and daunting at first. But it gives you lots of control. For the purposes here, we will use only a few of them.


PROPERTIES - GENERATOR

GENERATOR

Generator
MG Mask Editor (Outdated)

Attributes

Parameters

Global Invert Off

Global Blur 1.43

Global Balance 0.14

Global Contrast 0.45

Texture

Texture 1

Texture 2

Texture 2 0

Ambient Occlusion

Ambient Occlusion 0

Curvature

Curvature 1

World Space Normal

Figure 7-50. *Generator settings*

We now need to edit this generator to procedurally create the edge chipping. To do that, scroll down the Properties window where it says Image Inputs (see Figure 7-51).

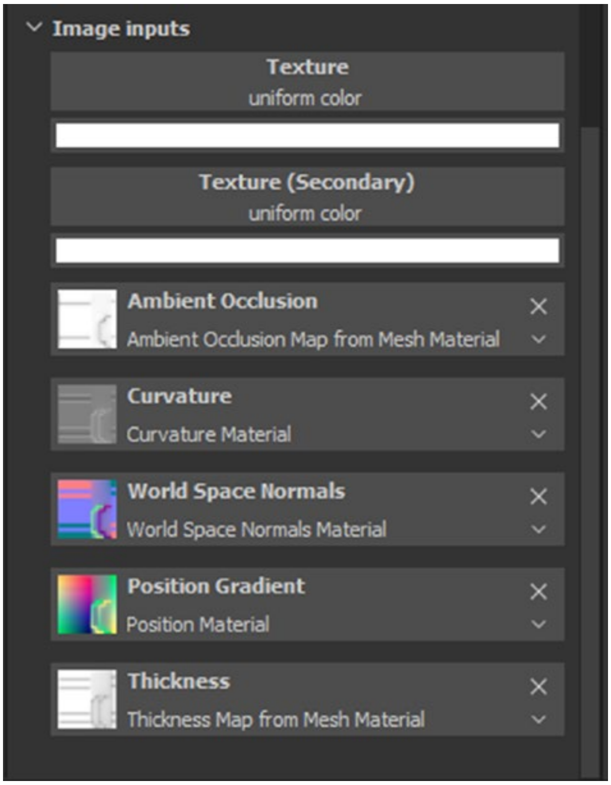


Figure 7-51. *Image input fields*

Click on the empty field that says Texture Uniform Color to open a new mini window. It will show a list of image inputs that you can apply to this field. In the search bar highlighted in Figure 7-52, type `concrete` next to procedural,texture.

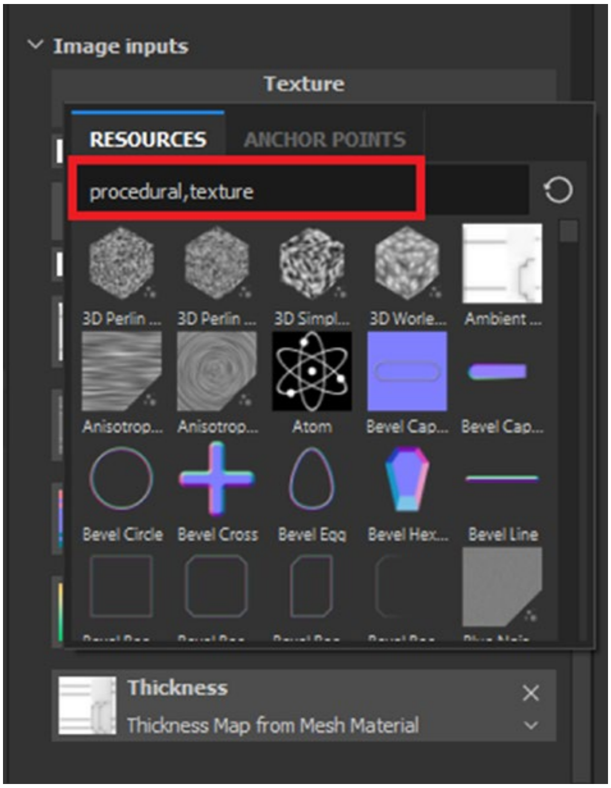


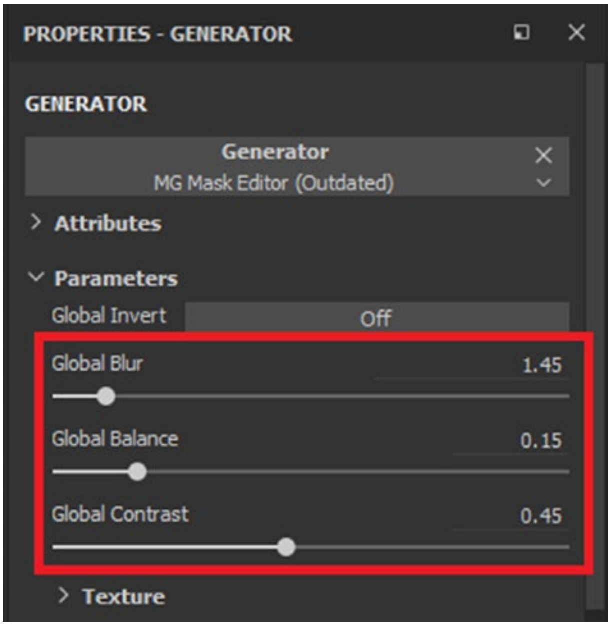
Figure 7-52. Image picker window

All the concrete grunge maps will appear when you type `concrete` in the search bar. Choose one from the list, the effect is mostly similar. Once you've applied the map, you will see something happening to the edges of your mesh. This will depend on your default settings, but we will tweak the values ourselves.

Scroll up to the top of the Properties window (see Figure 7-53) and change the following settings:

1. Set Global Blur to 1.45.
2. Set Global Balance to 0.15.
3. Set Global Contrast to 0.45.

You can also change the values according to the final result that you want. Finally, reduce the opacity of the layer to control the amount of color you want. You can also disable the color channel altogether if you want.


PROPERTIES - GENERATOR

GENERATOR

Generator
MG Mask Editor (Outdated)

Attributes

Parameters

Global Invert: Off

Global Blur: 1.45

Global Balance: 0.15

Global Contrast: 0.45

Texture

Figure 7-53. *Modifying the parameters*

If you have done everything correctly, your current result should look something similar to Figure 7-54.

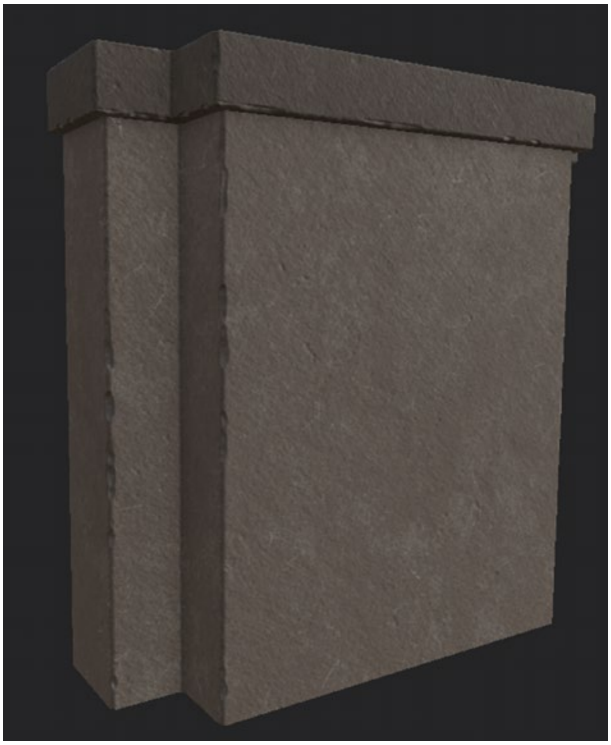


Figure 7-54. *Result so far*

You have now created edge damage using procedural generators. This adds another layer of realism to the mesh. Let's next see how you can create a Smart Material that can be reused on different meshes so that your work time and effort is reduced.

Start by creating a new folder by clicking on the Create Folder icon (see Figure 7-55). This will add an empty folder to your layer stack.

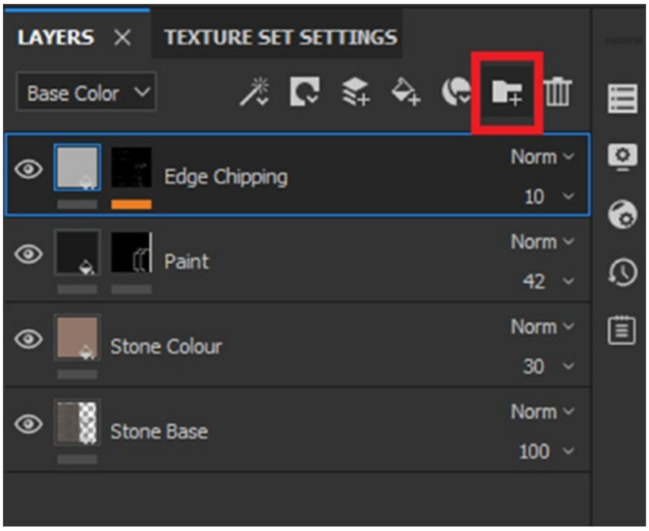


Figure 7-55. Creating a new folder

Select all your layers by Ctrl+clicking on them, then drag and drop them inside the new folder. Then double-click on the name of the folder so you can rename it (see Figure 7-56).

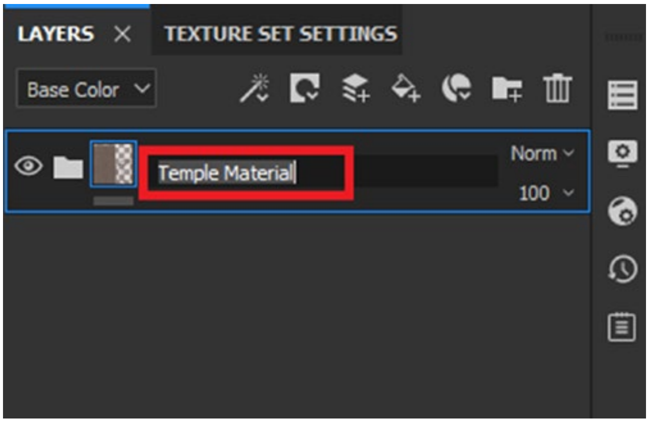


Figure 7-56. Renaming the layer

Give your layer a proper name, as the name of this layer will become the name of the Smart Material. In this case, I named it Temple Material.

Next, right-click the folder and choose Create Smart Material (see Figure 7-56). This will create a Smart Material with same name as the folder.

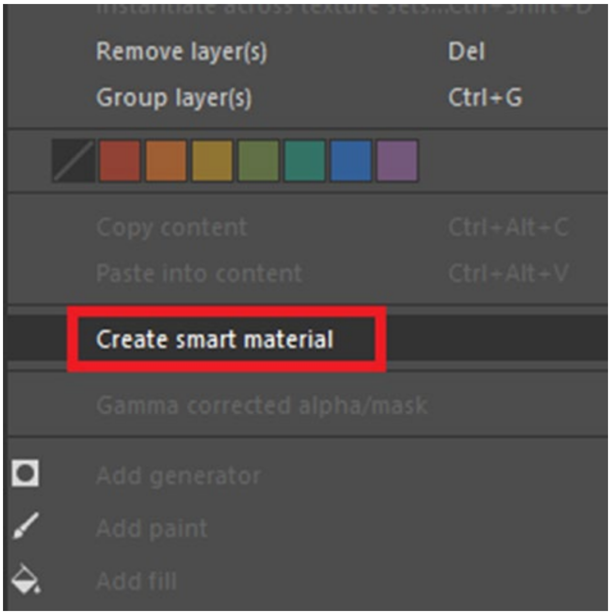


Figure 7-57. *Creating a Smart Material*

You will now be able to find this Smart Material in the Smart Materials tab of the Substance Painter “shelf” (see Figure 7-58).

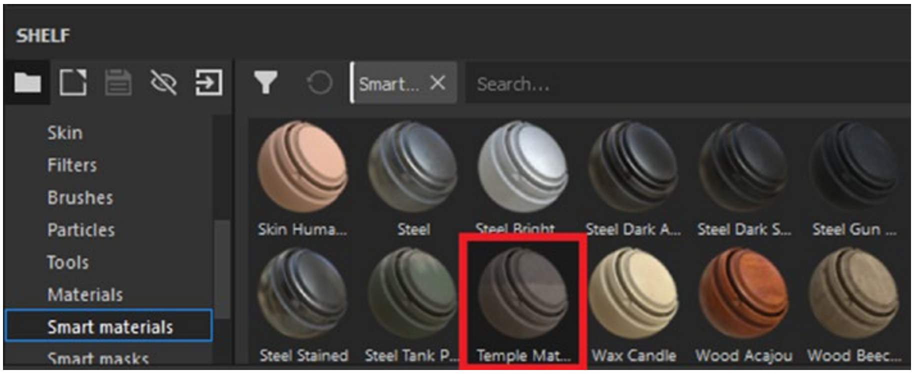


Figure 7-58. *The Smart Material in the shelf*

So the Smart Material is now in the shelf, ready to be applied to any mesh that you import into Painter. This material is adaptive and will automatically align itself to the topology of the imported mesh. The only thing that will not adapt to topology is the painting that we did to darken a specific part of the temple piece. But we still decided to include it. You can easily modify it or simply disable or remove it.

With that, we wrap up the texturing basics. We will now quickly texture the rest of the assets.

Texturing the Remaining Assets Using Smart Materials

Let's export another asset from Blender. This time it will be the bottom wall of the main temple body (see Figure 7-59).

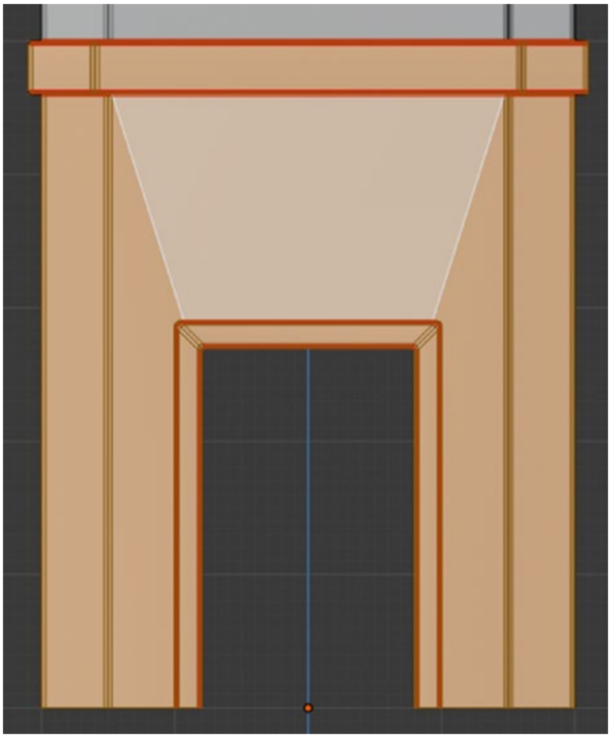
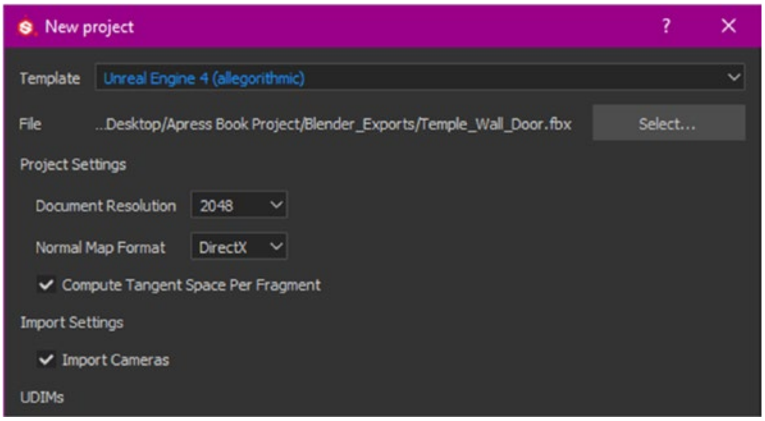


Figure 7-59. *The next asset to be textured*

Import it into Substance Painter, but this time we are going to change the import preset. Switch the preset from PBR Metallic Rough to Unreal Engine 4. Also check the Compute Tangent Space per Fragment option if it's not checked automatically. See Figure 7-60.



New project

Template: Unreal Engine 4 (allegorithmic)

File: ..Desktop/Apress Book Project/Blender_Exports/Temple_Wall_Door.fbx

Project Settings

Document Resolution: 2048

Normal Map Format: DirectX

✓ Compute Tangent Space Per Fragment

Import Settings

✓ Import Cameras

UDIMs

Figure 7-60. *New preset for this project*

Click on the Import button to import the mesh in Substance Painter. Bake the materials again, like you did before. This time, disable the ID and Thickness maps. Bake using the same settings that you used before. After the bake is complete, drag and drop the Smart Material that we created on the mesh. Your result may or may not be similar to Figure 7-61.

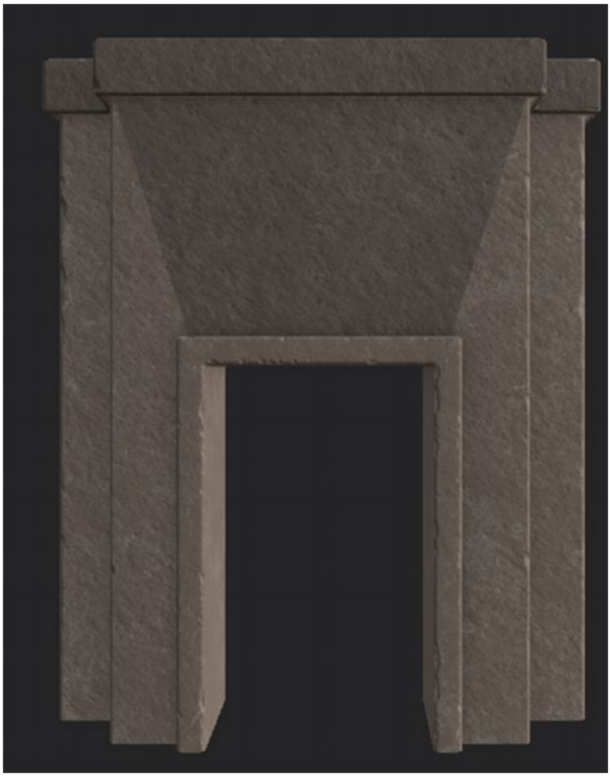


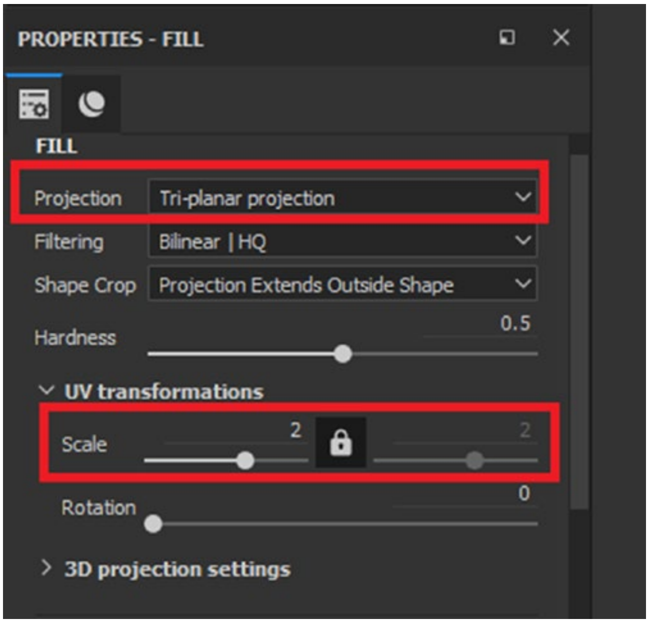
Figure 7-61. *Current result*

As you can see in Figure 7-61, the color is being mapped incorrectly. You might not have a similar problem, depending on many factors. But any problem like this can be fixed very easily. All you need to do is that edit the mask of the Paint layer.

Select the mask and then go to the Polygon Fill tool. With the mask color set to black, click on the wrongly colored polygon to clear the color. If any area does not have correct color, then adjust the mask and use the Polygon Fill tool to fill it in accordingly.

If you find the tiling is incorrect or the cracks are too large or small, there is way to fix them as well. Let's tackle this one by one.

1. First you will fix the tiling on the grainy stone material. To do this, select the layer that has the material. Go to Layer properties and change the Projection option to Tri-Planar Projection and increase the Scale to 2 (see Figure 7-62).


PROPERTIES - FILL

FILL

Projection: Tri-planar projection

Filtering: Bilinear | HQ

Shape Crop: Projection Extends Outside Shape

Hardness: 0.5

UV transformations

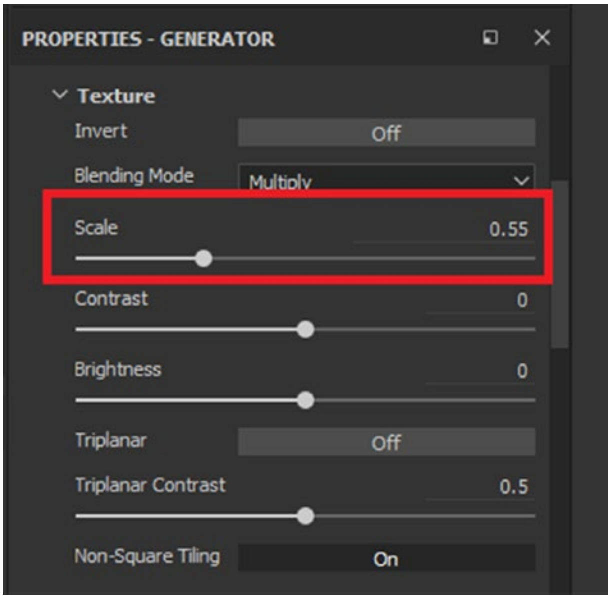
Scale: 2 | 2

Rotation: 0

3D projection settings

Figure 7-62. *Modifying the layer's parameters*

2. Next you are going to modify the tiling of the edge damage layer. Select the mask that contains the generator and click on the generator in the layer stack. Options related to it should appear in the Properties window. Scroll down until you see a collapsed category called Texture. Expand it by clicking on the arrow next to it. You should see the Scale parameter under it. Increase the scale until you are happy with the result of the cracks generated. See Figure 7-63.


PROPERTIES - GENERATOR

Texture

Invert: Off

Blending Mode: Multiply

Scale: 0.55

Contrast: 0

Brightness: 0

Triplanar: Off

Triplanar Contrast: 0.5

Non-Square Tiling: On

Figure 7-63. *Changing the scale of the texture used in the generator*

Hopefully you have the result that you want. You can modify the material in any way from this point. It is important to always have a reference so you can see what you want to achieve. Current results so far are shown in Figure 7-64. You can certainly try to achieve a result similar to it.

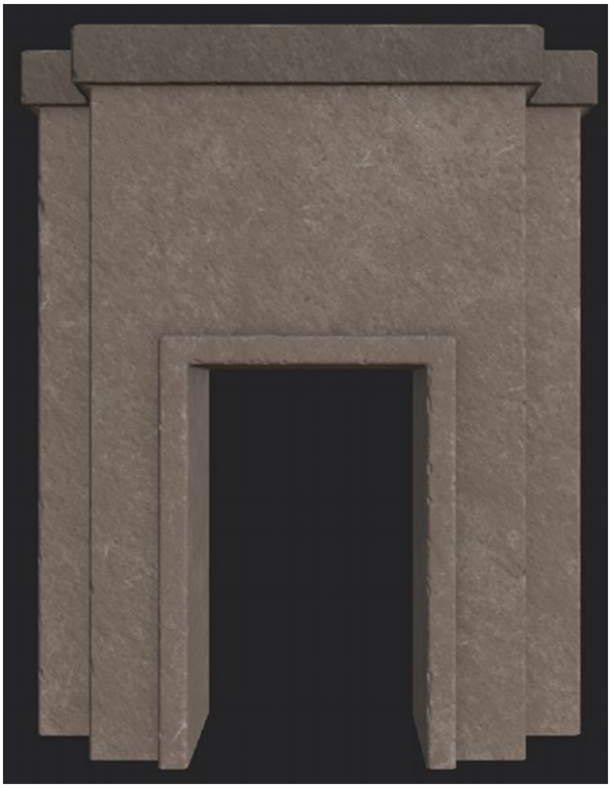


Figure 7-64. *Results so far*

Next, we are going to add carving details to the mesh. But before we do that, we will need alpha maps with carving information. In the previous chapters, we mentioned websites for getting such things. Look around until you find something good. In the next chapter, we will create game-ready foliage like grasses, bushes, and trees.

CHAPTER 8

Creating Foliage

In this chapter we are going to learn how to quickly create some foliage for our scene. Foliage is generally very difficult to create manually and a very time-consuming process, so this chapter introduces some tools and libraries that can make it easier. It is important to understand that creating foliage that has an organic, natural shape can be difficult at times, and there are several dedicated tools available at your disposal that allow you to overcome this problem. There are also some online libraries that have ready-made foliage available for you to download and use out-of-the-box.

Creating Grass

Let's start with the simplest foliage model first, grass. In game engines, grass is made up of planes with masked textures applied to them. We are going to use Quixel Bridge for this purpose. Go to `quixel.com` and download Quixel Bridge. This software and all its assets are completely free for use with Unreal Engine 4.

Once you have downloaded and installed Quixel Bridge, it's time to set it up to work with UE4. See Figure 8-1.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_8

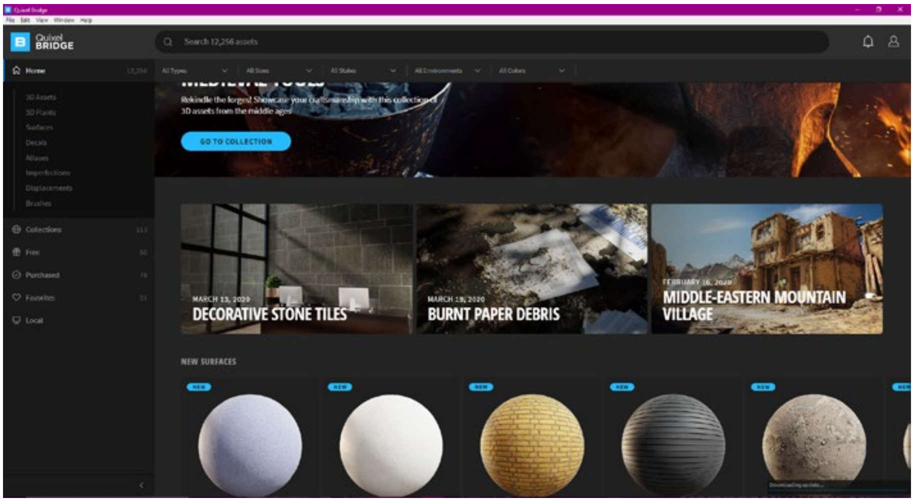
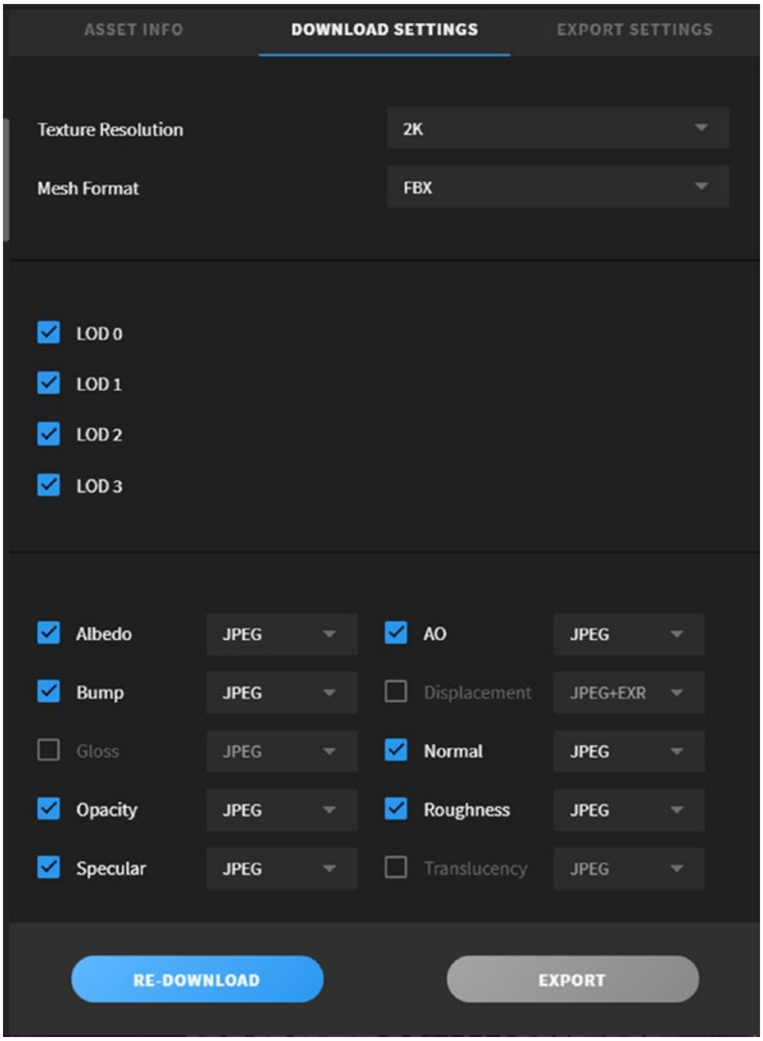


Figure 8-1. *Quixel Bridge*

Use the search bar to search for 3D grass assets. You can find plenty of them in Quixel Bridge. Use the download settings shown in Figure 8-2.


ASSET INFO | DOWNLOAD SETTINGS | EXPORT SETTINGS

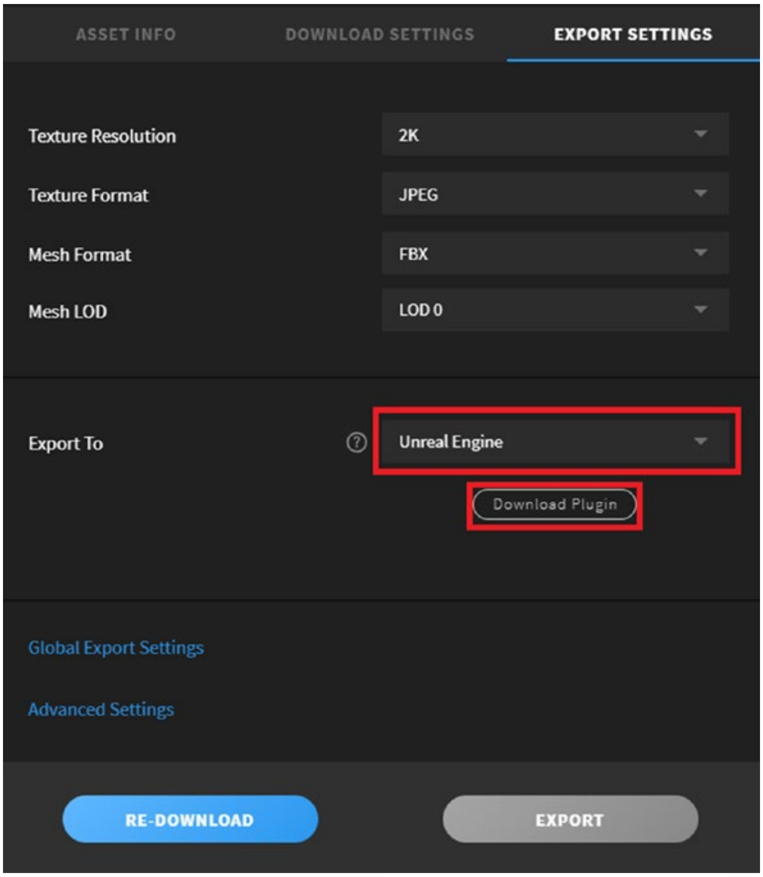
Texture Resolution	2K
Mesh Format	FBX

- [✓] LOD 0
- [✓] LOD 1
- [✓] LOD 2
- [✓] LOD 3

✓	Albedo	JPEG	✓	AO	JPEG
✓	Bump	JPEG	☐	Displacement	JPEG+EXR
☐	Gloss	JPEG	✓	Normal	JPEG
✓	Opacity	JPEG	✓	Roughness	JPEG
✓	Specular	JPEG	☐	Translucency	JPEG

Figure 8-2. *Download settings*

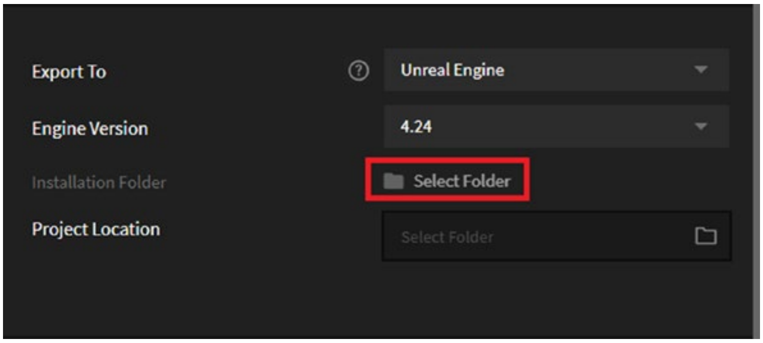
Next go to the Export Settings tab and change Export To to Unreal Engine. Then download the Unreal plugin. Set the Export settings to what you see in Figure 8-3.



Texture Resolution 2K
Texture Format JPEG
Mesh Format FBX
Mesh LOD LOD 0
Export To Unreal Engine

Figure 8-3. *Export settings*

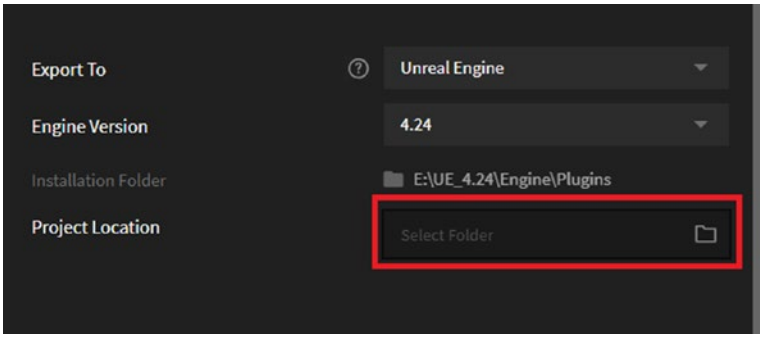
Once the plugin has been downloaded, set the Installation Folder to the directory where you installed UE4 (see Figure 8-4). UE4 may require a restart if an editor is running.



Export To Unreal Engine
Engine Version 4.24
Installation Folder Select Folder
Project Location Select Folder

Figure 8-4. *Set this to the Unreal Engine directory*

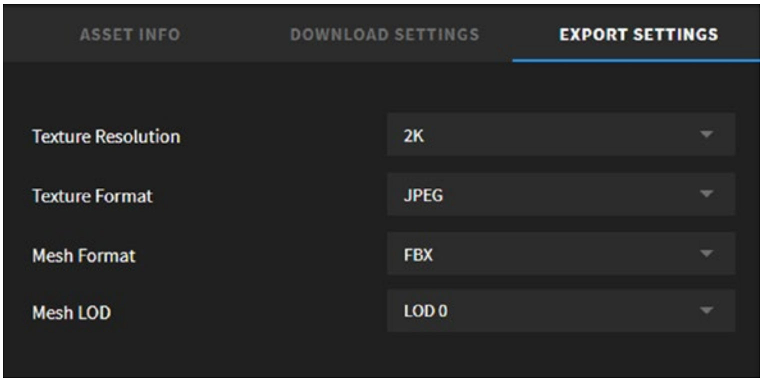
Next, you need to set the project location. For this, you need an Unreal Engine project. Go to the Unreal Engine section of this book to learn how to create a new project. It is really simple and you can create a temporary test project to see how this works. Next, click on the Project Location field and set the location to be the same as your project directory (see Figure 8-5).



Export To Unreal Engine
Engine Version 4.24
Installation Folder E:\UE_4.24\Engine\Plugins
Project Location Select Folder

Figure 8-5. *Project location*

Once you've completed these steps, the asset is ready for export. Set the Export settings as shown in Figure 8-6.



Texture Resolution 2K
Texture Format JPEG
Mesh Format FBX
Mesh LOD LOD 0

Figure 8-6. *Export settings for the asset*

Once everything has been set up, click on Export to send the file to UE4 and to your project. You should see the asset in your Content Browser right away.

Creating Trees

In this section we are going to create a simple tree using a free tool called TreeIt. Go to `www.evolved-software.com/treeit/treeit` to download it. This free tool allows you to create organic tree models using various easy controls. It also has a library of trees that you can download and randomize to get something that you want.

Once you have downloaded and installed the software, launch it and we will get started with the basics. The first thing that you will see when the software opens is a single tree trunk (see Figure 8-7).

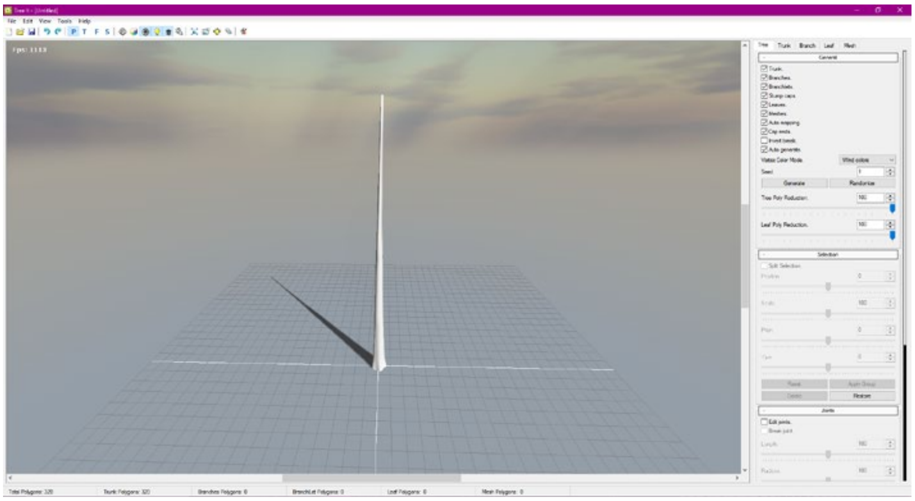
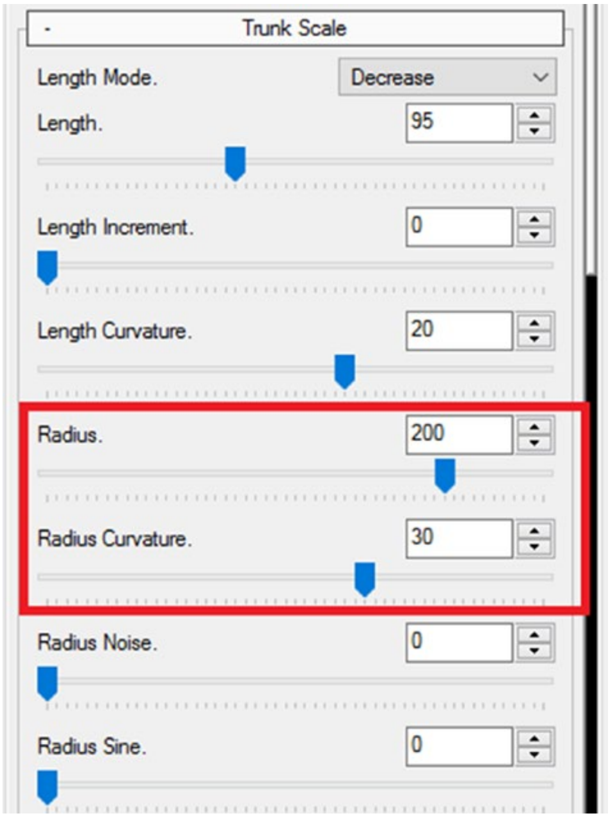


Figure 8-7. *TreeIt startup*

On the right side, you can see all the settings that you can modify that will alter the look of the tree you are creating. There are a ton of settings under different categories, namely: Tree, Trunk, Branch, Leaf, and Mesh. It is very easy to get lost and mess things up, so be very careful.

Let's create something now. We will begin by modifying the trunk. Set the Radius of the trunk to be something like 200 and reduce the Radius Curvature to 30 (see Figure 8-8).


- Trunk Scale

Length Mode. Decrease

Length. 95

Length Increment. 0

Length Curvature. 20

Radius. 200

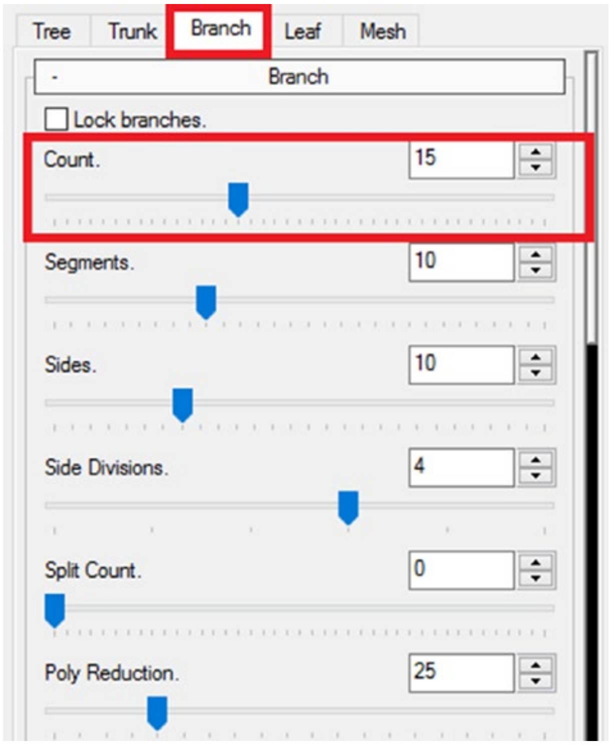
Radius Curvature. 30

Radius Noise. 0

Radius Sine. 0

Figure 8-8. *Modifying the trunk*

Next go to Branches and increase the Branch Count to about 15. Decrease the Branch Radius and Curvature values if the branches look too thick (see Figure 8-9).



Lock branches.
Count. 15
Segments. 10
Sides. 10
Side Divisions. 4
Split Count. 0
Poly Reduction. 25

Figure 8-9. *Increasing the branch count*

Let's apply some texture to the tree mesh. Go back to the Trunk tab and look for the Load Texture option (see Figure 8-10).

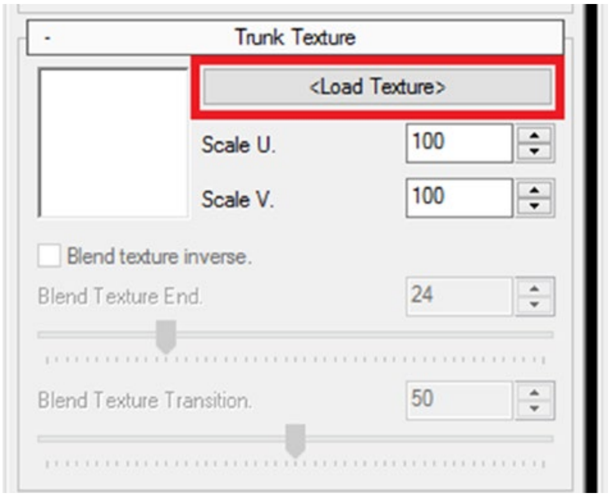


Figure 8-10. *The Load Texture button*

A new window will open, allowing you to choose textures. Use the drop-down menu to load `Bark_04` into the Base, Normal, and Roughness tabs (see Figure 8-11).

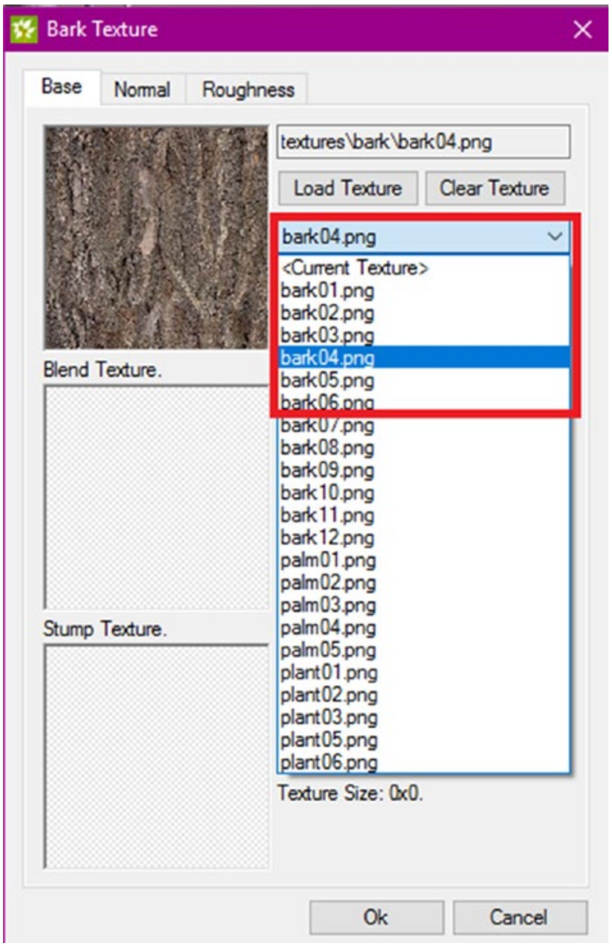


Figure 8-11. *Loading a texture*

For Blend Texture, choose any. Once you're done, click on OK. Your mesh will be textured and should look similar to Figure 8-12.

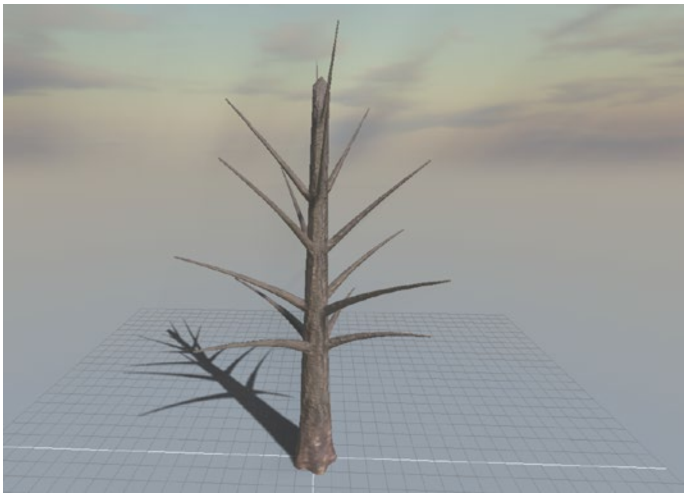
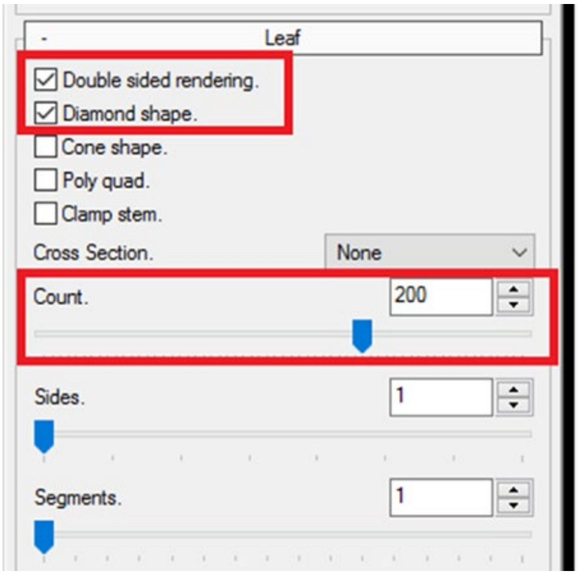


Figure 8-12. *Results so far*

Next let's add some leaves. Go to the Leaf tab, enable Double Sided and Diamond Shape, and increase the leaf Count to about 200 (see Figure 8-13).



Leaf

- [✓] Double sided rendering.
- [✓] Diamond shape.
- [ ] Cone shape.
- [ ] Poly quad.
- [ ] Clamp stem.

Cross Section. None

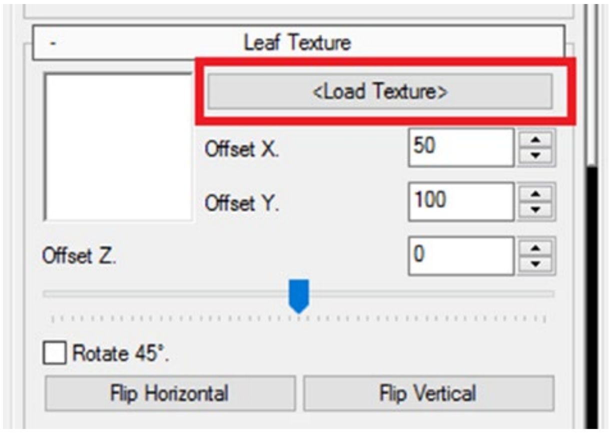
Count. 200

Sides. 1

Segments. 1

Figure 8-13. *Modifying leaf settings*

Next load the leaf textures by clicking on Load Texture under the Leaf Texture category (see Figure 8-14).



Leaf Texture

<Load Texture>

Offset X. 50

Offset Y. 100

Offset Z. 0

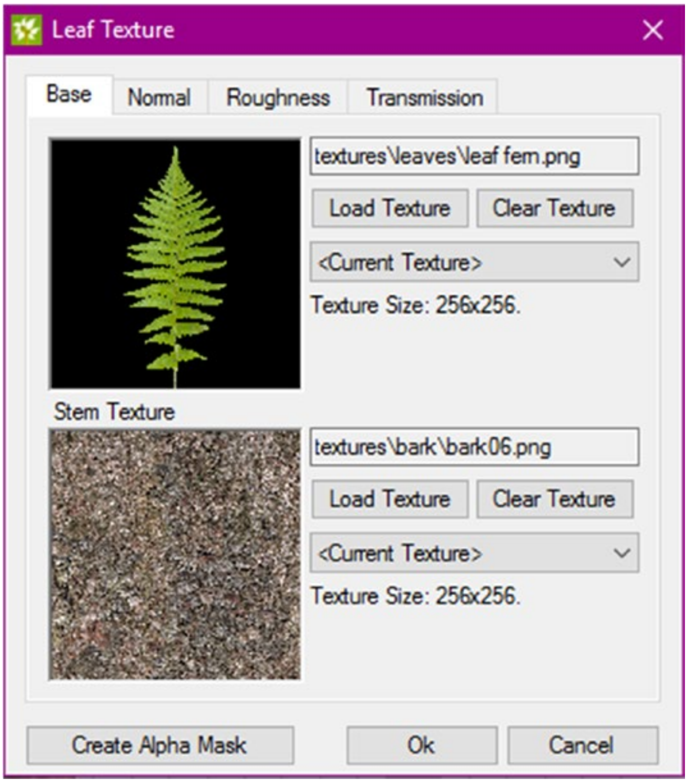
Rotate 45°.

Flip Horizontal

Flip Vertical

Figure 8-14. *Loading a leaf texture*

Just like you did for the trunk, choose a leaf texture of your choice for all the maps that the menu says. In this example, we chose the Fern texture (see Figure 8-15).



textures\leaves\leaf fern.png

<Current Texture>

Texture Size: 256x256.

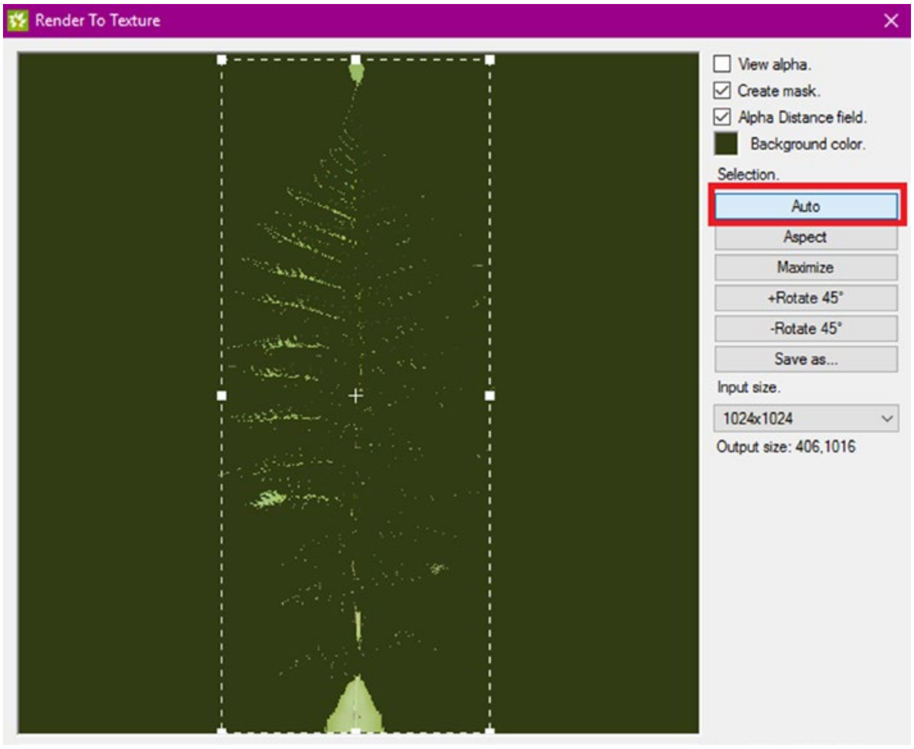
textures\bark\bark06.png

<Current Texture>

Texture Size: 256x256.

Figure 8-15. *Choosing a texture for the leaf*

Next choose Create Alpha Mask and, from the window that opens, choose Auto. Close the window (see Figure 8-16). Click on OK when you're done.



☐ View alpha.
☑ Create mask.
☑ Alpha Distance field.
Background color.
Selection.
Input size.
1024x1024
Output size: 406,1016

Figure 8-16. *Creating an alpha mask*

Your final result should look similar to Figure 8-17.

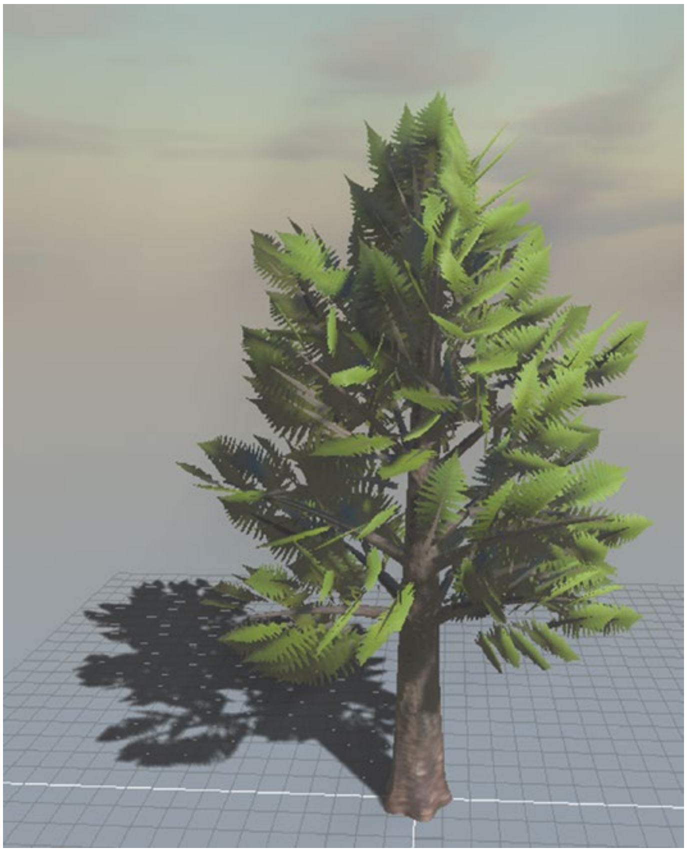


Figure 8-17. *Final result*

You can also go to TreeIt's website to download one of the pre-built tree templates and use it to get the result that you want.

After you are done, the only thing you need to do is export this tree. Go to File ➤ Export and choose an export format. I suggest you use the Autodesk Filmbox (.fbx) format (see Figure 8-18).

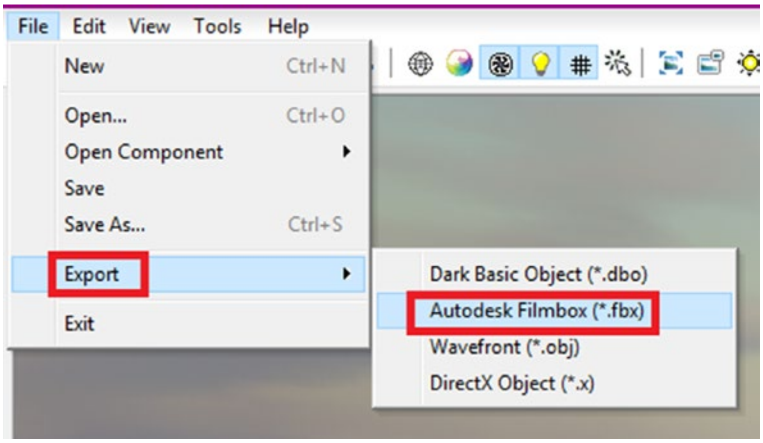


Figure 8-18. *Exporting the file*

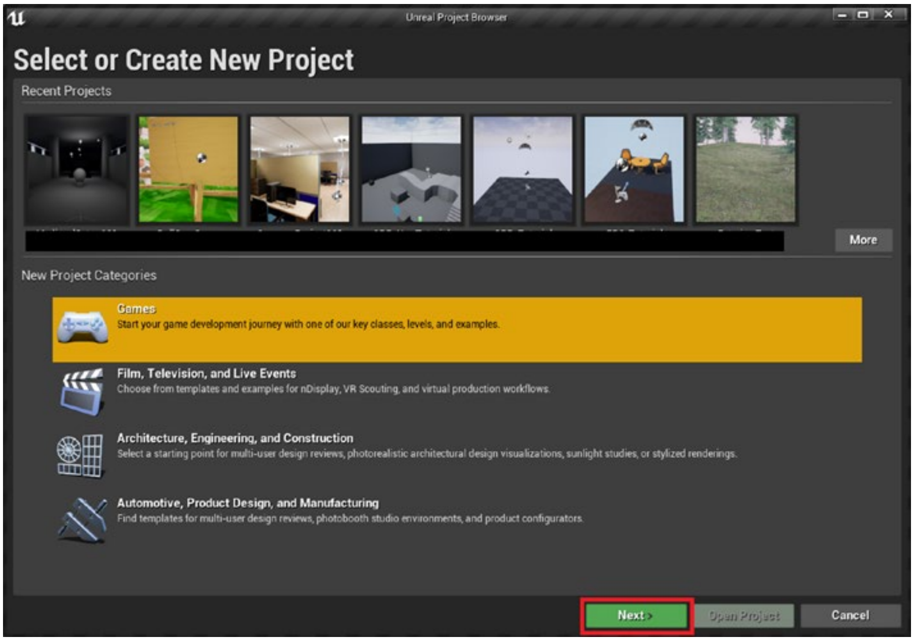
And that's all for this chapter. Make sure to check out the free foliage assets that are available at the Unreal Engine store and the templates on the TreeIt website in case you are not satisfied with your results. Making realistic assets from nature is difficult, because there is a lot of randomness present in nature, and that's very difficult to create manually. In the next chapter, you will see how to export the assets that you created in other tools for UE4.

CHAPTER 9

Working on Unreal Engine 4

This chapter covers the basics of using Unreal Engine 4. We will start by launching and setting up the engine. To access Unreal Engine 4, you need to have the Epic Games Launcher installed on your PC. You can get it from the official website of Epic Games. Download the launcher for Windows or Mac, whichever platform you are using. After the installation is done, launch the Epic Games Launcher and navigate to the Unreal Engine section. Go to the Library tab and download the engine version of your choice, preferably the most recent version. Unreal Engine is large, so it will take some time to download. Once the download is complete, launch the engine. The first window that will appear is shown in Figure 9-1.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_9



Select or Create New Project

[redacted]

New Project Categories

Games
Start your game development journey with one of our key classes, levels, and examples.

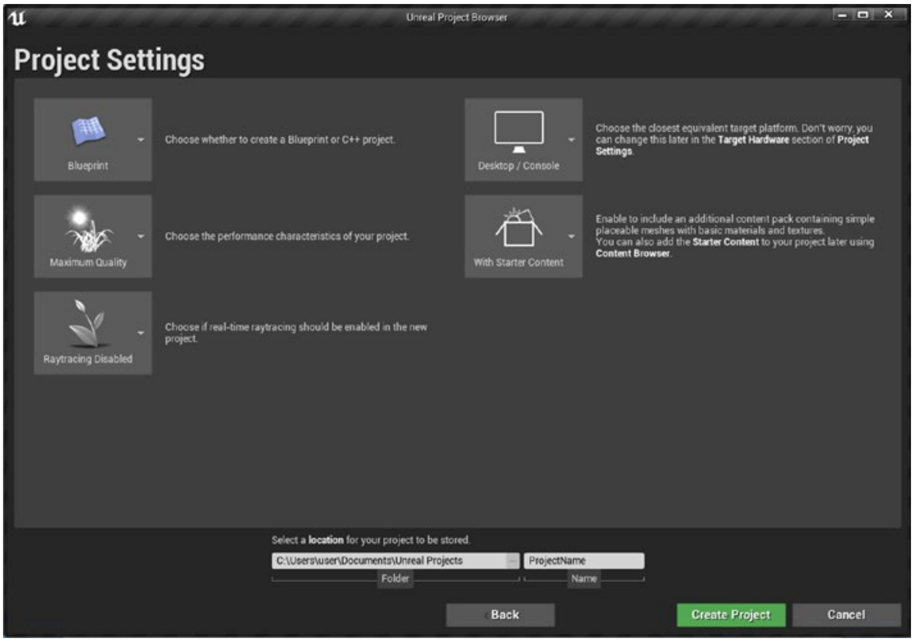
Film, Television, and Live Events
Choose from templates and examples for nDisplay, VR Scouting, and virtual production workflows.

Architecture, Engineering, and Construction
Select a starting point for multi-user design reviews, photorealistic architectural design visualizations, sunlight studies, or stylized renderings.

Automotive, Product Design, and Manufacturing
Find templates for multi-user design reviews, photobooth studio environments, and product configurators.

Figure 9-1. *Project browser window of Unreal Engine*

This is the project browser window, which allows you to open existing projects and create new ones. From here, select the Games category and click on Next. On the next screen, choose the First Person template and click on Next. In the final screen, make sure that the settings are as shown in Figure 9-2.



Unreal Project Browser

Project Settings

Blueprint — Choose whether to create a Blueprint or C++ project.

Maximum Quality — Choose the performance characteristics of your project.

Raytracing Disabled — Choose if real-time raytracing should be enabled in the new project.

Desktop / Console — Choose the closest equivalent target platform. Don't worry, you can change this later in the **Target Hardware** section of **Project Settings**.

With Starter Content — Enable to include an additional content pack containing simple placeable meshes with basic materials and textures. You can also add the **Starter Content** to your project later using **Content Browser**.

Select a **location** for your project to be stored.

Folder: C:\Users\user\Documents\Unreal Projects

Name: ProjectName

Figure 9-2. *Project settings*

Make sure to give your project a proper name and then click on Create Project. Unreal Engine will open up. Now let's get into the basics of the Unreal Engine 4 interface. Your default window should look similar to Figure 9-3.

- The Content Browser is at the bottom of the screen and it allows you to explore, manage, and import assets for your project.
- The World Outliner is on the top-right side of the screen. It shows you the list of all the assets that you have added to your scene.

- The Details panel is on the bottom-right side of your screen and it is basically the Properties Editor. It allows you to change various object and editor parameters.
- The top-left side of your screen is the Modes panel. It has various kinds of engine objects, lights, volumes, etc., that you need while developing your game.

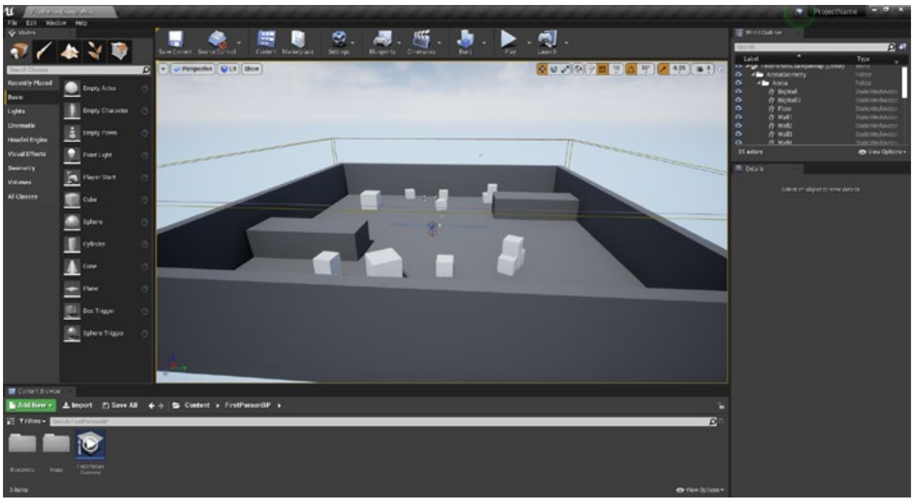


Figure 9-3. *Unreal Engine default layout*

Next, let's explore the basic navigation in Substance Painter. Press Alt+LMB (left mouse button) to rotate your view. Press Alt+MMB (middle mouse button) to pan the view. And finally, use Alt+RMB (right mouse button) to zoom your view. These are the basic controls, but the best way to navigate around Unreal Engine 4 is by right-clicking and pressing W, S, A, and D to move in the direction that you are looking at. Right-clicking and holding allows you to look around with your camera. This is best understood by using it practically inside the engine.

In order to test your game and see how everything will play out, you can click on the Play button (see Figure 9-4) or press the Alt+P hotkey.

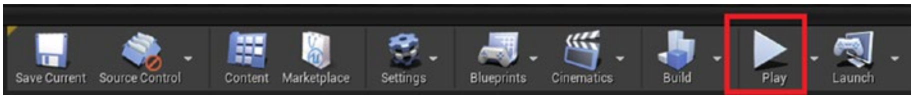


Figure 9-4. *The Play button for previewing your creation*

The following table lists some hotkeys that you need in order to use Unreal Engine 4 efficiently.

Hotkey	Function
W	Translate
E	Rotate
R	Scale
F	Focus selected
G	Game view
F11	Fullscreen viewport/Immersive mode
END	Snap to floor
Shift+Move	Move camera along with selected object
Alt+Move/Rotate/Scale	Duplicate object

The Material Graph has its own set of hotkeys, and we will discuss them as we get to it. As for now, try these out in the editor and familiarize yourself with the controls.

Creating Lightmap UVs

Let's switch back to Blender and prepare the assets to be exported into Unreal Engine 4. Open the scene that you have been working on and save a backup copy of it, as you are going to move things around a lot. When you export a scene to UE4, you need to center the objects. Otherwise, you'll have a lot of problems when you start moving them around in the engine.

First, you will create the Lightmap UVs for the objects. For that you need another UV map on top of the default one that is in the mesh. Choose the Object Data tab in the properties window, expand the UV Maps section, and then click on the + icon to create a new UV map (see Figure 9-5).

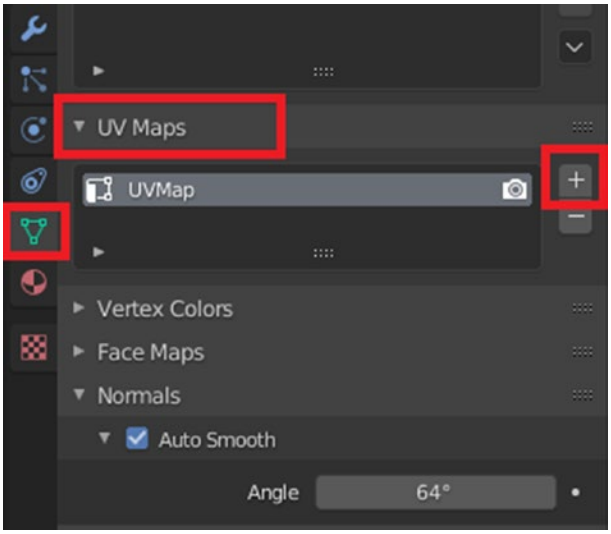


Figure 9-5. *Creating new UV maps*

Once a new UV map has been created, click on it to select it so that the changes to the UVs affect only the new UV map and don't ruin the original UV map. When you select this new UV map, the UV editing viewport will start displaying it so that you can edit it the way you want. One thing to note about Lightmap UVs here—they are not like standard UVs, where you have to keep them proportional to the 3D mesh. Lightmap UVs just store light data and all they require is a UV map where every face occupies "good enough" UV space. There's no need to be concerned about stretching or warping. Just make sure that every face of a mesh covers the maximum possible UV space.

For this, you are going to need an external add-on called UV-Squares. You can find it at the following link: `https://github.com/Radivarig/UvSquares`. If you can, support the developer by purchasing the premium version of the add-on.

Download it and then open Blender. Go to Edit ➤ Preferences ➤ Add-ons and click on the Install button (see Figure 9-6).

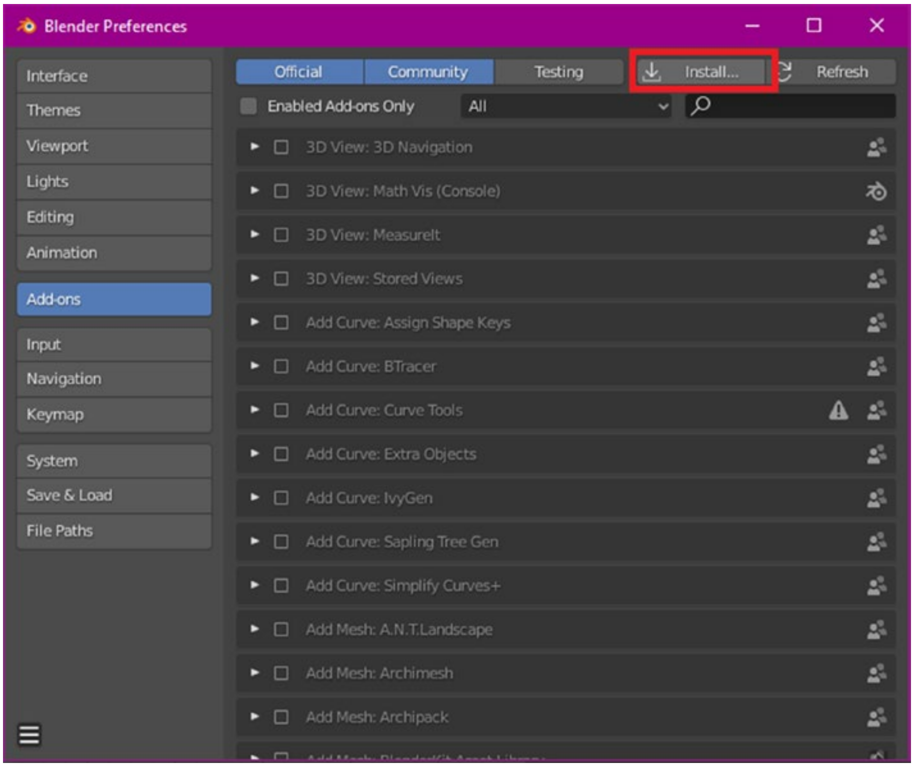


Figure 9-6. *The Add-Ons tab of Blender*

When you click on Install, the File Browser window will open. Navigate to where you have downloaded your add-on and click on the Install Add-on button toward the bottom of the window. The add-on will be installed within seconds. All you need to do is activate it by clicking on the check box on the left side of its name (see Figure 9-7).

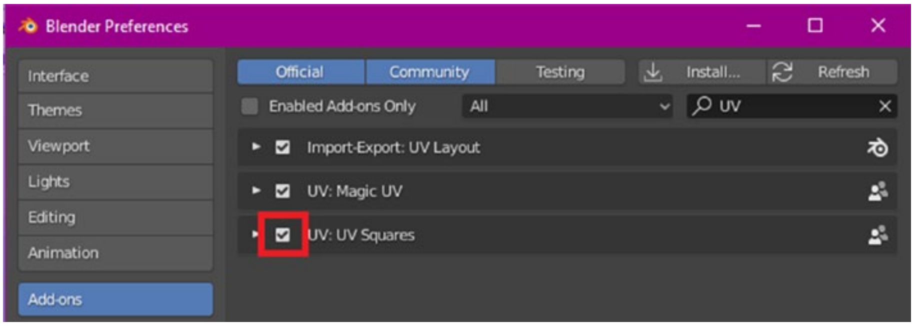


***Figure 9-7.** Enabling an add-on*

The add-on has been enabled and can be accessed in the UV editor when editing UVs. Let's now use the add-on to create Lightmap UVs. We will begin by selecting a UV island. Then press N to bring out the side menu. If the add-on has been installed correctly, you will see the UV Squares category there. Selecting it will reveal further details and functions (see Figure 9-8).

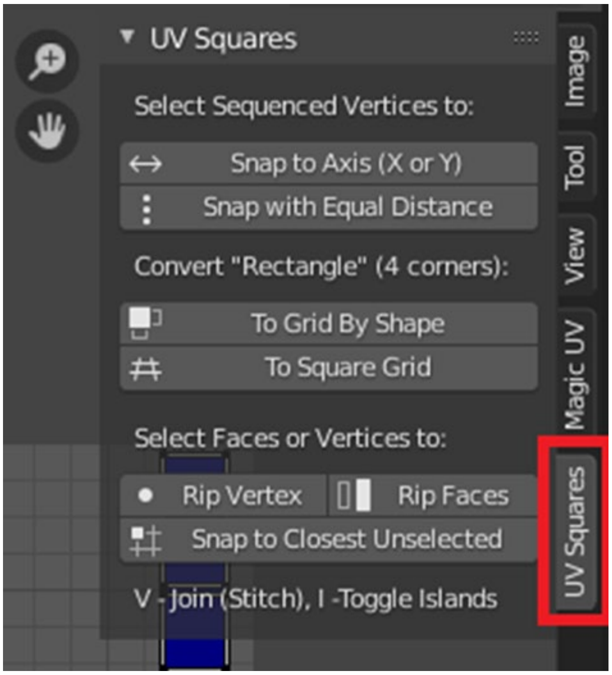

UV Squares
Select Sequenced Vertices to:
Snap to Axis (X or Y)
Snap with Equal Distance
Convert "Rectangle" (4 corners):
To Grid By Shape
To Square Grid
Select Faces or Vertices to:
Rip Vertex
Rip Faces
Snap to Closest Unselected
V - Join (Stitch), I -Toggle Islands
Image
Tool
View
Magic UV
UV Squares

***Figure 9-8.** UV Squares category*

Once you have selected a UV island, click on the To Square Grid option to convert it to an island whose faces are perfect squares (see Figure 9-9). The result might be large or small, so make sure to scale it to fit properly inside the UV space.

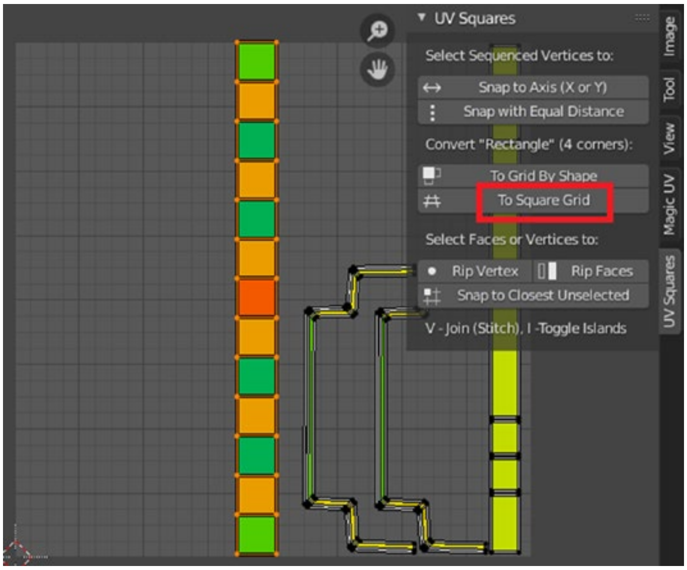

UV Squares
Select Sequenced Vertices to:
Snap to Axis (X or Y)
Snap with Equal Distance
Convert "Rectangle" (4 corners):
To Grid By Shape
To Square Grid
Select Faces or Vertices to:
Rip Vertex
Rip Faces
Snap to Closest Unselected
V - Join (Stitch), I -Toggle Islands
Image
Tool
View
Magic UV
UV Squares

Figure 9-9. *Converting UVs to a square grid*

Now do the same thing for the rest of the UVs and scale them roughly to the same size. Your results should be similar to Figure 9-10.

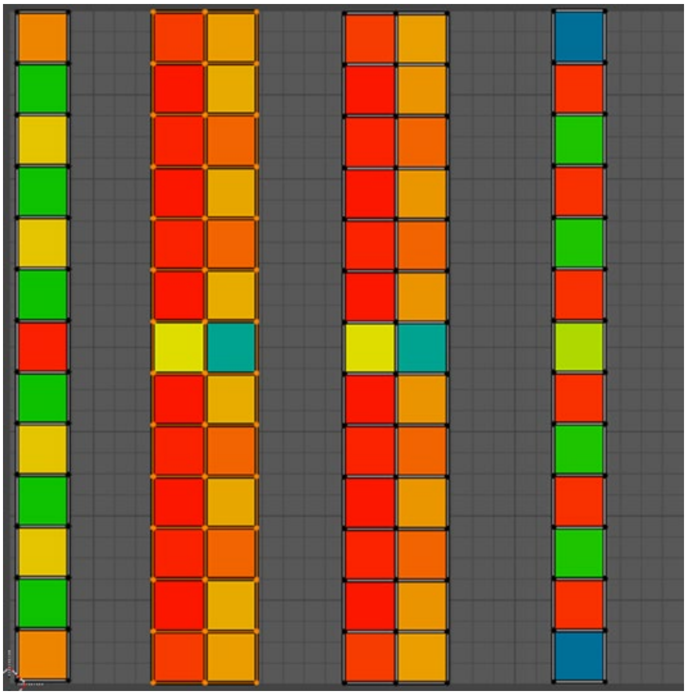


Figure 9-10. *Squared out UVs*

Next, we will scale these UV islands to stretch them out so that they occupy all of the UV space. No need to be precise; just scale them until all of the UV space is covered (see Figure 9-11).

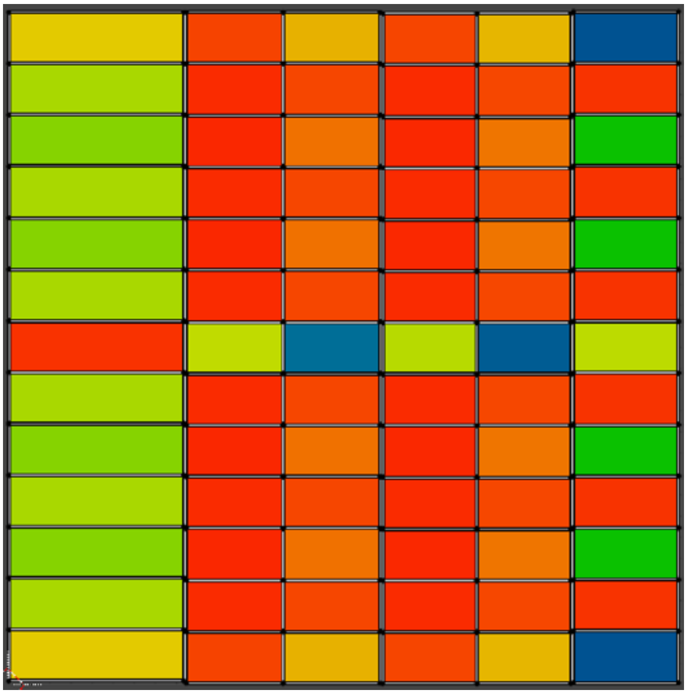


Figure 9-11. *Scaling and fitting UV islands into the UV space*

Do the same thing for the rest of your assets. By using the same UV map that you created by unwrapping, you can drastically reduce the Lightmap UV creation time.

Exporting Models from Blender

Next we will export the unwrapped models from Blender to UE4. You need to center the objects before you can export them to UE4. If you don't center the objects, the pivot of the object will go to the center of the world and the mesh will stay in its original position. This will make working with that mesh very, very difficult. Simply move your object to the center of the world, as shown in Figure 9-12.

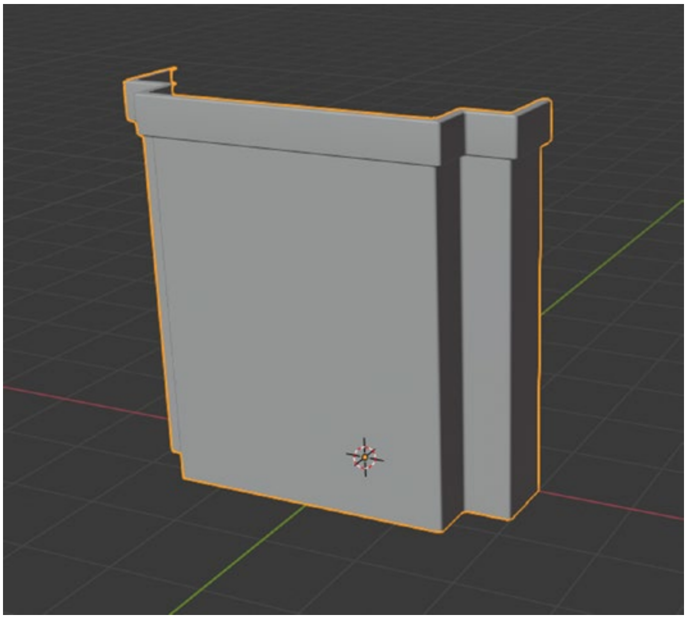
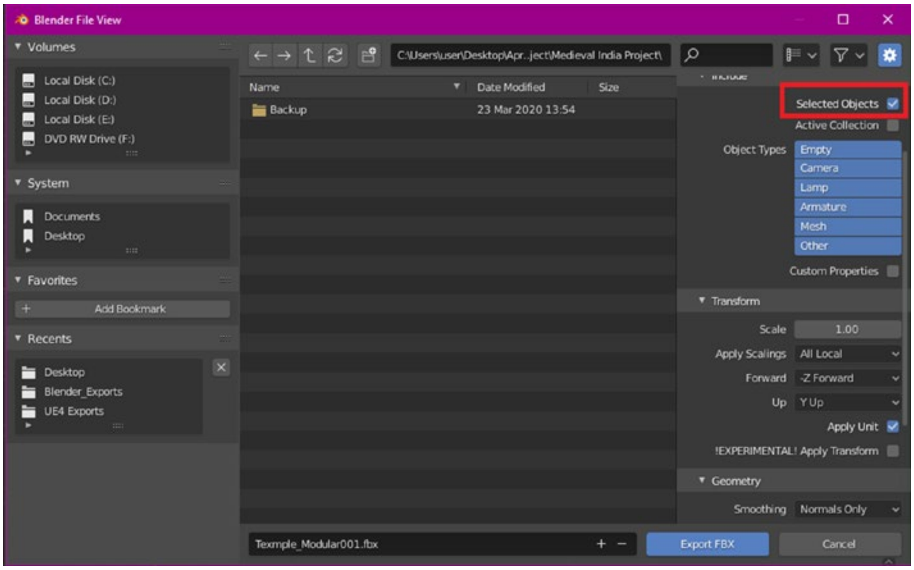


Figure 9-12. *Centering an object*

You need to do same thing for every object while exporting. Next go to File ➤ Export ➤ FBX to export your selected object as an FBX file. This will open the Exporter window, where you set the export parameters (see Figure 9-13).



C:\Users\user\Desktop\Apr...ject\Medieval India Project\

Backup 23 Mar 2020 13:54

Selected 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 ✓
!EXPERIMENTAL! Apply Transform

Geometry
Smoothing Normals Only

Texmple_Modular001.fbx

Figure 9-13. *Exporter window*

Check Selected Objects so that only the selected object is exported. Give the object a proper name. Then export it into a directory where it is easy to find.

Exporting Textures from Substance Painter

Finally let's export the textures from Substance Painter. Open the asset whose textures you want to export and press Ctrl+Shift+E to open the Exporter window. Or you can choose File ➤ Export Textures to do the same. Doing this will launch the Exporter window (see Figure 9-14).

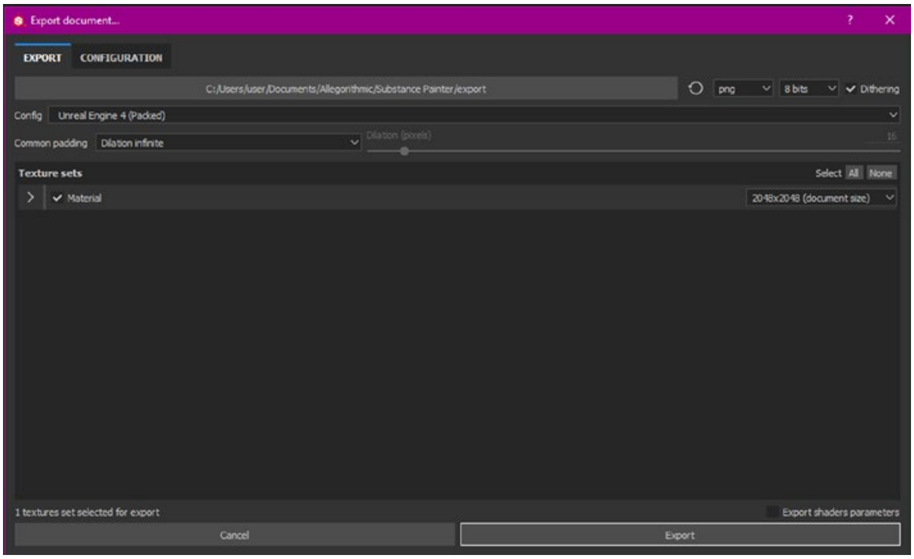

Export document...
EXPORT CONFIGURATION
C:/Users/user/Documents/Allegorithmic/Substance Painter/export png 8 bits ✓ Dithering
Config Unreal Engine 4 (Packed)
Common padding Dilation infinite Dilation (pixels) 16
Texture sets Select All None
✓ Material 2048x2048 (document size)
1 textures set selected for export
Export shaders parameters
Cancel Export

Figure 9-14. *Exporter window*

As you can see, all the templates should have already been set up for you by UE4. If not, click on the drop-down menu next to Config and choose Unreal Engine 4 (Packed) from the list (see Figure 9-15).

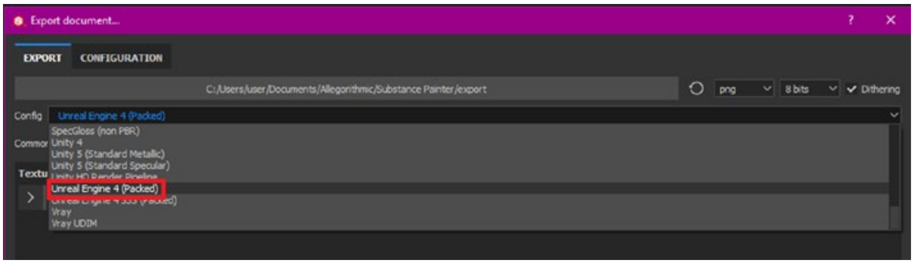

Export document...
EXPORT CONFIGURATION
C:/Users/user/Documents/Allegorithmic/Substance Painter/export png 8 bits ✓ Dithering
Config Unreal Engine 4 (Packed)
SpecGloss (non PBR)
Unity 4
Unity 5 (Standard Metallic)
Unity 5 (Standard Specular)
Unity HD Render Pipeline
Unreal Engine 4 (Packed)
Unreal Engine 4 SSS (Packed)
Vray
Vray UDIM

Figure 9-15. *Choosing an export template*

Change the image format to `bmp` and set the output resolution to 2048x2048. You can ramp up the export resolution to 4096x4096 for large assets that require that much detail. Otherwise, it is advisable to keep the export resolution low.

That's all about exporting assets for Unreal Engine. In the next chapter, we will learn how to import the exported assets and how to set up materials inside Unreal Engine 4.

CHAPTER 10

Importing into Unreal Engine 4

In this chapter, we learn how to import the assets that we exported from Blender and Substance Painter into Unreal Engine 4.

Import Settings

Importing into UE4 is very simple. There are two ways to do it. You can drag and drop your assets into the Content Browser of UE4 or click on the Import button in the Content Browser (see Figure 10-1).

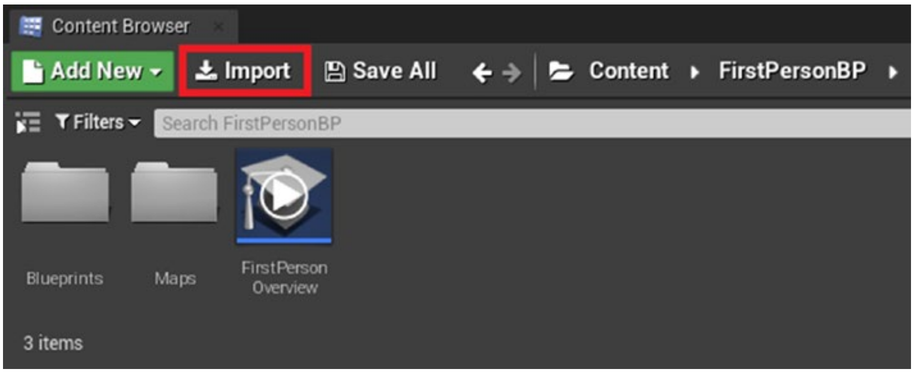


Figure 10-1. *Importing assets*

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_10

However, before we import anything, let's create some folders so that everything is organized. Right-click in the Content Browser to open the Create menu. There are lots of options in this menu and we will be using quite a lot of them for our project. For now, choose New Folder from the list, as shown in Figure 10-2.

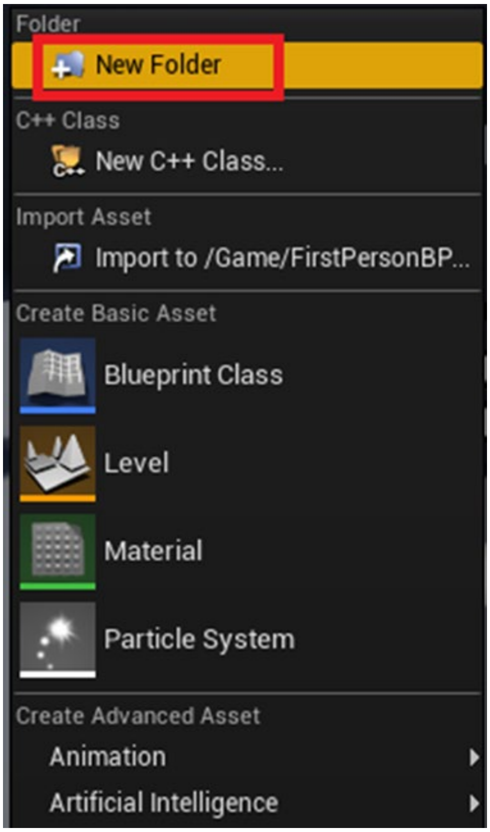
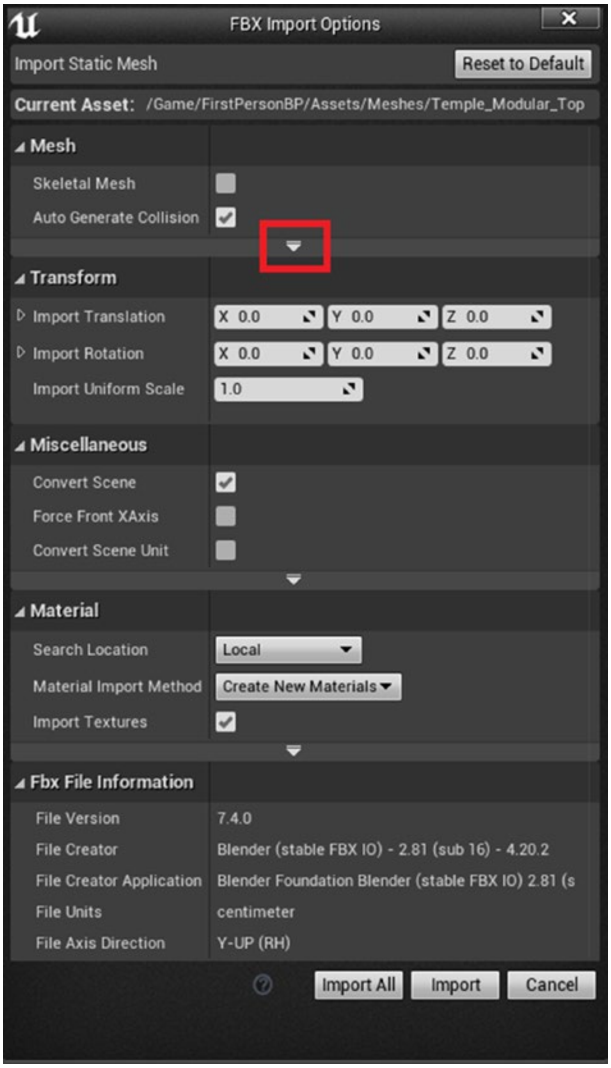


Figure 10-2. *Create menu in UE4*

Name this folder `Assets`. We will keep all our imported assets in this folder. Open the `Assets` folder and create two more folders called `Meshes` and `Materials`. Import meshes and textures inside the respective folders. This will organize your work and prevent things from getting messy.

Open the `Meshes` folder and click on Import to open the File Browser. Navigate to where you have been storing your exported FBX files from Blender and import one of them. A new window will open that will allow you to change the import parameters (see Figure 10-3).



Current Asset: /Game/FirstPersonBP/Assets/Meshes/Temple_Modular_Top

Mesh
- Skeletal Mesh: ☐
- Auto Generate Collision: ✓

Transform
- Import Translation: X 0.0, Y 0.0, Z 0.0
- Import Rotation: X 0.0, Y 0.0, Z 0.0
- Import Uniform Scale: 1.0

Miscellaneous
- Convert Scene: ✓
- Force Front XAxis: ☐
- Convert Scene Unit: ☐

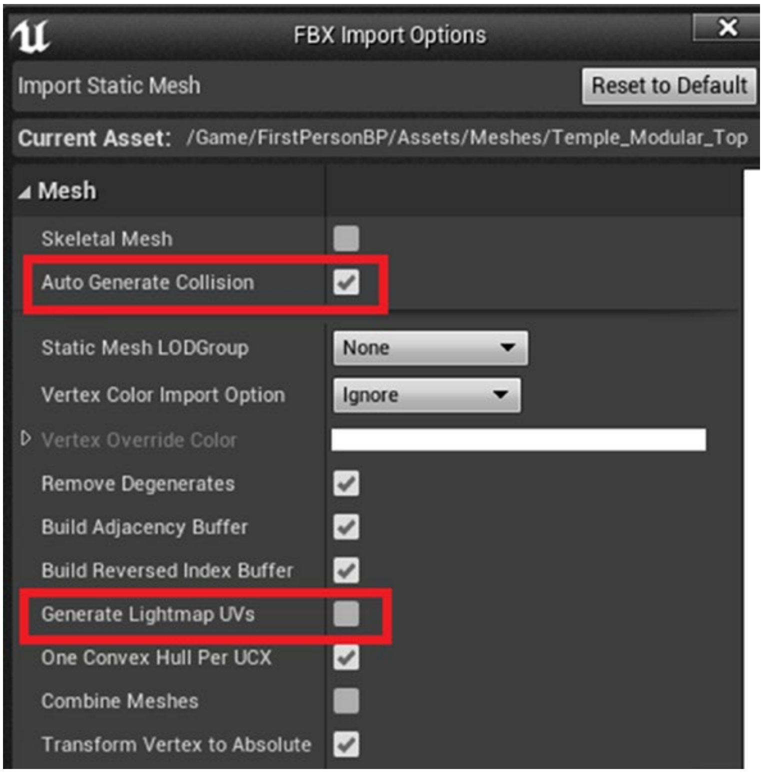
Material
- Search Location: Local
- Material Import Method: Create New Materials
- Import Textures: ✓

Fbx File Information
- File Version: 7.4.0
- File Creator: Blender (stable FBX IO) - 2.81 (sub 16) - 4.20.2
- File Creator Application: Blender Foundation Blender (stable FBX IO) 2.81 (s
- File Units: centimeter
- File Axis Direction: Y-UP (RH)

Figure 10-3. *Import options window*

You will see a ton of options that you can modify to personalize how you want your file to be imported. You need to click on the drop-down arrow under the Mesh category to expand the window and bring up a lot of new options. In this case, we are concerned with two main options (see Figure 10-4).

- **Auto Generate Collision:** Every static mesh that is a solid object needs to have a collision that tells UE4 that another solid object cannot pass through it. Generating collisions is actually pretty simple. All you have to do is create a simple 3D mesh that surrounds the object. Or you can check this option and let UE4 figure out the collision by itself, which we will do this time.
- **Generate Lightmap UVs:** Lightmap UVs are used for storing light and shadow data for any mesh. Since we created our own in Blender, we will uncheck this option.



FBX Import Options

Import Static Mesh — Reset to Default

Current Asset: /Game/FirstPersonBP/Assets/Meshes/Temple_Modular_Top

Mesh

Skeletal Mesh	☐
Auto Generate Collision	✓
Static Mesh LODGroup	None
Vertex Color Import Option	Ignore
Vertex Override Color	
Remove Degenerates	✓
Build Adjacency Buffer	✓
Build Reversed Index Buffer	✓
Generate Lightmap UVs	☐
One Convex Hull Per UCX	✓
Combine Meshes	☐
Transform Vertex to Absolute	✓

Figure 10-4. *Modifying the import parameters*

Now click on Import. This will be the setting for all the meshes that you import. If you haven't generated Lightmap UVs for any of your meshes, you can enable the Generate Lightmap UVs option. UE4 is generally not too good at generating Lightmap UVs by itself, so it is recommended that you do it yourself. Sometimes it does a serviceable job, so try it when you want decent results with less effort.

If your asset creates unwanted material files automatically, simply Force Delete them. We will create materials on our own.

As for importing textures, UE4 does not create much fuss. Simply drag and drop your exported textures inside the `Materials` folder and create more subfolders to better organize everything. Textures will be imported without any parameter windows, so just ignore any warnings that appear.

Note Always remember to save this file with the correct naming and in the right folder so it is easier to identify in later stages.

Exploring the Properties Editor

Any asset that you import has many properties that you can edit. This is done via the Properties Editor. You can access this by double-clicking on your asset. The Properties Editor window will look similar to Figure 10-5.

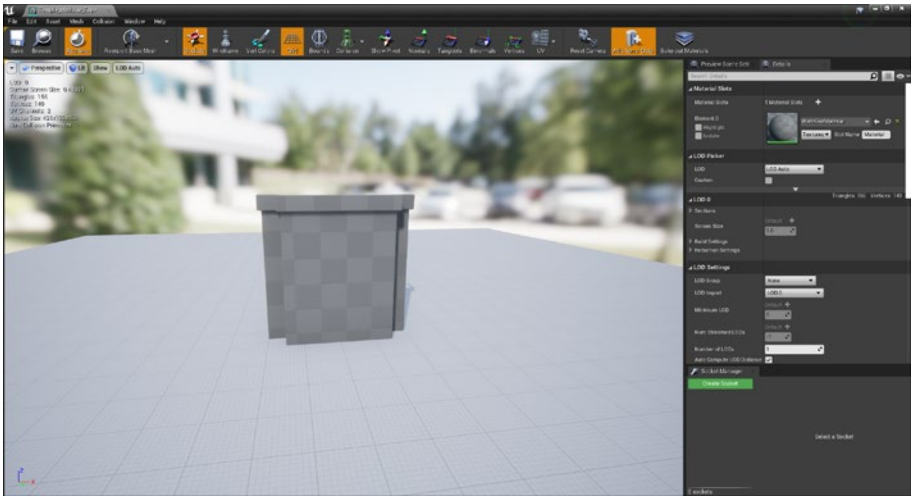


Figure 10-5. *Properties Editor*

Some of the important properties that we will work with include the following:

- **Material Slots:** In this category (see Figure 10-6), you can assign a material (which we will create in the next chapter) to your mesh. Then, whenever you create a new instance of your mesh, this material will be applied to it as well as to every old instance that was created previously.

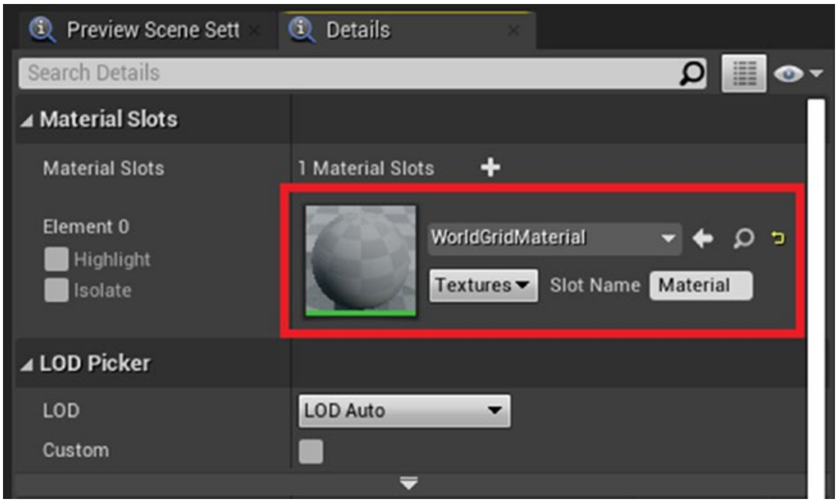
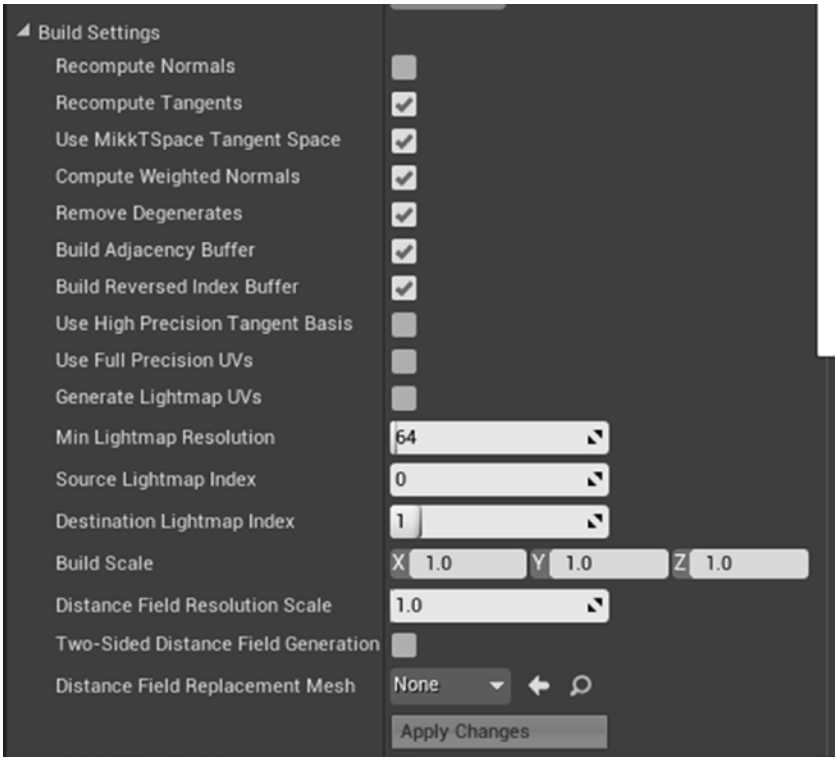


Figure 10-6. *Material slot of mesh properties*

- **Build Settings:** This allows you to build and rebuild various kinds of mesh-related options like recomputed tangents, to build light maps in case you don't have them, to designate an index of a Lightmap UV, and so on (see Figure 10-7). These options will make more sense once you start working with them.



Build Settings

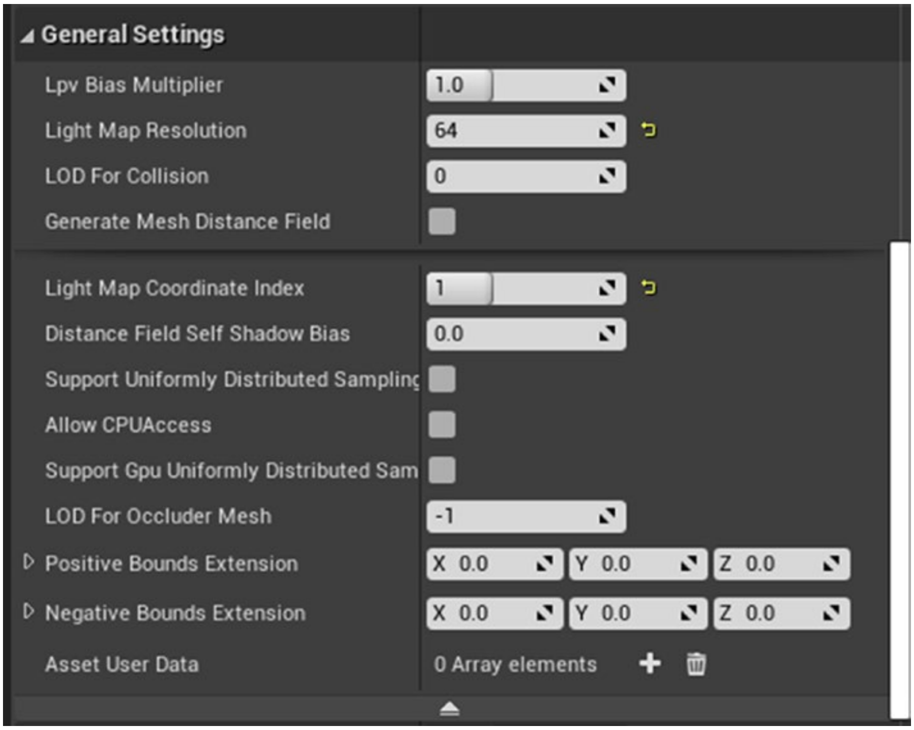
Recompute Normals	☐
Recompute Tangents	✓
Use MikkTSpace Tangent Space	✓
Compute Weighted Normals	✓
Remove Degenerates	✓
Build Adjacency Buffer	✓
Build Reversed Index Buffer	✓
Use High Precision Tangent Basis	☐
Use Full Precision UVs	☐
Generate Lightmap UVs	☐
Min Lightmap Resolution	64
Source Lightmap Index	0
Destination Lightmap Index	1
Build Scale	X 1.0 Y 1.0 Z 1.0
Distance Field Resolution Scale	1.0
Two-Sided Distance Field Generation	☐
Distance Field Replacement Mesh	None

Figure 10-7. *Build Settings*

- **General Settings:** These are other mesh-related settings that you will use (see Figure 10-8). Some of them are quality and optimization related and are important to understand because, when creating a game environment, you must maintain a balance between good quality and performance optimization. Some of these settings that you should know about include:
    - **Lightmap Resolution:** A higher lightmap resolution will produce a higher quality shadow bake and will eliminate any artifacts or errors occurring during light bakes on the mesh. Increasing this too much will increase the load on the system and

performance will decrease. In this case, you can increase it to something like 256. If problems persist then you may increase it to as high as 512 or 1024. But beyond that is not recommended.

- **Generate Mesh Distance Field:** This is an important setting for open world games. This generates a proxy mesh for objects used in light-related calculations and for generating shadows. Using this will optimize your game and reduce performance impacts of large scenes with lots of large objects. This may not be useful in this case, but nonetheless you should know about it.



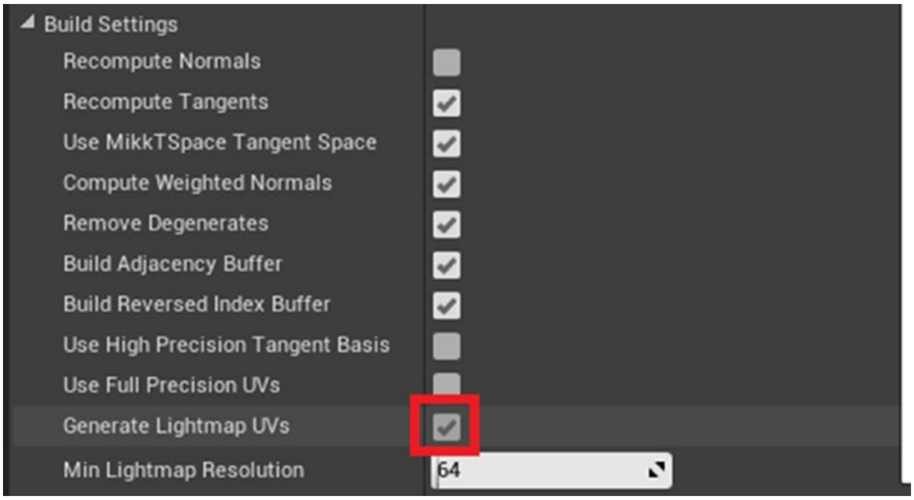
General Settings	
Lpv Bias Multiplier	1.0
Light Map Resolution	64
LOD For Collision	0
Generate Mesh Distance Field	☐
Light Map Coordinate Index	1
Distance Field Self Shadow Bias	0.0
Support Uniformly Distributed Samplinç	☐
Allow CPUAccess	☐
Support Gpu Uniformly Distributed Sam	☐
LOD For Occluder Mesh	-1
Positive Bounds Extension	X 0.0 Y 0.0 Z 0.0
Negative Bounds Extension	X 0.0 Y 0.0 Z 0.0
Asset User Data	0 Array elements

Figure 10-8. *General Settings*

Build Settings in Detail

Let's look at some build settings in detail and see how you can use them to bake Lightmap UVs.

Let's say you are not happy with the Lightmap UVs that you have generated. Instead of going back to Blender to create them again, you can have UE4 do it for you. You can do this very easily by clicking on the Generate Lightmap UVs checkbox to activate it (see Figure 10-9).



Build Settings

Recompute Normals	☐
Recompute Tangents	✓
Use MikkTSpace Tangent Space	✓
Compute Weighted Normals	✓
Remove Degenerates	✓
Build Adjacency Buffer	✓
Build Reversed Index Buffer	✓
Use High Precision Tangent Basis	☐
Use Full Precision UVs	☐
Generate Lightmap UVs	✓
Min Lightmap Resolution	64

Figure 10-9. *Generating Lightmap UVs*

Then set the Source Lightmap Index to be the index of the UV map that you want to be the reference for the Lightmap UV that UE4 generates. The value starts at `0`, with `0` being your UV map for the textures. If you want this to be your reference, then enter `0` in Source Lightmap Index field.

If you want the Lightmap UV that you created to be the source, then enter `1`, because ideally that should be its index.

Finally, you need to enter a Destination Lightmap Index value, which is the UV index that will store the newly created Lightmap UVs that UE4 will generate. You can either store this in one of the UV indexes that you have in the mesh or create a new one. To see the list of available UVs on your mesh, click on the UVs tab on the toolbar (see Figure 10-10).

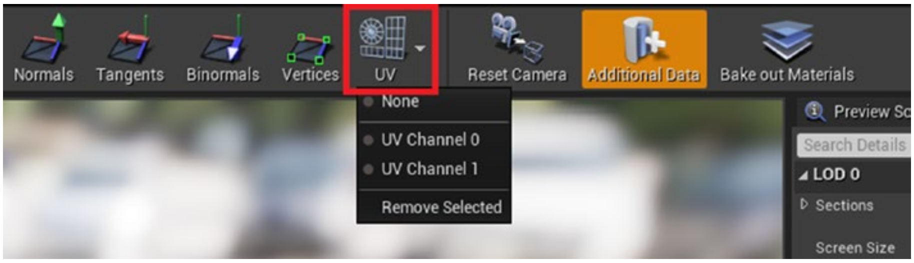
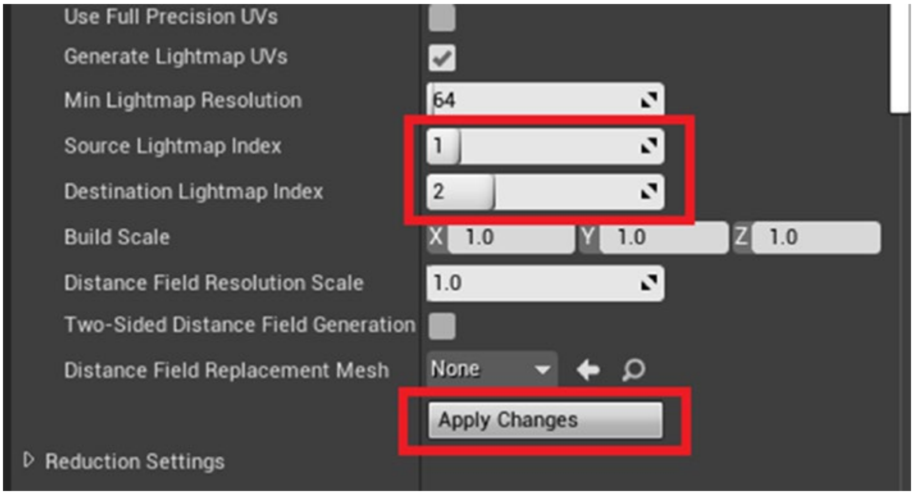


Figure 10-10. *List of available UVs*

Since we have only two in this case, we enter 2 in the Destination Lightmap Index to create a new UV index and store the generated lightmaps in that. Your settings should look similar to Figure 10-11. Once you're done, click on Apply Changes.

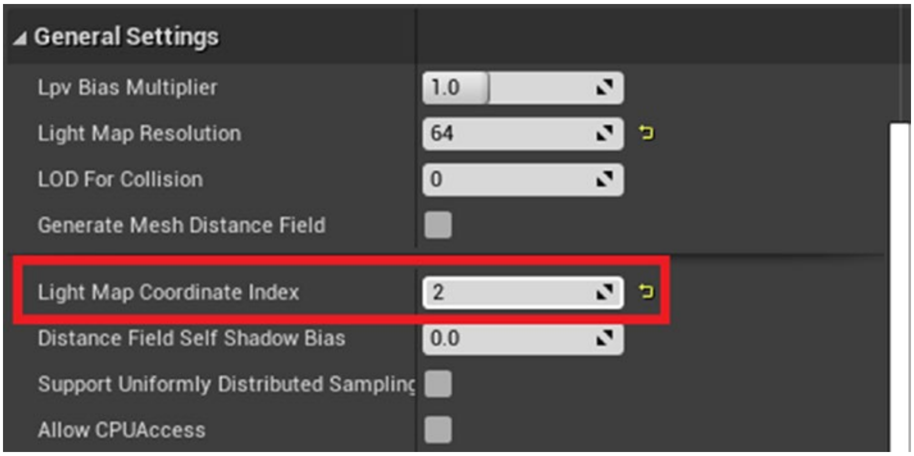


Use Full Precision UVs ☐
Generate Lightmap UVs ✓
Min Lightmap Resolution 64
Source Lightmap Index 1
Destination Lightmap Index 2
Build Scale X 1.0 Y 1.0 Z 1.0
Distance Field Resolution Scale 1.0
Two-Sided Distance Field Generation ☐
Distance Field Replacement Mesh None

Figure 10-11. *Generating new Lightmap UVs*

Once you click on Apply, you will see a new UV in the UV tab. If you are happy with the result then continue; otherwise, you can regenerate new UVs with different settings.

Now you need to set this newly generated Lightmap UV as the one used by the engine to store light and shadow information. To do that, scroll down to General Settings. Look for the Lightmap Coordinate Index option. Enter the index of the UV that you want to use as the Lightmap UV. In this case, it should be 2 (see Figure 10-12).



General Settings

Lpv Bias Multiplier	1.0
Light Map Resolution	64
LOD For Collision	0
Generate Mesh Distance Field	☐
Light Map Coordinate Index	2
Distance Field Self Shadow Bias	0.0
Support Uniformly Distributed Samplinc	☐
Allow CPUAccess	☐

Figure 10-12. *Setting the Lightmap coordinate index*

Editing Collisions

Let's see how we can edit the collision of our mesh. First click on the Collision tab on the toolbar (see Figure 10-13).

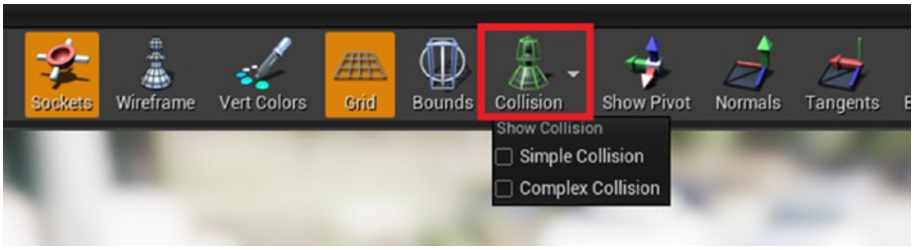


Figure 10-13. *Viewing the collision*

Check each of the box individually to visualize what your collisions look like. The Simple Collision option is simply a bounding box around the shape. It does not have much detail. While the Complex Collision option is generated according to the shape of the object, and it will follow the object's shape accurately. Simple Collision is not as performance heavy but is less accurate, while Complex Collision is obviously more accurate but will hamper performance if it's used too much in a scene.

By default, the collision is set to Simple but if you want to change it, scroll down to the Collisions category and click on the drop-down menu next to Collision Complexity. From the list, choose Use Complex Collision as Simple (see Figure 10-14).

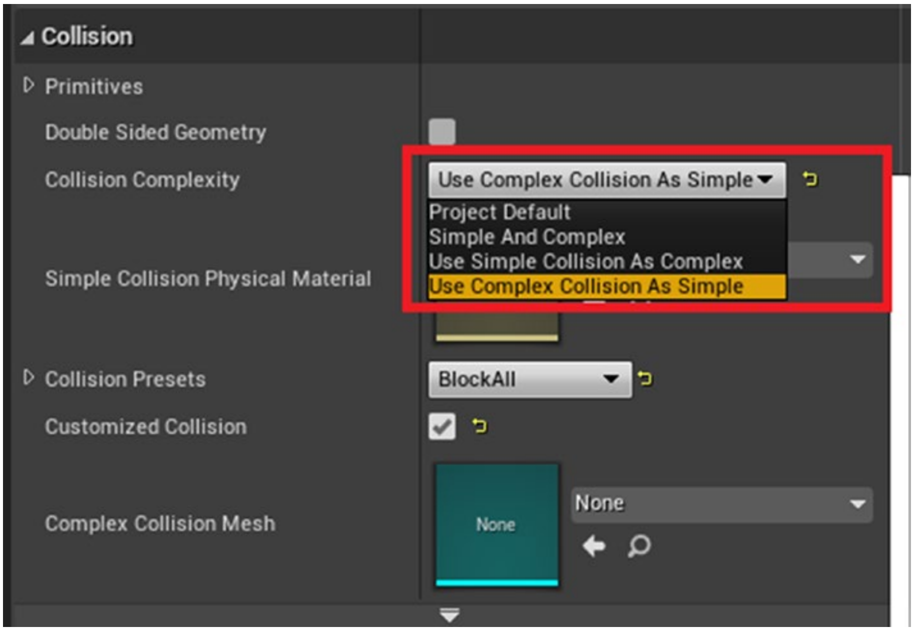


Figure 10-14. *Changing the collision type*

This will make the mesh use the Complex Collision option as its default. Everything has its uses and for objects like this large wall, it is better to use the Simple Collision option. But you will want to use Complex Collision option for walls with holes or doors so that you can traverse through the gap.

That's all for this chapter. I hope you have adequately learned how to import and set up your assets inside Unreal Engine 4. In the next chapter, you will learn about different ways to set up your materials.

CHAPTER 11

Material Setup in Unreal Engine 4

In Chapter 10, we imported the textures from Substance Painter into UE4. In this chapter, we will set up materials to use them.

Simple Material Setup

We will start with a very simple and basic material setup. Let's look at our materials first and learn what each of the maps means. You will have three main maps, each one named appropriately by the Substance Painter (see Figure 11-1).

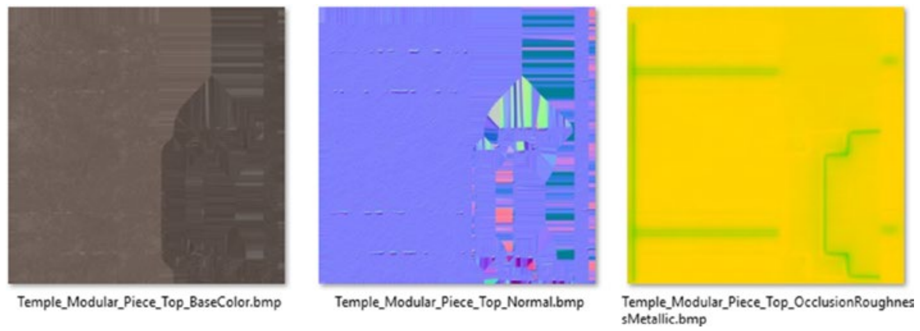

Temple_Modular_Piece_Top_BaseColor.bmp
Temple_Modular_Piece_Top_Normal.bmp
Temple_Modular_Piece_Top_OcclusionRoughnessMetallic.bmp

Figure 11-1. *Exported texture maps*

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_11

The three maps are Base Color, Normal, and OcclusionRoughnessMetallic. The third map is also called Mixed RMA, where RMA means Roughness, Metallic, and Ambient Occlusion. This map is created by storing Roughness, Metallic, and AO into the Green, Blue, and Red channels, respectively. We will be using the Material Editor to create the material.

Before you start creating your material, make sure to delete any residual material created automatically while importing the files. After that, go to the Assets ➤ Materials folder and create another folder for the asset whose material you are making. In this case, I name it `Temple_Modular_Top`. Since it is already inside the Materials folder, it is understood that this folder will contain files related to it. Drag and drop all the texture maps belonging to Temple_Modular_Top into this folder.

Now right-click to open the Create menu and choose Material from the list (see Figure 11-2). This will create a new material file in this location.

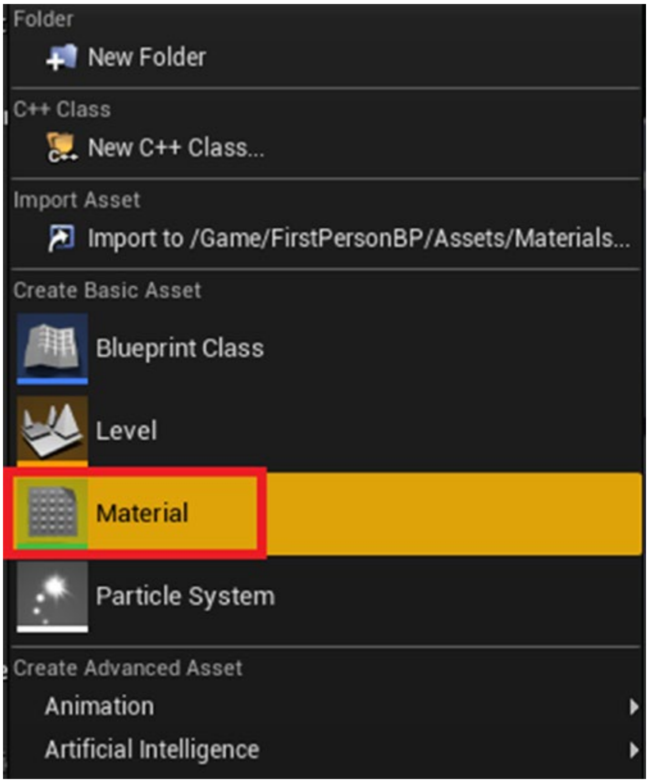


Figure 11-2. *Creating a new material file*

Rename this newly created material something appropriate, like M_Temple_Top, where "M" stands for "Master." This is our master material, which will store the material-related data. From this, we will create a material instance that allows us to make changes on the fly and apply them to our asset.

Double-click on the material to open the Material Editor window, which allows you to create and edit materials. Your Material Editor window should be similar to Figure 11-3.

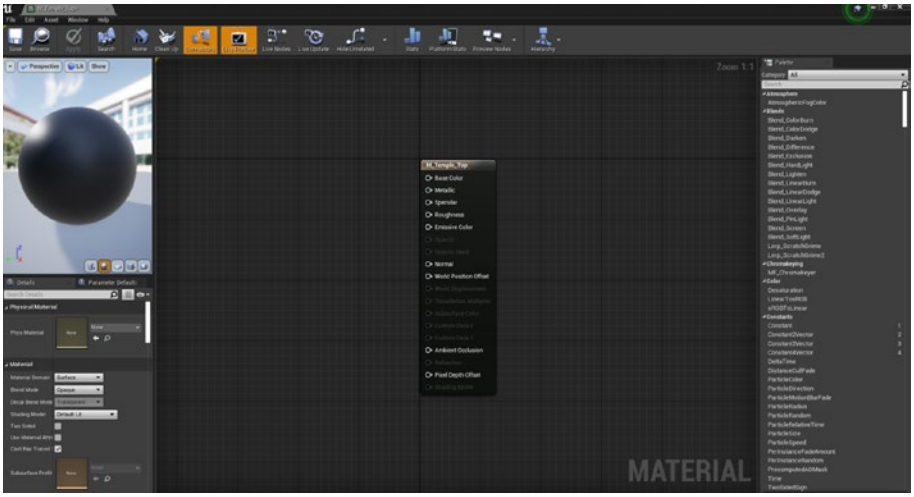


Figure 11-3. *The Material Editor window*

Add the texture files to the Material Editor window by dragging and dropping them from the Content Browser to the empty area of the window. They will appear on the Material Editor as node tiles; arrange them as shown in Figure 11-4.

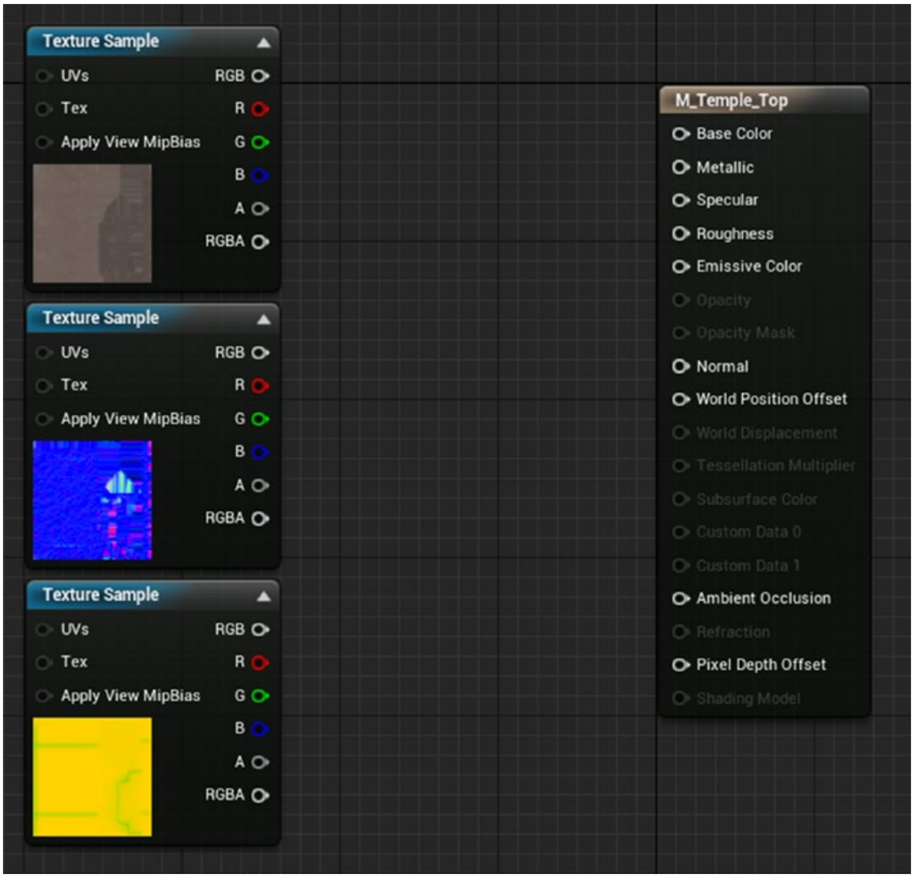


Figure 11-4. *Arranging the texture inputs*

The Material Editor in Unreal Engine 4 is node based, meaning you must connect the output pin of one node to the input of another. By doing this and adding lots of other nodes, you can create a very complex material right inside the UE4.

Note Node-based workflow is a very efficient and commonly used workflow in software programs now a days. It's very easy to understand and gives artists freedom to add many connections while being very intuitive. All that you need to do is drag a connection from an Output pin of a node and release it on an Input pin of the node to which you want to connect.

We are starting with the basic material setup, which is very simple and will get the job done in most cases, but only up to a certain extent. The basic setup lacks any kind of material customization. But since we have already finished the texturing in Substance Painter and are happy with result, we can go ahead with this method if we want to save some time.

With all that in mind, let's begin by creating the first connection. Click on the RGB pin of the Texture Sample node containing the Base Color and connect the "wire" to the Base Color of M_Temple_Top, as shown in Figure 11-5.

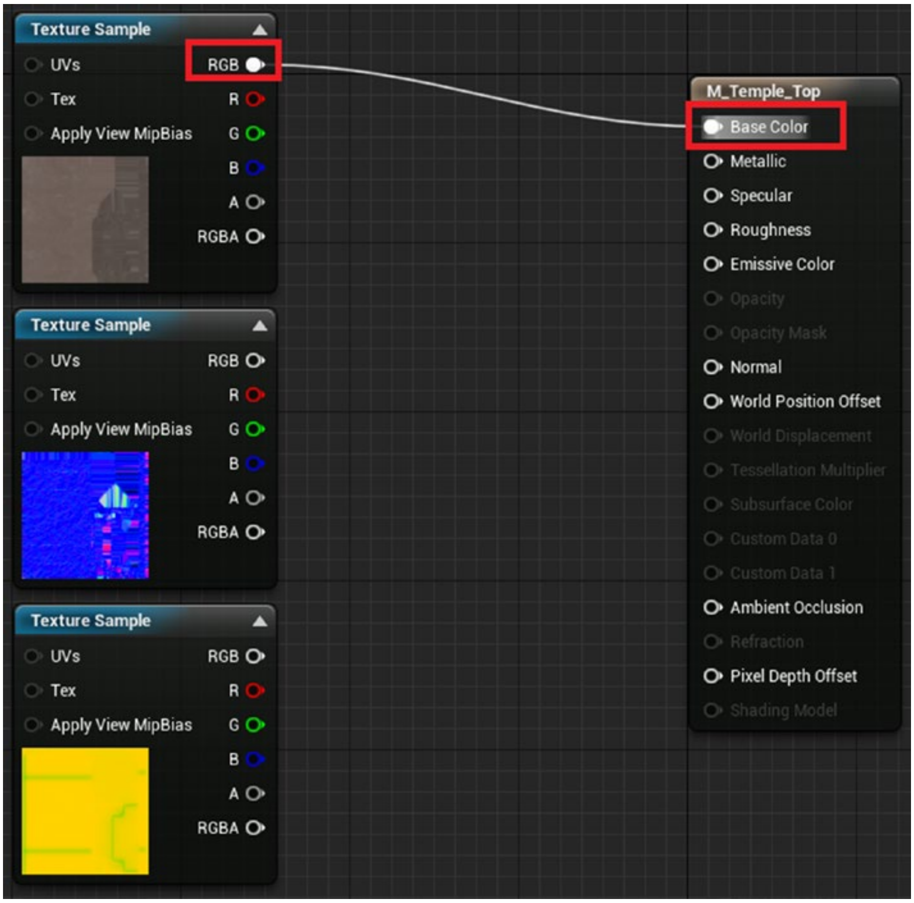


Figure 11-5. *Connecting the RGB output to the Base Color*

Next, connect the RGB pin of the Texture Sample containing the Normal map to the Normal pin of M_Temple_Top (we refer to this as the Material node from here on), as shown in Figure 11-6.

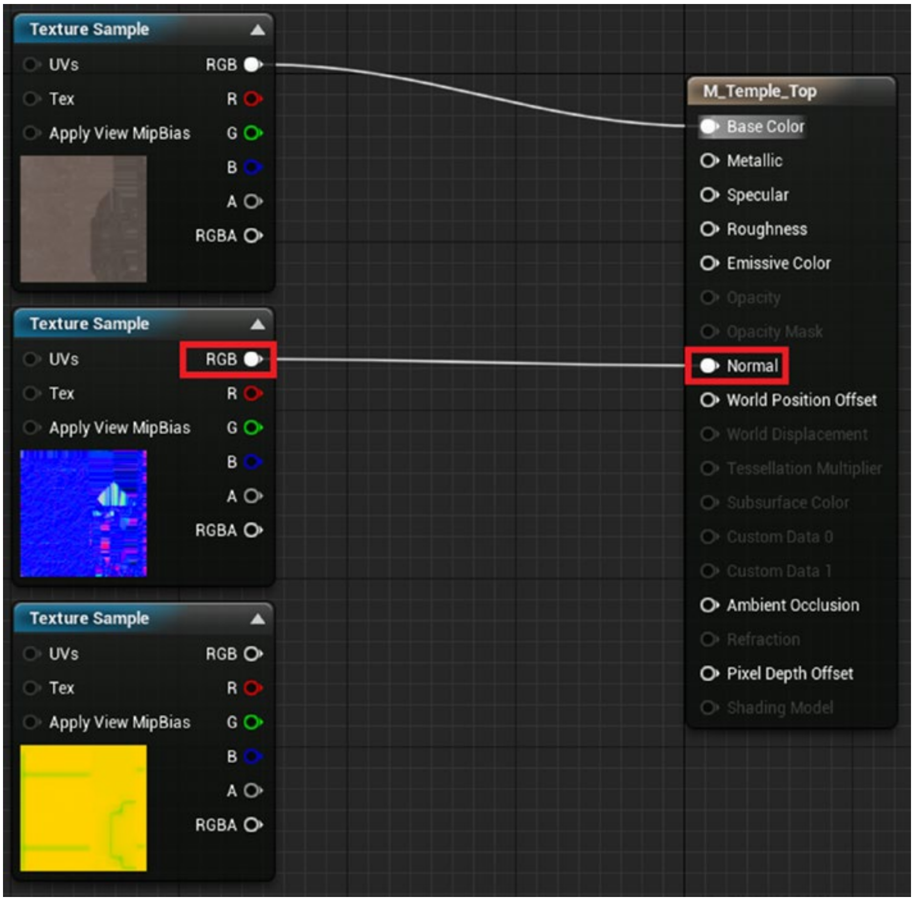


Figure 11-6. *Connecting Normal map pins*

For these two maps, we need the data stored in all three channels of the texture map. But for the next map, which is the MixedRMA or OcclusionRoughnessMetallic map, we will need the data stored in the individual texture channels (the Red, Green, and Blue channels). So for this one, the connection will go as follows:

Connect the Red or “R” pin to the Ambient Occlusion pin of the Material node, as shown in Figure 11-7.

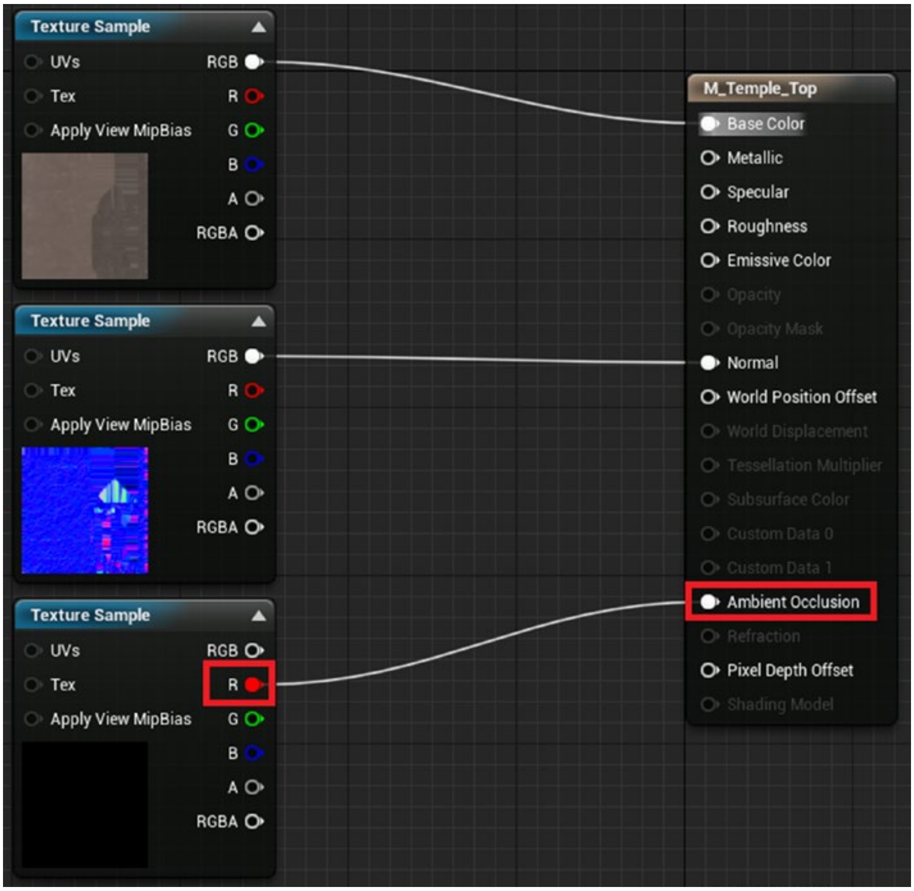


Figure 11-7. *Connection for the AO*

Connect the Green or "G" pin to the Roughness pin of the Material node. See Figure 11-8.

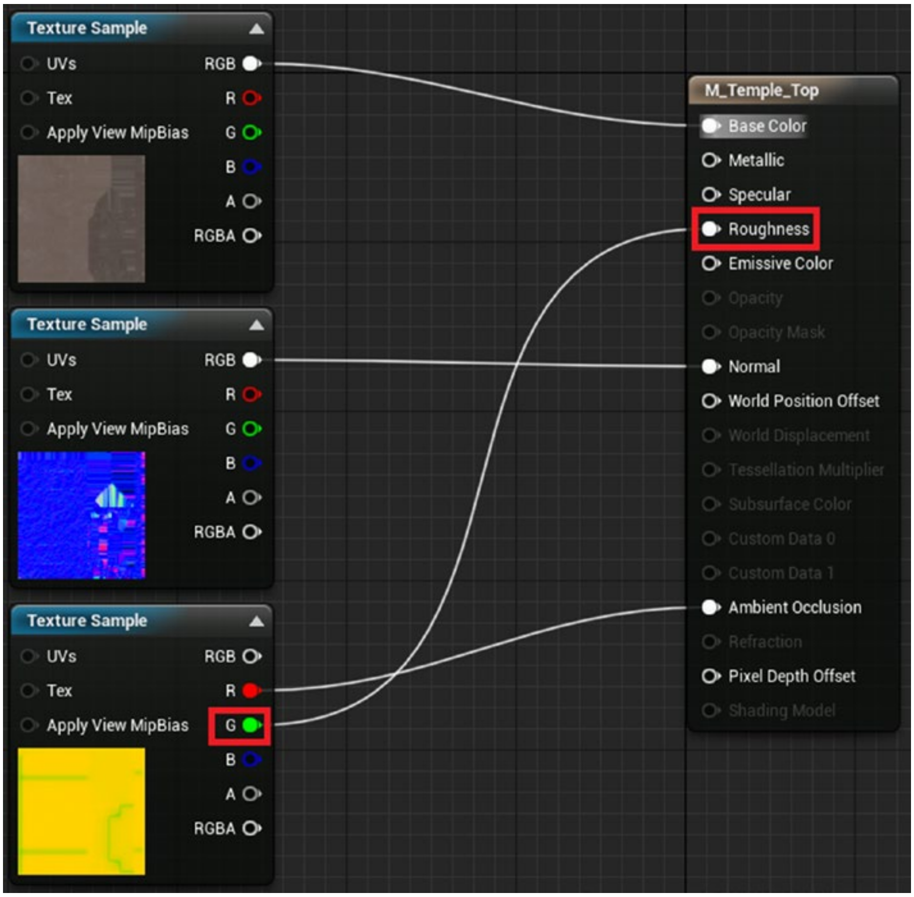


Figure 11-8. *Connecting pins to the Roughness map*

Finally, connect the Blue or "B" pin of the MixRMA node to the Metallic pin of the Material node, as shown in Figure 11-9.

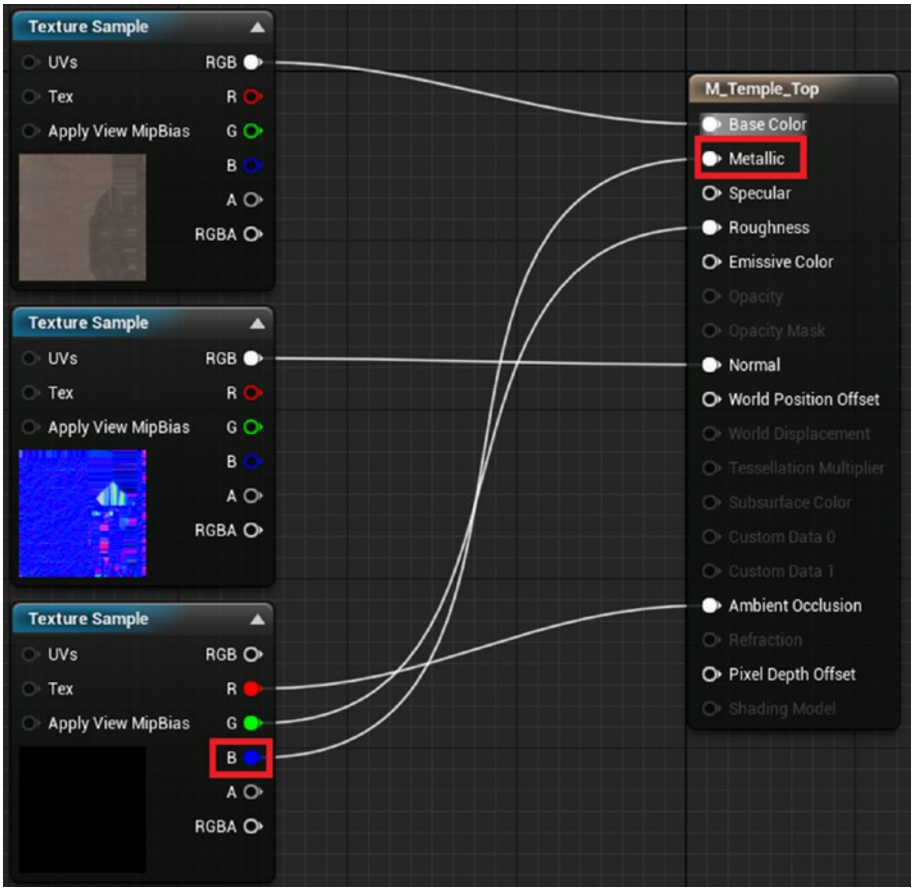


Figure 11-9. *Connection for the metallic map*

This is the final setup for our simple material. You can now apply this to the mesh and see how it looks. Click on the Save button in the top-left corner of the screen to save what you have created so far (see Figure 11-10).

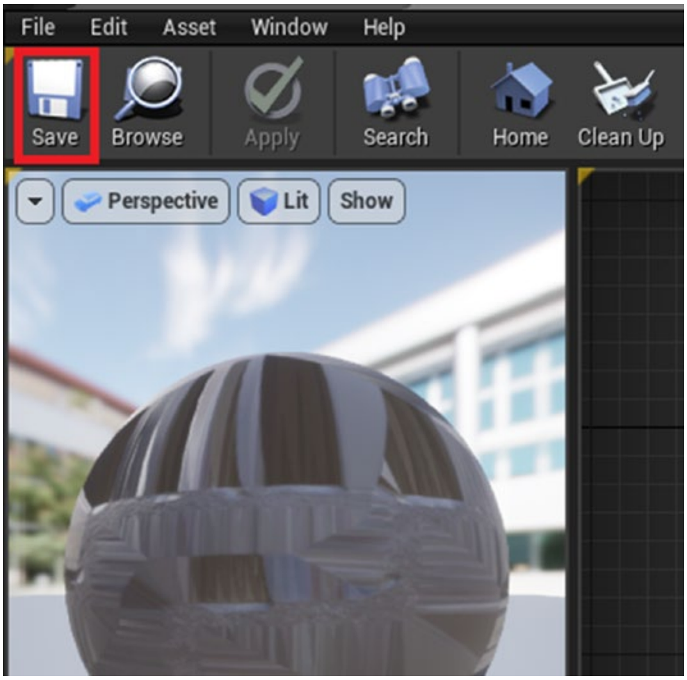


Figure 11-10. *Saving the material*

You'll want this material to be attached to the Temple_Modular_Top piece so that every time you create an instance of it, you'll already have this texture applied. To do that, open the Object Properties window by double-clicking on the mesh. Click on the Material drop-down list (see Figure 11-11).

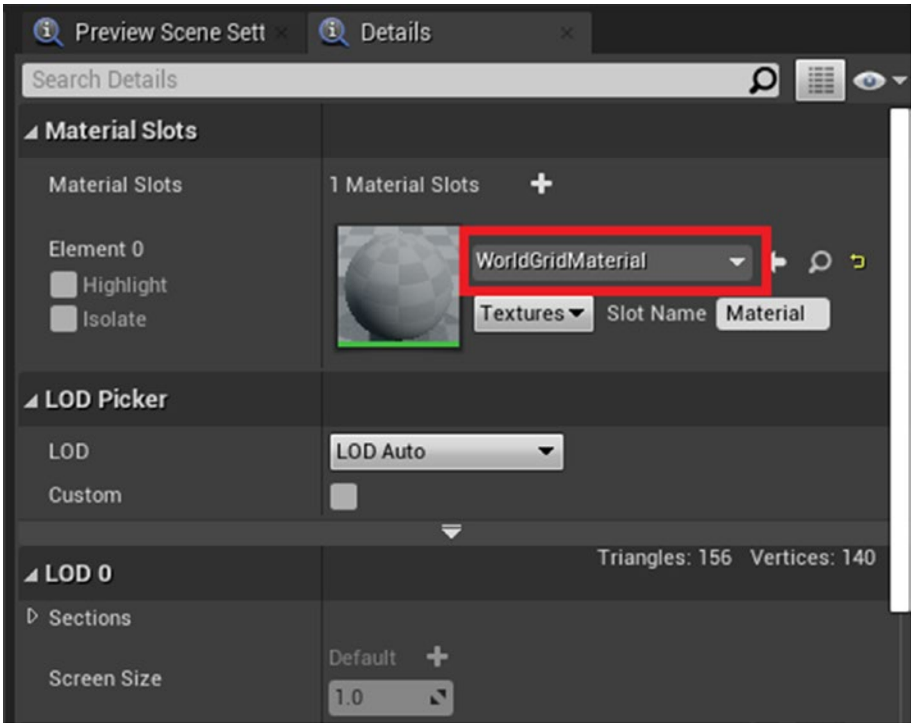


Figure 11-11. Material drop-down list

In the list, use the search bar to search for the material you just created by typing its name. When you see it in the list, click on it to apply it to the mesh (see Figure 11-12).

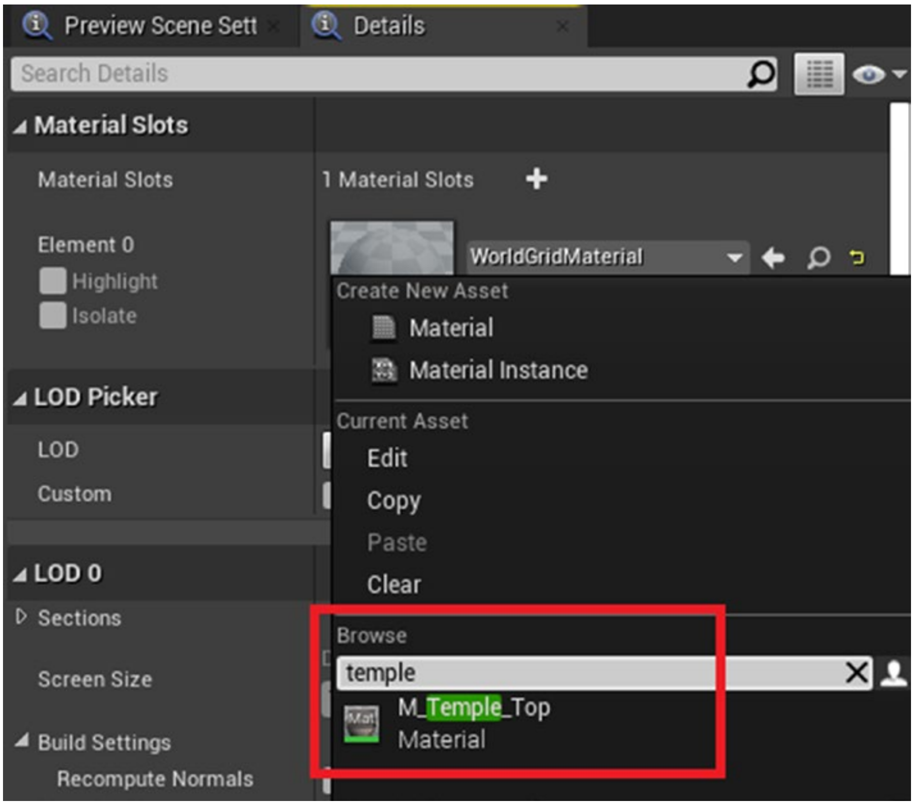


Figure 11-12. *Searching and applying the new material*

Once you're done, you will see that the mesh will be textured and will look like what you created inside the Substance Painter. Your results should be similar to Figure 11-13.

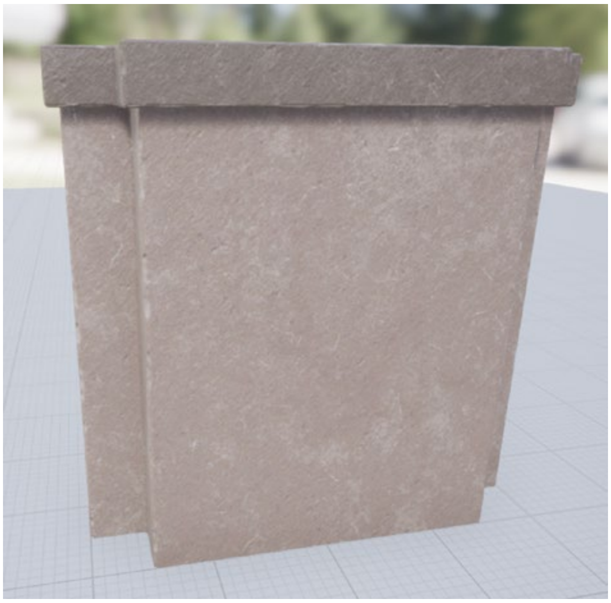


Figure 11-13. *Result so far*

This was the simple material setup for our textures. You can now drop a few pieces into the scene and test them.

Complex Material Setup

Let's now create a complex material setup, which will allow you to customize the material inside Unreal Engine 4. This section also explains the use of Material Instances. Open the Material Editor by double-clicking on M_Temple_Top. You are going to edit this material and add some nodes that will allow you to modify it.

First, we will break some connections. To break the connection, simply hold the Alt key and left-click on the connection pins. You can break all the connections or just the Base Color to start with. Either way, after doing that, move the Texture Input node for the Base Color back in the graph to make room for the additional nodes that you are going to add. Then right-click in an empty region of the graph. You will see a search menu. Search for CheapContrast_RGB (see Figure 11-14).

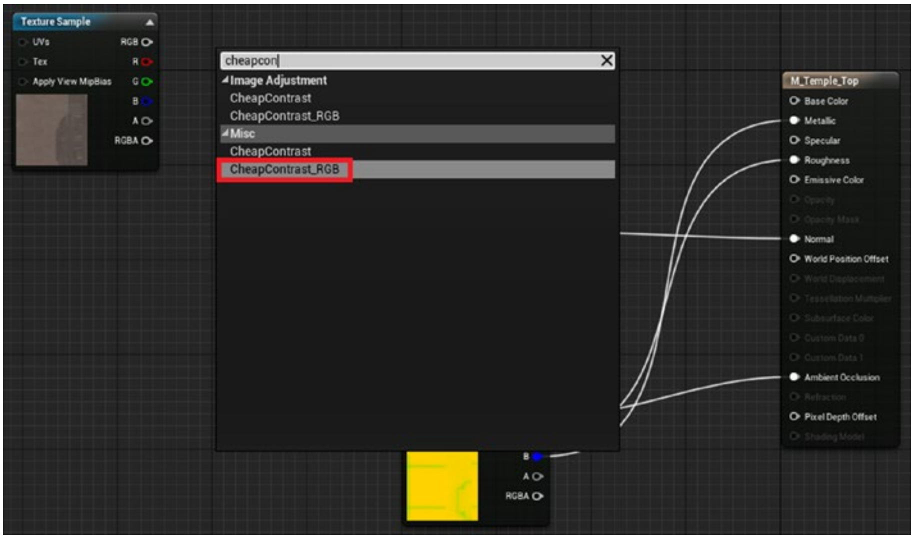


Figure 11-14. *Searching for nodes*

Once the node has been added, position it in front of the Base Color texture sample node. Let’s add another node; this time, a Constant node. Right-click again and search for Constant. Arrange all these nodes as shown in Figure 11-15.

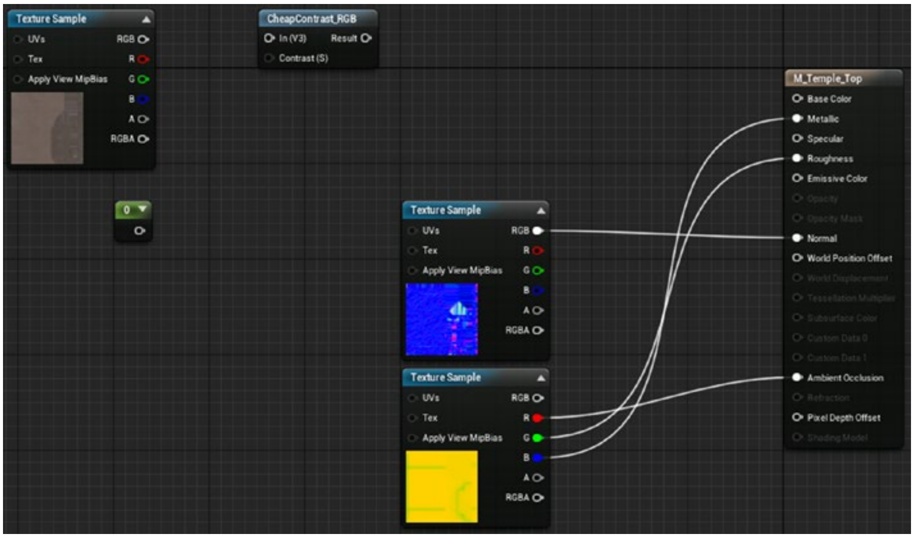


Figure 11-15. *Adding and arranging nodes*

Now let's form the connections. Connect the RGB of Texture Sample to the In (V3) of the CheapContrast_RGB node. Then connect the Constant node to the Contrast (S) of the CheapContrast_RGB node. Your connection should be similar to Figure 11-16.

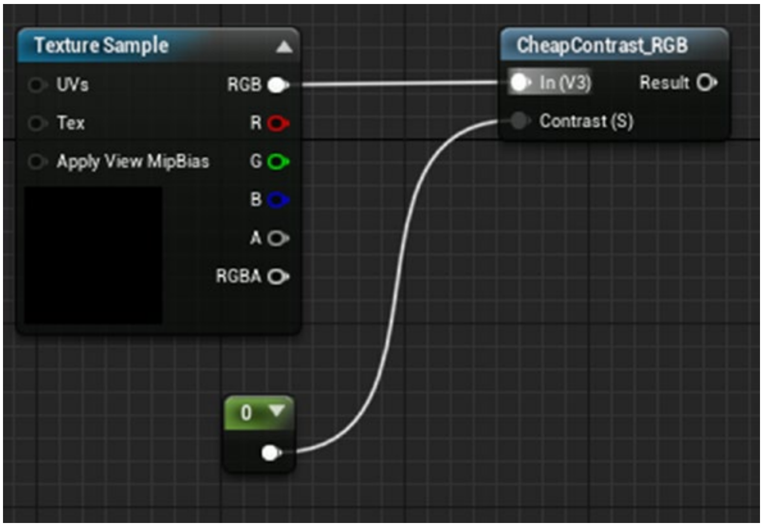


Figure 11-16. *Connection so far*

The CheapContrast_RGB node is used to control the contrast of any texture input node connected to it. The Constant node that we connected will control the strength of the CheapContrast node. We can convert this Constant node into a Parameter, node which will allow us to edit its value in real-time. To do that, right-click on it and choose Convert to Parameter from the list (see Figure 11-17).

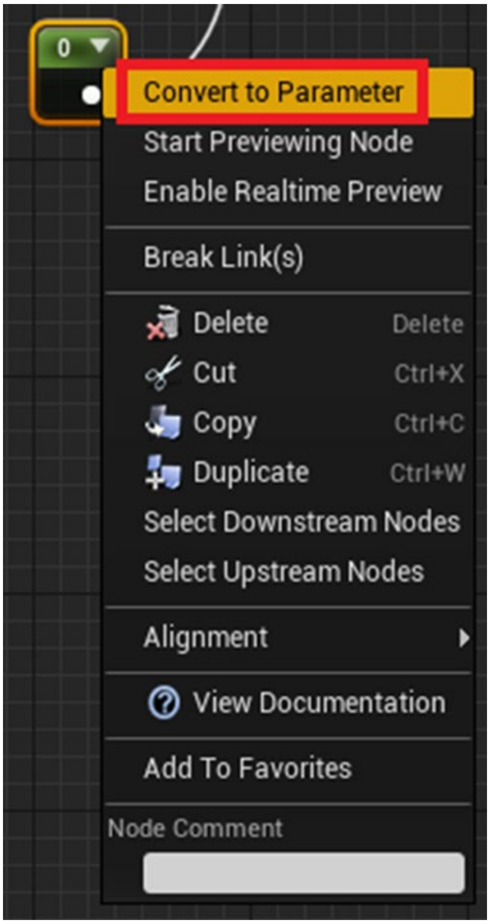
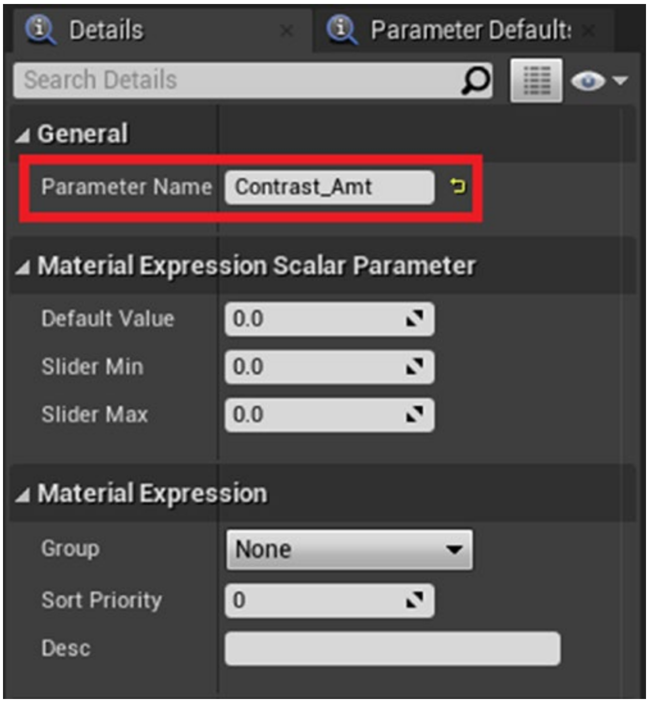


Figure 11-17. *Converting a constant node to a parameter*

As soon as you convert the node to a parameter, you will be given the option to name it. You can name it `Contrast_Amt` for Contrast Amount, as it controls just that. If you want to change the name later, you can do so from the left side Details panel (see Figure 11-18).



General
Parameter Name: Contrast_Amt

Material Expression Scalar Parameter
Default Value: 0.0
Slider Min: 0.0
Slider Max: 0.0

Material Expression
Group: None
Sort Priority: 0
Desc:

Figure 11-18. *Changing the parameter name*

Let's continue by adding a few more nodes. You will now add a Multiply node. Right-click and search for it in the Node Browser. Alternatively, you can press M and click on the graph to add it.

Next, search for Constant3Vector and add it to the graph as well. Alternatively, you can press 3 and left-click to add this node. Convert this node to a parameter and name it `Tint`. Connect all these nodes, as shown in Figure 11-19.

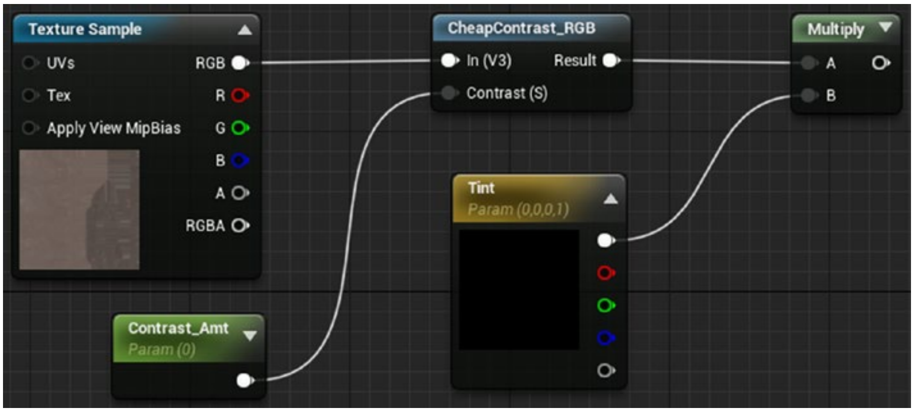
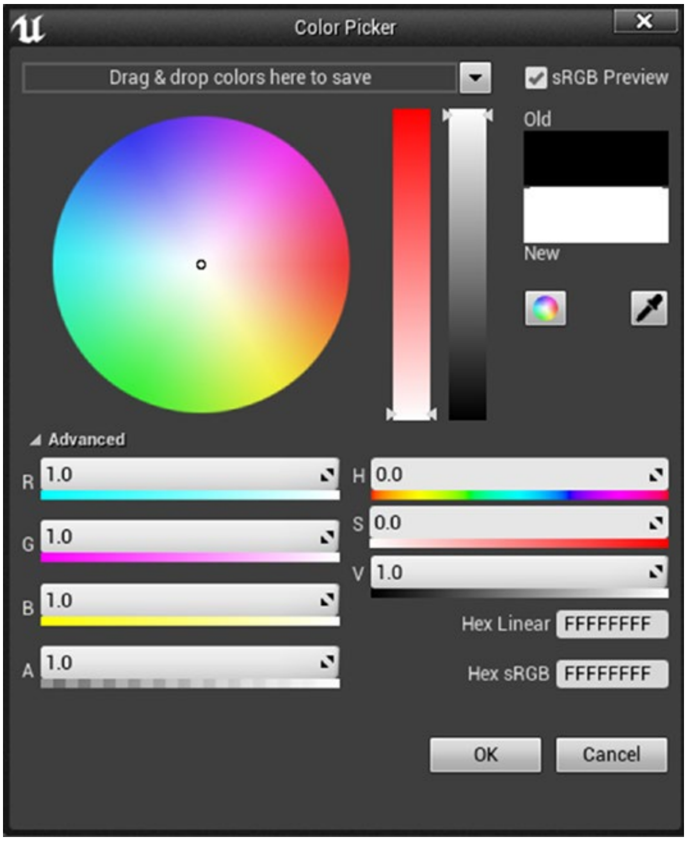


Figure 11-19. *Connecting the nodes*

Set the default value from the Details panel of the Tint node, as shown in Figure 11-20, so that the color becomes white.



R 1.0
G 1.0
B 1.0
A 1.0
H 0.0
S 0.0
V 1.0
Hex Linear FFFFFFFF
Hex sRGB FFFFFFFF

Figure 11-20. *Setting the default value of the Tint parameter*

Now connect the output of the Multiply node to the Base Color of the Material node. Your final setup should look similar to Figure 11-21.

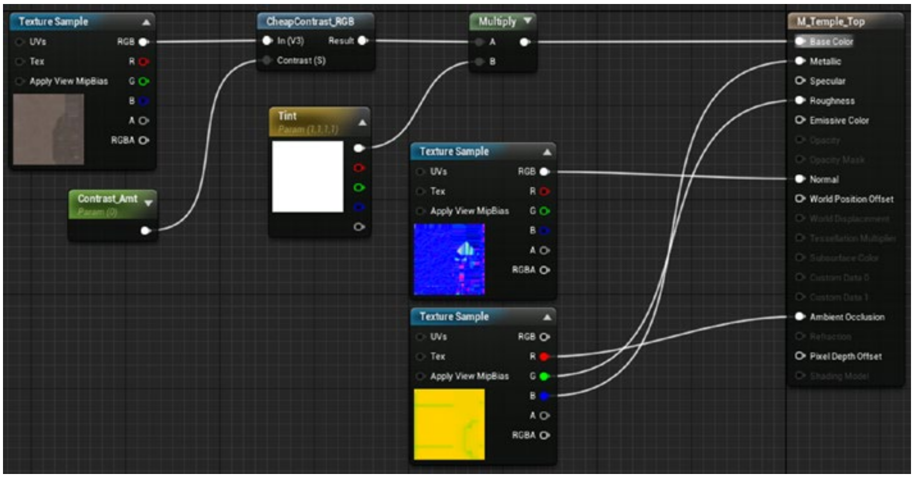


Figure 11-21. *Connection so far*

Click on Save to save the work you have done so far. Next try modifying the two parameter nodes and see how it affects the mesh. Make sure to reset their values to the defaults when you're done.

Let's now set up controls for the normal channel. This is going to be very simple. Break the connection of the normal node. Then search and add the FlattenNormal node. Next, add a Constant node and convert it to a parameter. Name it `Normal_Str` for Normal Strength. After that, search for the OneMinus node and add it to graph as well. Connect all of them, as shown in Figure 11-22.

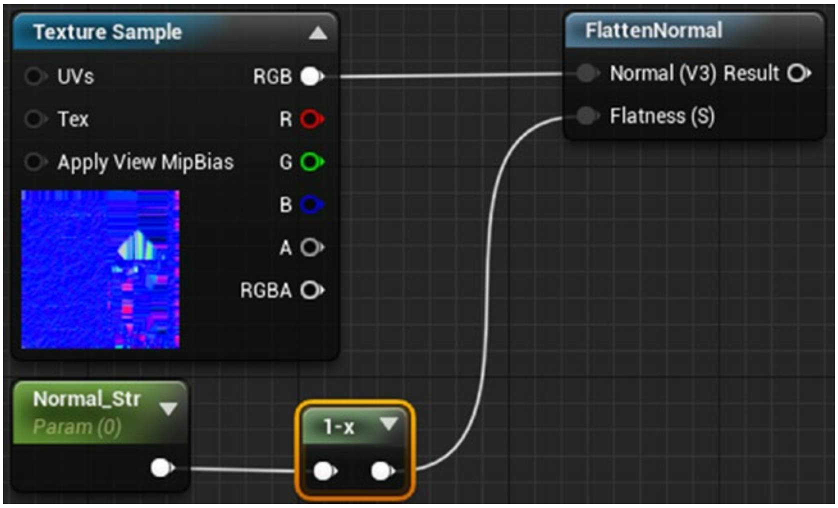


Figure 11-22. *Connections for normal map*

Connect the result of FlattenNormal to the Normal of the Material node.

Next, we will modify the roughness as well, which is equally simple. Simply add a Multiply and Constant node. Convert the Constant node to a parameter and rename it `Roughness_Amt`. Connect them as shown in Figure 11-23.

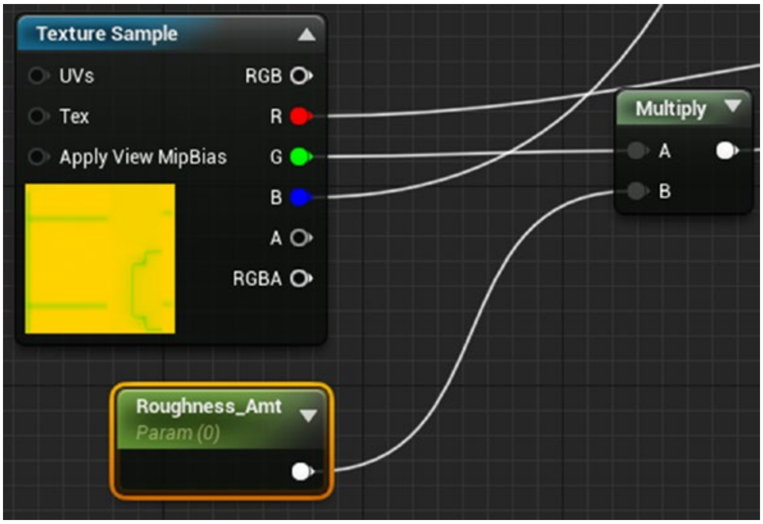


Figure 11-23. *Connections for the roughness map*

Your final connections should look similar to Figure 11-24.

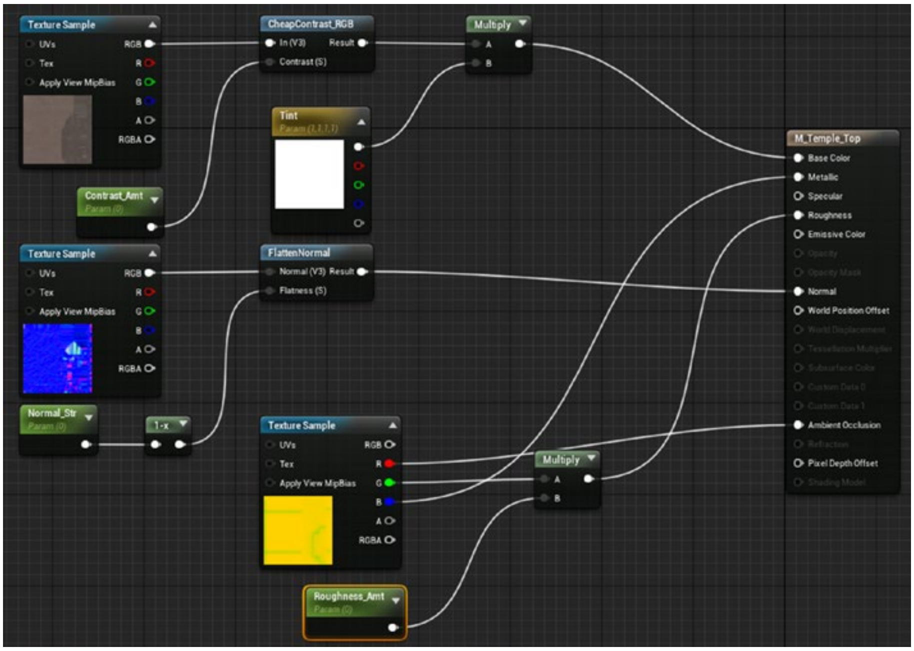


Figure 11-24. *Final setup*

Working with Master and Instance Materials

A material can be instanced to create a unique material created from specified parameters. To do that, make sure that the default values of `Normal_Str` and `Roughness_Amt` are both `1`. Save your file and create a material instance. Close the Material Editor window and right-click on your material file, then choose Create Material Instance from the list (see Figure 11-25).

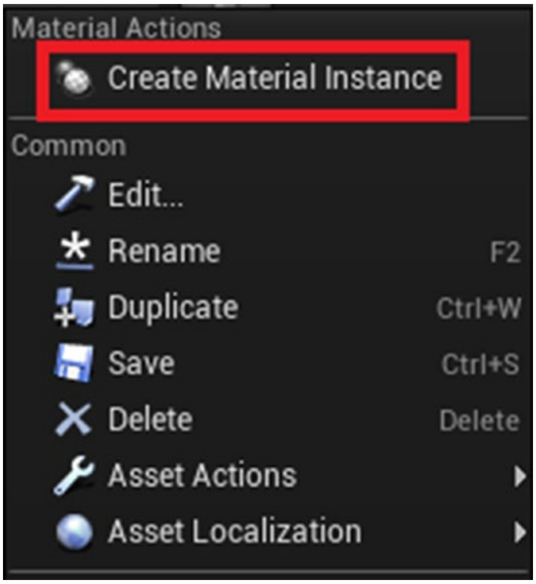


Figure 11-25. *Creating Material Instance*

Name this Material Instance `MI_Temple_Modular_Top`. Double-click on your mesh file to open the Object Properties window and switch the default material file from M_Temple_Modular_Top to MI_Temple_Modular_Top so that the mesh uses the Material Instance as the material file. Now double-click on the Material Instance file to see all the settings.

On the right side, the Details panel shows all the parameters that you created. You can now change the look of your material in real-time using these parameters (see Figure 11-26).

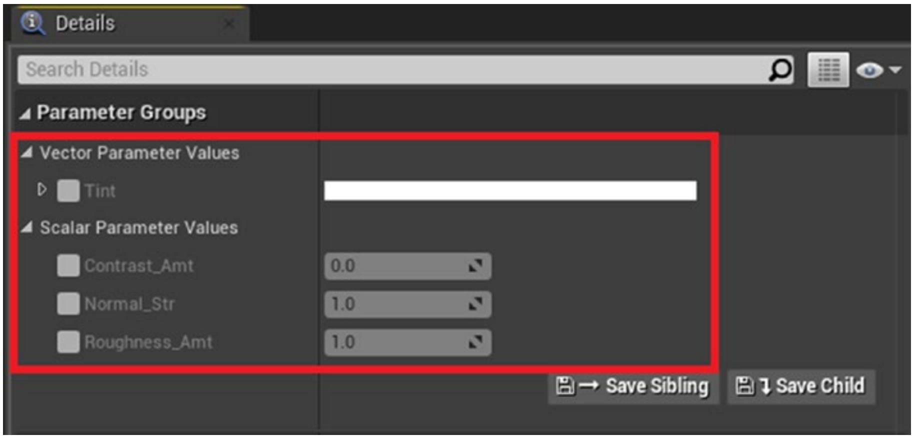

Details
Search Details
Parameter Groups
Vector Parameter Values
Tint
Scalar Parameter Values
Contrast_Amt 0.0
Normal_Str 1.0
Roughness_Amt 1.0

Figure 11-26. *Settings of the Material Instance*

This concludes the material setup chapter. Use this same process to create the rest of the materials. In the final chapter, we will set up the gameplay-related elements.

CHAPTER 12

Integration with VR

In this chapter, we will see how to set up and create a VR project using the assets that we created and imported. This is a pretty tricky chapter, especially for beginners and first-time coders. If you feel stuck at any point, I highly suggest that you go to the UE4 documentation page and forums to find solutions to your queries. Also go through your code and the code mentioned in the book to make sure that you have not missed any commas, full stops, or semicolons. It is very easy to make mistakes when coding; one typing error or missing symbol can cripple your code. So be vigilant and go very slowly through this section. With all that in mind, let's begin with the basics.

Setting Up Visual Studio

We will use C++ to create the VR project. When creating a C++ project, UE4 will notify you that you need an external compiler if you don't have one installed.

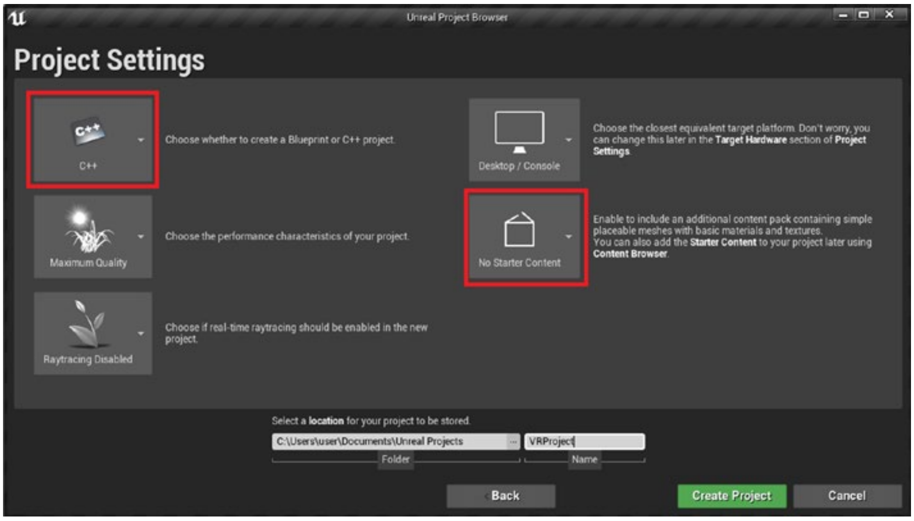
To use C++ with UE4, you need the Visual Studio external compiler. Go to Google and download Visual Studio Community. In this case we will be using the 2019 version, as it is the latest. Installing it is pretty simple, as it automatically handles everything. Close it once it is installed.

Make sure to set the compiler to Visual Studio 2019 (or to the version that you downloaded) when prompted by UE4.

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7_12

Next, let's create a new UE4 project with the following settings:

1. Launch UE4 from the Epic Games Launcher and choose Games from the New Projects Category.
2. From the Templates section, choose Blank.
3. In Project Settings, change Blueprints to C++.
4. Set Starter Content to No Started Content so that there is no starting content in your project.
5. Give your project a proper name (such as `VRProject`). Your final settings should look similar to Figure 12-1.



Project Settings

C++ — Choose whether to create a Blueprint or C++ project.

Maximum Quality — Choose the performance characteristics of your project.

Raytracing Disabled — Choose if real-time raytracing should be enabled in the new project.

Desktop / Console — Choose the closest equivalent target platform. Don't worry, you can change this later in the **Target Hardware** section of **Project Settings**.

No Starter Content — Enable to include an additional content pack containing simple placeable meshes with basic materials and textures. You can also add the **Starter Content** to your project later using **Content Browser**.

Select a **location** for your project to be stored.

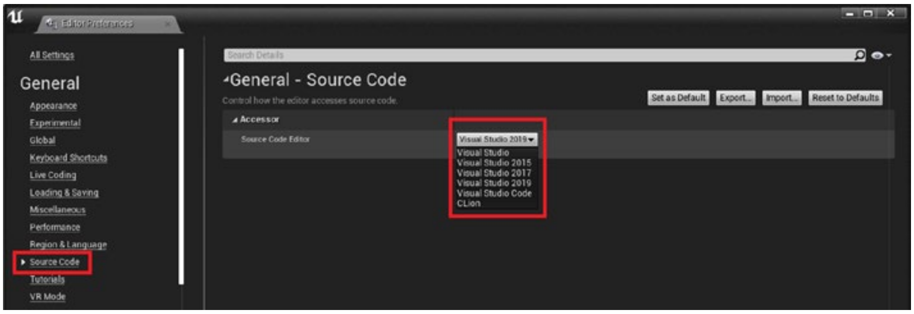
Folder: C:\Users\user\Documents\Unreal Projects

Name: VRProject

Figure 12-1. *Project Settings*

Click on Create Project to initiate project creation. If you have done everything properly, when you create your project, Visual Studio will automatically open. If something goes awry and Visual Studio does not open with UE4, you can troubleshoot the issue. (Even if everything is fine this time, you should know how to fix it in case something happens).

If your project opens in UE4 and you don't see Visual Studio running, click on Edit and choose Editor Preferences. In the Editor Preferences window, look for Source Code under General. When you click on Source Code, you will see Source Code Editor and a drop-down menu next to it (see Figure 12-2). Use this menu to change your Code Editor. In this case set it to Visual Studio 2019.



Source Code Editor: Visual Studio 2019

Figure 12-2. *Changing the source code compiler for UE4*

Next, go to File ➤ Open Visual Studio 2019 (see Figure 12-3) and VS will open with all necessary code loaded. Minimize it for now, as we will come back to it when needed.

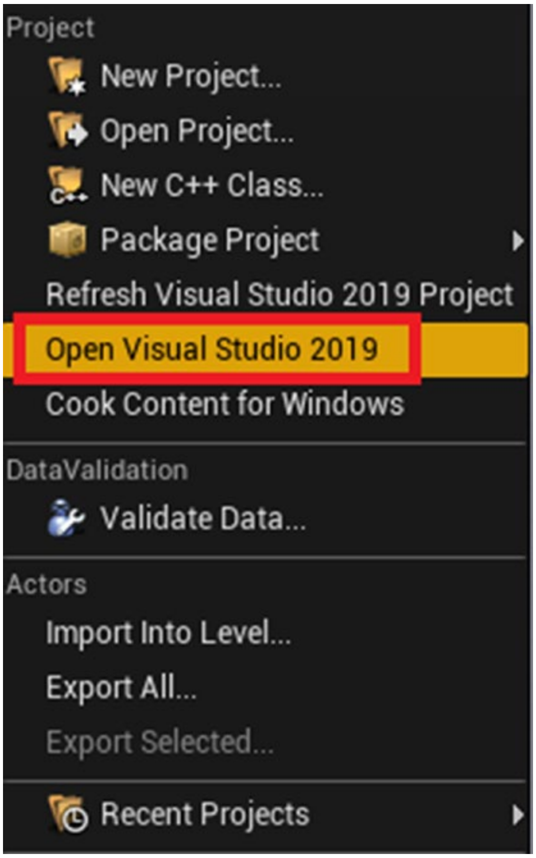


Figure 12-3. *Opening Visual Studio 2019*

Next, right-click on the Content Browser and choose New C++ Class from the list, as shown in Figure 12-4.

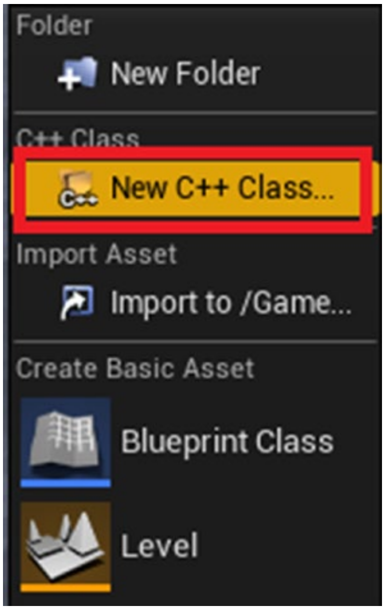
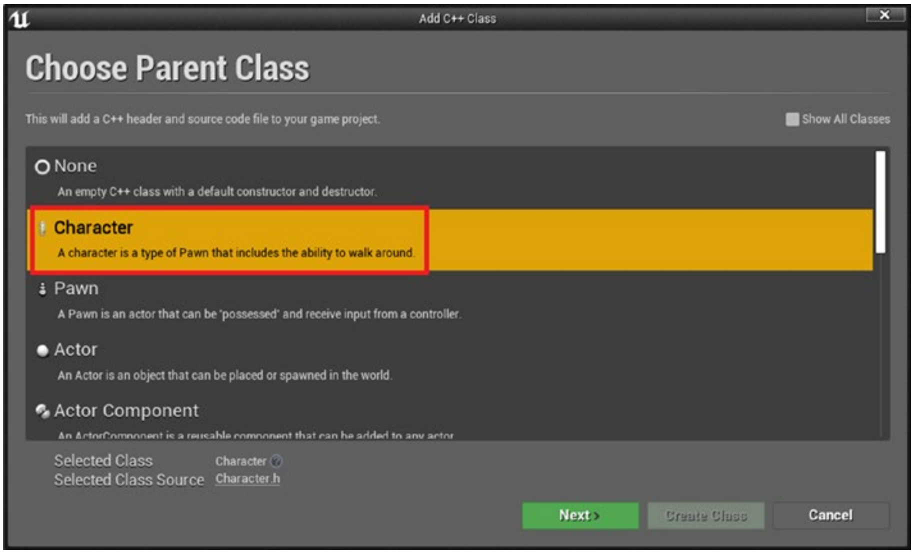


Figure 12-4. *Creating a new C++ class*

A new window will open. From there, choose Character (see Figure 12-5).



Add C++ Class

Choose Parent Class

This will add a C++ header and source code file to your game project.

Show All Classes

None
An empty C++ class with a default constructor and destructor.

Character
A character is a type of Pawn that includes the ability to walk around.

Pawn
A Pawn is an actor that can be 'possessed' and receive input from a controller.

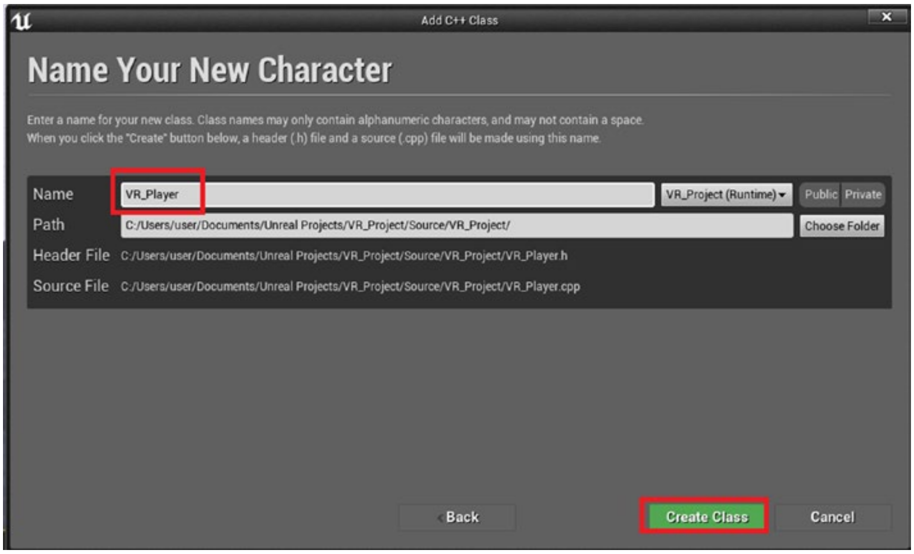
Actor
An Actor is an object that can be placed or spawned in the world.

Actor Component
An ActorComponent is a reusable component that can be added to any actor

Selected Class Character
Selected Class Source Character.h

Figure 12-5. *Choosing a C++ class type*

Click on Next. In the next window, give your class a proper name. In this case we can call it `VR_Player`, as it will contain data related to the Player character. Finally, click on Create Class (see Figure 12-6) to finish setting up. VS 2019 will launch with all the new codes.



Add C++ Class

Name Your New Character

Enter a name for your new class. Class names may only contain alphanumeric characters, and may not contain a space.
When you click the "Create" button below, a header (.h) file and a source (.cpp) file will be made using this name.

Name VR_Player VR_Project (Runtime) Public Private
Path C:/Users/user/Documents/Unreal Projects/VR_Project/Source/VR_Project/ Choose Folder
Header File C:/Users/user/Documents/Unreal Projects/VR_Project/Source/VR_Project/VR_Player.h
Source File C:/Users/user/Documents/Unreal Projects/VR_Project/Source/VR_Project/VR_Player.cpp

Figure 12-6. *Setting up the Player class*

You will see that VS 2019 will lead with two tabs containing different codes. One should be `VR_Player.cpp` and the other one should be `VR_Player.h`. The following code is present in `VR_Player.h`, which is a header file.

```
// Fill out your copyright notice in the Description page of
Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Character.h"
#include "VR_Player.generated.h"

UCLASS()
class VR_PROJECT_API AVR_Player : public ACharacter
{
        GENERATED_BODY()
public:
        // Sets default values for this character's properties
        AVR_Player();

protected:
        // Called when the game starts or when spawned
        virtual void BeginPlay() override;

public:
        // Called every frame
        virtual void Tick(float DeltaTime) override;

        // Called to bind functionality to input
        virtual void SetupPlayerInputComponent(class
        UInputComponent* PlayerInputComponent) override;

};
```

The main purpose of the header file is to declare our classes, functions, and variables. The #include statements at the top are called *preprocessor directives* and they import the necessary header files that contain certain codes and functions essential to developing a project.

One section that this is missing is a Private section, where we will declare our variables, classes, and functions. So let's add that now. Just below where the contents of public ends and above the } bracket, add the private section, as shown here.

```
public:
        // Called every frame
        virtual void Tick(float DeltaTime) override;

        // Called to bind functionality to input
        virtual void SetupPlayerInputComponent(class
        UInputComponent* PlayerInputComponent) override;

private:

};
```

Let's now look at the VR_Player.cpp file.

```
// Fill out your copyright notice in the Description page of
Project Settings.

#include "VR_Player.h"

// Sets default values
AVR_Player::AVR_Player()
{
         // Set this character to call Tick() every frame.  You
         can turn this off to improve performance if you don't
         need it.
        PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void AVR_Player::BeginPlay()
{
        Super::BeginPlay();

}

// Called every frame
void AVR_Player::Tick(float DeltaTime)
{
        Super::Tick(DeltaTime);

}

// Called to bind functionality to input
void AVR_Player::SetupPlayerInputComponent(UInputComponent*
PlayerInputComponent)
{
        Super::SetupPlayerInputComponent(PlayerInputComponent);

}
```

This is the place where we will add all the coding, logic, and definitions. The current setup gives us a head start toward doing that. We need to add a preprocessor directive to our cpp file. We want to work with the camera, so add the line #include "Camera/CameraComponent.h" just below the default #include command.

So with that, let's start adding our code to this. Go back to the header file and add the following line under the private category.

```
private:

        UPROPERTY(VisibleAnywhere)
                class UCameraComponent* Camera;

};
```

Next, we are going to add the following lines to VR_Player.cpp;

```
#include "VR_Player.h"
#include "Camera/CameraComponent.h"

// Sets default values
AVR_Player::AVR_Player()
{
         // Set this character to call Tick() every frame.  You
         can turn this off to improve performance if you don't
         need it.
        PrimaryActorTick.bCanEverTick = true;
        Camera = CreateDefaultSubobject
        <UCameraComponent>(TEXT("Camera"));
        Camera->SetupAttachment(GetRootComponent());
}
```

CreateDefaultSubobject allows us to create a component with the category set to Camera. We can add this component to our Actors in UE4.

Let's go back to UE4 and create GameModeBase, as shown in Figure 12-7.

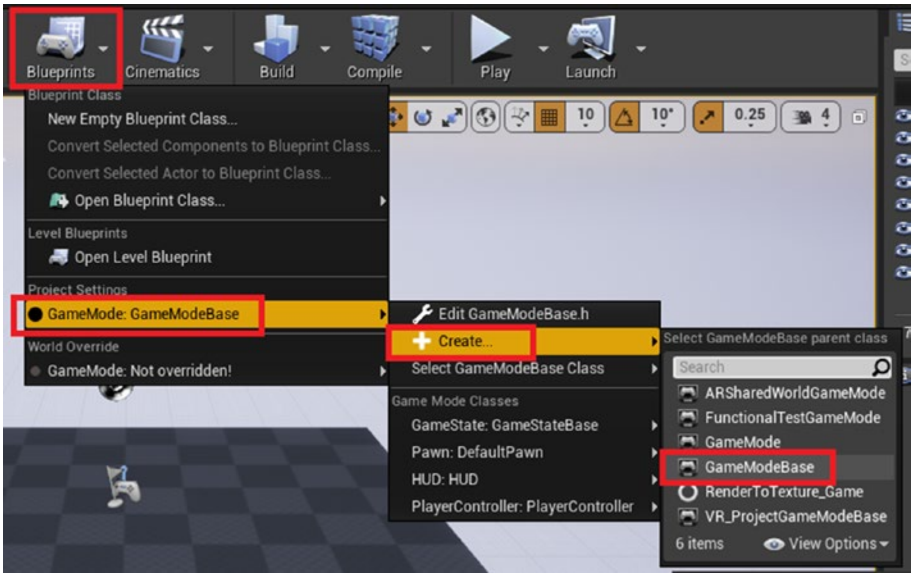
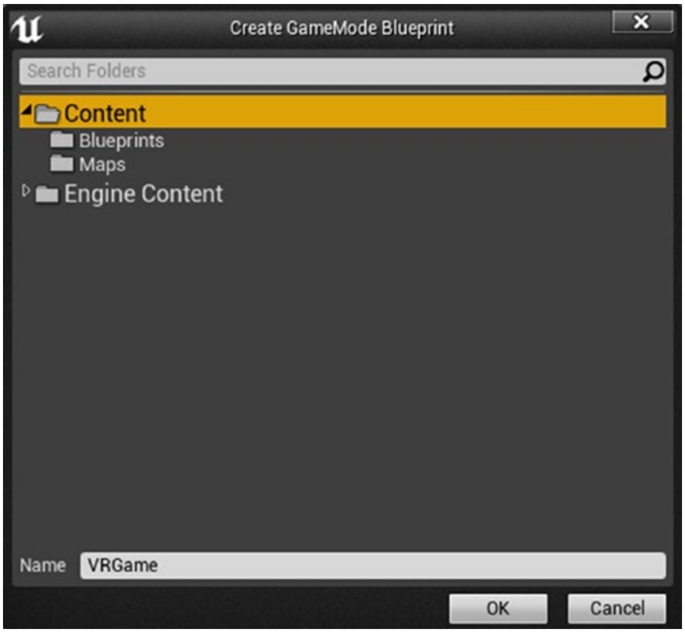


Figure 12-7. *Creating GameModeBase*

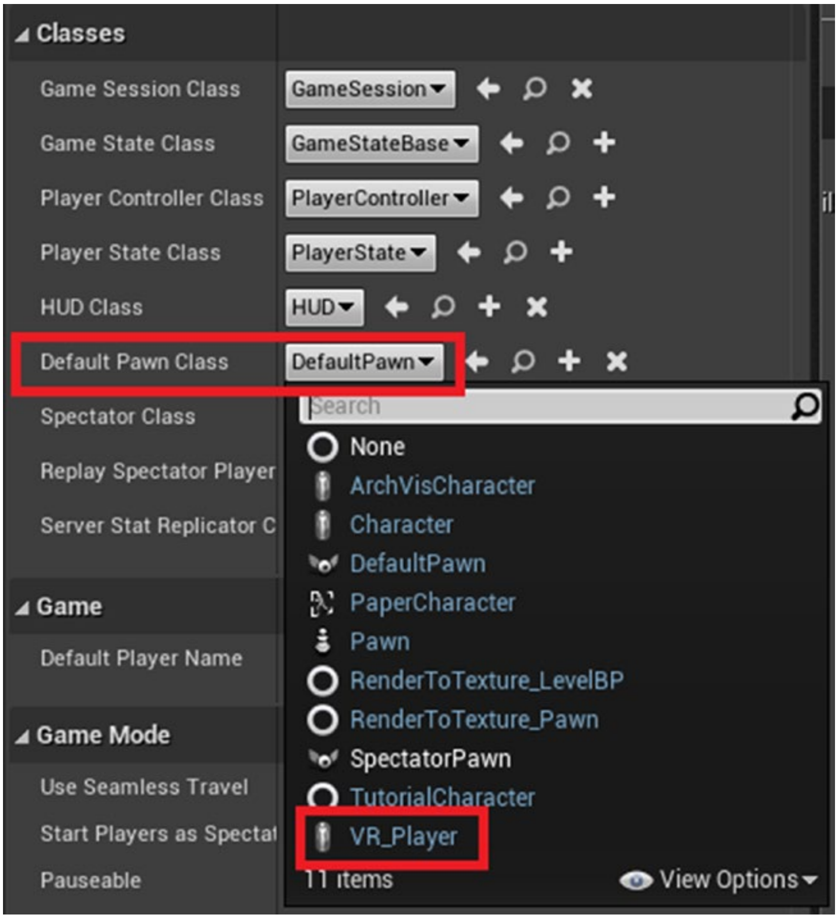
Name this `VRGame` or any name other name that you prefer, as shown in Figure 12-8.



Name: VRGame

Figure 12-8. *Naming the GameModeBase*

When you click OK, a new window will open, allowing you to edit the game mode. Change the Default Pawn Class option to `VR_Player`, as shown in Figure 12-9.


Classes

Game Session Class: GameSession

Game State Class: GameStateBase

Player Controller Class: PlayerController

Player State Class: PlayerState

HUD Class: HUD

Default Pawn Class: DefaultPawn

Spectator Class

Replay Spectator Player

Server Stat Replicator C

Game

Default Player Name

Game Mode

Use Seamless Travel

Start Players as Spectat

Pauseable

Search

None

ArchVisCharacter

Character

DefaultPawn

PaperCharacter

Pawn

RenderToTexture_LevelBP

RenderToTexture_Pawn

SpectatorPawn

TutorialCharacter

VR_Player

11 items

View Options

Figure 12-9. Setting the default pawn class

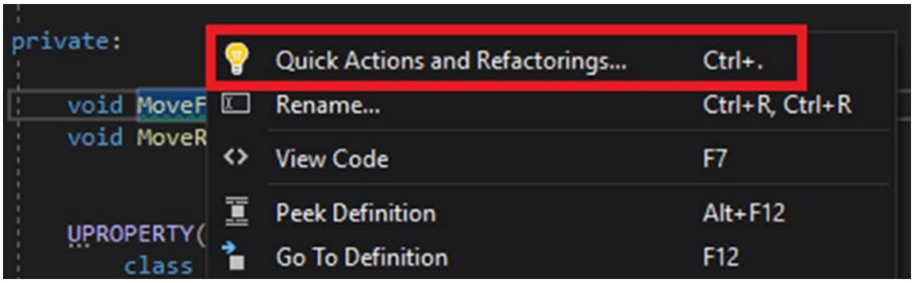
Click on Compile and close the window. After that, go back to VS and to the player header file. We will declare a couple of functions here—a function for moving forward and another for moving right. Add the following lines to your code, just below `private`.

```
private:

        void MoveForward(float move);
        void MoveRight(float move);
        UPROPERTY(VisibleAnywhere)
                class UCameraComponent* Camera;

};
```

Right-click on your class name. Let's begin with `MoveForward`. Choose Quick Actions and Refactorings (see Figure 12-10).



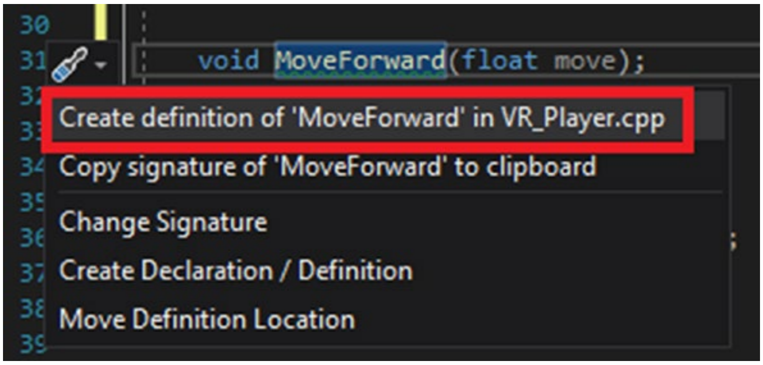
```
private:

    void MoveF
    void MoveR


    UPROPERTY(
        class
```

Figure 12-10. *Quick Actions and Refactorings*

Then choose Create Definition (see Figure 12-11).



```
    void MoveForward(float move);
```

Create definition of 'MoveForward' in VR_Player.cpp
Copy signature of 'MoveForward' to clipboard
Change Signature
Create Declaration / Definition
Move Definition Location

Figure 12-11. *Creating a definition*

This will add the following lines to VR_Player.cpp:

```
void AVR_Player::MoveForward(float move)
{
}
```

This is a very basic definition block. You can add the code inside those curly braces. Do the same thing for the MoveRight function.

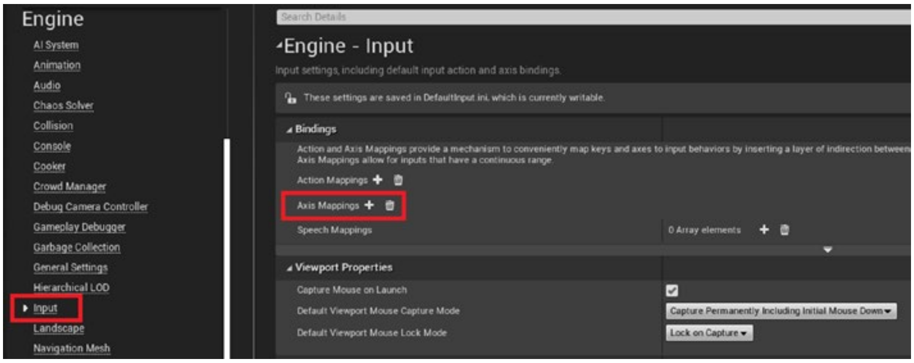
Next, we will create some binding for these controls. To do that, go to the cpp file and add the highlighted lines to your code within the SetupPlayerInputComponent function.

```
// Called to bind functionality to input
void AVR_Player::SetupPlayerInputComponent(UInputComponent*
PlayerInputComponent)
{
        Super::SetupPlayerInputComponent(PlayerInputComponent);

        PlayerInputComponent->BindAxis(TEXT("Forward"), this,
        &AVR_Player::MoveForward);

        PlayerInputComponent->BindAxis(TEXT("Right"), this,
        &AVR_Player::MoveRight);
}
```

Let's return to UE4 and create some key mappings for the project. Go to Edit ➤ Project Settings and look for Input under the Engine category (see Figure 12-12). Click on the + icon next to Axis Mappings to add a new input. Name this input Forward.



Engine - Input

Input settings, including default input action and axis bindings.

These settings are saved in DefaultInput.ini, which is currently writable.

Bindings

Action and Axis Mappings provide a mechanism to conveniently map keys and axes to input behaviors by inserting a layer of indirection between
Axis Mappings allow for inputs that have a continuous range.

Speech Mappings: 0 Array elements

Viewport Properties

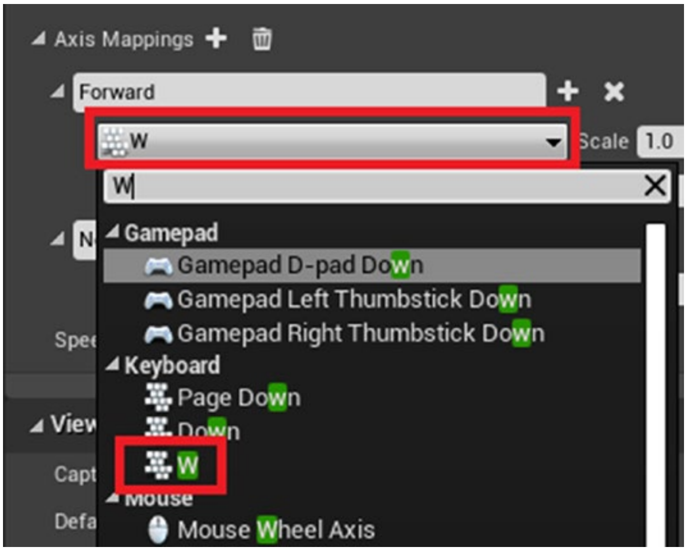
Capture Mouse on Launch: ✓

Default Viewport Mouse Capture Mode: Capture Permanently Including Initial Mouse Down

Default Viewport Mouse Lock Mode: Lock on Capture

Figure 12-12. *Creating axis bindings*

Use the drop-down menu (see Figure 12-13) to look for keys that you wish to bind. This example uses the keys W, S, A, and D.



Axis Mappings

Forward: W, Scale 1.0

Figure 12-13. *Adding a key binding*

Add another key binding to the same category by clicking on the + icon. Add an S input under Forward and set it to `-1.0`. Do the same for Right. Your final setup should look like Figure 12-14.

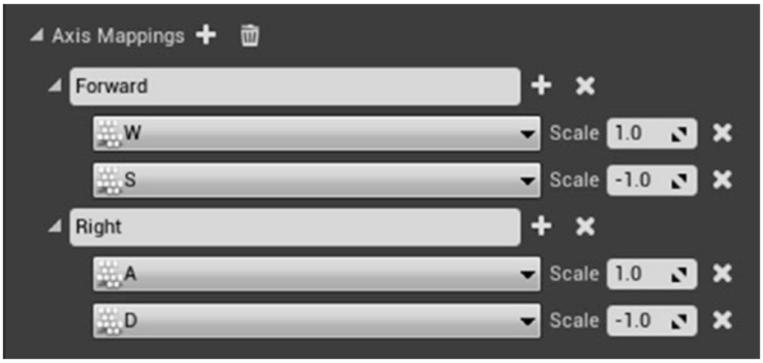


Figure 12-14. *Final setup of the key bindings*

Next, you add the following lines to the `MoveForward` and `MoveRight` functions to add the movement:

```
void AVR_Player::MoveForward(float move)
{
        AddMovementInput(move * Camera->GetForwardVector());
}

void AVR_Player::MoveRight(float move)
{
        AddMovementInput(move * Camera->GetRightVector());
}
```

Go back to UE4 and click on Compile to start compiling your code. Once it's completed, your project is ready for testing. Click on VR Preview to start testing in VR (see Figure 12-15).

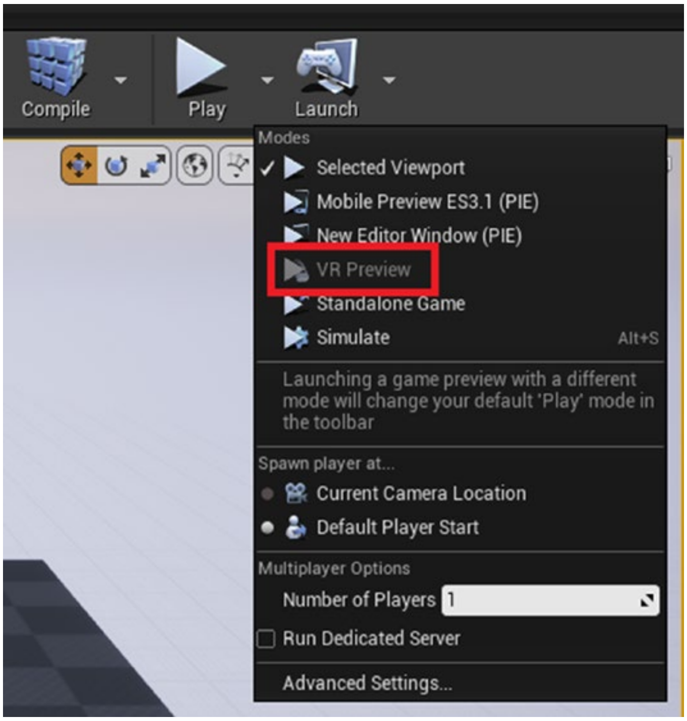


Figure 12-15. *VR Preview option*

Sculpting the Landscape

Let's create a landscape for this level. We will first delete all the unnecessary assets from the scene. It is better if the scene is completely clear to avoid clutter. Next, go to the Landscape tab in the Modes panel (see Figure 12-16).

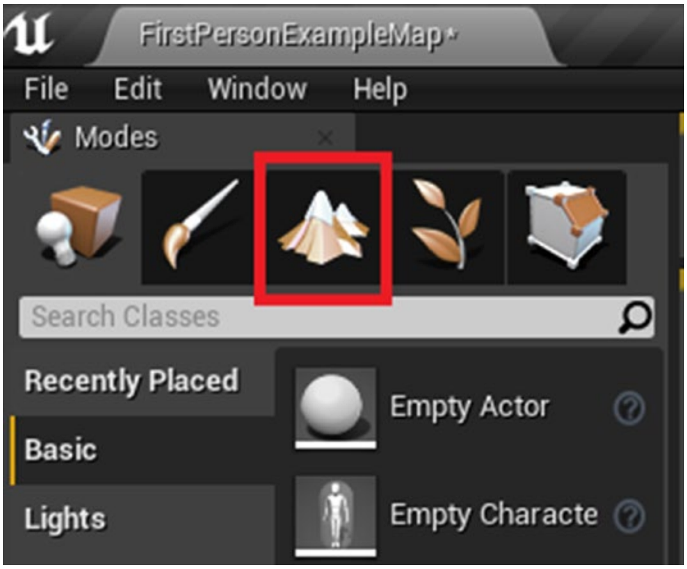
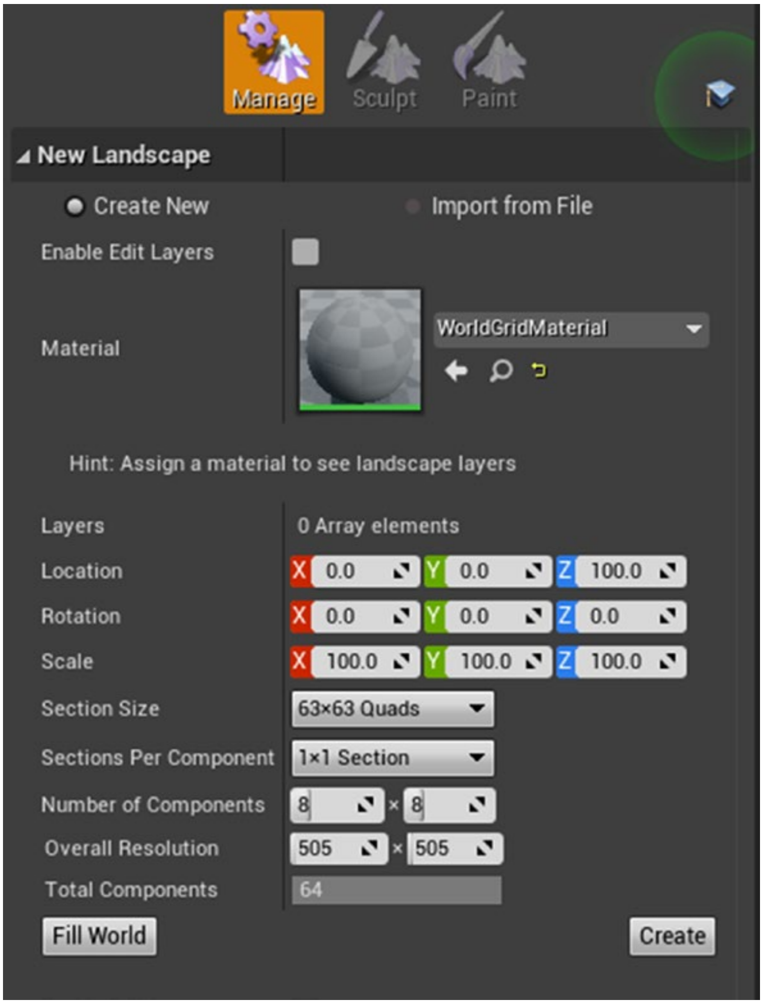


Figure 12-16. *The Landscape tab*

Once you click on the Landscape tab, you will be presented with various landscape settings. We will leave everything at the defaults in this project, as these settings are very complex and technical and out of the scope of this book. The default landscape settings should be similar to Figure 12-17.



Create New ● | Import from File

Enable Edit Layers	
Material	WorldGridMaterial

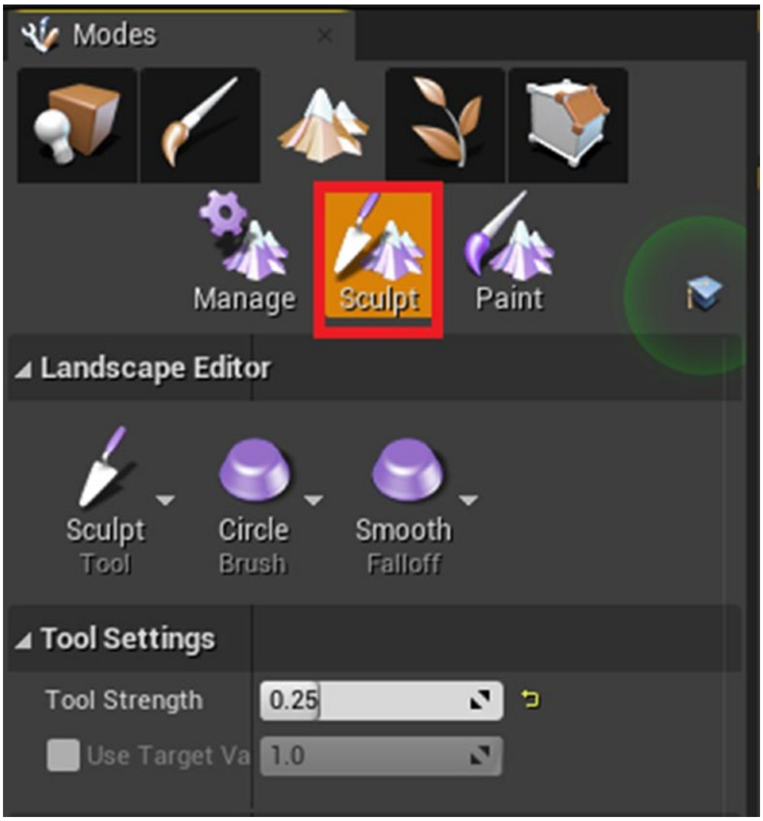
Hint: Assign a material to see landscape layers

Layers	0 Array elements
Location	X 0.0 Y 0.0 Z 100.0
Rotation	X 0.0 Y 0.0 Z 0.0
Scale	X 100.0 Y 100.0 Z 100.0
Section Size	63×63 Quads
Sections Per Component	1×1 Section
Number of Components	8 × 8
Overall Resolution	505 × 505
Total Components	64

Figure 12-17. *Default landscape settings*

A green wireframe of the landscape shows the preview of the landscape that will be created. Click on the Create button to initiate the landscape. An untextured landscape will appear. There will be black lines all across it, which are shadow artifacts. To fix them, simply bake the lighting and the lines will disappear.

Now let's see how we can edit this landscape. UE4 has landscape sculpting tools that we can use to modify the terrain. Let's review the sculpting tools in detail. First of all, in order to access the sculpting tools, you need to select a landscape. Then, in the Modes panel, click on Sculpt under the Landscape tab (see Figure 12-18).

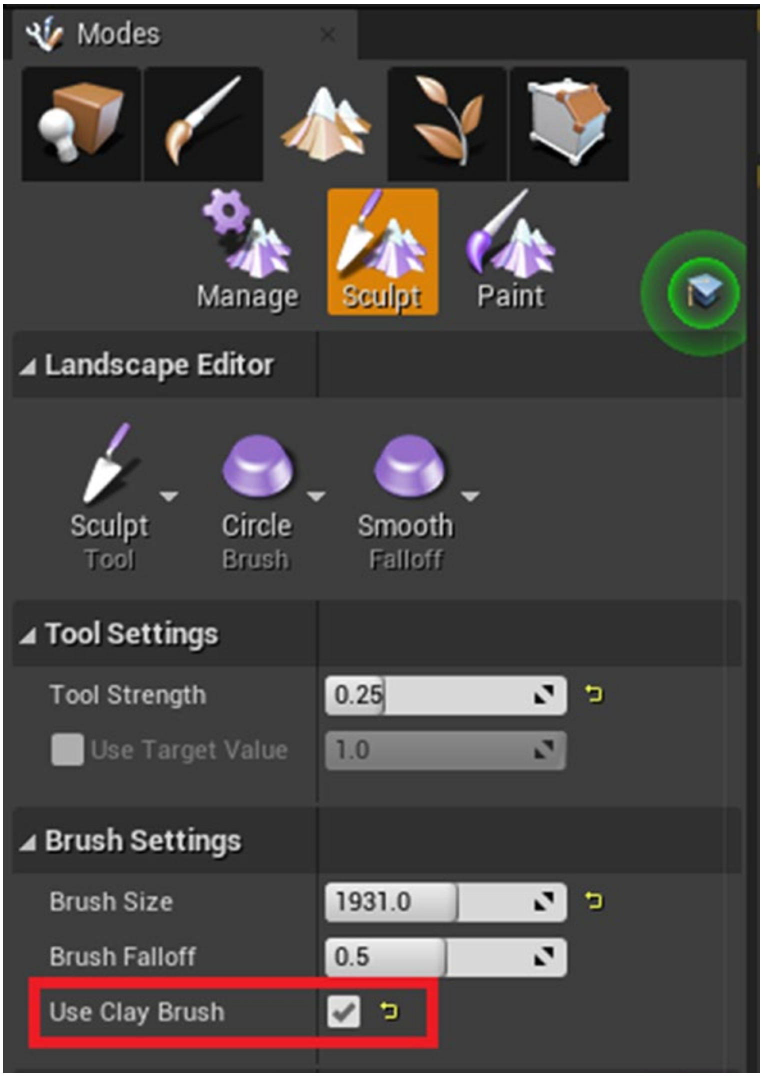


Tool Strength 0.25

Use Target Va 1.0

Figure 12-18. *Sculpt tools*

Next, let's look at all the options. By default, your brush should be set to Sculpt brush. Figure 12-19 shows all the default settings of the brush. Enable the Use Clay Brush checkbox because it makes it easier to control the painting details.



Tool Strength 0.25

Use Target Value 1.0

Brush Size 1931.0

Brush Falloff 0.5

Use Clay Brush ✓

Figure 12-19. *Brush settings*

Now try drawing something on the landscape. Don't worry too much about how it looks for now. Make something that roughly looks like a hill. You will see that it is slightly tricky to do this. There will be hard edges and shapes that you don't like. You can fix them with the Smooth brush. Click on the Brush drop-down menu (see Figure 12-20) and switch the brush to the Smooth brush.

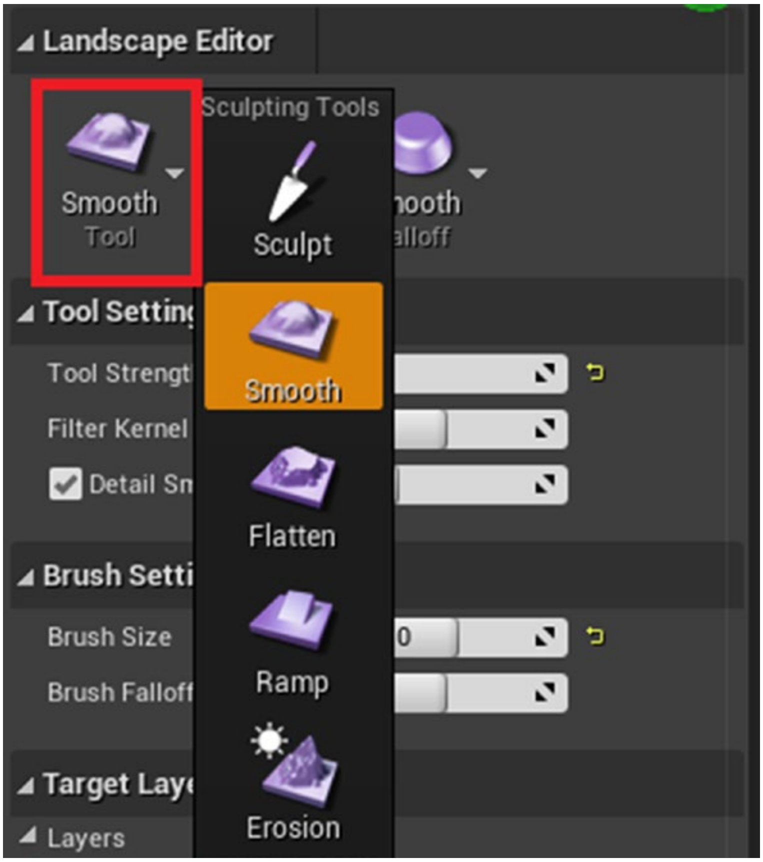
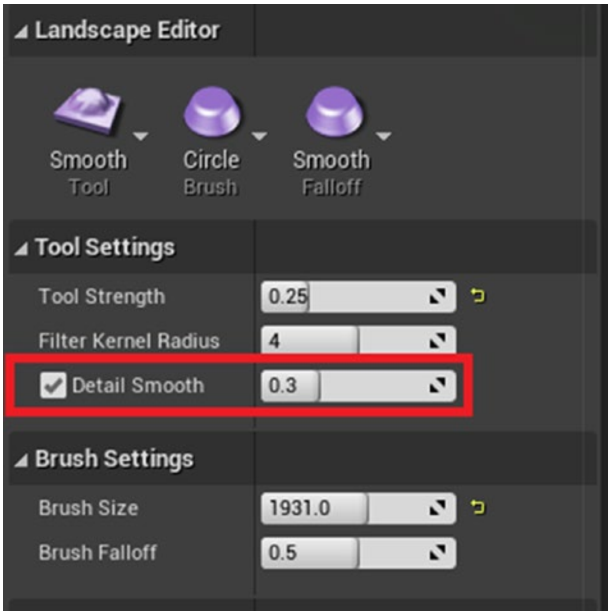


Figure 12-20. *Switching the brush type*

After selecting the Smooth brush, enable Detail Smooth (see Figure 12-21).


Landscape Editor

Smooth Tool | Circle Brush | Smooth Falloff

Tool Settings

Tool Strength 0.25

Filter Kernel Radius 4

✓ Detail Smooth 0.3

Brush Settings

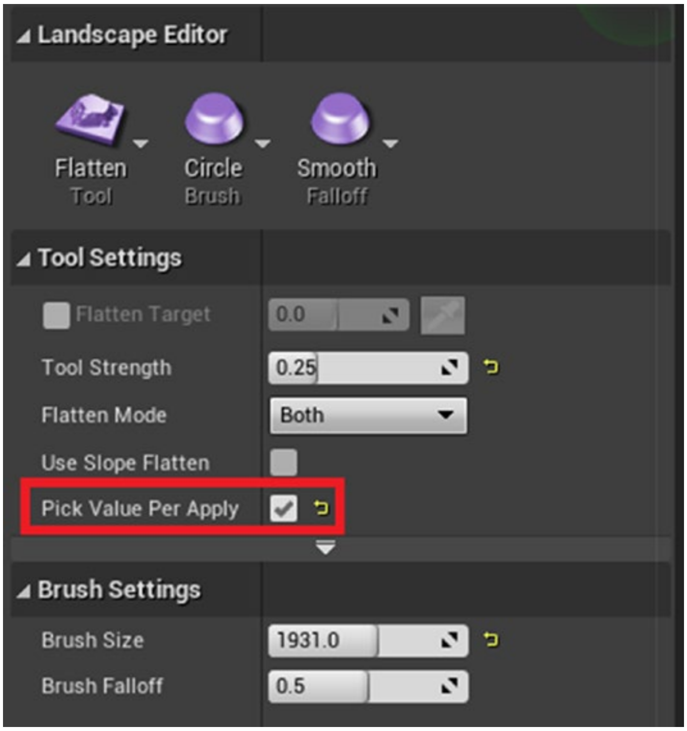
Brush Size 1931.0

Brush Falloff 0.5

Figure 12-21. *Smooth brush settings*

Keep the strength of the Smooth brush very low at all times because it will otherwise wipe out the details very quickly. Enabling Detail Smooth will ensure that any smaller details are preserved when smoothing.

Next is the Flatten brush. This brush will flatten the terrain based on the first place you clicked. Enable the Pick Value Per Apply checkbox here (see Figure 12-22).



Landscape Editor

Flatten Tool | Circle Brush | Smooth Falloff

Tool Settings

- Flatten Target: 0.0
- Tool Strength: 0.25
- Flatten Mode: Both
- Use Slope Flatten: ☐
- Pick Value Per Apply: ✓

Brush Settings

- Brush Size: 1931.0
- Brush Falloff: 0.5

Figure 12-22. *Flatten the brush settings*

This brush is good for creating flat surfaces for placing objects. Make sure to use this brush and see what it does before proceeding.

The last brushes we will discuss are the Erosion brushes, namely Hydro Erosion and Thermal Erosion. These brushes simulate erosion caused by wind, rain, and heat. It is best to use them and see what they do.

We will use these brushes to create a hill-like structure to place our temple on. This can be any shape or size that you want. The result that we have sculpted is shown in Figure 12-23.

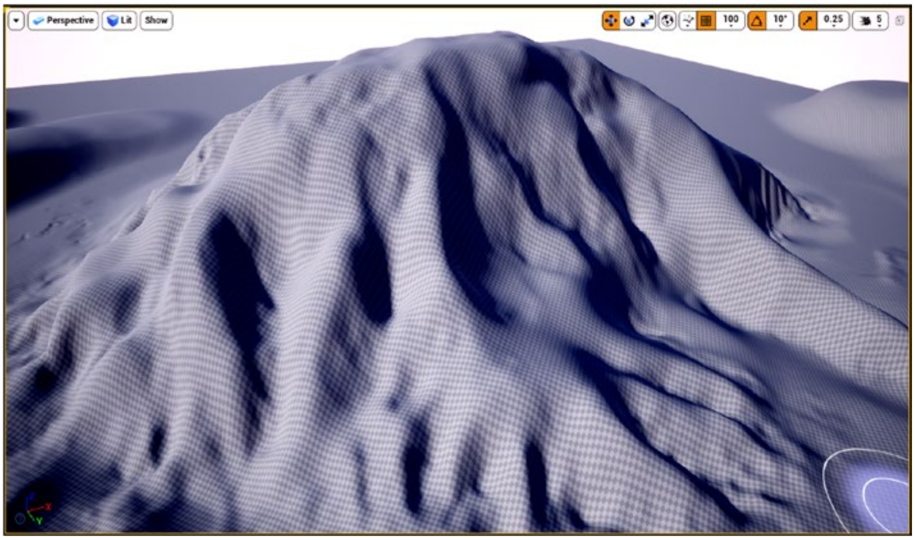


Figure 12-23. *Sculpted hill*

Creating the Landscape Material

We will now create the material for our landscape. First, we need to create a material class like we did previously. Name this class `M_LandscapeMaster`. Double-click on it to open the Material Editor window.

We will need some textures for the landscape. You can either import them from Quixel Megascans or use the starter content that ships with UE4. To import the starter content, simply click on Add New in the Content Browser (see Figure 12-24).

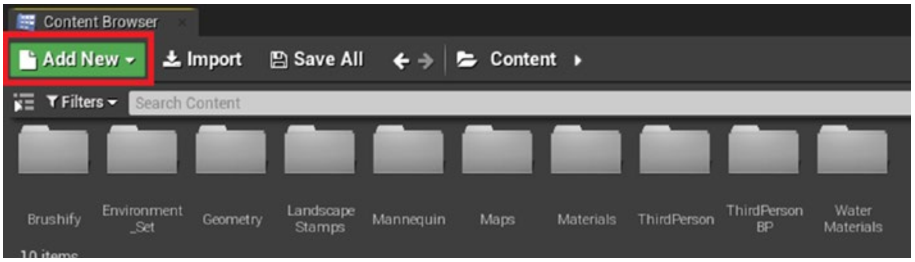
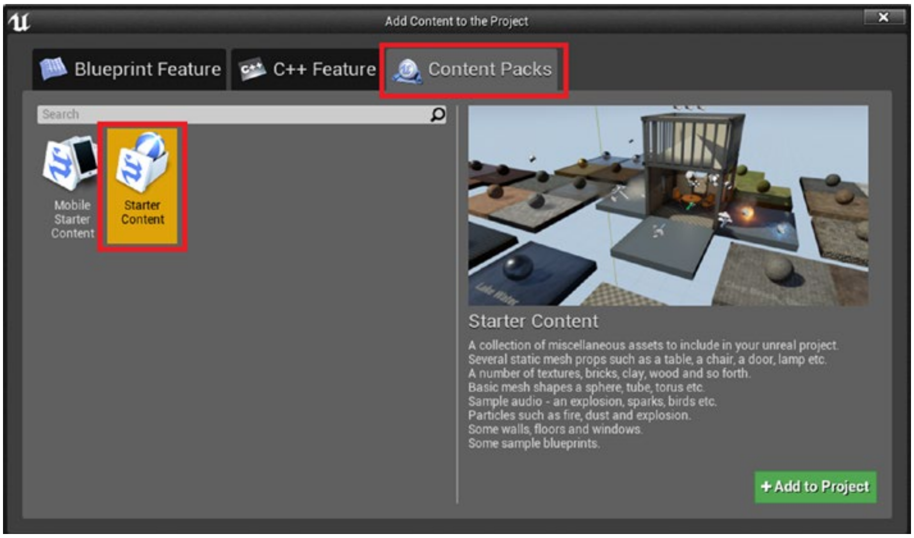


Figure 12-24. *Adding new content*

Next, click on Add Feature or Content Pack at the top and then, from the window, go to Content Packs and choose Starter Content (see Figure 12-25).



Starter Content

A collection of miscellaneous assets to include in your unreal project.
Several static mesh props such as a table, a chair, a door, lamp etc.
A number of textures, bricks, clay, wood and so forth.
Basic mesh shapes a sphere, tube, torus etc.
Sample audio - an explosion, sparks, birds etc.
Particles such as fire, dust and explosion.
Some walls, floors and windows.
Some sample blueprints.

Figure 12-25. *Adding starter content*

Once the import has finished, open the Material Editor window. Open the `Textures` folder of the starter content and search for Grass. Drag and drop T_GroundGrass_D into the Material Editor. Next, search for Rock and choose any rock texture that you want. Drag and drop the Name_D file into the Material Editor.

Next, right-click in the Material Editor window and search for LandscapeLayerBlend and add it to the graph. With the LayerBlend node selected, look at your Details panel and you will see Layers. Click on the + icon next to Array Elements two times to add two new layers (see Figure 12-26).

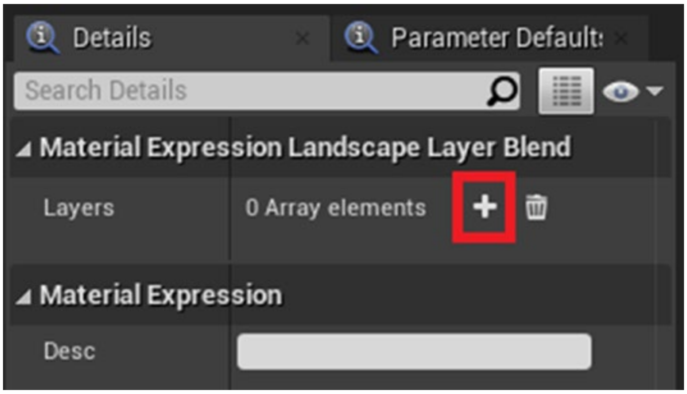
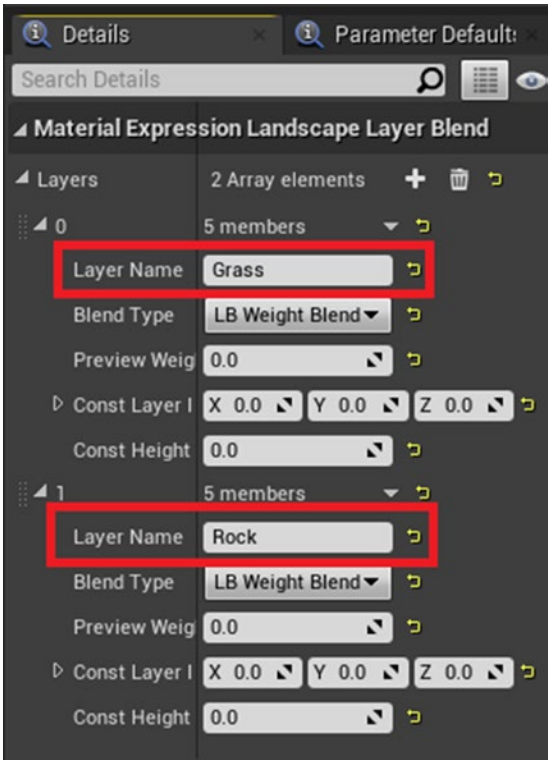


Figure 12-26. *Adding new layers*

Name the top and bottom layers `Grass` and `Rock`, respectively (see Figure 12-27).



Layers — 2 Array elements
0 — 5 members
Layer Name — Grass
Blend Type — LB Weight Blend
Preview Weig — 0.0
Const Layer I — X 0.0 Y 0.0 Z 0.0
Const Height — 0.0
1 — 5 members
Layer Name — Rock
Blend Type — LB Weight Blend
Preview Weig — 0.0
Const Layer I — X 0.0 Y 0.0 Z 0.0
Const Height — 0.0

Figure 12-27. *Naming the new layers*

You will see that these names are added to the LayerBlend node. Now connect the nodes as shown in Figure 12-28.

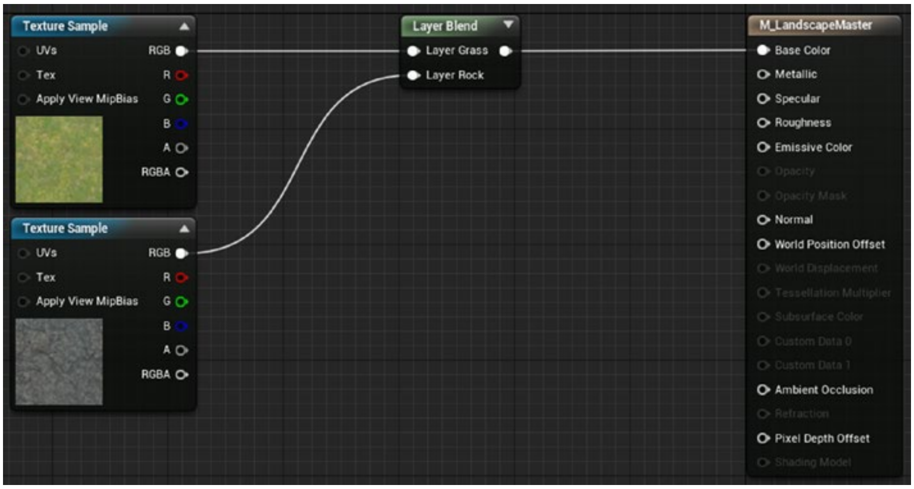


Figure 12-28. *Material setup*

Next, click on the LayerBlend node and, in the Details panel, set the Preview Weight of both layers to `0.5.` The normal maps must be set up the same way. Drag and drop the normal map files for the textures that you imported and then duplicate the LayerBlend node. You do this by right-clicking on it and choosing Duplicate. Set up the connections as shown in Figure 12-29.

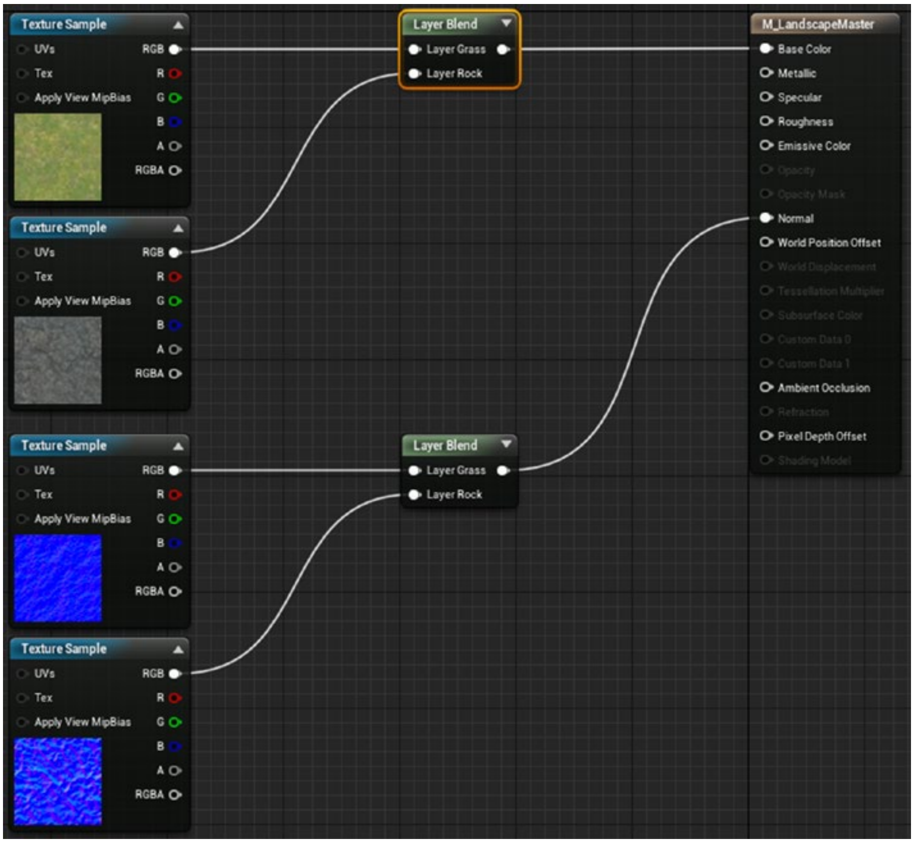


Figure 12-29. *Setting up node connections*

Now, right-click again and add a LandscapeLayerCoords node and connect it to the UVs pin of all the texture maps that we have. In its settings, set the Scale to 8. This will set the tiling of our material. You can change this according to what you want.

Once you're done, save and close the Material Editor and select your landscape. Go to the Details panel and use the Landscape Material drop-down menu to choose the material that you created (see Figure 12-30).

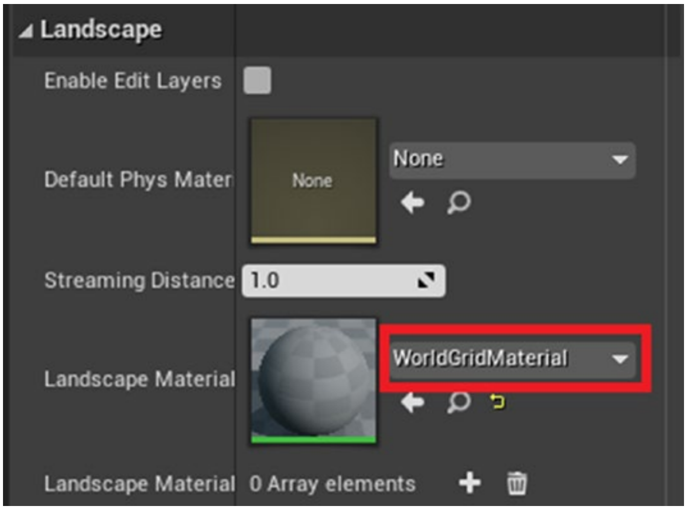
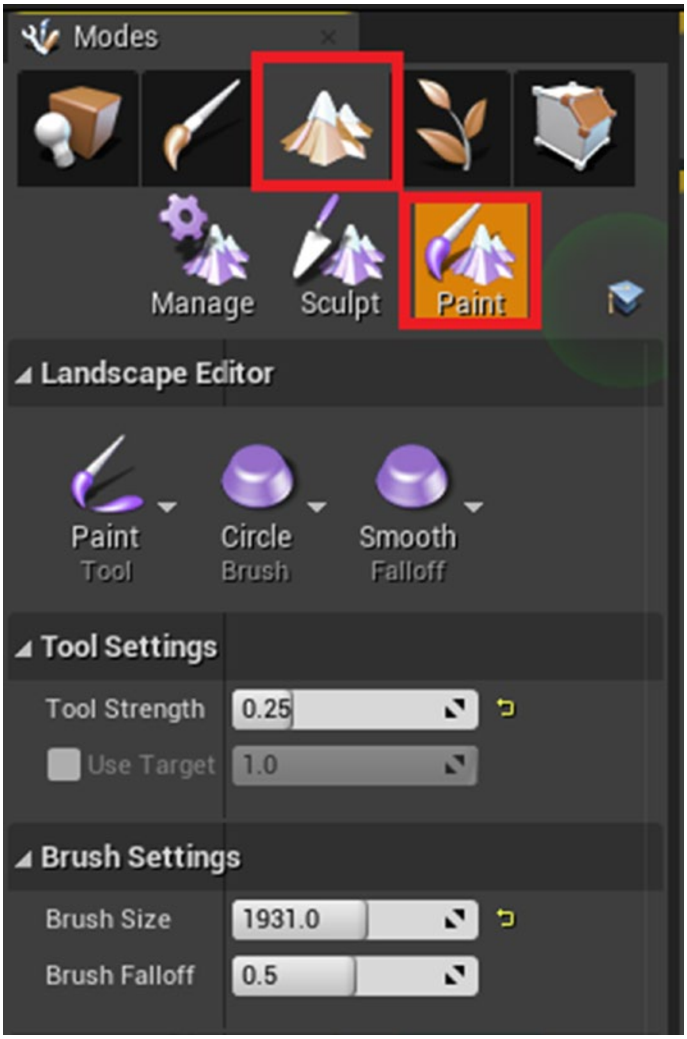

Landscape
Enable Edit Layers
Default Phys Mater None
Streaming Distance 1.0
Landscape Material WorldGridMaterial
Landscape Material 0 Array elements

Figure 12-30. *Selecting landscape material*

Once you're done, go to the Modes tab and, under Landscape, select Painting in order to start texture-painting the terrain.



Tool Strength 0.25
Use Target 1.0
Brush Size 1931.0
Brush Falloff 0.5

Figure 12-31. *Texture Paint tab*

Scroll down and look for the layers that you created. Click on the + icon and choose Weight Blended Layer (Normal) (see Figure 12-32).

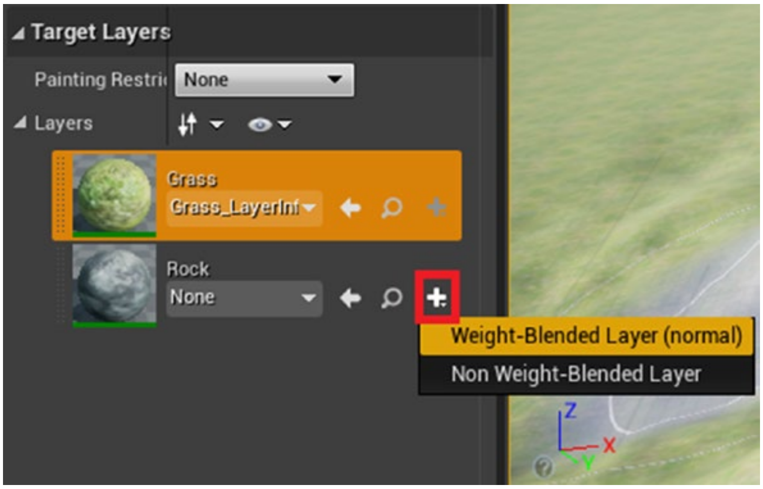


Figure 12-32. *Selecting a layer blend type*

A new window will open; just click OK there. Now you are ready to paint the textures. Paint the entire mountain with the rocks material and then paint grass on the flatter areas of the mountain. The results so far are shown in Figure 12-33.

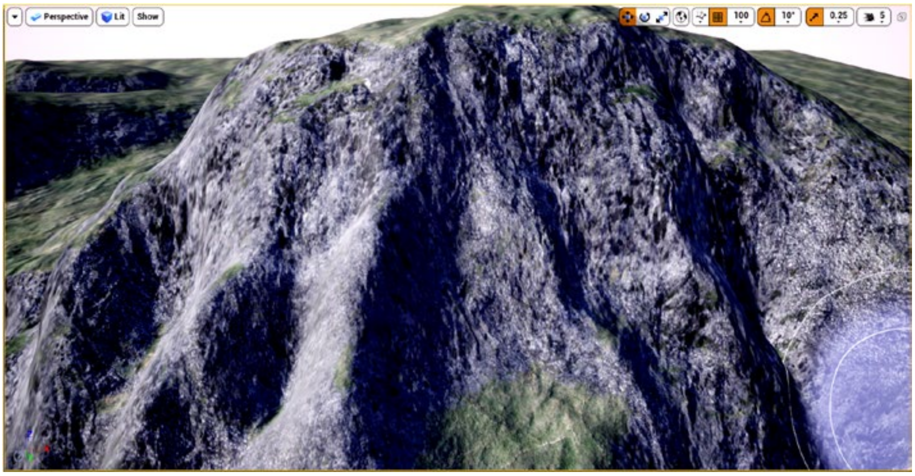


Figure 12-33. *Landscape texturing*

Importing and Using Foliage

Now we will import some foliage assets and use them to decorate the scene. First, go to the UE4 marketplace and locate the Free option at the top. In the Free submenu, click on Permanently Free Collection (see Figure 12-34).

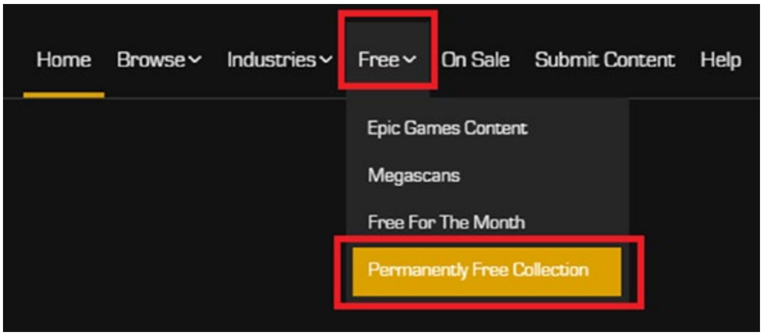
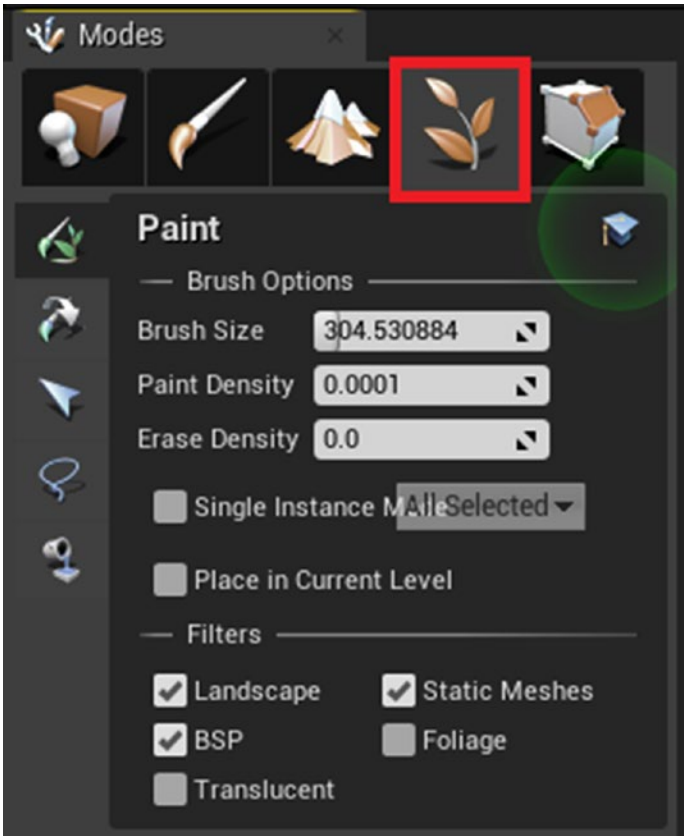


Figure 12-34. *Permanently free collection*

In the Permanently Free Collection, redeem all the vegetation and foliage packs. After adding them to your collection, go to your UE4 library and click on Add to Project. Add the vegetation pack of your choice to your project. After that, open the folder in Content Browser, which contains the foliage meshes. Click on the Foliage Painting tab in the Modes panel to open the Foliage Painter tool (see Figure 12-35).



Brush Size 304.530884
Paint Density 0.0001
Erase Density 0.0
Single Instance Mode All Selected
Place in Current Level
Filters
[✓] Landscape [✓] Static Meshes
[✓] BSP [ ] Foliage
[ ] Translucent

Figure 12-35. *Foliage Painter tool*

Now drag and drop some foliage meshes into the Drop Foliage Here section of the Foliage Painter tool. Adjust the tool settings by decreasing the brush size and reducing the paint density to a very small number. Now try painting some foliage on your hill. Figure 12-36 shows the result so far.

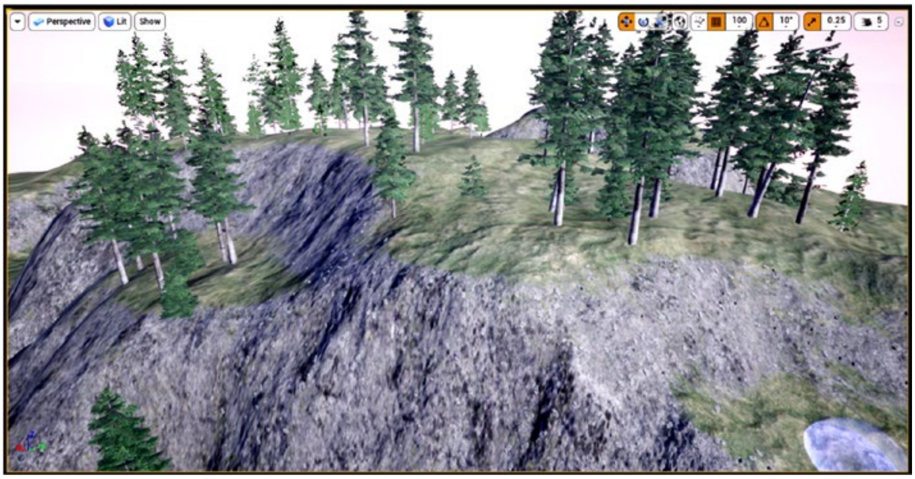


Figure 12-36. *Foliage painting result*

After importing the remaining assets, you should see results similar to Figures 12-37 and 12-38.

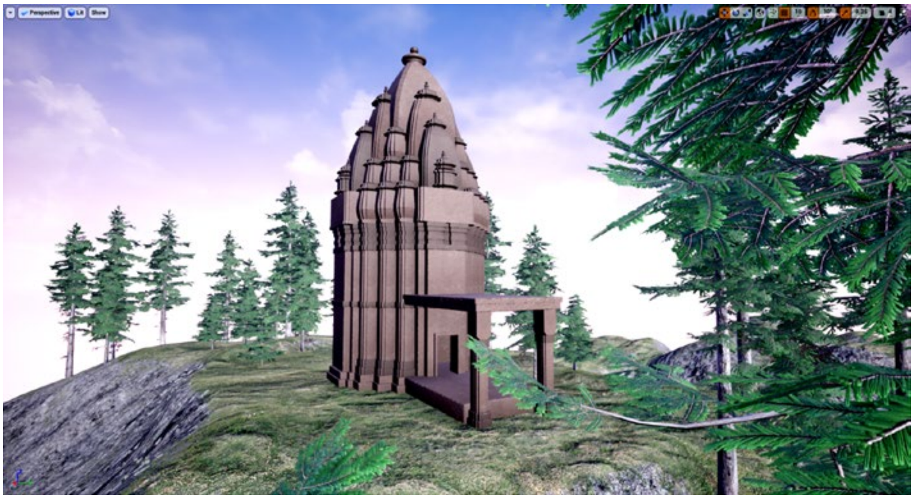


Figure 12-37. *Final result*

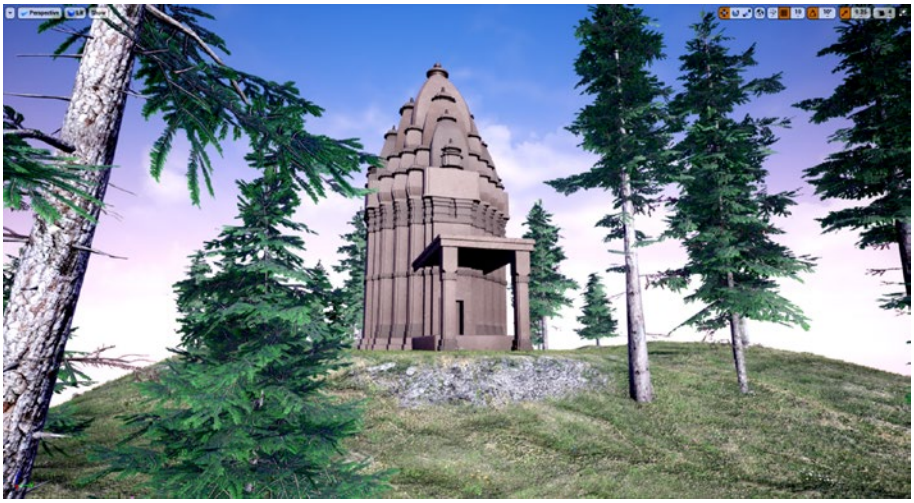


Figure 12-38. *Final result*

So what remains now is importing the previous project where you set up everything. This is easy; all you have to do is copy the contents of the previous project from its directory on your PC to the directory of the VR Project and you will be free to access everything. Or, if you are up for a challenge, try re-creating them, this time by yourself without looking at the book.

So this concludes the chapter on VR integration. I hope you found this informative and useful.

Index

© Abhishek Kumar 2020
A. Kumar, *VR Integrated Heritage Recreation*,
https://doi.org/10.1007/978-1-4842-6077-7