Tampereen ammattikorkeakoulu

Creating a 3D environment using Unreal Engine 5.1

Olivia Koivisto

BACHELOR'S THESIS
June 2023

Business Information Systems
Game Production

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Game Production

KOIVISTO, OLIVIA:
3D-ympäristön luominen Unreal Engine 5.1-pelimoottorilla

Opinnäytetyö 28 sivua
Kesäkuu 2023

Opinnäytetyön tavoitteena oli toteuttaa peliympäristö Unreal Engine 5.1 -pelimoottorissa. Tarkoituksena oli käyttää erilaisia metodeja pelivalmiiden 3D-mallien luomiseen, selvittää eri tapoja niiden optimointiin sekä teksturointiin ja näin määrittää prosessille optimoitu työnkulku. Opinnäytetyössä perehdytään myös eri elementtien ja efektien lisäämiseen Unreal Enginen sisällä sekä käyttämään pelimoottoria tehokkaasti niin, että siinä käsitellyt metodit ovat sovellettavissa myös muissa Unreal Enginellä tehtävissä projekteissa.

Prosessin jokainen työvaihe on jaettu omiin lukuihinsa. Ne käsitellään tekniikoineen, tarkastellaan pelituotannossa vallitsevia yleisiä käytäntöjä sekä huomioidaan projektissa käytettyjen ohjelmien ja Unreal Enginen välinen tekninen yhteentoimivuus.

Projektissa on käytetty 3D-mallien luomisessa pääasiassa Blender-mallinnusohjelmaa ja materiaalien tuottamiseen Adoben Substance Painteria. Opinnäytetyössä dokumentoidaan näiden lisäksi myös muita käytettyjä ohjelmia, jotka olivat projektin kannalta oleellisia sekä Unreal Engine 5.1:n käyttöä ja sen ominaisuuksien hyödyntämistä 3D-ympäristön luomisessa.

Opinnäytetyöprojektin toteuttaminen sujui aikataulussa, eikä projektissa kohdattu suurempia haasteita. Projekti oli todella laaja ja monivaiheinen, mutta sille määriteltiin selkeä ja helposti seurattava työnkulku vaiheineen, josta on hyötyä Unreal Engine-pelimoottoria käyttäville ympäristöartisteille.

Asiasanat: pelimoottori, Unreal Engine, Blender

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Production

KOIVISTO, OLIVIA:
Creating a 3D Environment Using Unreal Engine 5.1

Bachelor's thesis 28 pages
June 2023

The aim of this bachelor’s thesis was to create a 3D environment using the Unreal Engine 5.1 game engine. The intention was to use different methods to create game-ready 3D assets, examine different ways to optimize and texture them, and thus determine an optimal workflow for the process. This thesis will familiarize the reader with adding different elements and effects within Unreal Engine, as well as using the game engine effectively so that the methods discussed in it are also applicable in other projects made with Unreal Engine.

Each work phase of the process is divided into its own chapters, where used techniques are reviewed with the general practices prevailing in game production industry, and the technical interoperability between the programs used in the project and Unreal Engine is considered.

This project mainly uses the Blender modeling program to create 3D models and Adobe’s Substance Painter to produce the materials. In addition to these, the thesis also documents other used programs that were essential for the project, as well as the use of Unreal Engine 5.1 and the utilization of its features in creating a 3D environment.

The creation of the thesis project went smoothly and on schedule, and no major challenges were encountered in the process. The project was extensive and multi-phased, but a clear and easy-to-follow workflow with detailed work stages was designed, which is useful for environmental artists using the Unreal Engine game engine.

Key words: game engine, Unreal Engine, Blender

TABLE OF CONTENTS

ABBREVIATIONS AND TERMS

Blender	Free 3D modeling program
Game engine	Software designed for developing video games
High poly	3D model with high amount of polygons
LOD	Determines the level of detail of a 3D model
Low poly	3D model with low amount of polygons
Polycount	The number of polygons on a 3D model
Polygon	Surface made by 3 or more vertices
Rendering	Generating an image from 2D or 3D model
Substance 3D Painter	3D texturing program
Topology	Structure of a 3D model
Unreal Engine	Popular game engine by Epic Games
Vertex	Point in 3D space

1 INTRODUCTION

The gaming industry is a constantly growing field, and companies have an ongoing need for skilled artists. In addition to just 3D modeling and sculpting, a good game artist should know how to use the common game engines in the field, as the work of an artist is not limited to just creating a 3D model. Game engines have increasingly more graphical features that open possibilities in creating a stunning environment as development progresses. It is beneficial for a game artist to have some knowledge of these possibilities and features, as well as how to efficiently use them. This makes it possible to create complex graphics and effects that are optimized to the platforms, not being too heavy to run smoothly.

The Unreal Engine is a free game engine developed by Epic Games and it has gained a lot of popularity due to its features. It is constantly being developed and improved, as it is best known for its ability to create incredible immersion and have a great performance. Unreal Engine uses its own technologies, such as Lumen for photorealistic lighting, Nanite for automatic control of level of details (LODs), Virtual Shadow Maps for film-quality rendering of shadows, Niagara for particle effects and Chaos as a physics engine. These features are completely unique to Unreal Engine and enable it to be widely used as a platform for games, TV and film industries.

The thesis introduces the use of Unreal Engine 5.1 as a platform for creating a game-ready 3D environment. In this project, assets are modeled using Blender, which is a free and widely used modeling program. Then they are textured with Adobe's Substance 3D Painter and transferred to Unreal Engine. In the game engine, the environment is built using these 3D models, and plenty of the engine's internal features are utilized to create a beautiful world that is optimized for game use.

In this thesis each work phase is explained in detail and such a way that they can be easily applied in other projects. It is recommended that the reader has the basic knowledge of 3D modeling and texturing, as they are not explained too extensively.

2 PLANNING THE ENVIRONMENT

2.1 Finding references

Project planning is widely considered to be an important part of a successful project. The time spent on it reduces the risk of the project failing and increases the probability of its success. (Serrador, 2012.) The planning should begin by defining the goal of the project, setting a scope and deciding a schedule.

When creating a game environment as described in this thesis, it is necessary to find as many reference images suitable for the project as possible. Using these pictures, a general theme for the environment is planned, which we set out to implement. Reference pictures are also very important when modeling assets, as they help achieve the correct proportions.

Reference pictures can be found from free stock photo websites. They often contain various categories of images that can be freely used in different projects or studies. Gathering some of these images help plan and determine the wanted mood and general look of the environment

IMAGE 1. Reference pictures

2.2 Sketching and whiteboxing

Before starting with Unreal Engine, it can be beneficial to do some sketching and prototyping with primitives, sometimes called “whiteboxing” (West, 2021). Doing so gives a good idea of the general layout of the environment and the approximate size of assets in it. A mockup environment can be easily modeled for example using Blender and placing some basic shapes in the scene, representing the final assets.

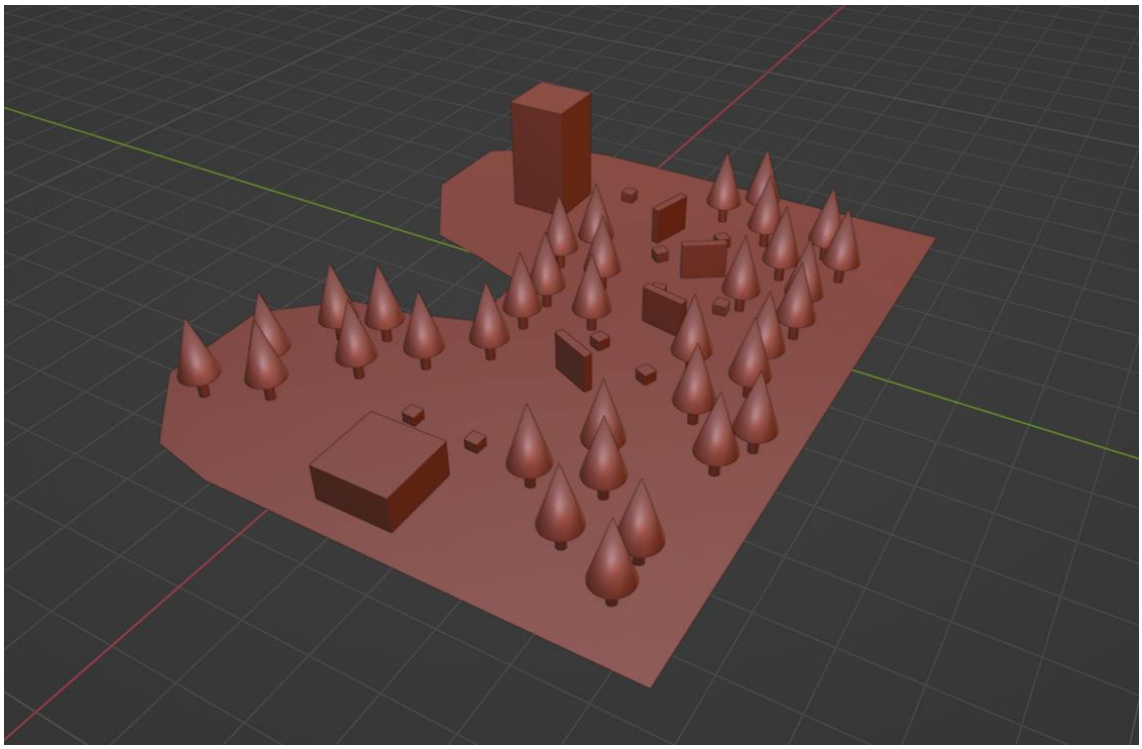

IMAGE 2. Simple environment prototype created by whiteboxing in Blender

Whiteboxing is especially important when designing a level for a video game. It helps the developers to plan the placement for buildings, obstacles, points of interest, enemies, and collectibles. Using this prototype, they can playtest the mechanics of the level before committing to the layout and replacing temporary assets with final 3D assets. This kind of prototyping also enables the developers to assess the flow and pacing of the level, helping to make the final environment engaging and challenging.

3 CREATING ASSETS

3.1 Modeling assets

Once the planning and prototyping is complete, the 3D artist has a clear idea of what they are going to create. The next step is to start the modeling process using any specialized software. The 3D model is created by constructing the asset's geometry according to the reference picture or concept art, setting up its dimensions and detailing. (Abhinay, 2022.)

In Blender, 2D reference pictures can be imported into the scene, so modeling from reference can be done easily.

IMAGE 3. A torii gate modeled on top of the reference picture

This asset's polycount is 728, so the geometry needs no optimization as the game engine will have no problem rendering it in real time.

3.1.1 Foliage

When modeling foliage, there are multiple specialized programs to speed up the process. In this project, Tree It was the program used to create the trees. It is a free and open-source tree generator software that allows the user to create 3D trees to use in their projects. Tree It generates realistic-looking trees procedurally, meaning the user can chance the parameters to create and customize the trees to their liking. There are also exportable textures available for the generated trees.

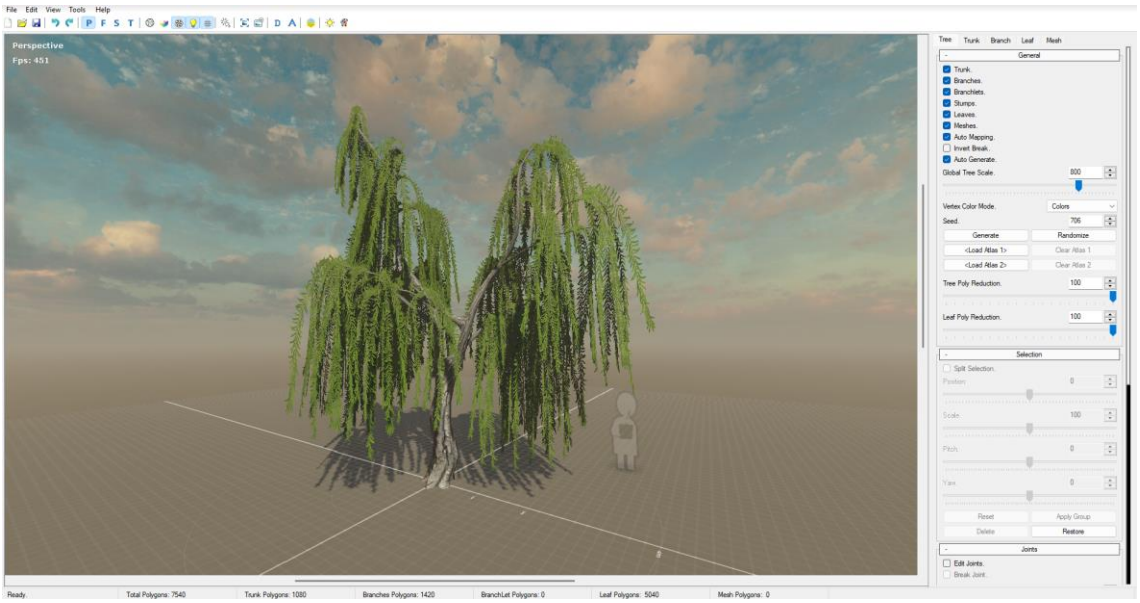

IMAGE 4. A default willow tree in Tree It-tree generator

Tree It allows the user to export their generated trees in formats, that can be imported easily into other 3D software. The polycount of these trees can get large quite fast, especially if the user wants to create something very realistic looking. This foliage gets hard to render for a game engine because of the amount of geometry it needs to calculate in real time. This causes performance issues, lowers the FPS and makes the finished game environment lag. However, when using Unreal Engine 5.1 the developers don’t have to worry about the polycount as much, as the game engine has its own way of handling large amounts of geometry.

In addition to the trees, there is a fern model used in the project. This model is very simple, constructed only with one sided planes to keep the geometry to a minimum. Like the tree leaves, the fern gets its realistic look from the material used instead of a complex, high poly model. As the Unreal Engine can make the materials double sided, there is no need to have geometry on both sides of the leaves.

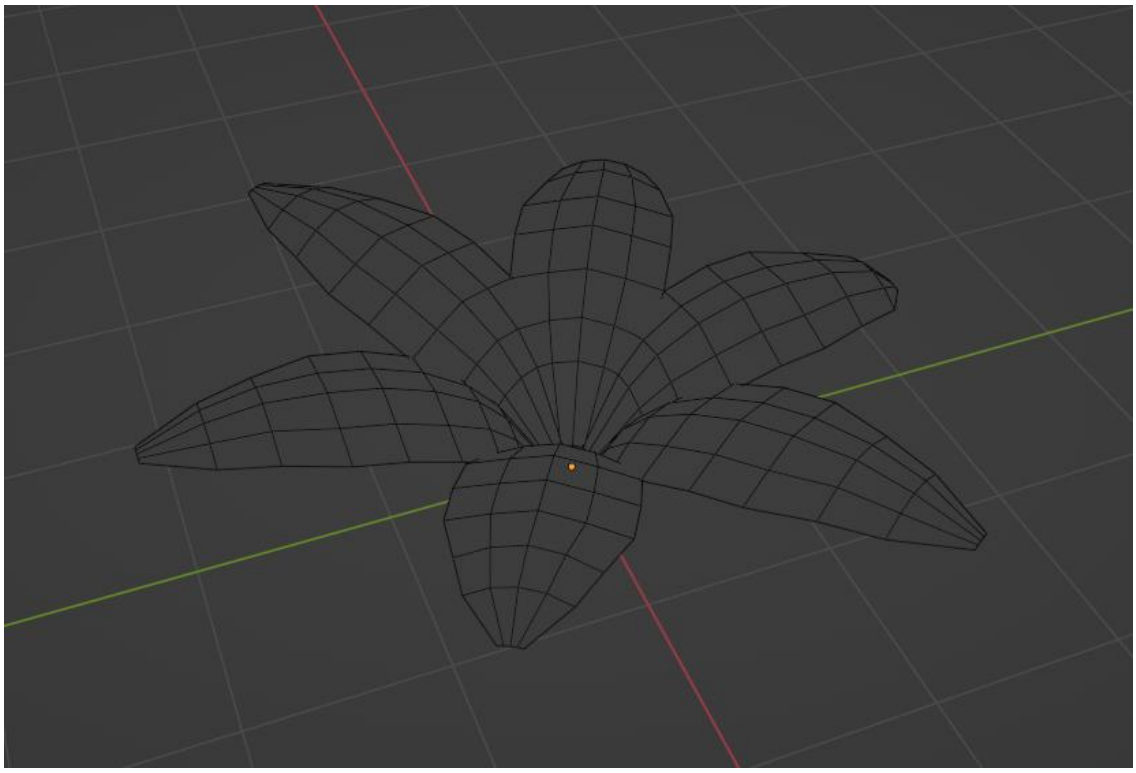

IMAGE 5. A very simple low poly 3D model of a fern

The fern leaves have a little variety on their geometry and size to make the plant look organic and realistic. Breaking the symmetry in realistic 3D models will make the asset look more credible and real, as there is not much real symmetry in nature.

3.1.2 UV mapping

UV mapping is the process of flattening a 3D object into 2D space so that texture maps can be applied. (Treehouse, 2018). When a 3D object is modeled, it has a geometry defined by vertices, edges and faces.

UV mapping defines the corresponding points between the 3D object and 2D texture. This phase lets the 3D object know how to apply the texture to itself.

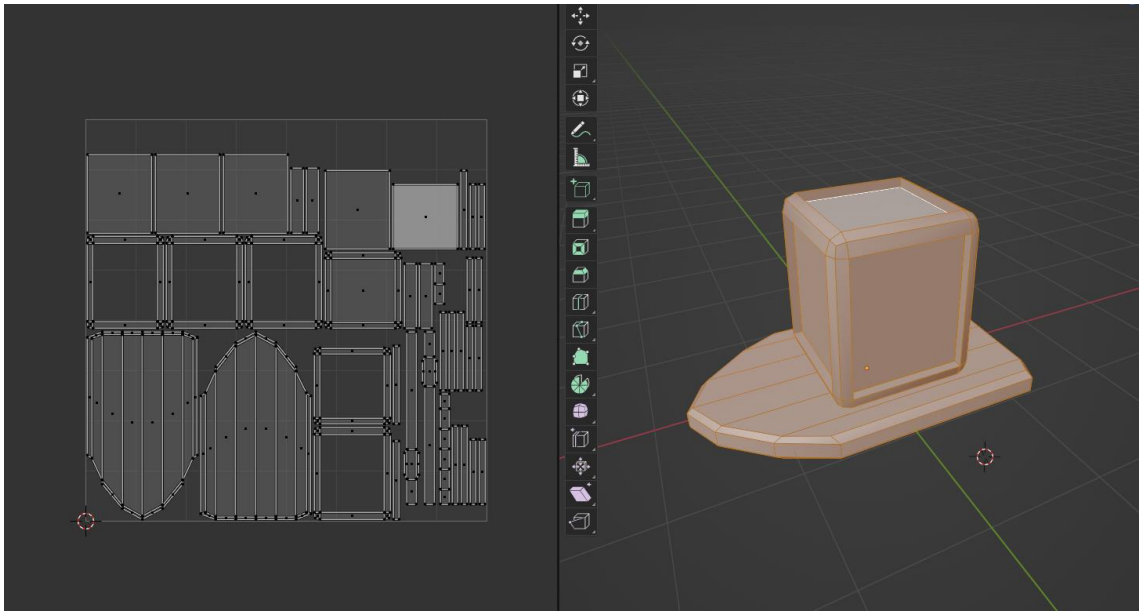

IMAGE 6. Unwrapped UVs of the floating lantern

When modeling objects with a simple geometry and only one texture, UV mapping can be an easy and automatic task, but with more complex models it needs to be done manually. This process includes marking the seams in the 3D model, and then flattening it to create a 2D texture map. Once the UV mapping is done to the mesh, it is unwrapped. Then the 2D texture created is ready to be texture painted.

When creating an asset with multiple materials such as wood, metal and rubber, every material instance needs to be defined as such. This can be done by first marking the seams to where they need to be, then individually UV unwrapping each material into the same 2D texture sheet. This way multiple materials can be baked to a single texture, reducing the storage required in the final game, improving rendering performance. (Treehouse, 2018).

3.2 Materials

As 2D texture is an image that creates a specific property on a 3D objects surface such as color, metallic and roughness, a material is a collection of these effects. A material is a set of instructions for how the object should look like with different lightings and angles. For example, when a bumpy textured mate-rial is added to a smooth object, the light interacts with it in such a way that it creates shadows, and the object looks bumpy instead of smooth. This is just an illusion, as the object itself still lacks any surface texture.

Textures can be created by multiple different techniques, including photography, 3D sculpting, digital painting, procedural generation, and displacement, all depending on the type of texture and the desired effect. (Shahbazi, 2023). As a potent and well-known tool in the field, this project used Adobe Substance 3D Painter as a tool for creating materials. It makes the process of producing high-quality effects simple because it enables painting materials directly onto a 3D object.

3.2.1 Texturing in Substance 3D Painter

First step is to import the 3D model into the software. Usually the FBX, or FilmBox format is the most efficient to use, as it contains a wide variety of data used with 3D. It is also supported by most 3D software, making it a standard format. When the model is imported to Substance Painter, is needs to be baked so that certain texture maps can be created. Then the model is ready for texturing.

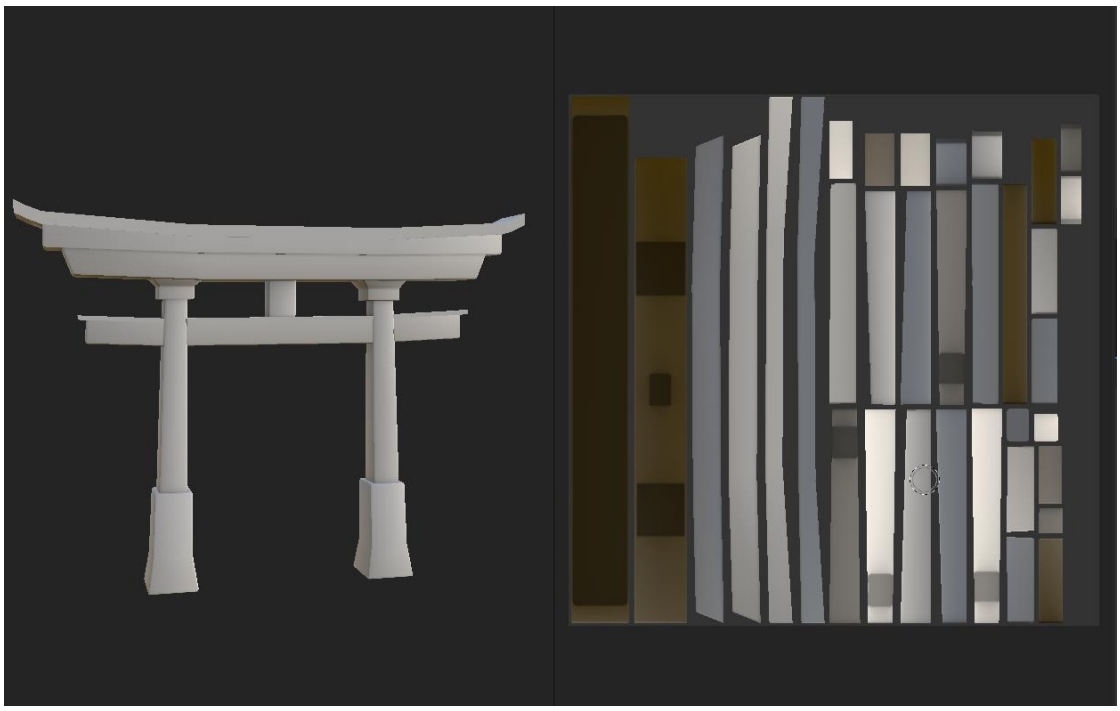

IMAGE 7. 3D model with baked ambient occlusion

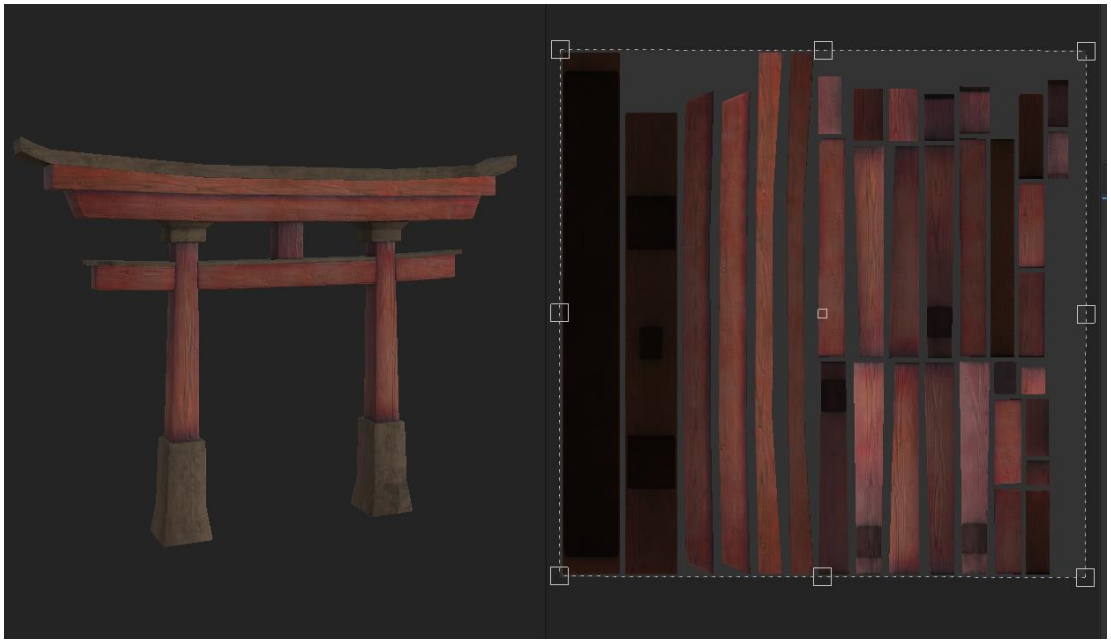

IMAGE 8. 3D model with wood materials

3.2.2 Exporting textures

When the texturing is complete, the materials need to be exported. Substance 3D Painter supports a variety of formats, such as .sbsar, .png, .bmp and .exr. In this project .png was used. After choosing the format and the output texture size, the texture images are exported to the chosen location.

4 CREATING A PROJECT IN UNREAL ENGINE

4.1 Setting up a new project

This project was started with a default first person template using blueprints. The template includes a player character which is viewed from first person perspective. The character can move and jump around the level using keyboard or other controllers. Starter content was also included, although most of it can later be removed as there are just a few assets used. Project's target platform was desktop, and the quality preset was set to maximum.

When the project has loaded, a new basic level is created. The default map can then be deleted from the project to prevent it using excessive space from the hardware.

4.1.1 Landscaping

When starting to create an island to a default basic level, the floor is unnecessary and can be deleted. After navigating to the landscape mode, the edit layers must be enabled. Below this option, there are settings that allow to change the size, location, and rotation of the landscape. Once these parameters have been adjusted, the landscape can be created and subsequently sculpted and painted as desired.

Now it is time to create the landmass. This method of making an island using a blueprint brush requires the enabling of the Landmass-plugin. Using a CustomBrush_Landmass blueprint brush is an easy way to create a shape for an island. As the landscape is clicked with the blueprint brush, it creates a pyramid shape. This shape can be altered from the details panel of the landmass, under the falloff -dropdown menu. Selecting the cap shape -option, the pyramid turns into a flat surface. Increasing the Z offset value makes the surface higher, making it resemble more of an island. Now moving the spline points alters the island shape. More spline points can be created by holding down alt key while dragging a point, helping create a more complex and natural island shape.

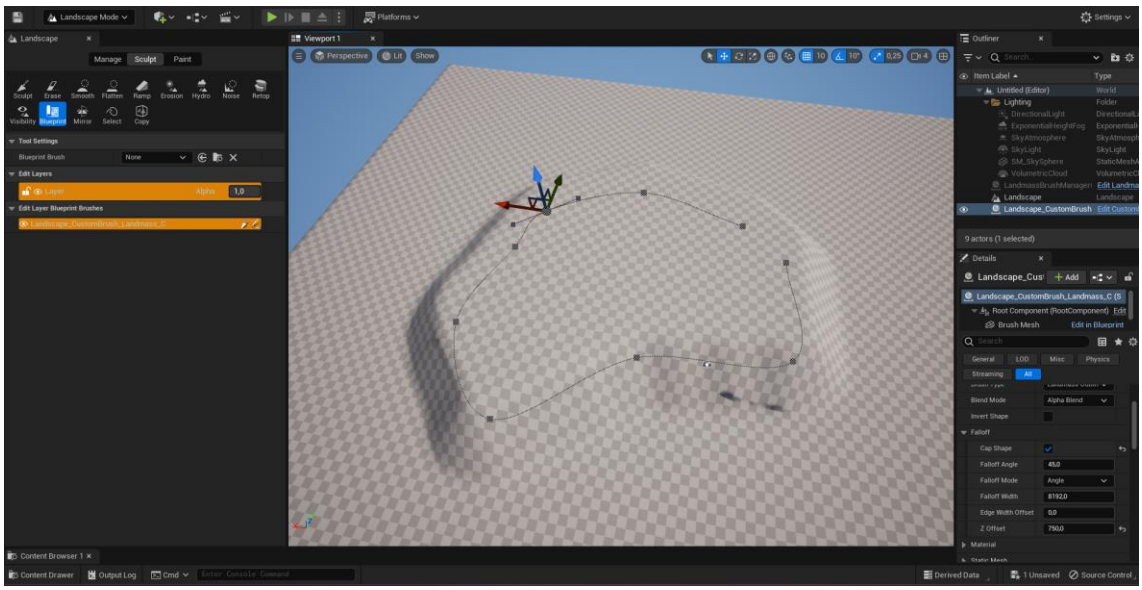

IMAGE 9. An island shape made using a blueprint brush

There are different options for making the shape look more organic and realistic. Adjusting some parameters in the details panel can turn a simple and basic shape into something more credible fast and easy. This can also be done by sculpting, but it can be a lot more tedious and time-consuming. Adjusting curl noise values makes the edges break, introducing some more complex shape and changing terracing strength makes the island surface have some texture instead of being completely flat. Displacement parameter also gives the surface some detailed texture, making it even more realistic looking. These values can be changed freely depending on what the desired outcome is.

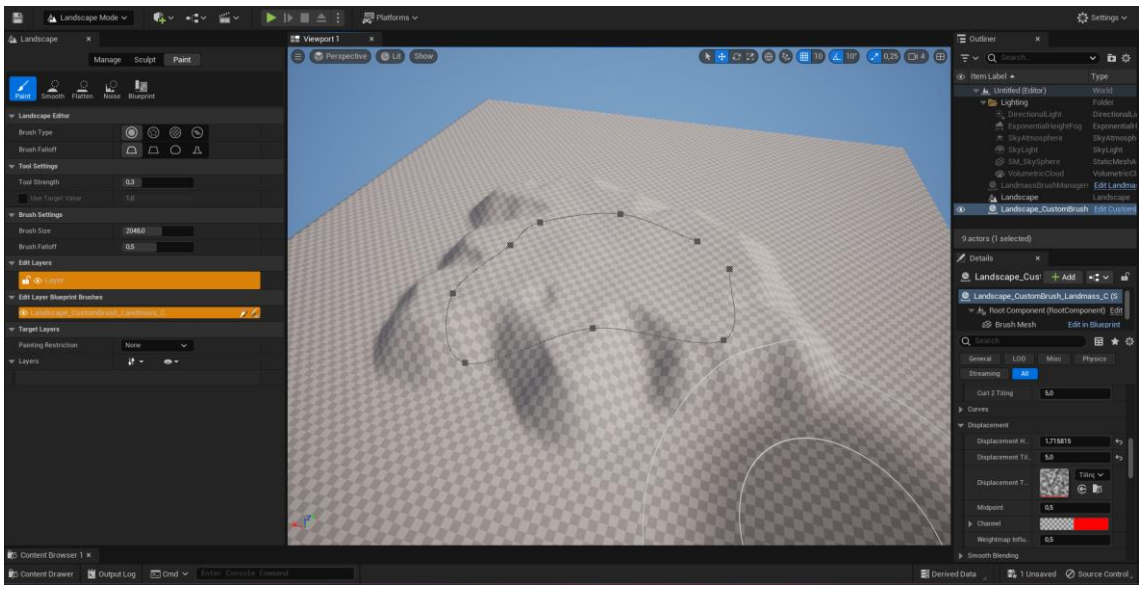

IMAGE 10. The same island with adjusted parameters

4.1.2 Painting the landscape

First step in painting the grass, dirt and rock to the landscape is to create a material for it. Because only one texture image can be plugged into the materials output node, a LandscapeLayerBlend -node needs to be plugged to the base color to be able to add all the textures to the material. In the LandscapeLayerBlend -node's details panel, three elements are added to be able to plug the grass, dirt, and rock textures in. The texture images used in this project are available in the project's starter contents, but any other textures can also be imported depending on the desired outcome.

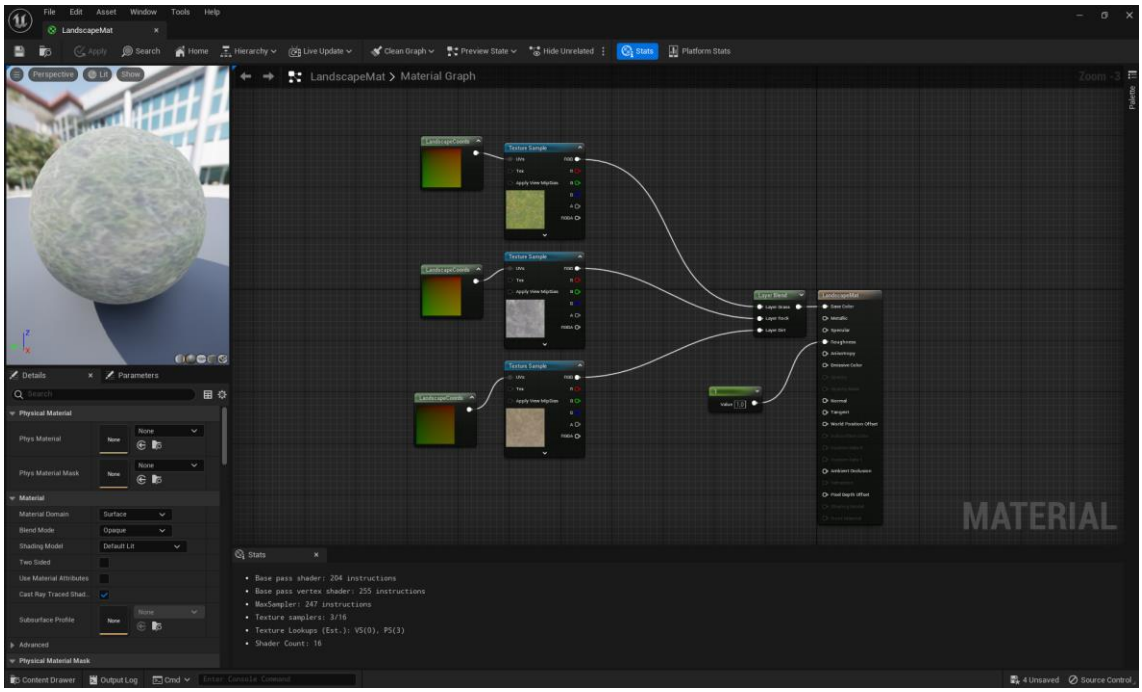

IMAGE 11. Landscape material with reduced tiling and increased roughness

After adding the material to the landscape, the layers need to be set up in the landscape mode. After changing each layer to a weight-blended layer, they are ready to be painted on to the landscape. Choosing a texture layer and a painting with a brush, the texture is applied straight on the landmass.

4.1.3 Adding the foliage

When making a realistic environment, foliage assets such as grass can be tedious to model. There are different methods in making real looking plants, but in this case the realism is achieved by materials.

Quixel is an organization also owned by Epic Games. They make a product called Megascans, which is an online library of high-quality assets created with a method called photogrammetry. It involves the scanning of real-world objects and turning them into textured 3D models ready to use in various projects. They offer a vast amount of free-to-use assets, like the grass added to this project.

Foliage can be added to a landscape in Foliage Mode. Dragging the meshes to the foliage library allows them to be painted directly to the ground using a brush, lasso tool or fill tool. It is important to set the collisions of the foliage from their foliage settings to match the wanted behavior, so that the player can for example walk through the grass and not go straight through the trees. Tweaking the settings of the foliage mesh by changing the scale range and other variables can make the outcome even more realistic. From there it is completely an artistic choice as to where paint different foliage, rocks and other meshes.

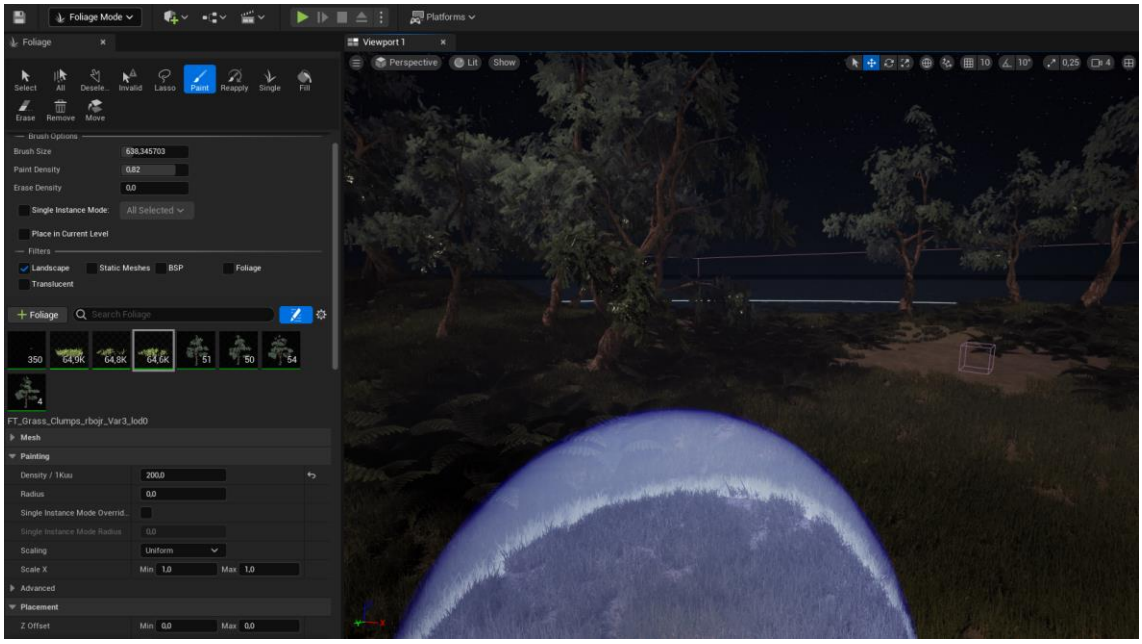

IMAGE 13. Foliage Mode viewport and painted foliage

4.2 Importing assets

When the 3d model is textured and exported, it can be imported to the Unreal Engine project. Choosing the appropriate location in content browser is important for clean and logical folder structure, so everything is found easily. Then by right clicking in the right location, the asset can be added to the project library. Dragging the asset to the project scene gives a responsive visual on how the asset looks on the game when modifying the settings. Importing the texture images works the same way.

Double-clicking the asset in content browser opens the asset editor. There are various settings including material, Nanite, collision and level of detail. The editor also displays the model's info and polycount. In Unreal Engine it is recommended to enable the Nanite support, as it increases the performance of the game by rendering complex geometries in real-time efficiently. It removes the need for traditional LODs and manual optimization techniques and allows the use of high-poly models without excessive memory usage or performance issues.

4.2.1 Collisions

Collisions are used in determining how objects interact physically with the environment. They enable the assets to respond to physical interactions with the player, projectiles, or other objects.

Collisions can be added to an asset in the asset editor window. Generally, it is recommended to use the simplified shape collisions, as they are not as computationally expensive. Sometimes though it is necessary to use a complex collision for more detailed result. After adding e. g. a simple box collision, the shape and size of the box can be modified to suit the geometry of the mesh. After the primitives are set, the collision can be applied to the asset. This generates the collision data based on the settings. The collision should then be tested in the game to check for any possible errors and make sure that they provide accurate interactions.

4.2.2 Creating materials

Unreal Engine 5.1 is unique and highly flexible in material creation as it allows for complex visual effects. Materials can be used to create stunning visual effects, such as leaves floating on water or swaying in the wind. The latter is a feature used in this project and will be explained more later.

Navigating to the appropriate folder in the content browser and right-clicking allows the creation of a basic material that is empty for now. Double-clicking the material opens the material editor. By default, it contains one output node. This is where more nodes can be added by right clicking the empty area and connected to the output to make stunning materials.

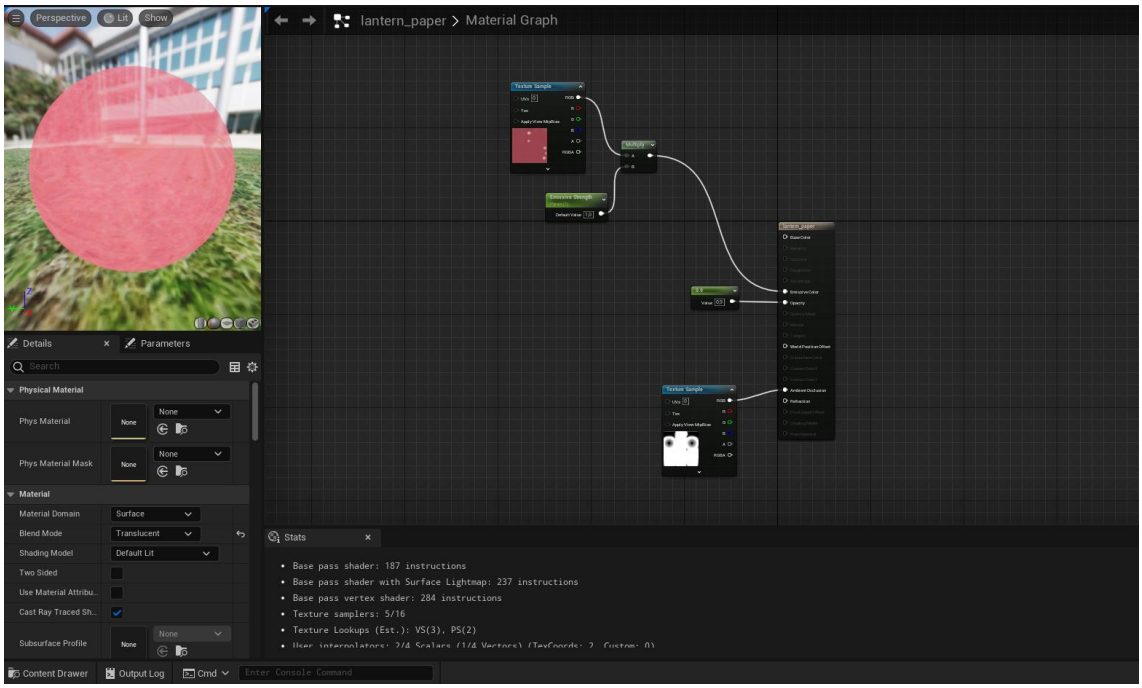

IMAGE 14. A simple translucent material for a paper lantern

The paper lantern material above is set to a translucent blend mode. It makes the material see-trough even though the mesh it is applied to is solid. Adding a parameter node and connecting it to the opacity output modifies the transparency to a wanted amount. The texture sample-node is the base color of the mesh, and it is combined with an emissive strength-node to make the lantern glow like it is lit. This gives the illusion of light without adding a light actor to the scene, making the game more lightweight to compute.

A material structure used most in this project is simple. It contains only the texture images made earlier. The appropriate images are connected to the output node, depending on what textures the material uses. The blend mode of these materials is opaque by default, as it is the mostly used option giving the mesh a solid material. Materials such as leaves and ferns that use a .png as a base color must have the blend mode set to masked to work properly. From the left side panel, the two-sided parameter can be checked to make the leaves double sided, as this option renders the material on both sides of a plane.

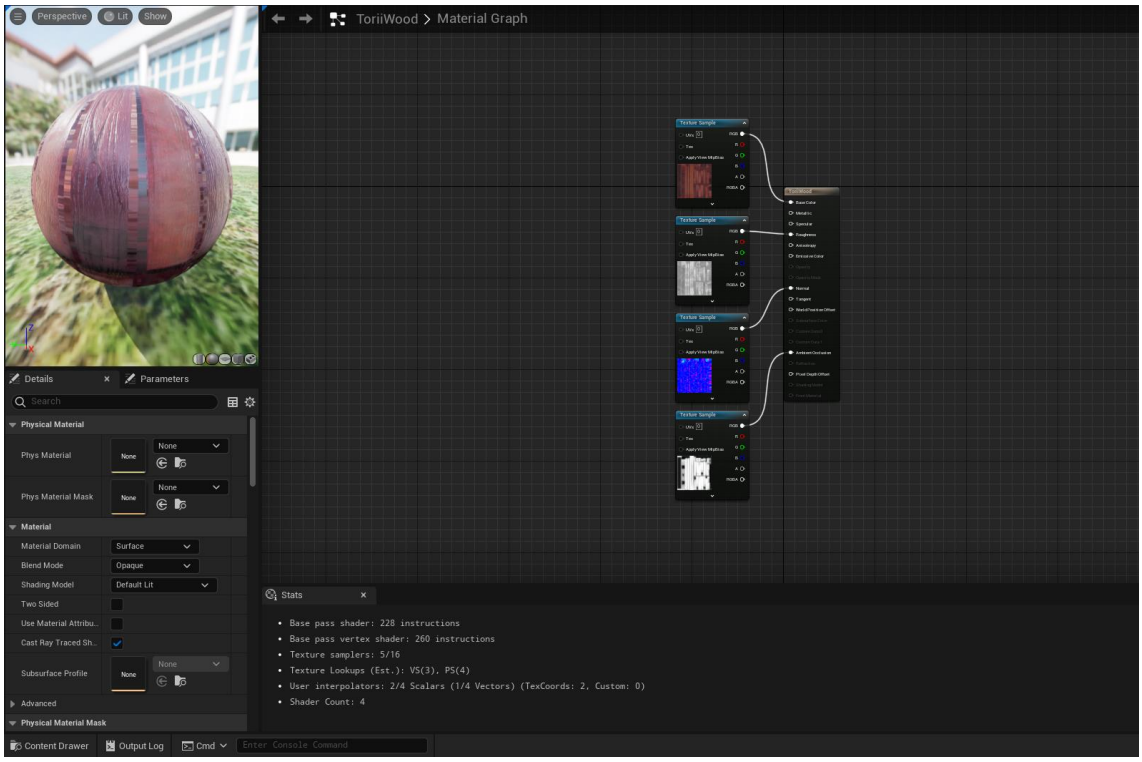

IMAGE 15. A wood material for the torii gates using texture images

As mentioned before, the Unreal Engine's material system opens possibilities to create visual effects. As an example of one, the trees and their leaves can be made to sway in wind, giving them a more realistic touch. The process is a little more intensive, but the main idea is to create a vertex offset with nodes that mimics the movement of the leaves. Parameters can be created and utilized in controlling the wind speed and direction of the effect. Adjusting the parameters as desired gives the effect variability.

4.3 Visual effects and simulation

4.3.1 VFX

Niagara is a VFX editor inside Unreal Engine. It allows the creation of complex and dynamic particle effects and simulations. It is based on nodes that control the visual effects, giving users control over the appearance and behavior of the effect. Niagara can be utilized in creating different effects. In this project the fire and fireflies are made in Niagara. If not already enabled, the Niagara plugin might need to be added to the project.

Adding a new fountain Niagara system by right clicking in the content browser and double-clicking it opens the Niagara editor. Making a fire simulation needs a few different emitters, starting from the flame emitter. Renaming the default emitter to flames, changing the particle shape to a flame texture image, and tweaking various settings of the emitter such as the spawn rate and sprite size and color makes a realistic looking fire. Now more effects can be added to the simulation. A real fire emits smoke and fire debris and makes a slight heat distortion to the surrounding air. All these can be added inside the Niagara editor by creating more emitters and modifying their parameters.

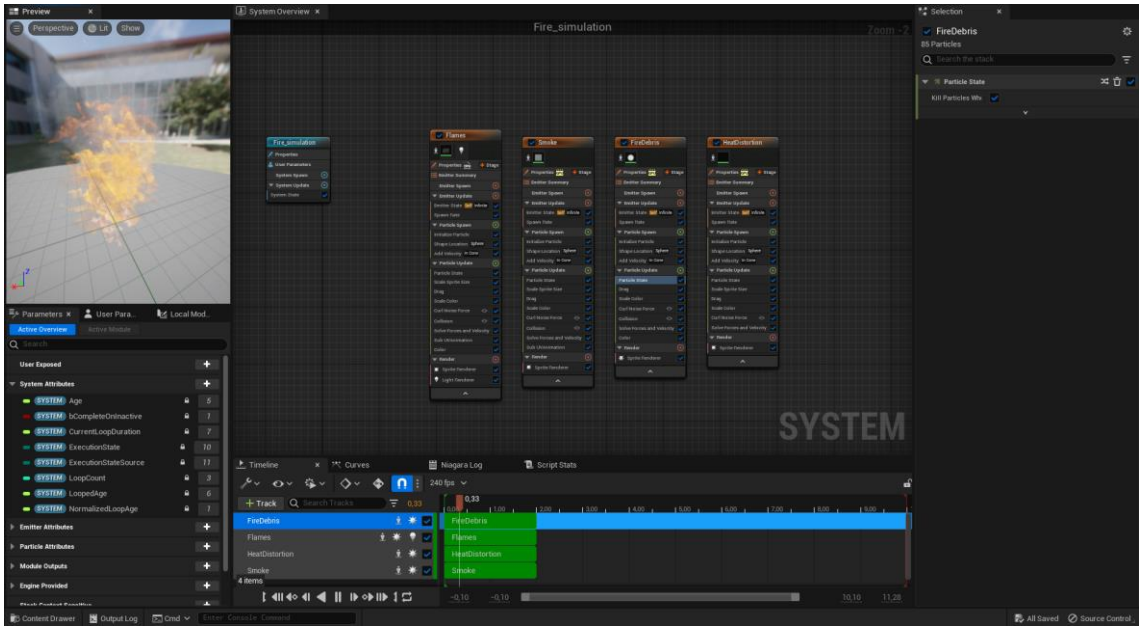

IMAGE 16. A fire simulation made in Niagara editor

The firefly simulation is a simple fountain emitter with modified parameters. The appearance of the sprites is changed to make them vary in size and color. This makes the effect a little more interesting to the eye. The sprites are made to spawn slowly and in a large area to make them look realistic.

These are only some of the effects possible to create with Niagara. There are several ways to utilize the editor and create gorgeous visual effects.

4.3.2 Cloth simulation using Chaos

This project includes pieces of cloth that interact with the wind. Creating a cloth simulation can be complex and time consuming, as it requires a notable amount of experimentation and fine tuning to work as desired.

First step is to create a mesh for the cloth. It is important to note that the mesh should be flat to prevent any glitching in the simulation. A flat subdivided plane should work. This is another asset that the double-sided material can be enabled for. When importing the mesh to Unreal Engine, it needs to be set as a skeletal mesh. From the editor a new clothing data can be added by right clicking the mesh. After applying the clothing data to the mesh, the cloth paint option can be activated. Painting the cloth controls how far each vertex can move away from the skinned mesh. A setting of 0 causes no simulation to be calculated, and a setting greater than 0 allows the mesh to move based on the max distance determined. (Stellwag, J. 2022.)

Dragging the skeletal mesh actor into the scene and simulating the game makes the mesh act like a cloth. A wind directional source can be added and positioned so that the cloth moves in the wind. Now the cloth simulation parameters will most likely need to be modified, depending on what material the cloth is made of and the way it should act. These settings can be found in the skeletal mesh editor, under config.

4.4 Lighting and atmosphere

4.4.1 Sky and fog

Creating a credible lighting in Unreal Engine is tricky. It is not the most obvious to get right, and it poses its own set of challenges. (Faucher, W. 2022.) It plays a crucial role in visually interesting and beautiful environments. Lumen is an illumination system in Unreal Engine that calculates the lighting in real time without the need for lightmaps or pre-baking. It is designed to keep the computation cost low while providing a high-quality lighting and performance in the game.

When creating a night sky, multiple components are needed. A blueprint for a sky sphere can be found in Unreal Engine's starter content. It contains the horizon, stars, and clouds. Adjusting horizon and zenith colors changes the color of the sky. A night sky is created by turning the colors dark and adding brightness to the stars. The scene needs a light source to illuminate it. A directional light is a great and realistic way to light up the environment, as it can be rotated to behave like a sun or a moon. Modifying the parameters can change the color, temperature, and intensity of the light, making it suitable for the wanted effect.

When the sky is set up, adding some atmospheric height fog can really make a difference in terms of realism. It makes the sky seem more credible, as it creates a layer of fog interacting with the light that illuminates the scene. An additional exponential height fog adds a layer of fog to the ground. It makes a night scene look like the air is cooler, and the environmental light is reflecting to the ground from the moon. Changing the parameters of the fog actors give the scene a different look, depending on what the desired effect is.

A simple moon is created to the sky by adding a sphere mesh with a moon material. Adding some emission to the material makes the moon look like it is reflecting the sunlight, as it should.

4.4.2 Light actors

Unreal Engine has different options for lights. This project used mainly point lights, as they illuminate a sphere around them rather than directing the light at a certain direction. Dragging the lights to the scene add illumination, and modifying the light's parameters changes its features.

Lighting can be very heavy to render for the engine, reducing the performance and frame rate. Keeping track of the computing cost is important, and Unreal Engine provides a visualization mode for checking the light complexity. Ensuring that the lights don't overlap, and that the scene doesn't contain any unnecessary light actors is mandatory for a smooth performing project.

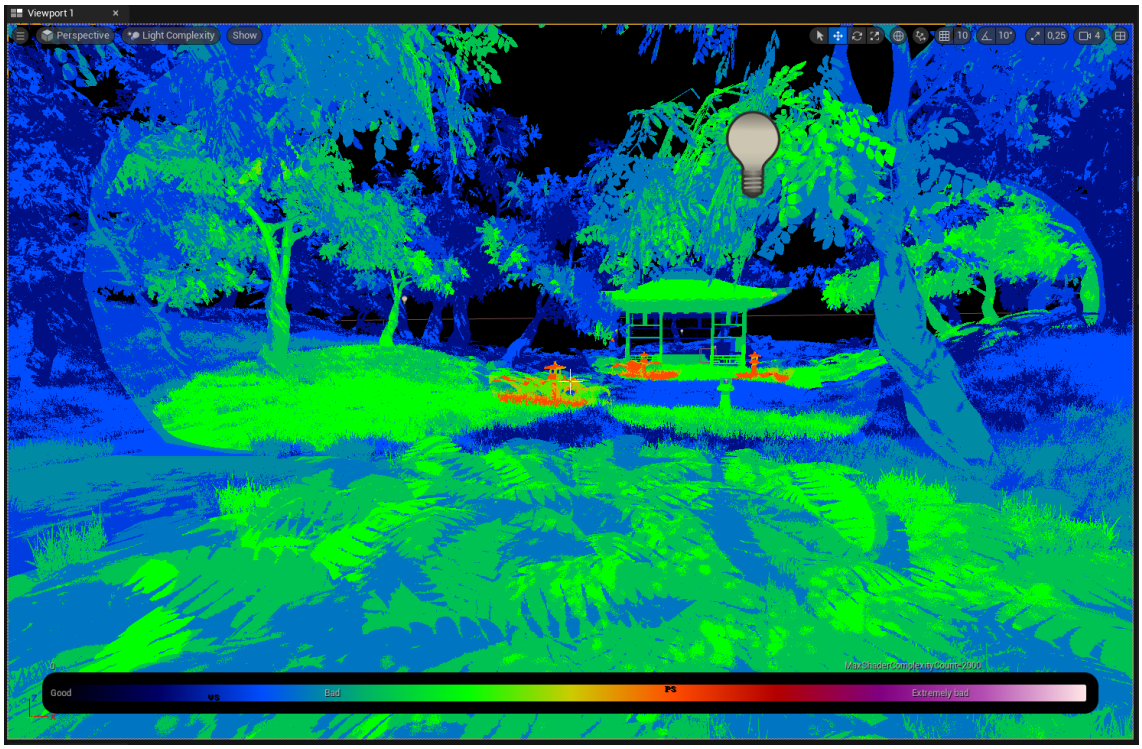

IMAGE 17. Light complexity visualization mode in Unreal Engine

4.5 Post processing

Finding and dragging a PostProcessVolume actor to the scene adds a box with adjustable settings for post processing. First notable thing is that the effect of the actor is limited to the cube representing it. From the details panel, the area of effect can be changed to infinite by checking the unbound box. Now it is a completely artistic choice as to what effects to add to the scene.

In this project the lighting was unstable in a way that made some places of the environment render pitch black. This problem was fixed in the post processing settings by adjusting the minimum and maximum exposure of the level to the desired amount. Colors were also adjusted to less saturated, and cooler toned to match the nighttime theme.

IMAGE 18. The scene with and without post processing

Post processing is a vital part of creating a stunning environment, as it enhances the visual quality of a scene. It enables the artist to enhance the final look and readability of their work, making the final experience more immersive and visually interesting.

5 CONCLUSION AND DISCUSSION

As technology advances, video games look better and better. It is increasingly easy to create beautiful environments, and the process gets faster with every new introduced feature. This opens opportunities for an artist to push the boundaries of what was previously possible. These new possibilities benefit both the developers and the players, making games more immersive and the player experience enhanced.

Getting into a new game engine can be very time consuming and difficult at the beginning, but staying consistent with learning is rewarding. With an engine like the Unreal Engine, there are limitless possibilities in creating. Following tutorials and reading the endless amounts of articles and manuals online make it possible to make even a complex environment with almost no previous experience. By utilizing the many robust tools of Unreal Engine 5.1, even someone with no coding skills can make their own game and bring their visions to life. While the creating process will almost certainly present some challenges and difficulties, there are many sources of learning surrounding the engine, making it an ideal choice for beginning developers to use.

The modeling and texturing software used in this project were Blender, TreeIt and Substance 3D Painter, but the result can be achieved by using any preferred programs. Every artist also has their own preferred art style. This project was created to be realistic, but the methods can be adjusted for different styles as well.

This thesis showed step-by-step the creating process of a game environment. It can be used as a gentle guide in projects made in Unreal Engine.

REFERENCES

Abhinay 2022 A detailed guide on Game Asset Creation & its workflow, Juego Studio. Available at: https://www.juegostudio.com/blog/game-asset-creation (Accessed: May 4, 2023).

Faucher, W. 2022 Lighting in Unreal engine 5 for Beginners, Evermotion.org. Available at: https://evermotion.org/tutorials/show/12868/lighting-in-unreal-engine-5-for-beginners (Accessed: 24 May 2023).

Mcdowell, K. 2022 Realistic Oceans & Lakes in Unreal engine 5, CGHero. Available at: https://cghero.com/tutorials/realistic-oceans-and-lakes-in-ue5 (Accessed: 24 May 2023).

Serrador, P. 2012 The importance of the planning phase to project success. Paper presented at PMI® Global Congress 2012—North America, Vancouver, British Columbia, Canada. Newtown Square, PA: Project Management Institute.

Shahbazi, N. 2023 3D texturing - A guide for beginners & professionals, Pixune. Available at: https://pixune.com/blog/3d-texturing/ (Accessed: 10 May 2023).

Stellwag, J. 2022 Unreal Cloth and hair simulation, CG GURU Icon. Available at: https://www.cgguru.com/unreal-cloth-and-hair-simulation (Accessed: 24 May 2023).

Treehouse 2018 Asset workflow for game art: 3D modeling, Treehouse Blog. Treehouse. Available at: https://blog.teamtreehouse.com/asset-workflow-game-art-3d-modeling (Accessed: May 5, 2023).

West, C. 2021 White Boxing? Grey Boxing? What?, Available at: https://gamedevchris.medium.com/white-boxing-grey-boxing-what-6aa9cfa3b0e4 (Accessed: May 3, 2023).